Package ‘photobiologyInOut’

April 2, 2020

Type Package
Title Read Spectral and Logged Data from Foreign Files
Version 0.4.22
Date 2020-04-02
Description Functions for reading, and in some cases writing, foreign files containing spectral data from spectrometers and their associated software, output from daylight simulation models in common use, and some spectral data repositories. As well as functions for exchange of spectral data with other R packages. Part of the 'r4photobiology' suite, Aphalo P. J. (2015) <doi:10.19232/uv4pb.2015.1.14>.
License GPL (>= 2)
VignetteBuilder knitr
Depends R (>= 3.5.0), photobiology (>= 0.9.30)
Imports methods, tools, stringr (>= 1.4.0), lubridate (>= 1.7.4), tibble (>= 2.1.3), dplyr (>= 0.8.1), tidyr (>= 0.8.3), readr (>= 1.3.1), readxl (>= 1.3.1), lazyeval (>= 0.2.2), colorSpec (>= 0.9-1)
Suggests hyperSpec (>= 0.99), pavo (>= 2.1.0), knitr (>= 1.23), rmarkdown (>= 1.1.3), ggplot2 (>= 3.1.1), ggspectra (>= 0.3.3), photobiologyWavebands (>= 0.4.3), testthat (>= 2.1.1)
LazyLoad yes
LazyData yes
ByteCompile true
Encoding UTF-8
URL http://www.r4photobiology.info/
BugReports https://bitbucket.org/aphalo/photobiologyinout/issues/
RoxygenNote 7.1.0
NeedsCompilation no
Author Pedro J. Aphalo [aut, cre] (https://orcid.org/0000-0003-3385-972X), Titta K. Kotilainen [ctb] (https://orcid.org/0000-0002-2822-9734), Glenn Davis [ctb]
photobiologyInOut-package

Description

Warning!

Most of the file formats supported are not standardized, and are a moving target because of changes in instrument firmware and support software. In addition the output format, especially with models, can depend on settings that users can alter. So do check that import is working as expected, and if not, please raise an issue and upload one example of an incorrectly decoded file.

Note

From version 0.4.4 the time zone (tz) used for decoding dates and times in files imported defaults to "UTC". In most cases you will need to pass the tz (or the locale) where the file was created as an argument to the functions!

Author(s)

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

Other contributors:

• Titta K. Kotilainen (ORCID) [contributor]
• Glenn Davis <gdavis@gluonics.com> [contributor]

References


See Also

Useful links:

• http://www.r4photobiology.info/
• Report bugs at https://bitbucket.org/aphalo/photobiologyinout/issues/

---

**as.colorSpec**

Convert into `colorSpec::colorSpec` objects

**Description**

Convert spectral objects (xxxx_spect, xxxx_mspct) as defined in package 'photobiology' into color-Spec objects preserving as much information as possible.
as.colorSpec

Usage

## S3 method for class 'generic_mspct'
as.colorSpec(x, spct.data.var = NULL, multiplier = 1, ...)

## S3 method for class 'generic_spct'
as.colorSpec(x, spct.data.var = NULL, multiplier = 1, ...)

## S3 method for class 'chroma_spct'
as.colorSpec(x, spct.data.var = NULL, multiplier = 1, ...)

Arguments

x 
R object

spct.data.var 
character The name of the variable to read spectral data from.

multiplier 
numeric A multiplier to be applied to the 'spc' data to do unit or scale conversion.

... 
currently ignored.

Methods (by class)

• generic_spct:
• chroma_spct:

Warning!

Always check the sanity of the returned data values, as guessing is needed when matching the different classes, and the functions defined here are NOT guaranteed to return valid data without help from the user through optional function arguments.

Note

Objects of class colorSpec::colorSpec do not contain metadata or class data from which the units of expression could be obtained. When using this function the user needs to use parameter multiplier to convert the data to what is expected by the object constructors defined in package 'photobiology' but should only rarely need to use parameter spct.data.var to select the quantity. colorSpec::colorSpec objects may use memory more efficiently than spectral objects of the classes for collections of spectra defined in package 'photobiology' as wavelengths are assumed to be the same for all member spectra, and stored only once while this assumption is not made for collections of spectra, allowing different wavelengths and lengths for the component spectra. Wavelengths are stored for each spectrum, but as spectral classes are derived from 'tbl_df' in many cases no redundant copies of wavelength data will be made in memory in spite of the more flexible semantics of the objects.

Examples

if (requireNamespace("colorSpec", quietly = TRUE)) {

}
as.generic_mspct  Convert into generic_mspct

Description

Convert into generic_mspct

Usage

## S3 method for class 'colorSpec'
as.generic_mspct(x, multiplier = 1, ...)

Arguments

x  
R object

multiplier  
numeric A multiplier to be applied to the spectral quantity data to do unit or scale conversion.

...  
currently ignored.

as.generic_spct  Coerce into generic_spct

Description

Coerce into generic_spct

Usage

## S3 method for class 'colorSpec'
as.generic_spct(x, multiplier = 1, ...)

Arguments

x  
R object

multiplier  
numeric A multiplier to be applied to the spectral quantity data to do unit or scale conversion.

...  
currently ignored.
Description

Convert 'colorSpec::colorSpec' objects into spectral objects (xxxx_spct, xxxx_mspct) as defined in package 'photobiology' and vice versa preserving as much information as possible.

Usage

colorSpec2mspct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.source_spct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.source_mspct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.response_spct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.response_mspct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.filter_spct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.filter_mspct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.reflector_spct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.reflector_mspct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.chroma_spct(x, multiplier = 1, ...)

colorSpec2spct(x, multiplier = 1, ...)

colorSpec2chroma_spct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.chroma_spct(x, multiplier = 1, ...)

## S3 method for class 'colorSpec'
as.chroma_mspct(x, multiplier = 1, ...)

mspct2colorSpec(x, spct.data.var = NULL, multiplier = 1, ...)

spct2colorSpec(x, spct.data.var = NULL, multiplier = 1, ...)

chroma_spct2colorSpec(x, spct.data.var = NULL, multiplier = 1, ...)

Arguments

x colorSpec object

multiplier numeric A multiplier to be applied to the 'spc' data to do unit or scale conversion.

... currently ignored.

spct.data.var character The name of the variable to read spectral data from.

Warning!

Always check the sanity of the imported or exported data values, as guessing is needed when matching the different classes, and the functions defined here are NOT guaranteed to return valid data without help from the user through optional function arguments.

Note

Objects of class colorSpec::colorSpec do not contain metadata or class data from which the units of expression could be obtained. When using this function the user needs to use parameter multiplier to convert the data to what is expected by the object constructors defined in package 'photobiology’ but should only rarely need to use parameter spct.data.var to select the quantity. colorSpec::colorSpec objects may use memory more efficiently than spectral objects of the classes for collections of spectra defined in package 'photobiology’ as wavelengths are assumed to be the same for all member spectra, and stored only once while this assumption is not made for collections of spectra, allowing different wavelengths and lengths for the component spectra. Wavelengths are stored for each spectrum, but as spectral classes are derived from ‘tbl_df’ in many cases no redundant copies of wavelength data will be made in memory in spite of the more flexible semantics of the objects.

Examples

# example run only if 'colorSpec' is available
if (requireNamespace("colorSpec", quietly = TRUE)) {
  library(colorSpec)
  colorSpec2mspct(Fs.5nm)
  colorSpec2spct(Fs.5nm)
  colorSpec2mspct(C.5nm)
  colorSpec2spct(C.5nm)
}
hyperSpec2mspct

Convert 'hyperSpec::hyperSpec' objects

Description

Convert hyperSpec::hyperSpec objects containing VIS and UV radiation data into spectral objects (xxxx_spct, xxxx_mspct) as defined in package 'photobiology' and vice versa, preserving as much information as possible. As hyperSpec can contain other kinds of spectral data, it does make sense to use these functions only with objects containing data that can be handled by both packages.

Usage

hyperSpec2mspct(x, member.class, spct.data.var, multiplier = 1, ...)

hyperSpec2spct(x, multiplier = 1, ...)

mspct2hyperSpec(x, spct.data.var, multiplier = 1, ...)

spct2hyperSpec(x, spct.data.var = NULL, multiplier = 1, ...)

Arguments

x
hyperSpec object

member.class
character One of the spectrum classes defined in package 'photobiology'.

spct.data.var
character The name to be used for the 'spc' data when constructing the spectral objects.

multiplier
numeric A multiplier to be applied to the 'spc' data to do unit or scale conversion. For example "a.u." units in some examples in package 'hyperSpec' seem to have scale factors applied.

...
currently ignored.

Warning!

Always check the sanity of the imported or exported data values, as guessing is needed when matching the different classes, and the functions defined here are NOT guaranteed to return valid data without help from the user through optional function arguments.

Note

Objects of class hyperSpec::hyperSpec contain metadata or class data from which the quantity measured and the units of expression can be obtained. However, units as included in the objects are not well documented making automatic conversion difficult. When using this function the user may need to use parameter multiplier to scale the data to what is expected by the object constructors defined in package 'photobiology' and use parameter spct.data.var to select the quantity.

hyperSpec::hyperSpec objects may use memory more efficiently than spectral objects of the classes for collections of spectra defined in package 'photobiology' as wavelengths are assumed
to be the same for all member spectra, and stored only once while this assumption is not made for collections of spectra, allowing different wavelengths and lengths for the component spectra. Wavelengths are stored for each spectrum, but as spectral classes are derived from `tbl_df` in many cases no redundant copies of wavelength data will be made in memory in spite of the more flexible semantics of the objects.

Examples

```r
# example run only if 'hyperSpec' is available
if (requireNamespace("hyperSpec", quietly = TRUE)) {
  library(hyperSpec)
  data(laser)
  wl(laser) <-
    list(wl = 1e7 / (1/405e-7 - wl(laser)),
         label = expression (lambda / nm))
  laser.mspct <- hyperSpec2mspct(laser, "source_spct", "s.e.irrad")
  class(laser.mspct)
}
```

Description

Reads and parses the header of a test file as available through the ASTER reflectance database. The Name field is retrieved and copied to attribute "what.measured". The header of the file is preserved as a comment.

Usage

```r
read_ASTER_txt(
  file,                
  date = NULL,        
  geocode = NULL,     
  label = NULL,       
  tz = NULL,          
  locale = readr::default_locale(),
  npixels = 2048
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>character string</td>
</tr>
<tr>
<td>date</td>
<td>a POSIXct object to use to set the &quot;when.measured&quot; attribute. If NULL, the default, the date is extracted from the file header.</td>
</tr>
<tr>
<td>geocode</td>
<td>A data frame with columns lon and lat used to set attribute &quot;where.measured&quot;.</td>
</tr>
</tbody>
</table>
Description

Reads and parses the header of a processed data file as output by the program Avaspec and then imports wavelength and spectral irradiance values. The file header has little useful metadata information.

Usage

```r
read_ivaspec_csv(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)
```

References

Arguments

- **file**: character string
- **date**: a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.
- **geocode**: A data frame with columns lon and lat used to set attribute "where.measured".
- **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 used for interpreting times saved in the file header.
- **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.
- **path**: Path to the xls/xlsx file

Value

A source_spct object.

References


Description

Reads and parses the header of a processed data file as output by the PC400 or PC200W programmes extracting variable names, units and quantities from the header. Uses the comment attribute to store the metadata.
Usage

```r
read_csi_dat(
  file,
  geocode = NULL,
  label = NULL,
  data_skip = 0,
  n_max = Inf,
  locale = readr::default_locale()
)
```

Arguments

- **file**: Path to file as a character string.
- **geocode**: A data frame with columns `lon` and `lat` used to set attribute "where.measured".
- **label**: character string, but if `NULL` the value of `file` is used, and if `NA` the "what.measured" attribute is not set.
- **data_skip**: integer Number of records (rows) to skip from the actual data block.
- **n_max**: integer Maximum number of records to read.
- **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.

Value

`read_csi_dat()` returns a `tibble::tibble` object.

Note

This function is not useful for .DAT and .PRN files from old CSI loggers and software. Those were simple files, lacking metadata, which was stored in separate .FLD files.

References

[https://www.r4photobiology.info](https://www.r4photobiology.info) [https://www.campbellsci.eu/](https://www.campbellsci.eu/)

---

**read_fmi2mspct**

Read multiple solar spectra from a data file.

Description

Read spectral irradiance file as output by Anders Lindors’ model based on libRadTrans for hourly simulation, or measured data from FMI’s Brewer spectrometer.
Usage

```r
read_fmi2mspct(
  file,
  scale.factor = 0.001,
  geocode = NULL,
  what.measured = NULL,
  how.measured = NULL,
  date.field = 2L,
  time.field = 3L,
  date.format = "ymd",
  time.format = "hms",
  tz = NULL,
  time.shift.min = 0,
  locale = readr::default_locale(),
  .skip = 0,
  .n_max = -1
)
```

Arguments

- `file` Either a path to a file, a connection, or literal data (either a single string or a raw vector).
- `scale.factor` numeric A multiplier to be applied to the spectral irradiance values.
- `geocode` A data frame with columns `lon` and `lat` used to set attribute "where.measured".
- `what.measured` character string, but if NULL the value of file is used, and if NA the "what.measured" attribute is not set.
- `how.measured` character string, but if NULL or NA the "how.measured" attribute is not set.
- `date.field`, `time.field` integer. Word positions in the header line.
- `date.format` character string. One of "ymd", "ydm", "dmy", or "mdy".
- `time.format` character string. One of "hms", "hm".
- `tz` character Time zone used for interpreting times saved in the file header.
- `time.shift.min`, numeric. Time shift with respect to TZ in minutes.
- `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.
- `.skip` Number of lines to skip before reading data.
- `.n_max` Maximum number of records to read.

Value

read_fmi2mspct() returns a `source_mspct` object containing `source_spct` objects as members, `time.unit` attribute set to "second" and `when.measured` attribute set to the date-time values extracted from the file body.
Note

See `read_table` for details of acceptable values for `file`. Individual spectra are names based on time and date in ISO format, at the time zone given by `tz` but the time shift subtracted. Say for times expressed in headers at UTC + 120 min, we use `tz = UTC` and `time.shift.min = 120` to convert times to UTC. This is different from using `tz = EET`, which is not invariant through the course of the year because of daylight saving time. Local time zones is not necessarily consistent across years because of changes in legislation. In contrast UTC is more consistent, making it preferable for time series.

---

**read_fmi_cum**

*Read daily cummulated solar spectrum data file(s).*

**Description**

Read one or more cumulated daily spectral irradiance file as output by Anders Lindors' model based on libRadTrans. Dates are read from the file header and parsed with the function supplied as `date.f`.

**Usage**

```r
read_fmi_cum(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = "UTC",
  locale = readr::default_locale(),
  .skip = 3,
  .n_max = -1,
  .date.f = lubridate::ymd
)
```

```r
read_m_fmi_cum(
  files,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = "UTC",
  .skip = 3,
  .n_max = -1,
  .date.f = lubridate::ymd
)
```

**Arguments**

- `file` Either a path to a file, a connection, or literal data (either a single string or a raw vector).
**date**

A POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.

**geocode**

A data frame with columns lon and lat used to set attribute "where.measured".

**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 used for interpreting times saved in the file header.

**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.

**.skip**

Number of lines to skip before reading data—i.e. the number of rows in the header.

**.n_max**

Maximum number of records to read.

**.date.f**

A function for extracting a date-time from the file header passed as character string to its first argument and which returns a POSIXct object.

**files**

List or vector of paths each one with the same requirements as described for argument file.

### Value

`read_fmi_cum()` returns a `source_spct` object with `time.unit` attribute set to "day" and `when.measured` attribute set to the date-time extracted from the header at the top of the read file.

`read_m_fmi_cum` returns a `source_mspct` containing one `source_spct` object for each one of the multiple files read.

### Note

See `read_table` for details of acceptable values for file.

### Examples

```r
file.name <- system.file("extdata", "2014-08-21_cum.hel", 
                         package = "photobiologyInOut", mustWork = TRUE)

fmi.spct <- read_fmi_cum(file = file.name)
```

# Description

**read_FReD_csv**

Read `.CSV` FReD database.

Reads a CSV data file downloaded from the FReD (Floral Reflectance Database) and then imports wavelengths and spectral reflectance values and flower ID.
Usage

read_FReD_csv(
  file,
  date = NA,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)

Arguments

  file character string
  date a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.
  geocode A data frame with columns lon and lat used to set attribute "where.measured".
  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 used for interpreting times saved in the file header.
  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.

Value

A reflectance_spect object.

References


read_licor_prn Read '.PRN' File(s) Saved by LI-COR's PC1800 Program.

Description

Reads and parses the header of a processed data file as output by the PC1800 program to extract the whole header remark field and also decode whether data is in photon or energy based units. The time field is ignored as it does not contain year information. This instrument is no longer being manufactured.
Usage

read_licor_prn(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale(),
  s.qty = NULL
)

read_m_licor_prn(
  files,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = Sys.timezone(),
  locale = readr::default_locale(),
  s.qty = NULL
)

Arguments

file Path to file as a character string.
date a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.
geocode A data frame with columns lon and lat used to set attribute "where.measured".
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 used for interpreting times saved in the file header.
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.
s.qty character The name of the spectral quantity to be read. One of "s.irrad", "Tfr", or "Rfr".
files A list or vector of character strings.

Details

Function read_m_licor_prn() calls red_licor_file() for each file in files. See read_table for a description of valid arguments for files.

Value

read_licor_prn() returns a source_spct object with time.unit attribute set to "second" and when.measured attribute set to the date-time extracted from the file name, or supplied.
Function `read_m_licor_prn()` returns a source_mspct object containing one spectrum per file read.

**Note**

The LI-1800 spectroradiometer does not store the year as part of the data, only month, day, and time of day. Because of this, in the current version, if `NULL` is the argument to `date`, year is set to 0000.

**References**

[https://www.r4photobiology.info](https://www.r4photobiology.info) [https://www.licor.com](https://www.licor.com)

---

**read_macam_dta**

Read `.DTA` File Saved by Macam's Software.

**Description**

Reads and parses the header of a processed data file as output by the PC program to extract the time and date fields and a user label if present, and then imports wavelengths and spectral energy irradiance values.

**Usage**

```r
read_macam_dta(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)
```

**Arguments**

- `file` character string
- `date` a POSIXct object to use to set the "when.measured" attribute. If `NULL`, the default, the date is extracted from the file header.
- `geocode` A data frame with columns `lon` and `lat` used to set attribute "where.measured".
- `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 used for interpreting times saved in the file header.
- `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.
**read_oo_jazirrad**

**Value**

A source_spec object.

**References**

https://www.r4photobiology.info http://www.irradian.co.uk/

---

**read_oo_jazirrad**  
*Read Files Saved by Ocean Optics’ Jaz spectrometer.*

**Description**

Reads and parses the header of processed data text files output by Jaz instruments extracting the spectral data from the body of the file and the metadata, including time and date of measurement from the header. Jaz modular spectrometers are manufactured by Ocean Optics, Dunedin, Florida, USA.

**Usage**

```r
read_oo_jazirrad(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)

read_oo_jazpc(
  file,
  qty.in = "Tpc",
  Tfr.type = c("total", "internal"),
  Rfr.type = c("total", "specular"),
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)

read_oo_jazdata(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)
```
Arguments

- **file**: character string.
- **date**: a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.
- **geocode**: A data frame with columns lon and lat used to set attribute "where.measured".
- **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 used for interpreting times saved in the file header.
- **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.
- **qty.in**: character string, one of "Tpc" (spectral transmittance, %), "A" (spectral absorbance), or "Rpc" (spectral reflectance, %).
- **Tfr.type**: character string, either "total" or "internal".
- **Rfr.type**: character string, either "total" or "specular".

Details

Function `read_oo_jazirrad` can read processed irradiance output files. Function `read_oo_jazpc` can read processed transmittance and reflectance output files (expressed as %s). Function `read_oo_jazdata` can read raw-counts data.

Value

A `source_spct` object, a `filter_spct` object, a `reflector_spct` object or a `raw_spct` object.

Note

Although the parameter is called `date` a date time is accepted and expected. Time resolution is 1 s.

References

https://www.r4photobiology.info https://oceanoptics.com/

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(),
  npixels = 2048,
  spectrometer.sn = "FLMS00673"
)

Arguments

- **file**: character string
- **date**: a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is set to the file modification date.
- **geocode**: A data frame with columns lon and lat used to set attribute "where.measured".
- **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.
- **npixels**: integer Number of pixels in spectral data.
- **spectrometer.sn**: character The serial number of the spectrometer needs to be supplied by the user as it is not included in the file header.

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 time 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. If no argument is passed to date this attribute is set to the file modification date obtained with file.mtime(). This date-time gives an upper limit to the real time of measurement as in some operating systems it is reset when the file is copied or even without any good apparent reason.

References

read_oo_sssirrad  
*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**

```r
read_oo_sssirrad(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)
```

```r
read_oo_sssdata(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)
```

**Arguments**

- **file**  
  character string
- **date**  
  a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.
- **geocode**  
  A data frame with columns lon and lat used to set attribute "where.measured".
- **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 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.

**Value**

- A source_spect object.
- A raw_spect object.
read_qtuv_txt

References

https://www.r4photobiology.info https://oceanoptics.com/

---

**Description**

Reads and parses the header of a text file output by the Quick TUV on-line web front-end at [http://cprm.acom.ucar.edu/Models/TUV/Interactive_TUV/](http://cprm.acom.ucar.edu/Models/TUV/Interactive_TUV/) to extract the header and spectral data. The time field is converted to a date.

**Usage**

```r
read_qtuv_txt(
  file,
  ozone.du = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)
```

**Arguments**

- `file` character string with the name of a text file.
- `ozone.du` numeric Ozone column in Dobson units.
- `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 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.

**Value**

A `source_spect` object obtained by finding the center of wavelength intervals in the Quick TUV output file, and adding variables `zenith_angle` and `date`.

**Note**

The ozone column value used in the simulation cannot be retrieved from the file. Tested only with Quick TUV version 5.2 on 2018-07-30. This function can be expected to be robust to variations in the position of lines in the imported file and resistant to the presence of extraneous text or even summaries.
References

https://www.r4photobiology.info http://cprm.acom.ucar.edu/Models/TUV/Interactive_TUV/

read_tuv_usrout  Read TUV output file.

Description

Reads and parses the header of a text file output by the TUV program to extract the header and spectral data. The time field is converted to a date.

Usage

read_tuv_usrout(
  file,
  ozone.du = NULL,
  date = lubridate::today(),
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale()
)

Arguments

file character string
ozone.du numeric Ozone column in Dobson units.
date a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.
geocode A data frame with columns lon and lat used to set attribute "where.measured".
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 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.

Value

a source_spct object obtained by 'melting' the TUV file, and adding a factor spct.idx, and vari-
ables zenith.angle and date.

Note

The ozone column value used in the simulation cannot be retrieved from the file. Tested only with
TUV version 5.0.

References

https://www.r4photobiology.info https://www2.acom.ucar.edu/modeling/tuv-download

read_uvspec_disort

Read libRadtran's uvspec output file.

Description

Read and parse a text file output by libRadtran’s uvspec routine for a solar spectrum simulation. The
output of uvspec depends among other things on the solver used. We define a family of functions,
each function for a different solver.

Usage

```r
read_uvspec_disort(
  file, 
  date = NULL, 
  geocode = NULL, 
  label = NULL, 
  tz = NULL, 
  locale = readr::default_locale(), 
  multiplier = 0.001, 
  qty = "irradiance"
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>character string</td>
</tr>
<tr>
<td>date</td>
<td>a POSIXct object to use to set the &quot;when.measured&quot; attribute. If NULL, the</td>
</tr>
<tr>
<td></td>
<td>default, the date is extracted from the file header.</td>
</tr>
<tr>
<td>geocode</td>
<td>A data frame with columns lon and lat used to set attribute &quot;where.measured&quot;.</td>
</tr>
</tbody>
</table>
read_uvspec_disort_vesa

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 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.
multiplier numeric A multiplier for conversion into W m^{-2} nm^{-1}, as the units of expression of the output from "uvspec" depend on the units in which the extraterrestrial solar spectrum data is expressed.
qty character "uvspec" returns both irradiance and intensity with solver "disort".

Value
A source_spct object.

Note
Currently only "irradiance" is supported as qty argument as intensity is not supported by classes and methods in package 'photobiology'.
Tested only with libRadtran version 2.0

References
https://www.r4photobiology.info http://www.libradtran.org

Description
Reads and parses the header and body of a text file output by a script used to run libRadtran’s uvspec in a batch job for a set of solar spectrum simulations. The header and time and date fields are converted into a datetime object.

Usage
read_uvspec_disort_vesa(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale(),
  multiplier = 1e-06,
  simplify = TRUE
)
Arguments

- file: character string
- date: a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date is extracted from the file header.
- geocode: A data frame with columns lon and lat used to set attribute "where.measured".
- 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 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.
- multiplier: numeric A multiplier for conversion into W m-2 nm-1, as the units of expression of the output from "uvspec" depend on the units in which the extraterrestrial solar spectrum data is expressed.
- simplify: logical Remove redundant columns from returned object.

Value

a source_spect object, possibly containing several spectra in long form and a datetime column.

References

https://www.r4photobiology.info http://www.libradtran.org

Description

Read '.CSV' file(s) dowloaded from YoctoPuce modules.

Usage

read_yoctopuce_csv(
  file,
  geocode = NULL,
  label = NULL,
  data_skip = 0,
  n_max = Inf,
  locale = readr::default_locale()
)
Arguments

- **file**: Path to file as a character string.
- **geocode**: A data frame with columns lon and lat used to set attribute "where.measured".
- **label**: character string, but if NULL the value of file is used, and if NA the "what.measured" attribute is not set.
- **data_skip**: integer Number of records (rows) to skip from the actual data block.
- **n_max**: integer Maximum number of records to read.
- **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.

Value

`read_yoctopuce_csv()` returns a `tibble::tibble` object.

Note

This function should be able to read data log files from any YoctoPuce USB interface module with data logging capabilities as the format is consistent among them.

References

https://www.r4photobiology.info  https://www.yoctopuce.com/

rspec2mspct

Convert "pavo::rspec" objects

Description

Convert between 'pavo::rspec' objects containing spectral reflectance data into spectral objects (xxxx_spct, xxxx_mspct) as defined in package 'photobiology'.

Usage

```r
rspec2mspct(
x,
  member.class = "reflector_spct",
  spct.data.var = "Rpc",
  multiplier = 1,
  ...
)
rspec2spct(x, multiplier = 1, ...)
```
Arguments

x                    rspec object
member.class         character One of the spectrum classes defined in package 'photobiology'.
spct.data.var        character The name to be used for the 'spc' data when constructing the spectral objects.
multiplier           numeric A multiplier to be applied to the 'rspc' data to do unit or scale conversion.
...                   currently ignored.

Warning!

Always check the sanity of the imported or exported data values, as guessing is needed when matching the different classes, and the functions defined here are NOT guaranteed to return valid data without help from the user through optional function arguments.

Note

Objects of class pavo::rspec do not contain metadata or class data from which the quantity measured and the units of expression could be obtained. When using this function the user needs to use parameter multiplier to convert the data to what is expected by the object constructors defined in package 'photobiology' and use parameter spct.data.var to select the quantity.
pavo::rspec objects may use memory more efficiently than spectral objects of the classes for collections of spectra defined in package 'photobiology' as wavelengths are assumed to be the same for all member spectra, and stored only once while this assumption is not made for collections of spectra, allowing different wavelengths and lengths for the component spectra. Wavelengths are stored for each spectrum, but as spectral classes are derived from 'tbl_df' in many cases no redundant copies of wavelength data will be made in memory in spite of the more flexible semantics of the objects.

Examples

# example run only if 'pavo' is available
if (requireNamespace("pavo", quietly = TRUE)) {
  library(pavo)
  data(sicalis, package = "pavo")
  sicalis.mspct <- rspec2mspct(sicalis)
  class(sicalis.mspct)

  data(teal, package = "pavo")
  teal.spct <- rspec2spct(teal)
  class(teal.spct)
  levels(teal.spct["spc.idx"])
  angles <- seq(from = 15, to = 75, by = 5) # from teal's documentation
  teal.spct["angle"] <- angles[as.numeric(teal.spct["spc.idx"])]
  teal.spct
}
Index

*Topic misc

read_avaspec_csv, 10
read_FReD_csv, 15
read_licor_prn, 16
read_macam_dta, 18
read_oo_ssirrad, 22
read_tuv_usrout, 24

as.chroma_mspct.colorSpec (colorSpec2mspct), 6
as.chroma_spct.colorSpec (colorSpec2mspct), 6
as.colorSpec, 3
as.filter_mspct.colorSpec (colorSpec2mspct), 6
as.filter_spct.colorSpec (colorSpec2mspct), 6
as.generic_mspct, 5
as.generic_spct, 5
as.reflector_mspct.colorSpec (colorSpec2mspct), 6
as.reflector_spct.colorSpec (colorSpec2mspct), 6
as.response_mspct.colorSpec (colorSpec2mspct), 6
as.response_spct.colorSpec (colorSpec2mspct), 6
as.source_mspct.colorSpec (colorSpec2mspct), 6
as.source_spct.colorSpec (colorSpec2mspct), 6

chroma_spct2colorSpec (colorSpec2mspct), 6
colorSpec2chroma_spct (colorSpec2mspct), 6
colorSpec2mspct, 6
colorSpec2spct (colorSpec2mspct), 6
hyperSpec2mspct, 8

hyperSpec2spct (hyperSpec2mspct), 8
locale, 10–13, 15–18, 20–23, 25–28

mspct2colorSpec (colorSpec2mspct), 6
mspct2hyperSpec (hyperSpec2mspct), 8
photobiologyInOut (photobiologyInOut-package), 2
photobiologyInOut-package, 2

read_ASTER_txt, 9
read_avaspec_csv, 10
read_avaspec_xls (read_avaspec_csv), 10
read_csi_dat, 11
read_fmi2mspct, 12
read_fmi_cum, 14
read_FReD_csv, 15
read_licor_prn, 16
read_m_fmi_cum (read_fmi_cum), 14
read_m_licor_prn (read_licor_prn), 16
read_macam_dta, 18
read_oo_jazdata (read_oo_jazirrad), 19
read_oo_jazirrad, 19
read_oo_jazpc (read_oo_jazirrad), 19
read_oo_pidata, 20
read_oo_SSdata (read_oo_ssirrad), 22
read_oo_ssirrad, 22
read_qtuv_txt, 23
read_table, 14, 15, 17
read_tuv_usrout, 24
read_tuv_usrout2mspct (read_tuv_usrout), 24
read_uvspec_disort, 25
read_uvspec_disort_vesa, 26
read_yoctopuce_csv, 27
rspec2mspct, 28
rspec2spct (rspec2mspct), 28

spct2colorSpec (colorSpec2mspct), 6
spct2hyperSpec (hyperSpec2mspct), 8

30