Package 'photobiologyInOut'

October 11, 2021

Type Package

Title Read Spectral and Logged Data from Foreign Files

Version 0.4.23

Date 2021-10-08

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 (>= 3.6.0), photobiology (>= 0.10.6)

Imports methods, tools, utils, stringr (>= 1.4.0), lubridate (>= 1.7.4), anytime (>= 0.3.9), 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 (>= 2.7.0), knitr (>= 1.31), rmarkdown (>= 2.7), ggplot2 (>= 3.3.3), ggspectra (>= 0.3.7), photobiologyWavebands (>= 0.4.3), testthat (>= 3.0.2)

LazyLoad yes

ByteCompile true

Encoding UTF-8

URL https://docs.r4photobiology.info/photobiologyInOut/

BugReports https://github.com/aphalo/photobiologyinout/issues/

RoxygenNote 7.1.2

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]

Maintainer Pedro J. Aphalo <pedro.aphalo@helsinki.fi>

Repository CRAN

Date/Publication 2021-10-11 04:10:01 UTC

R topics documented:

`photobiologyInOut-package`

photobiologyInOut: Read Spectral and Logged Data from Foreign Files

---

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>.

Data acquisition

The support for Ocean Insight, formerly Ocean Optics, spectrometers in package 'photobiologyInOut' is limited to the import of data acquired with Ocean Optics' software as is. In contrast, package 'ooacquire', part of these same suite, makes it possible to control, modify settings and acquire spectral data from Ocean Optics spectrometers directly from within R. 'ooacquire' also supports the conversion of raw-counts data into physical quantities.

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 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

Aphalo, Pedro J. (2015) The r4photobiology suite. UV4Plants Bulletin, 2015:1, 21-29. doi: 10.19232/uv4pb.2015.1.14.

See Also

Useful links:

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

as.colorSpec *Convert into 'colorSpec::colorSpec' objects*

Description

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

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 wihtout 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.

`colorSpec2mspct` *Convert 'colorSpec::colorSpec' objects*

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_mspct(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 wihtout 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 wihtout 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

```
# 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)
}
```

---

read_ASTER_txt *Read File downloaded from ASTER data base.*

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

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

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 Ignored.
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.

Value

A raw_spct object.

Note

The header in these files has metadata information, but mostly on the origin of the data. For a date and/or geocode are to be added to the return object it must be supplied by the user. as well as the date-time. Some metadata is extracted and added as attributes, while the whole header is copied to the comment attribute.

References

https://speclib.jpl.nasa.gov

Baldridge, A.; Hook, S.; Grove, C. & Rivera, G. (2009) The ASTER spectral library version 2.0. Remote Sensing of Environment. 113, 711-715

Examples

```
file.name <-
  system.file("extdata", "drygrass-spectrum.txt",
              package = "photobiologyInOut", mustWork = TRUE)

fred.spct <- read_ASTER_txt(file = file.name, npixels = Inf)

fred.spct
getWhatMeasured(fred.spct)
cat(comment(fred.spct))
```

read_avaspec_csv *Read '.csv' File Saved by Avantes' Software for AvaSpec.*

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

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

read_avaspec_xls(
  path,
  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.
path	Path to the xls/xlsx file

Value

A source_spct object.

References

https://www.avantes.com/

Examples

```
file.name <-
   system.file("extdata", "spectrum-avaspec.csv",
               package = "photobiologyInOut", mustWork = TRUE)

avaspec.spct <- read_avaspec_csv(file = file.name)

avaspec.spct
getWhatMeasured(avaspec.spct)
cat(comment(avaspec.spct))
```

---

read_csi_dat *Read '.DAT' file(s) saved by modern Campbell Scientific loggers.*

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

```
read_csi_dat(
  file,
  geocode = NULL,
  label = NULL,
  data_skip = 0,
  n_max = Inf,
  locale = readr::default_locale(),
  na = c("", "NA", "NAN"),
  ...
)
```

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.
na	character Vector of strings to interpret as missing values. Set this option to character() to indicate no missing values.
...	Further named arguments currently passed to read_csv().

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.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

```
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 suplied as `date.f`.

Usage

```
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
)

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 charecter sring 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

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

---

read_FReD_csv *Read '.CSV' FReD database.*

Description

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. Those relevant should match the format of the CSV file being read.

Value

A reflectance_spct object.

References

http://www.reflectance.co.uk Arnold SEJ, Faruq S, Savolainen V, McOwan PW, Chittka L, 2010 FReD: The Floral Reflectance Database - A Web Portal for Analyses of Flower Colour. PLoS ONE 5(12): e14287. doi:10.1371/journal.pone.0014287

Examples

```
file.name <-
  system.file("extdata", "FReDflowerID_157.csv",
              package = "photobiologyInOut", mustWork = TRUE)

fred.spct <- read_FReD_csv(file = file.name)

fred.spct
getWhatMeasured(fred.spct)
cat(comment(fred.spct))
```

---

read_li180_txt *Read '.TXT' File(s) Saved by LI-COR's LI-180 spectroradiometer.*

Description

Reads and parses the header of a data file as output by the LI-180 spectrometer (not to be confused with the LI-1800 spectrometer released in the 1980's by LI-COR) to extract the whole header remark field and also decode whether data is in photon or energy based units. This is a new instrument released in year 2020.

Usage

```
read_li180_txt(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale(),
  s.qty = "s.e.irrad"
)

read_m_li180_txt(
  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.e.irrad" or "s.q.irrad".
files	A list or vector of character strings.

Details

Function read_m_licor_espd() calls red_licor_espd() for each file in files. See read.table for a description of valid arguments for files.

Value

read_licor_espd() 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 header, or supplied by the user.

Spectrometer model, serial number and integration time are stored in attributes. The whole file header is saved as a `comment` while the footer is discarded.

Function `read_m_licor_espd()` returns a source_mspct object containing one spectrum per file read.

Note

The LI-180 spectroradiometer stores little information of the instrument and settings, possibly because they cannot be altered by the user or configured. The length of the file header does not seem to be fixed, so the start of the spectral data is detected by searching for "380nm".

References

LI-COR Biosciences, Environmental.

Examples

```
file.name <-
  system.file("extdata", "LI-180-irradiance.txt",
              package = "photobiologyInOut", mustWork = TRUE)

licor180.spct <- read_li180_txt(file = file.name)

licor180.spct
getWhenMeasured(licor180.spct)
getWhatMeasured(licor180.spct)
cat(comment(licor180.spct))
```

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

Description

Read and parse 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 irradiance in photon or energy based units, transmittance, reflectance or absorbance and then extract the wavelength and spectral data. PC1800 is an MS-DOS program provided for use with the LI-1800 spectrometer. This instrument was released in the 1980's and was sold until the early 2000's. It was very popular and several of them remain in use. (It should not be confused with the LI-180, a new spectrometer released released in 2020.)

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_prn() 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

LI-COR Biosciences, Environmental.

Examples

```
file.name <-
  system.file("extdata", "spectrum.PRN",
              package = "photobiologyInOut", mustWork = TRUE)

licor.spct <- read_licor_prn(file = file.name)

licor.spct
getWhenMeasured(licor.spct)
getWhatMeasured(licor.spct)
cat(comment(licor.spct))
```

---

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

```
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.

Value

A source_spct object.

References

https://www.irradian.co.uk/

Examples

```
file.name <-
  system.file("extdata", "spectrum.DTA",
              package = "photobiologyInOut", mustWork = TRUE)

macam.spct <- read_macam_dta(file = file.name)

macam.spct
getWhenMeasured(macam.spct)
getWhatMeasured(macam.spct)
cat(comment(macam.spct))
```

---

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 were manufactured by Ocean Optics. The company formerly named Ocean Optics is now called Ocean Insight.

Usage

```
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 if seconds are entered with a decimal fraction, such as "2021-10-05 10:10:10.1234".

References

https://www.oceaninsight.com/

Examples

```
file.name <-
  system.file("extdata", "spectrum.jaz",
              package = "photobiologyInOut", mustWork = TRUE)

jaz.spct <- read_oo_jazpc(file = file.name)

jaz.spct
getWhenMeasured(jaz.spct)
getWhatMeasured(jaz.spct)
cat(comment(jaz.spct))
```

---

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. The company formerly named Ocean Optics is now called Ocean Insight.

Usage

```
read_oo_pidata(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale(),
  npixels = Inf,
  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. 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. The user may wish to supply the date-time as an argument, but 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. The user may need to supply the number of pixels in the array although the default of npixels = Inf usually works and triggers no warnings.

References

https://www.oceaninsight.com/ https://www.raspberrypi.org/

Examples

```
file.name <-
  system.file("extdata", "spectrum.pi",
              package = "photobiologyInOut", mustWork = TRUE)

oopi.spct <- read_oo_pidata(file = file.name)

oopi.spct
getWhenMeasured(oopi.spct)
getWhatMeasured(oopi.spct)
cat(comment(oopi.spct))
```

---

read_oo_ssirrad *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. SpectraSuite was a program, now replaced by OceanView. The company formerly named Ocean Optics is now called Ocean Insight.

Usage

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

read_oo_ssdata(
  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_spct object.

A raw_spct object.

References

https://www.oceaninsight.com/

Examples

```
file.name <-
  system.file("extdata", "spectrum.SSIrrad",
              package = "photobiologyInOut", mustWork = TRUE)

ooss.spct <- read_oo_ssirrad(file = file.name)

ooss.spct
getWhenMeasured(ooss.spct)
getWhatMeasured(ooss.spct)
cat(comment(ooss.spct))
```

---

read_qtuv_txt *Read Quick TUV output file.*

Description

Reads and parses the header of a text file output by the Quick TUV on-line web front-end at UCAR to extract the header and spectral data. The time field is converted to a date.

Usage

```
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_spct 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 versison 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 resistent to the presence of extraneous text or even summaries.

References

https://www.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()
)

read_tuv_usrout2mspct(
  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 variables 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://www2.acom.ucar.edu/modeling/tropospheric-ultraviolet-and-visible-tuv-radiation-model

---

`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

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

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.
`qty`	character "uvspec" returns both irradiance and intensity with solver "disort".

Value

A source_spct object.

Note

Currently only "irradiance" is suported 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

`read_uvspec_disort_vesa` *Read libRadtran's uvspec output file from batch job.*

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

References

http://www.libradtran.org

---

read_wasatch_csv *Read File Saved by Wasatch's Enlighten.*

Description

Read wavelength and spectral data from the data section of a file as output by Enlighten importing them into R. Parse the header of a file to extract the acquisition time, instrument name and serial number, as well additional metadata related to the instrument and its settings. Function read_wasatch_csv() only accepts "column oriented" CSV files.

Usage

```
read_wasatch_csv(
  file,
  date = NULL,
  geocode = NULL,
  label = NULL,
  tz = NULL,
  locale = readr::default_locale(),
  s.qty = NULL,
  extra.cols = "keep",
  scale.factor = 1,
  ...
)
```

Arguments

file	character
date	a POSIXct object to use to set the "when.measured" attribute. If NULL, the default, the date and time are 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 that of the machine's locale.
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, possibly named. The name of the quantity using the conventions accepted used in package 'photobiology' that is to be imported from column "Processed" from the file.
extra.cols	character What to do non-processed data columns if present in file. One of "keep", "drop.pixel", "drop" or "split".
scale.factor	numeric vector of length 1, or length equal to the number of rows (= detector pixels). Numeric multiplier applied to returned spectral values.
...	additional arguments passed to the constructor of the spectrum object.

Details

Enlighten's column-wise CSV files contain at least two columns, Wavelength and Processed. In the header the Technique used is recorded. Additional data columns can be present. Column Pixel contains the pixel index in the array as integers. Columns Raw, Dark and Reference contain detector counts data. Technique is used to guess the type of spectrum stored in the column named Processed, which can be detector counts or derived values. By default the data are read into a single spectrum object and all columns retained, but only the data in Processed are interpreted as spectral data corresponding to the class of the object. If passed extra.cols = "drop", only Wavelength and Processed are copied to the returned object, while if passed extra.cols = "drop.pixel" only the contents of column Pixel are discarded. If passed extra.cols = "split" all columns containing spectral data are each read into a separate spectrum, these are collected and a "generic_mspct" object containing them returned. extra.cols can be a named vector of mappings, of length at least one but possibly longer. If longer a "generic_mspct" is returned, otherwise a spectrum object as inferred from the name each column is mapped to.

Value

An object of a class derived from generic_spct such as raw_spct or filter_spct. generic_spct is derived from tibble and data frame.

Acknowledgements

We thank Ruud Niesen from Photon Mission (https://www.photonmission.com/) for organizing the loan of the spectrometer used to produce the various files needed for the development of this function.

Note

Enlighten, the free software from Wasatch Photonics can save spectra in a variety of additional formats: different types of CSV files, plain text and JSON. Plain text files contain no metadata or even column headers and if the need arises can be read with R function read.table(). JSON files contain the most detailed metadata.

References

https://wasatchphotonics.com/ https://wasatchphotonics.com/product-category/software/

Examples

```
file.name <-
  system.file("extdata", "enlighten-wasatch-scope.csv",
              package = "photobiologyInOut", mustWork = TRUE)

wasatch.raw.spct <-
  read_wasatch_csv(file = file.name)
summary(wasatch.raw.spct)

wasatch.raw.spct <-
  read_wasatch_csv(file = file.name, s.qty = "counts")
summary(wasatch.raw.spct)

wasatch.raw.spct <-
  read_wasatch_csv(file = file.name, s.qty = c(Processed = "counts"))
summary(wasatch.raw.spct)

wasatch.raw.spct <-
  read_wasatch_csv(file = file.name, extra.cols = "drop")
summary(wasatch.raw.spct)
```

---

read_yoctopuce_csv *Read '.CSV' file(s) downloaded from YoctoPuce modules.*

Description

Reads and parses the header of processed data CSV files as output by the virtual- or hardware-hubs and modules from Yoctopuce. Uses the comment attribute to store the metadata.

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.

Details

Yoctopuce modules are small USB connected and USB powered, but isolated, very high quality miniature data acquisition and interface modules. All modules capable of data acquisition can log measured data autonomously and these data can be locally or remotely downloaded as a CSV file. (It is also possible and very easy to access these modules from R using package 'reticulate' and the Python library provided by Yoctopuce, or to send commands and retrieve data through the built-in HTML server of the modules or dedicated hubs.)

Value

read_yoctopuce_csv() returns a tibble::tibble object, with the number of columns dependent on the CSV file read.

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.yoctopuce.com/

Examples

```
 # We read a CSV file previously downloaded from a YoctoMeteo module.

file.name <-
  system.file("extdata", "yoctopuce-data.csv",
              package = "photobiologyInOut", mustWork = TRUE)

yoctopc.tb <- read_yoctopuce_csv(file = file.name)

yoctopc.tb
```

```
cat(comment(yoctopc.tb))
```

`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

```
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 wihtout 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[["spct.idx"]])
  angles <- seq(from = 15, to = 75, by = 5) # from teal's documentation
  teal.spct[["angle"]] <- angles[as.numeric(teal.spct[["spct.idx"]])]
  teal.spct
}
```

Index