Package: ooacquire (via r-universe)

July 6, 2024

Type Package

Title Acquire Data from OO Spectrometers

Version 0.4.6

Date 2024-03-29

Description Functions to acquire data directly from Ocean Optics spectrometers, and functions to read similar data from files. Functions to convert raw-counts into counts-per-second and physical quantities. Data are saved in objects of classes defined in package 'photobiology'. The instrument settings, instrument description, date-time of acquisition and optionally goecode are stored as attributes.

License GPL (>= 2)

Depends R (>= 4.2.0), photobiology (>= 0.11.1), ggspectra (>= 0.3.12), lubridate (>= 1.9.0)

Imports grDevices, utils, stats, methods, tibble (>= 3.1.7), ggplot2 (>= 3.4.0), caTools (>= 1.18.2), Rcpp (>= 1.0.10), photobiologyWavebands (>= 0.5.2), photobiologyInOut (>= 0.4.26), anytime (>= 0.3.9), polynom (>= 1.4-1), stringr (>= 1.5.0), dplyr (>= 1.1.1), readr (>= 2.1.4), magrittr (>= 2.0.3)

Suggests rOmniDriver (>= 0.1.20), mirai (>= 0.11.3), rstudioapi (>= 0.10), knitr (>= 1.45), rmarkdown (>= 2.25), testthat (>= 3.1.4), spacesXYZ (>= 1.2-1)

LazyData yes

LazyLoad yes

ByteCompile true

LinkingTo Rcpp

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

BugReports `https://github.com/aphalo/ooacquire/issues`

RoxygenNote 7.3.2

Encoding UTF-8

VignetteBuilder knitr

Repository https://aphalo.r-universe.dev

RemoteUrl https://github.com/aphalo/ooacquire

RemoteRef HEAD

RemoteSha 519a3e72881d5dc69fc01664fe2f40fd444ecb02

Contents

`ooacquire-package` *ooacquire: Acquire Data from OO Spectrometers*

Description

Functions to acquire data directly from Ocean Optics spectrometers, and functions to read similar data from files. Functions to convert raw-counts into counts-per-second and physical quantities. Data are saved in objects of classes defined in package 'photobiology'. The instrument settings, instrument description, date-time of acquisition and optionally goecode are stored as attributes.

Details

Processing of raw-counts data acquired following different protocols and corrections including integration-time bracketing and merging, slit function corrections, subtraction of stray light and adaptive smoothing corrections.

Acquisition of raw-counts spectra directly from Ocean Optics' spectrometers. A high level interface to the OmniDriver library for spectrometer control and data acquisition provided by Ocean Optics for their instruments. Built on top of the 'rOmniDriver' package which provides a very thin wrapper on the Java functions giving access to the proprietary driver.

Reading of raw-counts spectra from files saved by Ocean Optics' software: OceanView, Spectra-Suite, Raspberry Pi SDK, and Jaz firmware.

Note

This package is provided **without any warranty of suitability for any specific purpose or instrument** as it has been tested with only three different models of Ocean Optics spectrometers, each with one or two of many available hardware configurations.

Author(s)

Maintainer: Pedro J. Aphalo `<pedro.aphalo@helsinki.fi>` [translator]

Authors:

- Lasse Ylianttila

Other contributors:

- Titta K. Kotilainen [contributor]

See Also

Useful links:

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

acq_fraction_interactive

Acquire spectral fraction

Description

Interactive front-end allowing acquisition of spectral fractions using Ocean Optics spectrometers. Output of spectral data in R data files stored in objects suitable for use with packages 'photobiology' and 'ggspectra' as well as plots as PDF files and summaries as comma separated files.

Usage

```
acq_fraction_interactive(
  tot.time.range = c(5, 15),
  target.margin = 0.1,
 HDR.mult = if (light.source == "pulsed") c(short = 1) else c(short = 1, long = 10),
  protocols = NULL,
  correction.method = NA,
  descriptors = NA,
  stray.light.method = "simple",
  seq.settings = NULL,
  light.source = "continuous",
  ref.value = 1,
  qty.out = "Tfr",
  type = "total",
  plot.lines.max = 11,
  summary.type = "VIS",
  save.pdfs = TRUE,
  save.summaries = !interface.mode %in% c("series", "series-attr"),
  save.collections = !interface.mode %in% c("simple", "series", "series-attr"),
  async.saves = FALSE,
  show.figs = TRUE,
  interface.mode = ifelse(light.source == "pulsed", "manual", "auto"),
  num.exposures = ifelse(light.source == "pulsed", 1L, -1L),
  f.trigger.pulses = f.trigger.message,
 folder.name = paste("acq", qty.out, lubridate::today(tzone = "UTC"), sep = "-"),
  user.name = Sys.info()[["user"]],
  session.name = paste(user.name, strftime(lubridate::now(tzone = "UTC"),
    "%Y.%b.%d_%H.%M.%S"), sep = "_"),
  verbose = getOption("photobiology.verbose", default = FALSE),
  QC.enabled = TRUE
)
```

Arguments

`tot.time.range`	numeric vector Range of total times for a measurement in seconds.
`target.margin`	numeric (0..1) when tuning integration time, how big a head space to leave.

`HDR.mult`	numeric the integration time for each bracketed integration as a multiplier of the set or tuned integration time.
`protocols`	named list of character vectors, or a character vector with names of at least one member of the default list of protocols.
`correction.method`	list The method to use when applying the calibration
`descriptors`	list A list of instrument descriptors containing calibration data.
`stray.light.method`	character Used only when the correction method is created on-the-fly.
`seq.settings`	named list with numeric members `start.boundary`, `initial.delay`, `"step.delay"` and `"num.steps"`.
`light.source`	character One of "continuous", "pulsed".
`ref.value`	numeric or filter_spct/reflector_spct object.
`qty.out`	character One of "Tfr" (spectral transmittance as a fraction of one), "cps" (counts per second), or "raw" (raw sensor counts).
`type`	character Type of transmittance or reflectance measured.
`plot.lines.max`	integer Maximum number of spectra to plot as individual lines. Random sampling is used if number of spectra exceeds `plot.lines.max`.
`summary.type`	character One of "plant", "PAR" or "VIS".
`save.pdfs`, `save.summaries`, `save.collections`	logical Whether to save plots to PDFs files or not, and collection summaries to csv files or not, enable collections user interface or not.
`async.saves`	logical A flag enabling or disabling the use of concurrent processes to save data to files. Package 'mirai' must be installed before enabling this feature.
`show.figs`	logical Default for flag enabling display plots of acquired spectra.
`interface.mode`	character One of "auto", "simple", "manual", "full", "series", "auto-attr", "simple-attr", "manual-attr", "full-atr", and "series-attr".
`num.exposures`	integer Number or light pulses (flashes) per scan. Set to `-1L` to indicate that the light source is continuous.
`f.trigger.pulses`	function Function to be called to trigger light pulse(s). Should accept as its only argument the number of pulses, and return `TRUE` on success and `FALSE` on failure.
`folder.name`, `session.name`, `user.name`	character Default name of the folder used for output, and session and user names.
`verbose`	logical If TRUE additional messages are emitted, including report on memory usage.
`QC.enabled`	logical If FALSE return NA skipping QC.

Details

This function can be used to acquire spectral reflectance, spectral transmittance and/or spectral absorptance using different protocols for acquisition and stray light and dark corrections. Depending on the optical setup, solid or liquid samples can be measured. The kinetics of changes in optical properties can be captured as a time series of spectra, using ‘interface.mode = "series"‘.

The protocols are described in the vignettes and in the help for the lower level functions called, also from this same package.

Using this function only requires an Ocean Optics spectrometer to be connected to the computer where R is running and the OmniDriver runtime from Ocean Insight installed. The connection to the spectrometer and selection of channel, when relevant, is done from within these functions. A stable and continuous source of light is also needed as well as black, white and possibly grey reflectance patches.

The calculations for reflectance and transmittance are very similar, so we provide a single function capable of handling both. For transmittance the reference is usually direct exposure to radiation but for reflectance a white reference patch is normally used. In some cases one may want to use a grey reference. We provide an argument that allows the user to supply a constant or a spectrum describing the properties of the reference. It is also important to distinguish between total and internal transmittance, and between total and specular reflectance. In both cases which of these is measured depends on the measuring protocol (condition used as reference, use of an integrating sphere versus use of a probe with a narrow angle of aperture, etc.) and consequently the correct value should be entered to ensure that data are correctly tagged and later computations valid.

A wavelength calibration is needed, but being the measurements relative, no calibration of pixel responsiveness is required. A known linearization function is also needed.

Some protocols are available by default. They differ in the additional measurements done to correct for stray light and dark noise. The default protocols are usually suitable, if new protocols are passed, each character vector must contain strings "light", "filter" and "dark".

By default the integration time is set automatically so that the number of counts at the highest peak is close to 1 - `target.margin` times the maximum of the range of the instrument detector (retrieved from the calibration or the instrument memory). The minimum `tot.time` is obtained by increasing the number of scans. The maximum integration time supported by the spectrometer is not exceeded.

Plots are produced with functions from package ’ggspectra’ and respect the default annotations set with function `set_annotations_default()`, and default wavebands set with function `set_w.band_default()`.

The different interface modes available are suitable for different types of measurements.

Value

These functions return the acquired spectra through "side effects" as each spectrum is saved, both as raw counts data and optionally as spectral transmittance or counts-per-second data in an `.rda` file as objects of the classes defined in package ’photobiology’. Optionally, the plot for each spectrum is saved as a `.pdf` file. At any time, the current group of spectra can be saved as a collection. When a collection is created, spectral data for several spectra are saved together. Summaries are saved to a CSV file and joint plots to a `.pdf` file. The value returned by the function is that from closing the connection to the spectrometer.

Note

Calibration data needs in most cases to be imported into R and parameters entered for the special correction algorithms into a correction method descriptor. The corrections are skipped if the needed information is missing. If no wavelength calibration is available and attempt is made to retrieve it from the spectrometer.

The function is composed in a modular way from functions that can be reshuffled and combined with other functions to define new variations possibly better suited to users' needs and tastes. Even easier is to simply change the default arguments in a wrapper function or in a script.

See Also

This function calls functions `tune_interactive`, `protocol_interactive` and `set_attributes_interactive`.

Other interactive acquisition functions: `acq_irrad_interactive()`

Examples

```
# please, see also the example scripts installed with the package

## Not run:
# requires an Ocean Insight (former Ocean Optics) spectrometer to be
# connected via USB

acq_fraction_interactive()


## End(Not run)
```

`acq_irrad_interactive` *Acquire spectral irradiance or spectral fluence*

Description

Interactive front-end allowing acquisition of spectral irradiance and spectral fluence using Ocean Optics spectrometers. Output of spectral data in R data files stored in objects suitable for use with packages 'photobiology' and 'ggspectra' as well as plots as PDF files and summaries as comma separated files and R objects.

Usage

```
acq_irrad_interactive(
  tot.time.range = if (qty.out == "fluence") 5 else c(5, 15),
  target.margin = 0.1,
  HDR.mult = if (qty.out == "fluence") c(short = 1) else c(short = 1, long = 10),
  protocols = NULL,
  correction.method = NA,
  descriptors = NA,
  entrance.optics = NULL,
  stray.light.method = "none",
  seq.settings = NULL,
  area = NULL,
  diff.type = NULL,
  qty.out = "irrad",
  plot.lines.max = 11,
  summary.type = "plant",
  save.pdfs = TRUE,
  save.summaries = !interface.mode %in% c("series", "series-attr"),
  save.collections = !interface.mode %in% c("simple", "series", "series-attr"),
  async.saves = FALSE,
  show.figs = TRUE,
  interface.mode = ifelse(qty.out == "fluence", "manual", "auto"),
  num.exposures = ifelse(qty.out == "fluence", 1L, -1L),
  f.trigger.pulses = f.trigger.message,
  folder.name = paste("acq", qty.out, lubridate::today(tzone = "UTC"), sep = "-"),
  user.name = Sys.info()[["user"]],
  session.name = paste(user.name, strftime(lubridate::now(tzone = "UTC"),
    "%Y.%b.%d_%H.%M.%S"), sep = "_"),
  verbose = getOption("photobiology.verbose", default = FALSE),
  QC.enabled = TRUE
)
```

Arguments

`tot.time.range`	numeric vector Range of total times for a measurement in seconds.
`target.margin`	numeric (0..1) when tuning integration time, how big a head space to leave.
`HDR.mult`	numeric the integration time for each bracketed integration as a multiplier of the set or tuned integration time.
`protocols`	named list of character vectors, or a character vector with names of at least one member of the default list of protocols.
`correction.method`	list The method to use when applying the calibration
`descriptors`	list A list of instrument descriptors containing calibration data.
`entrance.optics`	character, name or geometry of diffuser, needed only if there is more than one for the same instrument.
`stray.light.method`	character Used only when the correction method is created on-the-fly.
`seq.settings`	named list with numeric members `start.boundary`, `initial.delay`, "`step.delay`" and "`num.steps`".
`area`	numeric Passed to `o_calib2irrad_mult()`.
`diff.type`	character Passed to `o_calib2irrad_mult()`.

`qty.out`	character One of "irrad" (spectral irradiance), "fluence" (spectral fluence), "cps" (counts per second), or "raw" (raw sensor counts).
`plot.lines.max`	integer Maximum number of spectra to plot as individual lines. Random sampling is used if number of spectra exceeds `plot.lines.max`.
`summary.type`	character One of "plant", "PAR" or "VIS".
`save.pdfs`, `save.summaries`, `save.collections`	logical Whether to save plots to PDFs files or not, and collection summaries to csv files or not, enable collections user interface or not.
`async.saves`	logical A flag enabling or disabling the use of concurrent processes to save data to files. Package 'mirai' must be installed before enabling this feature.
`show.figs`	logical Default for flag enabling display plots of acquired spectra.
`interface.mode`	character One of "auto", "simple", "manual", "full", "series", "auto-attr", "simple-attr", "manual-attr", "full-atr", and "series-attr".
`num.exposures`	integer Number or light pulses (flashes) per scan. Set to `-1L` to indicate that the light source is continuous.
`f.trigger.pulses`	function Function to be called to trigger light pulse(s). Should accept as its only argument the number of pulses, and return `TRUE` on success and `FALSE` on failure.
`folder.name`, `session.name`, `user.name`	character Default name of the folder used for output, and session and user names.
`verbose`	logical If TRUE additional messages are emitted, including report on memory usage.
`QC.enabled`	logical If FALSE return NA skipping QC.

Details

Function `acq_irrad_interactive()` supports measurement of spectral irradiance from continuous light sources and spectral fluence from discontinuous ones. For spectral irradiance it assumes that the duration of the measurement event is the relevant time base for expression of the flux of radiation. For spectral fluence the flux of radiation is expressed per pulse of illumination. The acquisition of both individual spectra and time series of spectra are supported.

A tutorial guiding on the use of this function, illustrated with diagrams, is available at `https://www.r4photobiology.info/pages/acq-irrad-tutorial.html`. A summary is provided below.

Different arguments passed to `interface.mode` modify which aspects of the user interface are available through menues, without altering the ability to control the behaviour through arguments passed to formal parameters when calling the function (see section Interface Modes for details).

This function can acquire spectra using different protocols for acquisition and stray light and dark corrections. The protocols are described in the vignettes and in the help for the low-level functions called by this function, also from this same package.

Opening the connection to the spectrometer and selection of the channel, when relevant, is done from within this function.

The irradiance calibration is retrieved from the spectrometer memory as a last resource if not supplied in any other way. Given that the factors are stored by Ocean Optics in a format that ignores the entrance optics, either the effective cosine diffuser area in xxx should be passed to parameter `area` or a character string with the type of the diffuser passed to `diff.type`. If no irradiance calibration is available, counts per second (cps) or raw counts are the only options available for the returned spectral data.

Three main protocols and two variations are available by default. They differ in the additional measurements done to correct for stray light and dark noise and in the sequence in which they are acquired. In most situations, at least one of the default protocols is suitable.

In the case of spectral irradiance, the default is to set the integration time automatically so that the number of counts at the highest peak is close to 1 - `target.margin` times the maximum raw-counts of the instrument detector (retrieved from the calibration or the instrument memory). The minimum `tot.time` is obtained by increasing the number of scans. The maximum integration time supported by the spectrometer cannot be exceeded but multiple scans can be averaged.

In the case of spectral fluence the default is for the integration time to be set manually and for a message to be displayed asking for the light pulse to be manually triggered. It is possible to override the default function by one that triggers the light source automatically when suitable hardware is available.

Repeated measurements are converted into physical units immediately after acquisition and saved to file on disk. Each repeated measurement can be either a single spectrum or a time series of spectra. To avoid long delays caused by saving large files, `async.saves` can be enabled.

Time series of spectra are acquired as fast as possible, converted into physical units after the acquisition of all individual raw-counts spectra and saved as a single `cps_spct` or `source_spct` in long form.

Time series of light measurements using single "dark" and "filter" measurements are scheduled by setting four members of the named list passed as argument to `seq.settings`, or interactively through the user interface.

The `initial.delay` is numeric and gives a minimum delay in seconds before the start of measurements with a default of 0s.

The `step.delay` is numeric and gives the length of time between successive "light" measurements. The value entered by the user is adjusted based on the estimated duration of individual spectrum acquisition. In most cases a vector of length one is used as time step lengths in seconds. Any vector shorter than the number of steps will be extended with `rep_len()`, and the values interpreted as the time increment in seconds between the start of successive measurements. If the length is the same as "num.steps", and the values are monotonically increasing, they are interpreted as time offsets from the start of the sequence. Member

The `start.boundary` must be set to a character string, wither "none", or a number of seconds, minutes or hours indicated by a number followed by S, M, or H (capital letters) the round value of the current time at which the measurement event will start. For example,`1H` indicates that measurements should be scheduled to start exactly at the hour, and `5M` at the next time the minutes in the current time are divisible by 5.

The `num.steps` must be an integer between 1 and an 100000 indicating the number of time points at which spectra should be acquired for the time series. The maximum value depends on the available memory and in many computers 5000 spectra is a more realistic limit than 100000. Applying corrections and applying the calibration are computation intensive, consequently for long series is wise to set ‘qty.out = "raw"‘ to speed up the measurement session.

Plots are produced with functions from package ’ggspectra’ and respect the defaults for plot annotations set in R options. The options can be easily set with functions `set_annotations_default()`, `set_w.band_default()`, `photon_as_default()`, and `energy_as_default()`. The ggplots are not saved as 'gg' objects as they contain redundant copies of the spectral data. They can be easily recreated using function `autoplot()` after attaching package 'ggspectra'.

The screen display of plots can be disabled, as in some cases the delay introduced by rendering can be a nuisance. Alternatively, the value of `plot.lines.max` can be changes from its default.

Value

This function returns the acquired spectra through "side effects" as each spectrum is saved, both as raw counts data and optionally as spectral irradiance, spectral fluence or counts-per-second spectral data in an `.rda` (R data) file as objects of the classes defined in package 'photobiology'. Optionally, the plot for each spectrum or a time series of spectra is saved as a `.pdf` file. At any time, the current group of spectra can be saved as a collection. When a collection is created, all recently measured spectra are saved together, decreasing the number of files and keeping related spectra in the same files. Summaries of the spectra in a collection are additionally saved to a CSV file and a plot of the collected spectra saved to a `.pdf` file.

The value returned by the function is that from closing the connection to the spectrometer.

Interface Modes

Mode **simple** displays a simplified user interface, supporting the acquisition of individual irradiance spectra. Integration time is adjusted automatically.

Mode **auto** displays a user interface supporting the acquisition of individual irradiance spectra and creation of collections of spectra. Integration time is adjusted automatically.

Mode **manual** displays a user interface supporting the acquisition of individual fluence or irradiance spectra and creation of collections of spectra. Integration time can adjusted automatically but also set manually.

Mode **series** displays a user interface supporting the acquisition of individual spectra and time series of irradiance spectra. Integration time can adjusted automatically but also set manually.

All these modes with **-attr** appended, enable a menu and dialogues that make it possible to set the values stored in attributes `comment` and `what.measure` interactively.

All modes support repeated measurements with unchanged acquisition settings reusing the reference spectra ('dark' and 'filter') from the most recent previous measurement.

Object names

Object names entered interactively are sanitized if necessary. Sequentially numbered object names are enabled by appending "#" to the desired base name. As long as no new name is entered, the sequence continues. If a new name is entered, numbering restarts at 001 or stops depending on whether the new name ends in "#" or not. In the case of repeated measurements, sequential numbering is enforced to ensure unique names.

Irradiance calibration

Calibration data needs in most cases to be imported into R and parameters entered for the special correction algorithms into a correction method descriptor. The corrections are skipped if the needed information is missing. If no spectral irradiance calibration is available and attempt is made to retrieve it from the spectrometer, but given the format used by Ocean Optics/Ocean Insight, in this case the effective `area` of the cosine diffuser used (or the model name if from Ocean Optics) should be supplied by the user.

Quality control of dark spectra

Disabling the quality control with `QC.enabled = FALSE` is necessary when the "dark" reference is a measurement of ambient light instead of true darkness; i.e., when the irradiance of one light source is measured as the difference between background illumination and background illumination plus the target light source.

Note

The function is composed in a modular way from functions defined in this some package, R or imported packages. The code of the function can be reshuffled combining the functions used here with other functions to create new variations, possibly better suited to users' needs and tastes.

A "light-weight" approach to tweaking the user interface is to implement new modes by simply changing which of the logical flags that control the display of menus are enabled or not. And even easier approach is to create a simple script that passes suitable arguments to the different formal parameters.

See Also

This function calls functions `tune_interactive`, `protocol_interactive`, `set_seq_interactive` and `set_attributes_interactive`. If irradiance calibration is retrieved from the instrument, functions `get_oo_descriptor` and `oo_calib2irrad_mult` are also called.

Other interactive acquisition functions: `acq_fraction_interactive()`

Examples

```
# please, see also the example scripts installed with the package

## Not run:
# requires an Ocean Optics spectrometer to be connected via USB

acq_irrad_interactive()
acq_irrad_interactive(qty.out = "cps")


## End(Not run)
```

`acq_raw_mspct` *Take one set of spectral readings*

Description

Take readings according to parameters from a list of settings and a protocol defined by a vector of names.

Usage

```
acq_raw_mspct(
  descriptor,
  acq.settings,
  f.trigger.pulses = f.trigger.message,
  seq.settings = list(initial.delay = 0, start.boundary = "none", step.delay = 0,
    num.steps = 1L),
  protocol = c("light", "filter", "dark"),
  user.label = "",
  where.measured = data.frame(lon = NA_real_, lat = NA_real_),
  pause.fun = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor()`.
`acq.settings`	list as returned by functions `tune_acq_settings()` or `retune_acq_settings()` or `acq_settings()`.
`f.trigger.pulses`	function Function to be called to trigger light pulse(s). Should accept as its only argument the number of pulses, and return `TRUE` on sucess and `FALSE` on failure.
`seq.settings`	list with members "initial.delay", "step,delay" numeric values in seconds, "num.steps" integer.
`protocol`	vector of character strings.
`user.label`	character string to set as label.
`where.measured`	data.frame with at least columns "lon" and "lat" compatible with value returned by `ggmap::geocode()`.
`pause.fun`	function used for handling protocol transitions.
`verbose`	ogical to enable or disable warnings.
`...`	passed to `pause.fun` (ignored by the default function).

Details

Function `acq_raw_mspct` acquires directly from a spectrometer a collection of spectra. The settings used for the acquisition of each member spectrum are the same, and are given by the argument passed to `acq.settings`. The number of numbers and their names are given by the argument passed to `protocol`.

Two types of light sources can be measured, for continuous-emission light sources, the integration time can at later steps used to compute irradiance. In the case of flashes, the duration of the exposure is unknown and irradiance cannot be computed, while spectral energy per flash can be computed if the number of flashes is known. The argument to `num.exposures` must be set to the number of flashes.

Two parameters accept functions as arguments, and default to functions that request the operator to trigger the flash or change the light conditions according to the names of the steps in the argument to `protocol`.

Sequences of light measurements using single "dark" and "filter" measurements are scheduled by setting the four members of the named list passed as argument to `seq.settings`. The member `initial.delay` is numeric and gives a minimum delay in seconds before the start of measurements with a default of 0s. Member `step.delay` is numeric and gives the delay in seconds between successive "light" measurements. In most cases a vector of length one is used as time delta in seconds. Any vector shorter than the number of steps will be extended with `rep_len()`, and the values interpreted as the time increment in seconds between the start of successive measurements. If the length is the same as "num.steps", and the values are monotonically increasing, they are interpreted as time offsets from the start of the sequence. Member `start.boundary` can take one of "none", "second", "minute" or "hour" indicating the unit to which the start of the series should be scheduled, e.g. the next minute and 0s, for "minute". Member `num.steps` must be an integer between 1 and small thousands indicating the number of time steps in the series.

Value

A raw_mspct object. The names and number of member spectra are determined by `protocol`, and the number of columns in each member spectrum is determined by `acq.settings`.

Note

Obviously the duration of the time steps must be longer than the time that a measurement takes. This time can be significantly more than the sum of integration times, as there is considerable overhead in both the OmniDriver Java code, in USB communication, in the spectrometer itself and in R. The overhead depends strongly on the model of spectrometer.

No multitasking is used or supported, so R waits for the spectrometer to answer. The operating system and other programs are not blocked, but the current R instance is.

See Also

`acq_raw_spct` which is used to acquire each member spectrum. Computations on date times are done with `lubridate`.

Other raw-counts-spectra acquisition functions: `acq_raw_spct()`, `hs_acq_raw_mspct()`

`acq_raw_spct` *Measure one raw spectrum*

Description

Take one spectral measurement which depending on the settings can consist in multiple raw spectra meant to represent a SINGLE observation after conversion into calibrated data, such as in the case of bracketing of integration time for HDR.

Usage

```
acq_raw_spct(
  descriptor,
  acq.settings,
  f.trigger.pulses = f.trigger.message,
  what.measured = NA,
  where.measured = data.frame(lon = NA_real_, lat = NA_real_),
  set.all = TRUE,
  verbose = TRUE
)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`.
`acq.settings`	list as returned by functions `tune_acq_settings`.
`f.trigger.pulses`	function Function to be called to trigger light pulse(s). Should accept as its only argument the number of pulses, and return `TRUE` on sucess and `FALSE` on failure.
`what.measured`	value used to set attribute.
`where.measured`	data.frame with at least columns "lon" and "lat" compatible with value returned by `ggmap::geocode()`.
`set.all`	logical resend or not all instrument settings.
`verbose`	logical to enable or disable warnings.

Value

A `raw_spct` object with one column `w.length` and one column `counts`, or two or more columns `counts1, counts2, ...` containing raw counts data. The number of columns with raw counts is determined by `acq.settings`, with multiple columns in the case of integration time bracketing or HDR.

See Also

`acq_raw_mspct` which can be used to acquire multiple spectra according to a user defined protocol.

Other raw-counts-spectra acquisition functions: `acq_raw_mspct()`, `hs_acq_raw_mspct()`

`acq_settings` *Settings for spectral measurement*

Description

Validate parameters for spectral measurements and return a list of values usable as input for functions `retune_acq_settings()`, `acq_sptc()`, and `acq_mspct()`.

Find optimal settings for spectral measurements under a given measurement protocol.

Usage

```
acq_settings(
  descriptor,
  integ.time = 0.01,
  num.scans = 10L,
  min.integ.time = -Inf,
  max.integ.time = Inf,
  tot.time.range = c(0, Inf),
  HDR.mult = ifelse(any(num.exposures != -1L), rep(1, length(num.exposures)), c(short =
    1, long = 10)),
  target.margin = 0.1,
  pix.selector = TRUE,
  corr.elect.dark = 0L,
  corr.sensor.nl = 0L,
  boxcar.width = 0L,
  force.valid = FALSE,
  num.exposures = -1L,
  verbose = TRUE
)

tune_acq_settings(descriptor, acq.settings, verbose = TRUE)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`
`integ.time`	numeric vaue in seconds
`num.scans`	integer
`min.integ.time`	numeric vaue in seconds
`max.integ.time`	numeric vaue in seconds
`tot.time.range`	numeric vector of length two with values in seconds
`HDR.mult`	a numeric vector with integ.time multipliers to be used for "bracketing".
`target.margin`	numeric (0..1)
`pix.selector`	a logical or numeric vector used as subscript to select pixels
`corr.elect.dark`, `corr.sensor.nl`	integer 0L (FALSE) or 1L (TRUE)
`boxcar.width`	integer Number of pixels to average
`force.valid`	logical Accept all spectra as valid, for example do not treat clipping as an error condition.
`num.exposures`	integer Number of flashes triggered per scan.
`verbose`	a logical to enable or disable warnings
`acq.settings`	list as returned by a previous call to `acq_settings()`, or `tune_acq_settings()`.

Details

`acq_settings()` is used to create a complete set of instrument settings in a way that they can be reused as needed for repated acquisition of spectra. `acq_settings()` is an object constructor and `tune_acq_settings()` takes as argument a stored object containing settings and tunes them to optimize them for the measurement of the current spectral irradiance.

This function searches for the optimal integration time for a given condition by trial and error helped by interpolation and extrapolation when readings are not saturated. In the case of clipping or sensor signal saturation the integration time is decreased until clipping is avoided and then it is increased until optimal.

Value

a list.

a list Containing the tuned settings.

Note

`pixel.selector` can be used for two different purposes: to ignore bad pixels and to restrict integration-time tuning to the response from a range of pixels. The interpretation of `tot.time.range` is as follows: first value is minimum time, second value is maximum time. If both values are the same, then an exact measurement time is computed.

Ocean Optics spectrometers can be queried for the maximum and minimum supported integration times. This function modifies the user supplied values if outside these bounds. The defaults of -Inf and Inf force the use of the whole valid range of integration time supported by the connected intrument. `pixel.selector` can be used for two different purposes: to ignore bad pixels and to restrict integration-time tuning to the response from a range of pixels.

The returned value can be used as argument to `acq.settings` in other functions like `acq_raw_spct` and `acq_raw_mspct`

See Also

Other acquisition-settings related functions: `set_integ_time()`, `set_linearized()`, `set_num_exposures()`

`bleed_nas` *Expand NA's to neighbouring pixels*

Description

Replace "good" data from pixels adjacent to NAs with NAs as data from pixels not saturated but located in the neighbourhood of saturated pixels can return unreliable data. This correction is needed by a phenomenon similar to "blooming" in camera sensors whereby when a sensor well gets saturated some of the charge migrates to adjacent wells in the detector increasing their readings.

Usage

```
bleed_nas(x, n = 10)
```

Arguments

`x`	raw_spct object
`n`	integer Number of pixels to set to NAs.

Value

a copy of x with values replaced by NAs as needed in all counts columns present.

Note

Avoid using very large n values as n pixels at each end of the array are assumed not to be ever saturated. The value of n needed for each detector type/instrument needs to be found through testing. As rule of thumb use 5 < n < 10 for Sony's ILxxx and 8 < n < 14 for Hamamatsu xxxx. At the moment we use a symmetric window although "blooming" could be asymmetric.

`blue_filter.raw_mspct` *Raw counts data for a filter measurement*

Description

A blue interference filter from UQG measured using a household incandescent lamp as light source. The spectrometer used was an Ocean Optics Maya2000 Pro.

Usage

```
blue_filter.raw_mspct
```

Format

A raw_mspct

See Also

Other objects containing example raw-counts data: `halogen.raw_mspct`, `red_filter.raw_mspct`, `sun001.raw_mspct`, `white_LED.raw_mspct`, `xenon_flash.raw_mspct`

check_sn_match *Check consistency of serial number*

Description

This method checks that the sn of the instrument used to acquire the raw counts matches that in the definition of a correction method.

Usage

```
check_sn_match(x, correction.method, missmatch.action = stop)
```

Arguments

`x`	a generic_mspct object, in normal use a raw_mspct object.
`correction.method`	a list describing the correction method. Must incluse a member named `"spectrometer.sn"`.
`missmatch.action`	a function, in practice one of `stop`, `warning`, `message` or `NULL`.

Value

A logical vector of length one, with `FALSE` indicating a missmatch.

See Also

Other spectral data-processing functions: `MAYP112785_tail_correction()`, `MAYP11278_tail_correction()`, `linearize_counts()`, `merge_raw_mspct()`, `new_correction_method()`, `ref_correction()`, `trim_counts()`, `uvb_corrections()`

choose_ch_interactive *Interactively select a channel*

Description

Choice of channel to be used if spectrometer has more than one channel.

Usage

```
choose_ch_interactive(
  instruments,
  sr.index = 0L,
  prompt.text = "Channels available: "
)
```

Arguments

`instruments`	the return value of `list_instruments(w)`
`sr.index`	integer The index to the spectrometer, starting from zero, following Omni Driver indexing conventions.
`prompt.text`	character string to use as prompt.

Value

A numeric index suitable for use in calls to functions defined in package 'rOmniDriver' based on which package 'ooacquire' is coded.

See Also

Other interactive acquisition utility functions: `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

`choose_sr_interactive` *Interactively select an instrument*

Description

Choice of spectrometer from a list of serial numbers, allowing the user to correct the selection if needed.

Usage

```
choose_sr_interactive(instruments)
```

Arguments

`instruments`	the return value of `list_instruments(w)`.

Value

A numeric index suitable for use in calls to functions defined in package 'rOmniDriver' based on which package 'ooacquire' is coded.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

collect_spct_files *Collect spectra into a collection*

Description

Read .Rda files as saved during data acquisition and build a collection with the spectra read, possibly first trimming the range of wavelengths and/or smoothing the spectral data.

Usage

```
collect_spct_files(
  path = ".",
  range = NULL,
  method = NULL,
  strength = 0,
  name.root = "",
  na.rm = FALSE
)
```

Arguments

`path`	a character string giving the name of the folder from wich to load .Rda files containing spectra.
`range`	a numeric vector of length two, or any other object for which function range() will return two.
`method`	a character vector of length 1 or 2, containing the smoothing method to use, one of "custom", "lowess", or "supsmu".
`strength`	numeric value to adjust the degree of smoothing.
`name.root`	character string to prepend to the name of the spectra in the collection.
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Note

If the argument passed to `method` is of length 2, the first member will apply to `source_spct` objects and the second one to `filter_spct` and `reflector_spct` objects. A numeric vector of length 2 passed to `strength` is treated in the same way. This is only relevant when we are collecting spectra belonging to different classes and need to treat them differently with respect to smoohting. It also allows different defaults, as transmittance spectra tend to lack the fine structure of some emission spectra.

See Also

Function `trim_wl` is used to trim the range of the data to plot, and function `smooth_spct` is used for smoothing, and prameters `range`, and `strength` and `method` are passed to them, respectively.

Other functions for importing spectral data from files: map_oofile_header_rows(), oofile_data_rows(), plot_spct_file(), read_files2mspct(), read_oo_data(), read_oo_ovdata(), read_oo_pidata(), read_oo_ssdata(), set_oo_ssdata_descriptor(), set_oo_ssdata_settings()

---

compute_irrad_calibration

Compute calibration multipliers.

Description

Function to calculate from a set of three (or optionally five or two) output files from the spectrometer, as produced by SpectraSuite, for each of two calibration lamps (D2 and FEL), and calibration data for the lamps, a set of calibration multipliers for the instrument. This function applies a nonlinearity correction, an slit-function correction, and stray-light correction, and optionally smoothing by means of process_maya_arrays(). Optionaly a HDR ("high dynamic range") merging of spectra (based on integration time bracketing) using two different integration times is done.

Usage

```
compute_irrad_calibration(
  FEL.raw.counts,
  D2.raw.counts,
  pix.wavelengths = NULL,
  wl.range = c(250, 900),
  FEL.k,
  D2.k,
  method,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

`FEL.raw.counts`	source_mscpt The raw counts and integration time data for the FEL light source.
`D2.raw.counts`	source_mscpt The raw counts and integration time data for the D2 light source.
`pix.wavelengths`	numeric vector The "true" wavelengths in nanometres at each pixel in the detector array.
`wl.range`	numeric vector of length two Range of wavelengths for which to compute the calibration.
`FEL.k`	numeric vector a numeric vector with n constants for the polynomial for the FEL lamp.
`D2.k`	numeric vector a numeric vector with n constants for the polynomial for the D2 lamp.
`method`	list Method variant to be used.
`verbose`	Logical indicating the level of warnings wanted. Defaults to `FALSE`.

Value

An instrument descriptor as a list containing the updated wavelengths and multipliers from the calibration.

Note

To calculate new multipliers we set the calibration multipliers equal to 1. Do the calculations as usual and calculate new multipliers based on the known spectral irradiance for the calibration lamps.

`end_session` *Disconnect from spectrometer*

Description

Close the connection to the spectrometer pointed at by the Java object referenced in `x`.

Usage

```
end_session(w)
```

Arguments

`w`	a java wrapper as returned by `start_session()`

See Also

Other spectrometer-connection functions: `list_instruments()`, `start_session()`

`f.trigger.message` *Manual trigger pulses request*

Description

This function is used by default. It prints a message asking the operator to manually trigger the flash. A more elaborate function, using specific hardware can be used to automatically trigger the light source, or to enable hard triggering of the light source by the spectrometer itself.

Usage

```
f.trigger.message(n = 1L)
```

Arguments

`n`	integer Number of pulses (flashes) to trigger per call.

Note

When using this function, set an integration time that gives enough time for the manual triggering of the flash to reliably fall within the integration.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

`filter_correction` *Correct for stray light*

Description

Correct cps readings for stray light, using either measured stray light, or using a non-excited region of the detector array.

Usage

```
filter_correction(
  x,
  flt,
  stray.light.method = "original",
  stray.light.wl = c(218.5, 228.5),
  flt.dark.wl = c(193, 209.5),
  flt.ref.wl = c(360, 379.5),
  flt.Tfr = 1,
  trim = 0.05,
  filter.nir.adjust = FALSE,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.1),
  verbose = getOption("photobiology.verbose", default = FALSE)
)

no_filter_correction(
  x,
  stray.light.wl = c(218.5, 228.5),
  flt.dark.wl = c(193, 209.5),
  flt.ref.wl = NULL,
  flt.Tfr = 1,
  trim = 0,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.1),
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

`x, flt`	cps_spct objects, containing spectral data from which to subtract stray light, and measured stray light, respectively.
`stray.light.method`	Method variant used, "original" (Ylianttila), "simple", "full", "sun", "raw", "none".
`stray.light.wl`	numeric vector of length 2 giving the range of wavelengths to use for the final stray light correction.
`flt.dark.wl, flt.ref.wl`	numeric vectors of length 2 giving the ranges of wavelengths to use for the "dark" and "illuminated" regions of the array in the filter correction.
`flt.Tfr`	numeric fractional transmittance of the filter to the source of stray light, used only for method "simple".
`trim`	a numeric value to be used as argument for mean
`filter.nir.adjust`	logical Flag indicating if the cps in the "filter" reference spectrum need to be adjust based on NIR region cps in the "light" spectrum. EXPERIMENTAL!!
`hdr.tolerance`	numeric Tolerance for mean deviation among cps columns as a fraction of one. Used in check of HDR consistency.
`verbose`	Logical indicating the level of warnings wanted.

`FLMS00416_descriptors` *Flame S spectrometer s/n FLMS00416*

Description

Calibration data, descriptors and methods for correction of stray light and high dynamic range based on multiple integrations times per spectrum.

Usage

```
FLMS00416_descriptors

FLMS00416_cal.spct

FLMS00416_calib_dates.df

FLMS00416_ylianttila.mthd

FLMS00416_sun.mthd

FLMS00416_simple.mthd

FLMS00416_none.mthd
```

Format

Lists and data frames.

An object of class `calibration_spct` (inherits from `generic_spct`, `tbl_df`, `tbl`, `data.frame`) with 2048 rows and 2 columns.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1 rows and 6 columns.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 10.

Note

Some of the method definitions are fill-ins for methods not implemented. In this case no characterizations of the slit function are available.

See Also

Other objects containing instrument-specific data: `FLMS00440_descriptors`, `FLMS00673_descriptors`, `FLMS04133_descriptors`, `JAZA3098_calib_dates.df`, `MAYP112785_descriptors`, `MAYP11278_descriptors`, `MAYP114590_descriptors`

`FLMS00440_descriptors` *Flame S spectrometer s/n FLMS00440*

Description

Calibration data, descriptors and methods for correction of stray light and high dynamic range based on multiple integrations times per spectrum.

Usage

```
FLMS00440_descriptors

FLMS00440_cal.spct

FLMS00440_calib_dates.df

FLMS00440_ylianttila.mthd

FLMS00440_sun.mthd

FLMS00440_simple.mthd

FLMS00440_none.mthd
```

Format

Lists and data frames.

An object of class `calibration_spct` (inherits from `generic_spct`, `tbl_df`, `tbl`, `data.frame`) with 2048 rows and 2 columns.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1 rows and 6 columns.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 10.

Note

Some of the method definitions are fill-ins for methods not implemented. In this case no characterizations of the slit function are available.

See Also

Other objects containing instrument-specific data: `FLMS00416_descriptors`, `FLMS00673_descriptors`, `FLMS04133_descriptors`, `JAZA3098_calib_dates.df`, `MAYP112785_descriptors`, `MAYP11278_descriptors`, `MAYP114590_descriptors`

`FLMS00673_descriptors` *Flame S spectrometer s/n FLMS00673*

Description

Calibration data, descriptors and methods for correction of stray light and high dynamic range based on multiple integrations times per spectrum.

Usage

```
FLMS00673_descriptors

FLMS00673_cal.spct

FLMS00673_calib_dates.df

FLMS00673_ylianttila.mthd

FLMS00673_sun.mthd

FLMS00673_simple.mthd

FLMS00673_none.mthd
```

Format

Lists and data frames.

An object of class `calibration_spct` (inherits from `generic_spct`, `tbl_df`, `tbl`, `data.frame`) with 2048 rows and 2 columns.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1 rows and 6 columns.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 10.

Note

Some of the method definitions are fill-ins for methods not implemented. In this case no characterizations of the slit function are available.

See Also

Other objects containing instrument-specific data: `FLMS00416_descriptors`, `FLMS00440_descriptors`, `FLMS04133_descriptors`, `JAZA3098_calib_dates.df`, `MAYP112785_descriptors`, `MAYP11278_descriptors`, `MAYP114590_descriptors`

`FLMS04133_descriptors` *Flame S spectrometer s/n FLMS04133*

Description

Calibration data, descriptors and methods for correction of stray light and high dynamic range based on multiple integrations times per spectrum.

Usage

```
FLMS04133_descriptors

FLMS04133_cal.spct

FLMS04133_calib_dates.df

FLMS04133_ylianttila.mthd

FLMS04133_sun.mthd

FLMS04133_simple.mthd

FLMS04133_none.mthd
```

Format

Lists and data frames.

An object of class `calibration_spct` (inherits from `generic_spct`, `spec_tbl_df`, `tbl_df`, `tbl`, `data.frame`) with 2048 rows and 2 columns.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1 rows and 6 columns.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 10.

An object of class `list` of length 10.

Note

Some of the method definitions are fill-ins for methods not implemented. In this case no characterizations of the slit function are available.

See Also

Other objects containing instrument-specific data: `FLMS00416_descriptors`, `FLMS00440_descriptors`, `FLMS00673_descriptors`, `JAZA3098_calib_dates.df`, `MAYP112785_descriptors`, `MAYP11278_descriptors`, `MAYP114590_descriptors`

`get_oo_descriptor` *Get the instrument description and EEPROM data*

Description

Model, configuration, serial number, and calibration data stored in the EEPROM of an Ocean Optics spectrometer are retrieved and returned in a list.

Usage

```
get_oo_descriptor(
  w,
  sr.index = 0L,
  ch.index = 0L,
  area = NULL,
  diff.type = NULL
)
```

Arguments

`w`	an open Wrapper object from OmniDriver.
`sr.index`	an index to address the spectrometer for the time being not exported.
`ch.index`	an index to address the channel in a spectrometer with more than one channel.
`area`	numeric Passed to `o_calib2irrad_mult()`.
`diff.type`	character Passed to `o_calib2irrad_mult()`.

Value

a list

Note

One and only one of `area` or `diff.type` is needed if an irradiance calibration stored in the EEPROM is to be retrieved. If both are null, as by default, the irradiance calibration factors will not be retireved even if present in the EEPROM.

`get_oo_settings` *Get the current values of instrument settings*

Description

Query the spectrometer for the settings currently in use for corrections, smotthing and acquisition parameters integration time and number of scans.

Usage

```
get_oo_settings(descriptor)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`

Value

a list

`halogen.raw_mspct` *Raw counts data for a lamp measurement.*

Description

A household Osram tungsten halogen 25W E27 lamp measured at short distance. The spectrometer used was an Ocean Optics Maya2000 Pro.

Usage

```
halogen.raw_mspct
```

Format

A raw_mspct

See Also

Other objects containing example raw-counts data: `blue_filter.raw_mspct`, `red_filter.raw_mspct`, `sun001.raw_mspct`, `white_LED.raw_mspct`, `xenon_flash.raw_mspct`

`hs_acq_raw_mspct` *Acquire spectra at high speed*

Description

Take one set of spectra at high speed using unchanged instrument settings special OmniDriver API functions for buffered acquisition. No HDR bracketing is possible, and synchronization with a pulsed light source is not supported.

Usage

```
hs_acq_raw_mspct(
  descriptor,
  acq.settings,
  num.spectra = 100L,
  base.name = NULL,
  f.trigger.pulses = f.trigger.message,
  what.measured = NA,
  where.measured = data.frame(lon = NA_real_, lat = NA_real_),
  set.all = TRUE,
  verbose = TRUE,
  return.list = FALSE
)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`.
`acq.settings`	list as returned by functions `tune_acq_settings`.
`num.spectra`	integer Number of individual spectra to acquire.
`base.name`	character The name given to individual spectra is formed by this string followed by a sequential numeric index.
`f.trigger.pulses`	function Function to be called to trigger an action. Should accept as its only argument the number of pulses, and return `TRUE` on success and `FALSE` on failure.
`what.measured`	value used to set attribute.
`where.measured`	data.frame with at least columns "lon" and "lat" compatible with value returned by `ggmap::geocode()`.
`set.all`	logical resend or not all instrument settings.
`verbose`	logical to enable or disable warnings.
`return.list`	logical Return a `list` instead of a `raw_mspct` object.

Value

A `raw_mspct` containing one `raw_spct` object with one column `w.length` and one column `counts` for each spectrum. The number of columns with raw counts is always one and integration time bracketing or HDR values in `acq.settings` are ignored except for the smallest value.

See Also

For normal speed acquisition of a single spectrum and multiple spectra according to a user defined protocol see `acq_raw_spct` and `acq_raw_mspct`.

Other raw-counts-spectra acquisition functions: `acq_raw_mspct()`, `acq_raw_spct()`

`irrad_summary_table` *Summarize spectral irradiance or fluence*

Description

Compute irradiance or fluence by waveband and energy or photon ratios between wavebands of interest to plants' and human visual responses to light.

Usage

```
irrad_summary_table(
  mspct,
  unit.out = getOption("photobiology.radiation.unit", default = "energy"),
  scale.factor = ifelse(unit.out == "photon", 1e+06, 1),
  attr2tb = "when.measured",
  summary.type = "plant",
  digits = 3L
)
```

Arguments

`mspct`	A source_mspct, or a source_spct object containing spectral irradiance for one or more sources.
`unit.out`	character One of "photon" or "energy".
`scale.factor`	numeric A multiplicative factor used to rescale data.
`attr2tb`	character Vector with one or more of "when.measured", "what.measured", "where.measured", "how.measured" and "comment".
`summary.type`	character One of "plant", "PAR" or "VIS".
`digits`	integer The number of significant digits in the output.

Details

This function packages different functions from pacakge 'photobiology' and returns a typical set of summaries for different purposes.

Value

A tibble with one row per spectrum and one column per summary quantity and attribute and a column with the names of the spectra.

See Also

See the documentation for functions `irrad`, `q_ratio`, `e_ratio`, `add_attr2tb`, `spct_CRI`, `spct_CCT` and `signif` which are called to build the summary table.

Examples

```
irrad_summary_table(sun.spct)
irrad_summary_table(sun.spct, attr2tb = c("what.measured", "where.measured"))
irrad_summary_table(sun.spct, summary.type = "plant", unit.out = "photon")
irrad_summary_table(sun.spct, summary.type = "PAR", unit.out = "photon")
# temporary kludge until fixed in photobiologyInOut
# irrad_summary_table(sun.spct, summary.type = "VIS", unit.out = "energy")
```

`JAZA3098_calib_dates.df`

Jaz spectrometer s/n JAZA3098

Description

Calibration data, descriptors and methods for correction of stray light and high dynamic range based on multiple integrations times per spectrum.

Usage

```
JAZA3098_calib_dates.df

JAZA3098_ch1_descriptors

JAZA3098_ch1_ylianttila.mthd

JAZA3098_ch1_sun.mthd

JAZA3098_ch1_simple.mthd

JAZA3098_ch1_none.mthd

JAZA3098_ch2_descriptors

JAZA3098_ch2_ylianttila.mthd

JAZA3098_ch2_sun.mthd

JAZA3098_ch2_simple.mthd

JAZA3098_ch2_none.mthd
```

Format

Lists and data frames.

An object of class `list` of length 1.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 9.

An object of class `list` of length 9.

An object of class `list` of length 1.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 9.

An object of class `list` of length 9.

Note

Some of the method definitions are fill-ins for methods not implemented. In this case no calibration for spectral irradiance or characterizations of the slit function are available.

See Also

Other objects containing instrument-specific data: FLMS00416_descriptors, FLMS00440_descriptors, FLMS00673_descriptors, FLMS04133_descriptors, MAYP112785_descriptors, MAYP11278_descriptors, MAYP114590_descriptors

---

linearize_counts   *Function to apply linearization correction to raw counts data.*

Description

Uses a function stored as an attribute of x, by default retrieved from the instrument's firmware at the time of data acquisition or possibly a replacement set by the user.

Usage

```
linearize_counts(
  x,
  force.zero = TRUE,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

x	raw_spct object.
force.zero	A logical indicating whether to change negative count values to zero.
verbose	Logical Currently ignored.

Value

A raw_spct object containing the adjusted values, still as uncalibrated counts. The object is tagged with the with attribute "linearized" set to the function used for linearization.

Note

In contrast to other classes defined in package 'photobiology', class "raw_spct" can have more than one column of raw counts in cases where the intention is to merge these values as part of the processing at the time the calibration is applied. In other words these columns are not separate observations but instrumental replicates or bracketing readings part of the same "logical" or "effective" observation. The contents of any column whose name starts with "counts" will have the intrument response linearization function applied.

See Also

Other spectral data-processing functions: MAYP112785_tail_correction(), MAYP11278_tail_correction(), check_sn_match(), merge_raw_mspct(), new_correction_method(), ref_correction(), trim_counts(), uvb_corrections()

`list_instruments` *List connected spectrometers*

Description

List all connected spectrometers by model name and serial number, plus the number of channels in each of them.

Usage

```
list_instruments(w)
```

Arguments

`w`	an open Wrapper object from Omnidriver

Value

a list

See Also

Other spectrometer-connection functions: `end_session()`, `start_session()`

`list_srs_interactive` *Get list of connected instruments*

Description

Get list of spectrometers requesting user to connect one or abort if none found.

Usage

```
list_srs_interactive(w)
```

Arguments

`w`	handle to Omni Driver, used to test if an spectrometer is still connected.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

`map_oofile_header_rows`

Parse a file header to locate metadata items

Description

Locate in which lines of the file header different metadata features are located. Parsing is needed because the format used varies with among Ocean Optics programs and instruments. Even the number of lines in the header, and the position of the first line with spectral data vary.

Usage

```
map_oofile_header_rows(lines, header.end = NULL, grammar = oo.minimum.gr)
```

Arguments

`lines`	character The data file read line by line.
`header.end`	integer The index to the last line of the longest header expected.
`grammar`	data.frame With character variables "feature" and "pattern" that will be used to locate the lines containing each type of metadata.

Value

A data frame with two variables, "feature" of class character, and "line.idx" of class integer.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `oofile_data_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

`MAYP112785_descriptors`

Maya2000 Pro spectrometer s/n MAYP112785

Description

List of instrument descriptors, each one containing the calibration data needed to convert raw data acquired on different dates. Calibration data from STUK's non-ionising radiation laboratory. Correction method algorithms for stray light and slit function by Lasse Ylianttila, and variations by Pedro J. Aphalo..

Usage

```
MAYP112785_descriptors

MAYP112785_calib_dates.df

MAYP112785_ylianttila.mthd

MAYP112785_sun.mthd

MAYP112785_simple.mthd
```

Format

Lists and data frames.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 4 rows and 8 columns.

An object of class `list` of length 10.

An object of class `list` of length 10.

An object of class `list` of length 10.

See Also

Other objects containing instrument-specific data: `FLMS00416_descriptors`, `FLMS00440_descriptors`, `FLMS00673_descriptors`, `FLMS04133_descriptors`, `JAZA3098_calib_dates.df`, `MAYP11278_descriptors`, `MAYP114590_descriptors`

`MAYP112785_tail_correction` *Function to compute the tail correction*

Description

Function to compute the tail correction

Usage

```
MAYP112785_tail_correction(w_length, cts_second)
```

Arguments

`w_length`	numeric vector of wavelengths in nm.
`cts_second`	numeric vector of counts per second (stray-light corrected).

Value

tail returned as numeric vector within a list.

Tail correction is a reimplementation of the calculations developed by Lasse Ylianttila (STUK, Finland), originally in Excel.

See Also

Other spectral data-processing functions: `MAYP11278_tail_correction()`, `check_sn_match()`, `linearize_counts()`, `merge_raw_mspct()`, `new_correction_method()`, `ref_correction()`, `trim_counts()`, `uvb_corrections()`

`MAYP11278_descriptors` *Maya2000 Pro spectrometer s/n MAYP11278*

Description

List of instrument descriptors, each one containing the calibration data needed to convert raw data acquired on different dates. Calibration data from STUK's non-ionising radiation laboratory. Correction method algorithms for stray light and slit function by Lasse Ylianttila, and variations by Pedro J. Aphalo..

Usage

```
MAYP11278_descriptors

MAYP11278_calib_dates.df

MAYP11278_ylianttila.mthd

MAYP11278_short_flt_ref.mthd

MAYP11278_sun.mthd

MAYP11278_simple.mthd
```

Format

Lists and data frames.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 9 rows and 8 columns.

An object of class `list` of length 10.

An object of class `list` of length 10.

An object of class `list` of length 10.

An object of class `list` of length 10.

See Also

Other objects containing instrument-specific data: `FLMS00416_descriptors`, `FLMS00440_descriptors`, `FLMS00673_descriptors`, `FLMS04133_descriptors`, `JAZA3098_calib_dates.df`, `MAYP112785_descriptors`, `MAYP114590_descriptors`

`MAYP11278_tail_correction` *Function to compute the tail correction*

Description

Function to compute the tail correction

Usage

```
MAYP11278_tail_correction(w_length, cts_second)
```

Arguments

`w_length`	numeric vector of wavelengths in nanometres (nm).
`cts_second`	numeric vector of counts per second (stray-light corrected).

Value

tail returned as numeric vector within a list.

Tail correction is a reimplementation of the calculations developed by Lasse Ylianttila (STUK, Finland), originally in Excel.

See Also

Other spectral data-processing functions: `MAYP112785_tail_correction()`, `check_sn_match()`, `linearize_counts()`, `merge_raw_mspct()`, `new_correction_method()`, `ref_correction()`, `trim_counts()`, `uvb_corrections()`

`MAYP114590_descriptors`

Maya2000 Pro spectrometer s/n MAYP114590

Description

List of instrument descriptors, each one containing the calibration data needed to convert raw data acquired on different dates. Calibration data from STUK's non-ionising radiation laboratory. Correction method algorithms for stray light and slit function by Lasse Ylianttila, and variations by Pedro J. Aphalo..

Usage

```
MAYP114590_descriptors

MAYP114590_cal.spct

MAYP114590_calib_dates.df

MAYP114590_ylianttila.mthd

MAYP114590_sun.mthd

MAYP114590_simple.mthd

MAYP114590_none.mthd
```

Format

Lists and data frames.

An object of class `calibration_spct` (inherits from `generic_spct`, `spec_tbl_df`, `tbl_df`, `tbl`, `data.frame`) with 2068 rows and 2 columns.

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1 rows and 6 columns.

An object of class `list` of length 0.

An object of class `list` of length 0.

An object of class `list` of length 10.

An object of class `list` of length 10.

Note

Some of the method definitions are fill-ins for methods not implemented. In this case no characterization of the slit function is available.

See Also

Other objects containing instrument-specific data: FLMS00416_descriptors, FLMS00440_descriptors, FLMS00673_descriptors, FLMS04133_descriptors, JAZA3098_calib_dates.df, MAYP112785_descriptors, MAYP11278_descriptors

---

merge_cps *Merge counts per second data*

Description

In a cps_spct object with multiple columns of CPS data, each acquired using a different integration time, merge columns into a single column.

Usage

```
merge_cps(
  x,
  tolerance = 0.05,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

x	cps_spct object
tolerance	numeric Tolerance for mean deviation among cps columns as a fraction of one.
verbose	Logical indicating the level of warnings and messages wanted.

Details

Pixels affected directly, or by neighbourhood, by clipping, should be set to NA in each variable of CPS values, before passing the spectrum as argument to parameter x of this function.

The merging of the CPS variables starts from the one corresponding to the longest integration time, expected to contain the largest number of NA values, by replacing NA values by cps values from the variable corresponding to the next shorter integration. The procedure is repeated until no NA remains or until no shorter integration time data are available. *The process stops when all NAs have been replaced, and, even when available, CPS values for unnecesarilly short integration times discarded.*

When measuring daylight different exposures for HDR are taken sequentially, and if light conditions change rapidly the cps values may be inconsistent. If the mean ratio of cps values is outside plus/minus the tolerance, instead of merging, the data for the longer of the two exposures is discarded instead of merged (or spliced) with the longer exposure, in which case a message is emitted. Ratio is computed after discarding low signal pixels as these readings are affected by noise, distorting the ratio when the light source-spectrum has only narrow peaks of emission.

Value

a copy of x with values replaced as needed in all counts columns present.

merge_raw_mspct *Merge raw spectra into a single multicolumn spectrum*

Description

Member spectra are sorted according to integration time stored in the inst.settings attribute and merged into a single raw_spct object with raw counts variables named counts_1, counts_2, etc.

Usage

```
merge_raw_mspct(x)
```

Arguments

x	raw_mspct

Note

The individual raw_spct objects contained in x must have identical values in w.length.

See Also

Other spectral data-processing functions: MAYP112785_tail_correction(), MAYP11278_tail_correction(), check_sn_match(), linearize_counts(), new_correction_method(), ref_correction(), trim_counts(), uvb_corrections()

new_correction_method *Make default method from descriptor*

Description

A function that builds a default method from an instrument descriptor. Useful when the spectrometers has not been characterized as needed for the more sofisticated methods. Can use stray light correction but not slit function correction. Stray light correction is valid only if it was also used during irradiance callibration. Suitablity of wavelengths and method depends on the instrument configuration so they are set to NA as default..

Usage

```
new_correction_method(
  descriptor,
  stray.light.method = "none",
  stray.light.wl = c(NA_real_, NA_real_),
  flt.dark.wl = c(NA_real_, NA_real_),
  flt.ref.wl = c(NA_real_, NA_real_),
  flt.Tfr = NA_real_
)
```

Arguments

`descriptor`	list, as returned by `get_oo_descriptr()`
`stray.light.method`	character Name of method.
`stray.light.wl`, `flt.dark.wl`, `flt.ref.wl`	numeric vector with wavelengths (nm).
`flt.Tfr`	numeric Transmittance of filter as a fraction (0...1).

Details

The currently recognized methods for stray-light correction are `"none"`, `"original"`, `"sun"`, and `"simple"`. With the default method `"none"`, the values of the remaining parameters are ignored.

Value

a list

Note

Defaults for indexes are for the first channel of the first spectrometer currently connected.

See Also

Other spectral data-processing functions: `MAYP112785_tail_correction()`, `MAYP11278_tail_correction()`, `check_sn_match()`, `linearize_counts()`, `merge_raw_mspct()`, `ref_correction()`, `trim_counts()`, `uvb_corrections()`

`oofile_data_rows` *Find range of lines in file containing spectral data.*

Description

Ocean Optics files can have varying numbers of rows containing spectral data, but these rows are fenced between two lines containing recognizable character strings. This function, detects these strings.

Usage

```
oofile_data_rows(lines)
```

Arguments

`lines`	character The data file read line by line.

Value

integer vector of length two, containing the indexes to the first last rows containing spectral data.

Note

This is a rather slow operation, so if all files to be read have the same format, it is inefficient to call this function for each file.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

`oo_calib2irrad_mult` *Convert an OO calibration*

Description

Convert irradiance calibration values as supplied by Ocean Optics into multipliers expressed in the units and format expected by the functions in this package.

Usage

```
oo_calib2irrad_mult(
  x,
  area = NULL,
  diff.type = NULL,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

`x`	generic_spct object with variables `w.length` and `oo.cal`.
`area`	numeric Area of the cosine diffuser (mm2).
`diff.type`	character Value giving type of diffuser as in OO's documents, case insensitive. Ignored unless `area = NULL`.
`verbose`	Logical indicating the level of warnings wanted.

Value

a `calibration_spct` object of the same number of rows as `x` containing wavelengths in varable `w.length` and the re-scaled calibration factors in variable `irrad.mult`.

See Also

`read_oo_caldata`

`plot_spct_file`	*Read a file and plot spectrum*

Description

Read an .Rda file as saved during data acquisition and plot it, possibly first trimming the range of wavelengths and/or smoothing the spectral data.

Usage

```
plot_spct_file(file, range = NULL, method = NULL, strength = 0, na.rm = FALSE)
```

Arguments

`file`	a (readable binary-mode) connection or a character string giving the name of the file to load (when tilde expansion is done).
`range`	a numeric vector of length two, or any other object for which function range() will return two.
`method`	a character string "custom", "lowess", "supsmu".
`strength`	numeric value to adjust the degree of smoothing.
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Note

If the argument passed to `method` is of length 2, the first member will apply to `source_spct` objects and the second one to `filter_spct` and `reflector_spct` objects. A numeric vector of length 2 passed to `strength` is treated in the same way. This is only relevant when we are collecting spectra belonging to different classes and need to treat them differently with respect to smoohting. It also allows different defaults, as transmittance spectra tend to lack the fine structure of some emission spectra.

See Also

Function `trim_wl` is used to trim the range of the data to plot, and function `smooth_spct` is used for smoothing, and prameters `range`, and `strength` and `method` are passed to them, respectively.

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

`protocol_interactive` *Interactively select a measurement protocol*

Description

Choose a protocol by name from a list of protocols, allowing the user to correct the selection if needed. Protocols are not hard-wired, but instead defined by the list passed as argument to `protocols`.

Usage

```
protocol_interactive(protocols, default = names(protocols)[[1]])
```

Arguments

`protocols`	named list Measuring protocol defifinitions and names.
`default`	character Name of the default protocol.

Details

A protocol is defined as a named list of character strings, and consist in multiple acquisition of spectra contributing to the same logical measurement. The name of the list member is the name of the protocol, while the members of each character vector correspond to spectra to be acquired, after some change in the measurement conditions. For example the list `list(rsd = c("reference", "sample", "dark"), rs = c("reference", "sample"))` defines two protocols.

Value

The member vector corresponding to the protocol selected by the user.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

`QC_dark` *Quality control of dark spectra*

Description

Function used to check the number of pixels that deviate from the median.

Usage

```
QC_dark(
  x,
  range = NULL,
  tol.margin = 1,
  max.hot = 60,
  max.cold = 20,
  spct.label = "Spectrum",
  verbose = getOption("photobiology.verbose", default = TRUE),
  QC.enabled = getOption("ooacquire.qc.enabled", default = TRUE)
)
```

Arguments

`x`	cps_spct or raw_spct A spectrum measured in darkness.
`range`	numeric A wavelength range [nm].
`tol.margin`	numeric A multiplier applied to MAD, the default 1 corresponds to 1 pixel per 1000 pixels as the expected random Normal noise.
`max.hot, max.cold`	integer Maximum number of hot and cold pixels exceeding the tolerance per 1000 array pixels.
`spct.label`	character A character string to use in message.
`verbose`	logical If true a message is emitted in addition to returning the outcome.
`QC.enabled`	logical If FALSE return NA skipping QC.

Details

The expectation is that in a spectrum measured in the dark, to be used as a reference, the variation among pixel counts is small. This also applies in some cases to ranges of pixels protected from radiation by a long pass or short pass filter. By default the whole spectrum is included in the QC check, but if needed an argument can be passed to `range` to select a smaller region.

Value

A logical value.

Note

Disabling the quality control is necessary when the "dark" reference is a measurement of ambient light instead of true darkness; i.e., when the irradiance of one light source is measured as the difference between background illumination and background illuminations plus the target light source.

`raw2corr_cps` *Convert raw detector counts into counts-per-second*

Description

Replace data from bad pixels with interpolated values, replace data from saturated and nearby pixels withs NAs, apply linearization function if data is not already linearized, optionally use a range of pixels as dark reference, convert the raw counts for each integration time used into counts-per-second, if data from bracketed intergartion times is available, splice the different spectra.

Usage

```
raw2corr_cps(x, ref.pixs.range, ...)

## Default S3 method:
raw2corr_cps(x, ref.pixs.range = NULL, ...)

## S3 method for class 'raw_spct'
raw2corr_cps(
  x,
  ref.pixs.range = c(1, 100),
  despike = FALSE,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.05),
  ...
)

## S3 method for class 'raw_mspct'
raw2corr_cps(x, ref.pixs.range = c(1, 100), despike = FALSE, ...)
```

Arguments

`x`	raw_spct object.
`ref.pixs.range`	integer vector of length 2.
`...`	passed to `photobiology::despike`.
`despike`	logical flag, if TRUE despiking will be attempted.
`hdr.tolerance`	numeric Passed as tolerance argument to merge_cps().

Value

a cps_spct object with one spectrum preserving the metadata present in `x`.

Methods (by class)

- `raw2corr_cps(default)`: Default method
- `raw2corr_cps(raw_spct)`: raw_spct method
- `raw2corr_cps(raw_mspct)`: raw_spct method

See Also

Other functions for conversion of raw-counts data: `raw2cps()`, `s_fraction_corrected()`, `s_irrad_corrected()`

`raw2cps` *Convert raw detector counts into counts per second (cps).*

Description

These functions simply divide "raw counts" by the integration time used for acquisition.

Usage

```
raw2cps(x, ...)

## Default S3 method:
raw2cps(x, ...)

## S3 method for class 'raw_spct'
raw2cps(x, ...)

## S3 method for class 'raw_mspct'
raw2cps(x, ...)
```

Arguments

`x`	an R object
`...`	not used in current version

Value

an object of class "cps_spct"

an object of class "cps_mspct"

Methods (by class)

- `raw2cps(default)`: Default method
- `raw2cps(raw_spct)`: Method for spectral data expressed as raw instrument counts.
- `raw2cps(raw_mspct)`: Method for collections of raw-counts spectra

Note

In the case of objects of class "raw_spct" the columns with names starting with "counts" are processed and renamed to columns with names starting with "cps". All other columns are left unchanded.

See Also

Other functions for conversion of raw-counts data: `raw2corr_cps()`, `s_fraction_corrected()`, `s_irrad_corrected()`

`read_files2mspct` *Read multiple files into raw_mspct object*

Description

Read multiple files and return a collection of raw spectra as a raw_spct object.

Usage

```
read_files2mspct(files, read.f = ooacquire::read_oo_data, ...)
```

Arguments

`files`	a named list of character strings of file names
`read.f`	a function which expects in its first parameter a file name or file path
`...`	additional arguments passed by name to the function passed as argument to `read.f`

Note

Depending of the function passed to `read.f` more or less complete metadata information will be stored as attributes in the raw_spct objects.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `plot_spct_file()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

`read_oo_caldata` *Read OO irradiance calibration.*

Description

Reads a calibration data file as supplied by Ocean Optics. Wavelength and calibration values are stored as data and the metadata parsed from the header or supplied as arguments as attributes of the same object, including the time and date of the calibration. The whole file header is in addition stored as a comment.

Usage

```
read_oo_caldata(
  file,
  time = NULL,
  geocode = NULL,
  label = NULL,
  descriptor = NULL,
  tz = NULL,
  locale = NULL,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

`file`	character string Path or file to read.
`time`	a `POSIXct` object, but if `NULL` the date stored in file is used, and if `NA` the `when.measured` attribute is set to `NA`.
`geocode`	A one row data frame with numeric columns `lon` and `lat`, and optionally a character column `address`.
`label`	character string, but if `NULL` the value of `file` is used, and if `NA` the "what.measured" attribute is set to `NA`.
`descriptor`	list A list describing the instrument, used to set attribute `instr.descriptor`.
`tz`	character Time zone is by default read from the file.
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`verbose`	Logical indicating the level of warnings wanted.

Value

A generic_spct object, with columns `w.length` and `oo.cal`.

See Also

`oo_calib2irrad_mult`

read_oo_data *Read Files Saved by Ocean Optics' instruments or software.*

Description

Reads and parses the header of a processed data file as output by SpectraSuite to extract the whole header remark field. The time field is retrieved and decoded.

Usage

```
read_oo_data(
  file,
  time = NULL,
  geocode = NULL,
  label = NULL,
  descriptor = NULL,
  tz = NULL,
  locale = NULL,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

`file`	character string
`time`	a `POSIXct` object, but if `NULL` the date stored in file is used, and if `NA` no date variable is added
`geocode`	A data frame with columns `lon` and `lat`.
`label`	character string, but if `NULL` the value of `file` is used, and if `NA` the "what.measured" attribute is not set.
`descriptor`	list A list describing the instrument used.
`tz`	character Time zone is by default read from the file.
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`verbose`	Logical indicating the level of warnings wanted.

Value

A raw_spct object.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_ovdata()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

`read_oo_ovdata` *Read File Saved by Ocean Optics' OceanView.*

Description

Reads and parses the header of a processed data file as output by OceanView to extract the whole header remark field. The time field is retrieved and decoded.

Usage

```
read_oo_ovdata(
  file,
  time = NULL,
  geocode = NULL,
  label = NULL,
  descriptor = NULL,
  tz = NULL,
  locale = NULL,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

`file`	character string
`time`	a `POSIXct` object, but if `NULL` the date stored in file is used, and if `NA` no date variable is added
`geocode`	A data frame with columns `lon` and `lat`.
`label`	character string, but if `NULL` the value of `file` is used, and if `NA` the "what.measured" attribute is not set.
`descriptor`	list A list describing the instrument used.
`tz`	character Time zone is by default read from the file.
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`verbose`	Logical indicating the level of warnings wanted.

Value

A raw_spct object.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

read_oo_pidata — *Read File Saved by Ocean Optics' Raspberry Pi software.*

Description

Reads and parses the header of a raw data file as output by the server running on a Raspberry Pi board to extract the whole header remark field. The time field is retrieved and decoded.

Usage

```
read_oo_pidata(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale(),
  descriptor = NULL,
  corr.sensor.nl = FALSE,
  spectrometer.name = "Unknown via Raspberry Pi",
  spectrometer.sn = "Unknown via Raspberry Pi",
  npixels = 2048
)
```

Arguments

`file`	character string
`date`	a `POSIXct` object, but if `NULL` file modification date is used with a warning and if `NA` date is set to NA.
`geocode`	A data frame with columns `lon` and `lat`.
`label`	character string, but if `NULL` the value of `file` is used, and if `NA` the "what.measured" attribute is not set.
`tz`	character Time zone is not saved to the file.
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`descriptor`	list as returned by function `get_oo_descriptor`.
`corr.sensor.nl`	logical, indicating if spectral data is already linearized. If TRUE the spectrum is marked as linearized, and linearization skipped during processing.
`spectrometer.name`, `spectrometer.sn`	character.
`npixels`	integer Number of pixels in spectral data.

Value

A raw_spct object.

Note

The header in these files has very little information, so the user needs to supply the number of pixels in the array as well as the date-time. The file contains a date in milliseconds but as the Raspberry Pi board contains no real-time clock, it seems to default to number of milliseconds since the Pi was switched on.

References

`http://www.r4photobiology.info`

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

`read_oo_ssdata` *Read File Saved by Ocean Optics' SpectraSuite.*

Description

Reads and parses the header of a processed data file as output by SpectraSuite to extract the whole header remark field. The time field is retrieved and decoded.

Usage

```
read_oo_ssdata(
  file,
  time = NULL,
  geocode = NULL,
  label = NULL,
  descriptor = NULL,
  tz = NULL,
  locale = NULL,
  verbose = getOption("photobiology.verbose", default = FALSE)
)
```

Arguments

`file`	character string
`time`	a `POSIXct` object, but if `NULL` the date stored in file is used, and if `NA` no date variable is added
`geocode`	A data frame with columns `lon` and `lat`.

`label`	character string, but if `NULL` the value of `file` is used, and if `NA` the "what.measured" attribute is not set.
`descriptor`	list A list describing the instrument used.
`tz`	character Time zone is by default read from the file.
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`verbose`	Logical indicating the level of warnings wanted.

Value

A raw_spct object.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_pidata()`, `set_oo_ssdata_descriptor()`, `set_oo_ssdata_settings()`

`red_filter.raw_mspct` *Raw counts data for a filter measurement*

Description

A red long-pass filter from Heliopan (695 nm) measured using a household incandescent lamp as light source. The spectrometer used was an Ocean Optics Maya2000 Pro.

Usage

```
red_filter.raw_mspct
```

Format

A raw_mspct

See Also

Other objects containing example raw-counts data: `blue_filter.raw_mspct`, `halogen.raw_mspct`, `sun001.raw_mspct`, `white_LED.raw_mspct`, `xenon_flash.raw_mspct`

ref_correction — *Apply a correction to spectral data.*

Description

These functions simply subtract (by default) detector counts data (raw or cps) of one spectrum by the corresponding columns in another spectrum, or apply a user supplied operator or function to them.

Usage

```
ref_correction(x, y, .oper, ...)

## Default S3 method:
ref_correction(x, y, .oper, ...)

## S3 method for class 'numeric'
ref_correction(x, y, .oper = `-`, ...)

## S3 method for class 'raw_spct'
ref_correction(x, y, .oper = `-`, ...)

## S3 method for class 'cps_spct'
ref_correction(x, y, .oper = `-`, ...)

## S3 method for class 'cps_mspct'
ref_correction(x, y = x, .oper = `-`, ref_name = "dark", ...)
```

Arguments

`x, y`	R objects
`.oper`	a function with its first two formal parameters accepting numerical vectors of equal length (e.g. a binary numerical operator).
`...`	not used in current version
`ref_name`	character Name of variable to substract from all other columns.

Value

a numeric vector of the same length as x

an object of class "cps_spct"

an object of class "cps_spct"

an object of class "cps_mspct"

Methods (by class)

- `ref_correction(default)`: Default method
- `ref_correction(numeric)`: Numeric method
- `ref_correction(raw_spct)`: Method for spectral data expressed as raw instrument counts.
- `ref_correction(cps_spct)`: Method for spectral data expressed as counts per second.
- `ref_correction(cps_mspct)`: Method for collections of spectral data objects containing data expressed as counts per second.

Note

In the case of objects of class "raw_spct" the columns with names starting with "counts" are processed. All other columns are left unchanded.

If `x` and `y` are both cps_mspct objects, `y[[ref_name]]` will be used as reference, otherwise `y` itself should be a cps_spct and will be used as is. In all cases variables in ref.name will be skipped in `x`.

See Also

Other spectral data-processing functions: `MAYP112785_tail_correction()`, `MAYP11278_tail_correction()`, `check_sn_match()`, `linearize_counts()`, `merge_raw_mspct()`, `new_correction_method()`, `trim_counts()`, `uvb_corrections()`

`rm_jwrapper` *Remove java wrapper from descriptor*

Description

Clean up left-behind wrappers used to communicate with Java code using 'rJava'.

Usage

```
rm_jwrapper(x)
```

Arguments

`x`	raw_spct or raw_mspct object with attribute `instr.desc` set.

Details

Field `w` contains a wrapper on a Java object used to communicate with the OmniDriver driver during data acquisition. Once the current connection between R and a spectrometer ends, these wrappers are invalidated, becoming useless. However, if present they create a dependency on 'rJava', possibly triggering errors. In recent versions of 'ooacquire' this wrapper is removed immediately after acquisition. However, the instrument descriptor of spectral objects created with versions of 'ooacquire' for some years ago can contain a member storing a useless Java wrapper. This function removes this field if present.

Value

a copy of `x` possibly with an updated `instr.desc` attribute embedded.

Note

Method `getInstrDesc()` removes member field `w` from the returned value but does not modify its argument.

`set_attributes_interactive`
Interactively set user attributes

Description

Allow user to provide values for "user supplied" attribute values `"comment"` and `"what.measured"`.

Usage

```
set_attributes_interactive(
  user.attrs = list(what.measured = "", comment.text = "")
)
```

Arguments

`user.attrs`	character Default values for the attributes.

Value

a named list of character vectors.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

set_descriptor_bad_pixs

Add bad pixel information to an instrument description

Description

A integer vector of indexes to bad pixels in the instrument array. Data from these array pixels will be discarded.

Usage

```
set_descriptor_bad_pixs(descriptor, bad.pixs)

update_spct_bad_pixs(spct, bad.pixs)

update_mspct_bad_pixs(mspct, bad.pixs)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`
`bad.pixs`	numeric vector of positional indexes corresponding to individual pixels in the instrument array, or an object of class `"instr_desc"` from where to copy the `bad.pixs` member.
`spct`	an object of class `generic_spct` or derived.
`mspct`	an object of class `generic_mspct` or derived.

Value

A copy of the argument passed to `descriptor`, `spct` or `mspct` with the descriptor with indexes to bad pixels set to the new values.

Note

Following R's syntax the first pixel in the detector array has index 1.

set_descriptor_entrance_optics

Add or replace entrance optics data to descriptor

Description

Replace or add a description of the fibre and diffuser or other entrance optics to the instrument descriptor.

Usage

```
set_descriptor_entrance_optics(
  descriptor,
  make = NA_character_,
  model = NA_character_,
  geometry = NA_character_,
  serial.number = NA_character_,
  area = NA_real_
)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`
`make`, `model`, `geometry`, `serial.number`	character.
`area`	numeric (m^2).

Value

a copy of the argument passed for `oo_descriptor` with the `entrance.optics` field of the calibration data adde or replace by the new list.

`set_descriptor_integ_time` *Replace integration time limits in instrument descriptor*

Description

This function can be needed in exceptional cases such as when the limits stored in the intrument's persistent memory are wrong. In other cases in can be used to restrict the range of values allowed to be set to a smaller range than natively supported by the spectrometer.

Usage

```
set_descriptor_integ_time(
  descriptor,
  min.integ.time = NA_integer_,
  max.integ.time = NA_integer_,
  force.change = FALSE
)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`
`min.integ.time`, `max.integ.time`	numeric values in seconds in seconds.
`force.change`	If FALSE values outside the range returned by a query to the instrument trigger an error. If TRUE this test is overriden.

Value

a copy of the argument passed for `oo_descriptor` with the integration time fields of the descriptor modified.

Note

This function should not be needed, but for some instruments the query may fail or return the wrong value. Values should be within the range in the instrument's specifications. Setting wrong values can result in invalid data without an error being triggered.

`set_descriptor_irrad_mult` *Add spectral irradiance calibration*

Description

Adds calibration data expressed as multipliers for each pixel stores in a numeric vector.

Usage

```
set_descriptor_irrad_mult(
  descriptor,
  irrad.mult,
  wl.range = NULL,
  start.date = lubridate::today(tzone = "UTC") - lubridate::days(1),
  end.date = lubridate::today(tzone = "UTC") + lubridate::days(1),
  cal.source = "unknown"
)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`
`irrad.mult`	numeric vector of the same length as the number of pixels or of length one.
`wl.range`	numeric Range of wavelengths for which the calibration is valid.
`start.date, end.date`	range of dates when calibration is valid.
`cal.source`	character Identifier or source of the calibration data, used for traceability.

Value

A copy of the argument passed for `oo_descriptor` with the irrad.mult field of the calibration data replaced by the new values.

`set_descriptor_nl` *Replace linearization function in instrument description.*

Description

Uses a user supplied function, possibly that supplied by a manufacturer like Ocean Optics stored in firmware or in any other form.

Usage

```
set_descriptor_nl(descriptor, nl.coeff = NA_real_, nl.fun = NULL)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`
`nl.coeff`	numeric vector, if nl.fun is missing, assumed to be a polynomial.
`nl.fun`	A function or a polynom::polynomial object containing the linearization to be applied.

Value

A copy of the argument passed for `oo_descriptor` with the wavelengths field of the calibration data replaced by the new values.

`set_descriptor_wl` *Replace wavelength values in an instrument description*

Description

Replace wavelength values in an instrument descriptor for an Ocean Optics spectrometer with new values. (e.g. when wavelngth calibration is not stored in firmware).

Usage

```
set_descriptor_wl(descriptor, wl)
```

Arguments

`descriptor`	list as returned by function `get_oo_descriptor`
`wl`	numeric vector of sorted wavelengths values corresponding to each pixel in the instrument array.

Value

a copy of the argument passed for `oo_descriptor` with the wavelengths field of the calibration data replaced by the new values.

set_folder_interactive

Interactively get folder to use

Description

Enter values for "user supplied" folder.

Usage

```
set_folder_interactive(folder.name = NULL)
```

Arguments

`folder.name`	character Default name of the folder.

Details

If the requested folder does not already exist it will be created. The name of the folder is returned, but NOT set as working directory.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

set_oo_ssdata_descriptor

Set the instrument description.

Description

Model and serial number are retrieved from the file header, the remaining unknown parameter values are set to NA. These values are used to set the "inst.desc" atrribute of `x`.

Usage

```
set_oo_ssdata_descriptor(
  x,
  file.header = comment(x),
  descriptor = photobiology::getInstrDesc(x),
  action = "merge"
)
```

Arguments

`x`	generic_spct, although with defaults only raw_spct.
`file.header`	character string The header of the file output by SpectraSuite.
`descriptor`	list An already built instrument descriptor, to be used as basis when action == "merge" or action == "keep". Defaults, to that stored in `x`.
`action`	character A flag indicating if an existing instrument descriptor in `x` should be overwritten, merged or kept as is if present.

Value

A copy of `x` with updated attributes.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_settings()`

`set_oo_ssdata_settings` *Set the values of instrument settings from file header*

Description

Parse the header of the file returned by SpectraSuite for corrections, smotthing and acquisition parameters used. These values are used to set the "inst.settings" atrribute of `x`.

Usage

```
set_oo_ssdata_settings(x, file.header = comment(x), overwrite = TRUE)
```

Arguments

`x`	generic_spct, although with defaults only raw_spct.
`file.header`	character string The header of the file output by SpectraSuite.
`overwrite`	logical A flag indicating if an existing valid instrument descriptor in `x` should be overwritten.

Value

A copy of `x` with updated attributes.

See Also

Other functions for importing spectral data from files: `collect_spct_files()`, `map_oofile_header_rows()`, `oofile_data_rows()`, `plot_spct_file()`, `read_files2mspct()`, `read_oo_data()`, `read_oo_ovdata()`, `read_oo_pidata()`, `read_oo_ssdata()`, `set_oo_ssdata_descriptor()`

set_seq_interactive *Interactively set sequential measurements*

Description

Enter settings defining a sequence of spectra to be measured as a time series.

Usage

```
set_seq_interactive(
  seq.settings = list(start.boundary = "second", initial.delay = 0, step.delay = 0,
    num.steps = 1),
  measurement.duration = 0,
  minimum.step.delay = measurement.duration,
  time.division = 0
)
```

Arguments

`seq.settings`	numeric Definition of time steps for a sequence of repeated measurements. Named vector with member named `start.boundary`, `"initial.delay"`, `"step.delay"`, and `"num.steps"`.
`measurement.duration`	numeric Duration of one measurement event (s).
`minimum.step.delay`	numeric Minimum duration of `"step.delay"` (s).
`time.division`	numeric The step is forced to be a multiple of this time duration, because spectrometers normally are constantly acquiring spectra and thbey return the most recently acquired one. Should be set to the integration time plus a very small overhead (s).

Details

Function `seq.settings()` allows users to enter values needed to define a sequence of spectral acquisitions. These are the time unit boundary to synchronize to, the delay or duration of the time step between successive acquisitions and the number of acquisitions in the series. The `measurement.time` determines the minimum length for `"step"`.

A sequence of measurements are expected to share a single reference or dark scan, and be done as a sequence. With a single time point, the initial delay or time unit boundary can be used to schedule a single timed measurement.

Value

A named numeric vector of length two.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_session_name_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

`set_session_name_interactive` *Interactively get session name to set*

Description

Enter values for "session" name.

Usage

```
set_session_name_interactive(session.name = NULL)
```

Arguments

`session.name`	character Default name of the folder.

Details

Validate seesionr name, and allow user to change the default.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_user_name_interactive()`, `tune_interactive()`

`set_user_name_interactive` *Interactively get user name to set*

Description

Enter values for "user supplied" name.

Usage

```
set_user_name_interactive(user.name = NULL)
```

Arguments

`user.name`	character Default name of the folder.

Details

Validate user name, and allow user to change the default.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `tune_interactive()`

`skip_bad_pixs` *Replace bad pixels*

Description

Replace bad pixels with the average of the counts from the two neighbouring pixels.

Usage

```
skip_bad_pixs(x)
```

Arguments

`x`	raw_spct object

Value

a copy of x with values replaced as needed in all counts columns present.

`start_session` *Connect to Maya spectrometer*

Description

Open a connection to the first spectrometer found and initialize an object with a reference to the Java object returned by Omni Driver.

Usage

```
start_session(error.action = stop)
```

Arguments

`error.action`	function, usually one of `stop()`, `warning()` or `message()`.

Value

On success a java wrapper which allows access to the driver with an open connection to the instrument.

See Also

Other spectrometer-connection functions: `end_session()`, `list_instruments()`

`sun001.raw_mspct` *Raw counts data for a lamp measurement.*

Description

Solar radiation at ground level measured in Helsinki Finland. The spectrometer used was an Ocean Optics Maya2000 Pro.

Usage

```
sun001.raw_mspct
```

Format

A raw_mspct

See Also

Other objects containing example raw-counts data: `blue_filter.raw_mspct`, `halogen.raw_mspct`, `red_filter.raw_mspct`, `white_LED.raw_mspct`, `xenon_flash.raw_mspct`

`s_fraction_corrected` *Convert raw counts data into a spectral fraction*

Description

Convert raw counts data into spectral transmittance or spectral reflectance.

Usage

```
s_fraction_corrected(x, ...)

## Default S3 method:
s_fraction_corrected(x, ...)

## S3 method for class 'list'
s_fraction_corrected(
  x,
  reference.value = 1,
  type = "internal",
  time = NULL,
  correction.method,
  qty.out = "Tfr",
  descriptor = NULL,
  dyn.range = NULL,
  locale = NULL,
  verbose = getOption("photobiology.verbose", default = FALSE),
  ...
)

## S3 method for class 'raw_mspct'
s_fraction_corrected(
  x,
  spct.names = c(sample = "sample", reference = "reference", dark = "dark"),
  reference.value = 1,
  type = switch(qty.out, Tfr = "internal", Rfr = "total"),
  correction.method,
  dyn.range = NULL,
  qty.out = "Tfr",
  verbose = getOption("photobiology.verbose", default = FALSE),
  ...
)
```

Arguments

`x`	A named list of one to three vectors of file names, with names "sample", "reference", and "dark". Or a raw_mspt object, or a raw_spct object.
`...`	Named argument passed to `photobiology::cps2irrad` which is the final calculation step.
`reference.value`	numeric or filter_spct or reflector_spct object, with the fractional transmittance or reflectance of the reference.
`type`	character One of "internal" or "total".
`time`	a `POSIXct` object, but if `NULL` the date stored in file is used, and if `NA` no date variable is added

`correction.method`	A named list of constants and functions defining the method to be sued for stray light and dark signal corrections.
`qty.out`	character, one of "Tfr", "Rfr".
`descriptor`	A named list with a descriptor of the characteristics of the spectrometer (if serial number does not agree an error is triggered).
`dyn.range`	numeric Effective dynamic range of the spectrometer.
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`verbose`	Logical indicating the level of warnings wanted.
`spct.names`	named character vector of length three, to map names in `x` to those expected.

Methods (by class)

- `s_fraction_corrected(default)`: Default for generic function.
- `s_fraction_corrected(list)`: Default for generic function.
- `s_fraction_corrected(raw_mspct)`: Default for generic function.

Note

Currently `s_fraction_corrected.list` allows processing of files written by OceanOptics' SpectraSuite software, from protocols with integration-time bracketing or not, with a dark reference measurement or not. Four measurements components are recognized: a "sample" measurement, a "referenece" measurement using a clear or white, a "filter" measurement with a UV-blocking filter in the light pass, and a "dark" measurement. Only "sample" and "reference" are mandatory. Data should be raw counts, either corrected for detector non-linearity or not. All three spectra should be acquired using the same instrument settings to achieve good accuracy.

See Also

Other functions for conversion of raw-counts data: `raw2corr_cps()`, `raw2cps()`, `s_irrad_corrected()`

`s_irrad_corrected` *Convert raw counts data into spectral irradiance or fluence*

Description

Convert raw counts data into spectral irradiance or fluence

Usage

```
s_irrad_corrected(x, ...)

## Default S3 method:
s_irrad_corrected(x, ...)

## S3 method for class 'list'
s_irrad_corrected(
  x,
  time = NULL,
  correction.method,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.05),
  return.cps = FALSE,
  trim.descriptor = !return.cps,
  descriptor,
  locale = NULL,
  verbose = getOption("photobiology.verbose", default = FALSE),
  ...
)

## S3 method for class 'raw_mspct'
s_irrad_corrected(
  x,
  spct.names = c(light = "light", filter = "filter", dark = "dark"),
  correction.method,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.05),
  return.cps = FALSE,
  trim.descriptor = !return.cps,
  verbose = getOption("photobiology.verbose", default = FALSE),
  ...
)

## S3 method for class 'raw_spct'
s_irrad_corrected(
  x,
  time = NULL,
  correction.method,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.05),
  return.cps = FALSE,
  trim.descriptor = !return.cps,
  verbose = getOption("photobiology.verbose", default = FALSE),
  ...
)
```

Arguments

`x`	A named list of one to three vectors of file names, with names "light", "filter", and "dark". Or a raw_mspt object, or a raw_spct object.

`...`	Named arguments passed to `photobiology::cps2irrad` which is the final calculation step.
`time`	a `POSIXct` object, but if `NULL` the date stored in file is used, and if `NA`, date is set to NA.
`correction.method`	A named list of constants and functions defining the method to be used for stray light and dark signal corrections.
`hdr.tolerance`	numeric Tolerance for mean deviation among cps columns as a fraction of one. Used in check of HDR consistency. A negative value disables merging using only the data for the shortest integration time.
`return.cps`	logical Useful when there is no need to apply a calibration, such as when computing new calibration multipliers.
`trim.descriptor`	logical If `TRUE` the spectrometer calibration constants, pixel wavelengths, slit-function "tail-correction" function code and other calibration-related information is deleted.
`descriptor`	A named list with a descriptor of the characteristics of the spectrometer (if serial number does not agree an error is triggered).
`locale`	The locale controls defaults that vary from place to place. The default locale is US-centric (like R), but you can use `locale` to create your own locale that controls things like the default time zone, encoding, decimal mark, big mark, and day/month names.
`verbose`	Logical indicating the level of warnings and messages wanted.
`spct.names`	named character vector of length three, to map names in `x` to those expected.

Methods (by class)

- `s_irrad_corrected(default)`: Default for generic function.
- `s_irrad_corrected(list)`: Default for generic function.
- `s_irrad_corrected(raw_mspct)`: Default for generic function.
- `s_irrad_corrected(raw_spct)`: Default for generic function.

Note

Currently `s_irrad_corrected.list` allows processing of files written by OceanOptics' SpectraSuite software, from protocols with integration-time bracketing or not, with a dark reference measurement or not. Three measurements components are recognized: a "light" measurement, a "filter" measurement using a polycarbonate filter and a dark measurement. Only "light" is mandatory. Data should be raw counts, either corrected for detector non-linearity or not. All three spectra should be acquired using the same instrument settings to achieve good accuracy.

Enabling `trim.descriptor` ensures that the data objects are free of references to code in 'ooacquire', which is crucial for the portability of the spectral data.

See Also

Other functions for conversion of raw-counts data: `raw2corr_cps()`, `raw2cps()`, `s_fraction_corrected()`

Tfr_summary_table *Summarize spectral transmittance*

Description

Compute transmittance by waveband of interest to plants' and human visual responses to light.

Usage

```
Tfr_summary_table(
  mspct,
  quantity = "average",
  attr2tb = "when.measured",
  summary.type = "VIS",
  digits = 3L
)
```

Arguments

`mspct`	A filter_mspct, or a filter_spct object containing spectral transmittance for one or more sources.
`quantity`	character Passed to `transmittance`.
`attr2tb`	character Vector with one or more of "when.measured", "what.measured", "where.measured", "how.measured" and "comment".
`summary.type`	character One of "plant", "PAR" or "VIS".
`digits`	integer The number of significant digits in the output.

Details

This function calls different functions from package 'photobiology' and returns a typical set of summaries.

Value

A tibble with one row per spectrum and one column per summary quantity and attribute and a column with the names of the spectra.

See Also

See the documentation for functions `transmittance`, `add_attr2tb` and `signif` which are called to build the summary table.

Examples

```
Tfr_summary_table(yellow_gel.spct)
Tfr_summary_table(yellow_gel.spct, attr2tb = c("what.measured", "where.measured"))
Tfr_summary_table(yellow_gel.spct, summary.type = "plant")
Tfr_summary_table(yellow_gel.spct, summary.type = "PAR")
Tfr_summary_table(yellow_gel.spct, summary.type = "VIS")
```

`trim_counts` *Replace out-of-range instrument counts*

Description

Trim out-of-range counts values in data stored in a "raw_spct" object. The values are replaced with the supplied `fill` value.

Usage

```
trim_counts(x, range = c(NA, getInstrDesc(x)[["max.counts"]] - 1), fill = NA)
```

Arguments

`x`	raw_spct object
`range`	integer vector of length two
`fill`	an integer value (including NA) to be used to replace the values that are outside range.

Value

a copy of x with values replaced as needed in all counts columns present.

See Also

Other spectral data-processing functions: `MAYP112785_tail_correction()`, `MAYP11278_tail_correction()`, `check_sn_match()`, `linearize_counts()`, `merge_raw_mspct()`, `new_correction_method()`, `ref_correction()`, `uvb_corrections()`

`tune_interactive` *Interactively adjust the integration time settings*

Description

Adjust integration time settings, allowing the user to repeat the tuning, and to change some of the parameters used for tuning such as total compound integration time and integration time bracketing.

Usage

```
tune_interactive(
  descriptor,
  acq.settings,
  start.int.time = 0.1,
  interface.mode = "auto"
)
```

Arguments

`descriptor`	list Descriptor of the instrument, including wrapper to Java object used to access the instrument.
`acq.settings`	list containing starting values for instrument settings.
`start.int.time`	numeric Integration time to use as starting guess when tuning the settings.
`interface.mode`	character One of "simple", "auto", or "manual".

Details

This function implements three different user interfaces: 1) "simple" is an interface suitable for the most usual measurements using automatic tuning of integration time and hides some of the less frequently used options, 2) "auto" gives access to all available options offering maximum flexibility when using automatic tuning of integration time, and 3) "manual" supports use of fixed integration times directly entered by the user.

Tuning of the integration time takes into account the range of times supported by the connected instrument, read from the instrument descriptor. The algorithm also makes use of the linearisation function when extrapolating to guess the integration time needed. Initial (default) values are read from `acq.settings` while `start.int.time` provides a default starting value for integration time for tuning when the user chooses not to use the value stored in `acq.settings`.

See Also

Other interactive acquisition utility functions: `choose_ch_interactive()`, `choose_sr_interactive()`, `f.trigger.message()`, `list_srs_interactive()`, `protocol_interactive()`, `set_attributes_interactive()`, `set_folder_interactive()`, `set_seq_interactive()`, `set_session_name_interactive()`, `set_user_name_interac`

`update_bad_pixs` *Update bad-pixels in instrument descriptor*

Description

Update field `bad.pixs` of the instrument descriptor embedded in `raw_spct` and `raw_mspct` objects.

Usage

```
update_bad_pixs(x, bad.pixs = NULL, action = "replace")
```

Arguments

`x`	raw_spct or raw_mspct object with attribute `instr.desc` set.
`bad.pixs`	numeric New vector of indexes to bad pixels in the detector array. If `NULL`, bad pixels are retrieved from calibration data in the current version of 'ooacquire'.
`action`	character One of "replace" or "add".

Details

Spectral objects, including those with raw counts data can contain an instrument descriptor. One member of this attribute is a vector of indexes to bad pixels. New bad pixels can be identified in some cases after data are acquired. Recomputing of physical quantities from raw counts normally reuses the embedded calibration data. Function `update_bad_pixs()` makes it possible to update the embedded bad pixels information before recomputing derived quantities. This function can be applied only to `raw_spct` objects created with functions from package 'ooacquire'.

With defaults arguments for formal parameters `bad.pixs` and `action`, the `bad.pixs` field of the descriptor is updated to match that in the matching calibration data in the version of 'ooacquire' currently loaded. However, if a numeric vector to positions in the detector array is passed as argument, depending on the argument passed to `action`, this vector will be used either to replace the existing one, or the indexes in the vector "added" to those already stored, using `union()`.

Value

a copy of `x` with an updated `instr.desc` attribute embedded.

Note

Only objects of class `raw_spct`, individually or as members of a `raw_mspct` object, are supported as the update must precede any conversion into physical units, and will propagate to returned values when computations are applied to the updated `raw_spct` objects.

uvb_corrections *Apply filter-based stray-light correction*

Description

Apply to a counts-per-second spectrum corrections based of a paired reading obtained with a polycarbonate filter (long-pass with cut-in at 400 nm). This is a bit more sophisticated than simple subtracting the filter reading from the measurement as the effect of the filter itself on stray light is corrected for.

Usage

```
uvb_corrections(
  x,
  spct.names = c(light = "light", filter = "filter", dark = "dark"),
  stray.light.method = "original",
  stray.light.wl = c(218.5, 228.5),
  flt.dark.wl = c(193, 209.5),
  flt.ref.wl = c(360, 379.5),
  flt.Tfr = 0.9,
  inst.dark.pixs = 1:4,
  worker.fun = NULL,
  trim = 0.05,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.05),
  verbose = getOption("photobiology.verbose", default = FALSE),
  ...
)

slit_function_correction(
  x,
  worker.fun = NULL,
  hdr.tolerance = getOption("ooacquire.hdr.tolerance", default = 0.05),
  verbose = getOption("photobiology.verbose", default = FALSE),
  ...
)
```

Arguments

x	raw_mspct The raw counts from measurements.
spct.names	named character vector of length three.
stray.light.method	Method variant used, "original" (Ylianttila), "simple", "full", "sun", "raw", "none".
stray.light.wl	numeric vector of length 2 giving the range of wavelengths to use for the final stray light correction.
flt.dark.wl, flt.ref.wl	numeric vectors of length 2 giving the ranges of wavelengths to use for the "dark" and "illuminated" regions of the array in the filter correction.

`flt.Tfr`	numeric fractional transmittance of the filter to the source of stray light, used only for method "simple".
`inst.dark.pixs`	numeric vector with indexes to array pixels that are in full darkness by instrument design.
`worker.fun`	function actually doing the correction on the w.lengths and counts per second vectors, or the name of the function as a character string.
`trim`	a numeric value to be used as argument for mean
`hdr.tolerance`	numeric Tolerance for mean deviation among cps columns as a fraction of one. Used in check of HDR consistency.
`verbose`	Logical indicating the level of warnings wanted.
`...`	additional parameters passed to worker.fun.

Value

A `cps_spct` object with the corrected count-per-second values in coulmn "cps".

Note

`stray.light.method = "none"` is a valid argument only for function `uvb_corrections()`. The default `worker.fun` is just an example. Corrections are specific to each individual spectrometer unit (not just type or configuration) and code needs to be written for most individual use cases. The slit function tail correction requires the characterization of the shape of the slit function by measuring one or more laser beams at suitable wavelengths.

Author(s)

Algorithm for method "original" developed by Lasse Ylianttila. Other methods are modified from Ylianttila's method by Pedro J. Aphalo.

References

`http://www.r4photobiology.info`

See Also

Other spectral data-processing functions: `MAYP112785_tail_correction()`, `MAYP11278_tail_correction()`, `check_sn_match()`, `linearize_counts()`, `merge_raw_mspct()`, `new_correction_method()`, `ref_correction()`, `trim_counts()`

which_descriptor — *Select which instrument descriptor to use*

Description

Select from a list of instrument descriptors which one to use based on date of measurement.

Usage

```
which_descriptor(
  date = lubridate::now(tzone = "UTC"),
  descriptors = ooacquire::MAYP11278_descriptors,
  verbose = getOption("photobiology.verbose", TRUE),
  strict.calib = getOption("photobiology.strict.calib", FALSE),
  entrance.optics = NULL,
  ...
)
```

Arguments

`date`	Any object that `anytime::anydate()` will decode as a date or convert to a date. Used to select a descriptor containing calibration data valid for a given day.
`descriptors`	A named list of descriptors of the characteristics of the spectrometer including calibration data.
`verbose`	Logical indicating the level of warnings wanted.
`strict.calib`	Logical indicating the level of validity checks.
`entrance.optics`	character The name or geometry of the diffuser or entrance optics to select. Only required when there are calibration with multiple entrance optics for the same spectrometer.
`...`	Currently ignored.

Details

Calibrations for instruments stored in a list and passed as argument to `descriptors`, also store the dates between which they are valid. This function walks the list searching for a calibration valid for `date`. If no valid calibration is found and `strict.calib = FALSE`, the calibration valid closest in time is returned with a warning while if no valid calibration is found and `strict.calib = TRUE` an error is triggered.

If a character string is passed as argument to `date`, it must be in a format suitable for `anytime::anydate()`. One needs to be careful with months and days of the month when supplying them as numbers, so using months names or their abbreviations can be safer.

Note

The default argument for `verbose` is for this function `TRUE` as conversion of other objects to a date may fail.

`white_LED.raw_mspct` *Raw counts data for a lamp measurement.*

Description

A zhaga standard module with Nichia "horticulture" LEDs measured at short distance. The spectrometer used was an Ocean Optics Maya2000 Pro.

Usage

```
white_LED.raw_mspct
```

Format

A raw_mspct

See Also

Other objects containing example raw-counts data: `blue_filter.raw_mspct`, `halogen.raw_mspct`, `red_filter.raw_mspct`, `sun001.raw_mspct`, `xenon_flash.raw_mspct`

`xenon_flash.raw_mspct` *Raw counts data for a lamp measurement.*

Description

A photography flash Godox AD200 with a bare lamp in a head with a reflector fitted. The spectrometer used was an Ocean Optics Maya2000 Pro.

Usage

```
xenon_flash.raw_mspct
```

Format

A raw_mspct

See Also

Other objects containing example raw-counts data: `blue_filter.raw_mspct`, `halogen.raw_mspct`, `red_filter.raw_mspct`, `sun001.raw_mspct`, `white_LED.raw_mspct`

Index