photobiologyInOut Version 0.4.6.9000
User Guide

Pedro J. Aphalo

July 19, 2016

Contents

1 Introduction

```
# this may be needed in some geographic locations as some Windows TZ strings are
# not recognized by all versions of R
Sys.setenv(TZ = 'UTC')
library(photobiology)
library(photobiologyWavebands)
library(photobiologyInOut)
library(lubridate)
library(ggplot2)
library(ggmap)
library(ggspectra)
library(hyperSpec)
library(colorSpec)
library(pavo)
library(readr)
```

```
options(tibble.print_max = 5)
options(tibble.print_min = 3)
```

This package defines functions for importing spectral data from different instruments, simulation models, and for data exchange with R packages 'hyperSpec' and 'pavo' (Table 1).

Table 1: Functions for importing measured and simulated spectral emission data.

R function	Instrument	Program	`class` of value
read_oo_ssirrad()	Ocean Optics spectrom.	SpectraSuite	source_spct
read_oo_ssdata()	Ocean Optics spectrom.	SpectraSuite	raw_spct
read_oo_jazirrad()	Ocean Optics Jaz	*instrument*	source_spct
read_oo_jazdata()	Ocean Optics Jaz	*instrument*	raw_spct
read_oo_pidata()	Ocean Optics spectrom.	STS DK (Raspbian)	raw_spct
read_avaspec_csv()	Avantes spectrom.	*instrument*	source_spct
read_macam_file()	Macam	*instrument*	source_spct
read_licor_file()	LI-COR LI-1800	PC1800 (MS-DOS)	source_spct
read_m_licor_file()	LI-COR LI-1800	PC1800 (MS-DOS)	source_mspct
R function	Simulation model	Version	`class` of value
read_tuv_usrout()	TUV (S. Madronich)	version 5.0	source_spct
read_fmi_cum()	(A. Lindfors)	daily cumulated	source_spct
read_m_fmi_cum()	(A. Lindfors)	daily cumulated	source_mspct
read_libradtran_vesa()	(T. & V. Kotilainen)	irradiance	source_spct
R function	R package	Function	`class` of value
hyperSpec2mspct()	'hyperSpec'	import	source_mspct
mspct2hyperSpec()	'hyperSpec'	export	hyperSpec
rspec2mspct()	'pavo'	import	source_mspct

All functions attempt to decode and store as metadata as much of the information present in file headers as possible. In most cases, the unchanged header of the file is stored as is as a comment in the constructed objects.

It should be remembered, though, that this package has been developed based on the example files I had access to. Files from the same instruments with different hardware configurations, different firmware versions, or even settings may differ substantially. In many cases the output is produced by software in a host computer rather by the instrument itself, adding further uncertainties and possible differences due to for example the operating system of the host computer. A further complication is that in some cases the format of dates, times and numbers depends on the locale settings in use at the time of data acquisition, or analysis. For all those reasons, do expect to have to do some debugging, and most importantly always validate the imported data against the original file (remembering to run a new validation each time there is a software or firmware update) or update of this package as I test each version before release only with the example files I have available, which are not many.

2 Examples

2.1 Ocean Optics Jaz

2.1.1 Raw detector counts

Reading a raw data file generated by Ocean Optics' Jaz spectrometer. The light source was the Jaz PX pulsed Xenon light module.

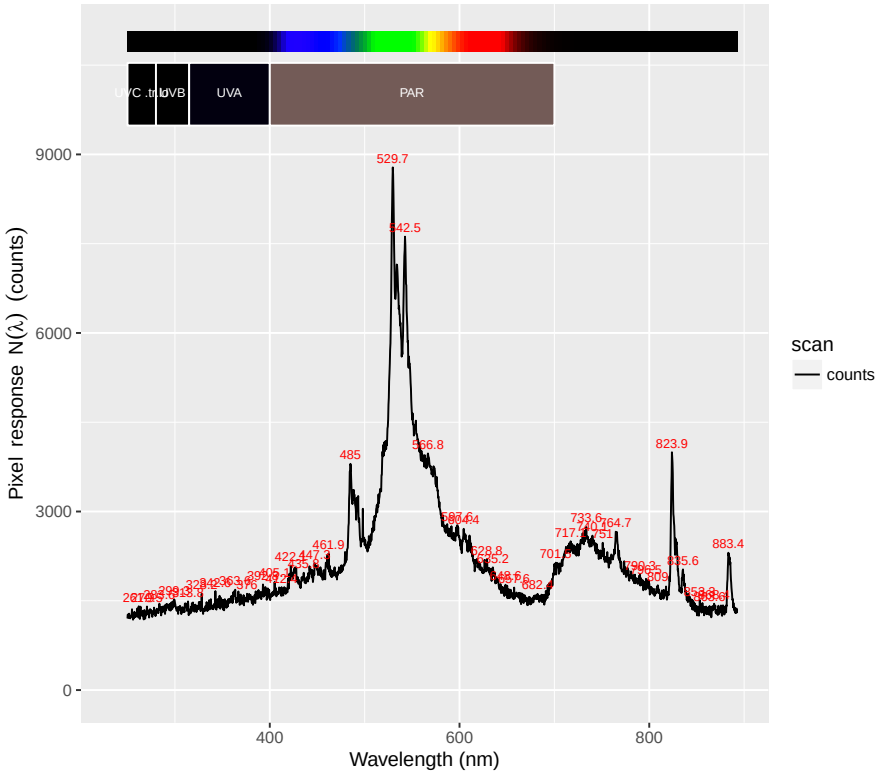
The first few lines of the file look like this, with W for wavelength, D for dark, R for reference, S for sample and P for processed (all spectral data values are raw detector counts):

```
Jaz Data File
++++++++++++++++++++++++++++++++++++
Date: Mon Apr 25 12:49:11 2016
User: jaz
Dark Spectrum Present: Yes
Reference Spectrum Present: Yes
Processed Spectrum Present: Yes
Spectrometers: JAZA3098
Integration Time (usec): 748000 (JAZA3098)
Spectra Averaged: 1 (JAZA3098)
Boxcar Smoothing: 0 (JAZA3098)
Correct for Electrical Dark: No (JAZA3098)
Strobe/Lamp Enabled: Yes (JAZA3098)
Correct for Detector Non-linearity: No (JAZA3098)
Correct for Stray Light: No (JAZA3098)
Number of Pixels in Processed Spectrum: 2048
>>>>>Begin Processed Spectral Data<<<<<
W D R S P
190.313904 0.000000 0.000000 0.000000 0.000000
190.695511 0.000000 0.000000 0.000000 0.000000
191.077087 1138.953125 1123.134277 1102.795898 228.570541
191.458633 1184.149658 1227.086426 1059.859131 -289.473419
191.840149 1175.110352 1193.188965 1132.173584 -237.500336
...
```

```
jazraw.spct <- read_oo_jazdata(file = "data-vignettes/spectrum.jaz")
jazraw.spct <- trim_wl(jazraw.spct, range = c(250, 900))
```

Plotting the spectrum.

```
plot(jazraw.spct)
```



The metadata stored in attributes can be accessed with functions. It is clear, that not all settings can be recovered from the file. However, we store the record will all the fields which would have been filled if the data had been acquired directly from R using package 'ooacquire'.

```
getWhenMeasured(jazraw.spct)

## [1] "2016-04-25 12:49:11 UTC"
```

```
getInstrDesc(jazraw.spct)

## $time
## [1] "2016-04-25 12:49:11 UTC"
##
## $w
```

```
## NULL
##
## $sr.index
## [1] NA
##
## $ch.index
## [1] NA
##
## $spectrometer.name
## [1] "Jaz"
##
## $spectrometer.sn
## [1] "JAZA3098"
##
## $bench.grating
## [1] NA
##
## $bench.filter
## [1] NA
##
## $bench.slit
## [1] NA
##
## $min.integ.time
## [1] NA
##
## $max.integ.time
## [1] NA
##
## $max.counts
## [1] NA
##
## $wavelengths
## [1] NA
##
## $bad.pixs
## numeric(0)
##
## $inst.calib
## list()
```

```
getInstrSettings(jazraw.spct)
```

```
## $time
## [1] "2016-04-25 12:49:11 UTC"
##
## $w
## NULL
##
## $sr.index
## [1] NA
##
## $ch.index
## [1] NA
##
## $correct.elec.dark
```

```
## No
##  0
##
## $correct.non.lin
## No
##  0
##
## $correct.stray.light
## No
##  0
##
## $boxcar.width
## [1] "0"
##
## $integ.time
## [1] "748000"
##
## $num.scans
## [1] "1"
```

2.1.2 Spectral energy irradiance

Reading an "Absolute Irradiance File" (sic) generated by Ocean Optics' Jaz spectrometer results in a `source_spct` object. In this example, the light source measured was a 'white' fluorescent tube.

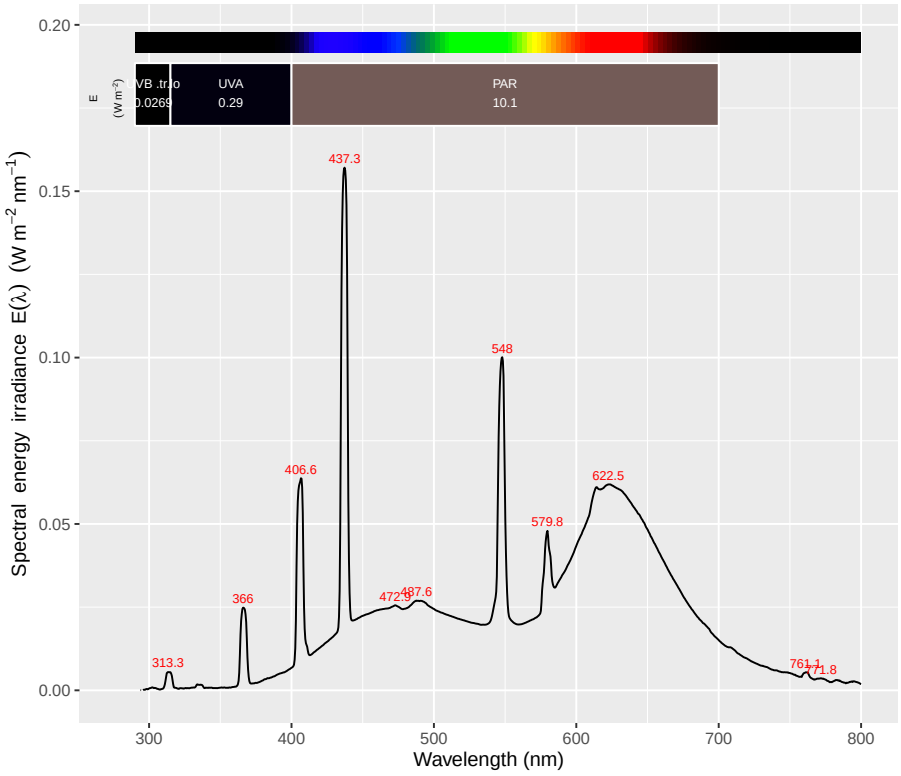
The first few lines of the file look like this:

```
Jaz Absolute Irradiance File
++++++++++++++++++++++++++++++++++++
Date: Tue Feb 03 09:44:41 2015
User: jaz
Dark Spectrum Present: Yes
Processed Spectrum Present: Yes
Spectrometers: JAZA1065
Integration Time (usec): 193000 (JAZA1065)
Spectra Averaged: 3 (JAZA1065)
Boxcar Smoothing: 5 (JAZA1065)
Correct for Electrical Dark: Yes (JAZA1065)
Strobe/Lamp Enabled: No (JAZA1065)
Correct for Detector Non-linearity: Yes (JAZA1065)
Correct for Stray Light: No (JAZA1065)
Number of Pixels in Processed Spectrum: 2048
Fiber (micron): 3900
Collection Area: 0.119459
Int. Sphere: No
>>>>>Begin Processed Spectral Data<<<<<
W D S P
188.825226 0.000000 0.000000 0.000000
189.284851 0.000000 0.000000 0.000000
189.744415 -89.659378 -90.917900 -0.000000
190.203964 -106.165916 -96.419785 0.000000
...
```

```
jaz.spct <- read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad")

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032

jaz0.spct <- jaz.spct
jaz.spct <- trim_wl(jaz.spct, range = c(290, 800))

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032
```

Plotting the spectrum.

```
plot(jaz.spct)

## Warning:  Removed 9 rows containing non-finite values
## (stat_peaks).
## Warning:  Removed 9 rows containing non-finite values
## (stat_wb_irrad).
## Warning:  Removed 2 rows containing missing values (geom_path).
```

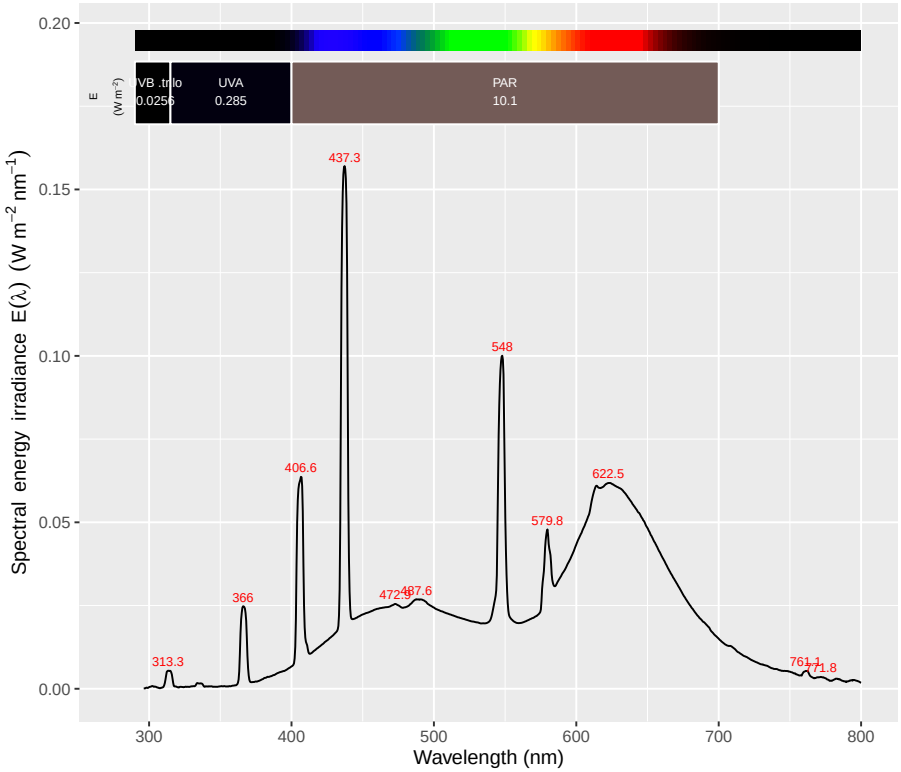


2.1.3 Cleaning spectral data

We can see that the data have problems. We get a warning because the data include negative values for spectral irradiance. We will use some methods from package 'photobiology' to correct the problem. As the data are noisy we cannot just shift the scale so that the most negative value becomes zero. Neither can we replace all negative values with zeros, as this would create bias.

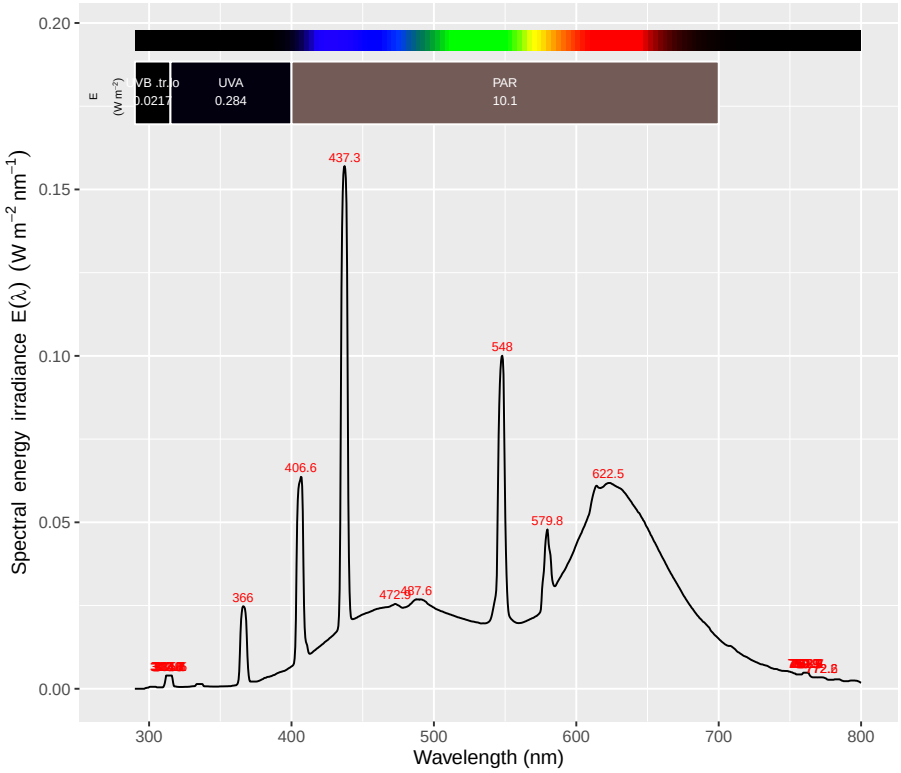
In the following code chunk we will use a region of the spectrum in which spectral irradiance is known to be equal to zero as reference to shift the scale zero. Afterwards we discard data "known" to be zero, and for which the instrument calibration is not valid, and finally we plot the spectrum.

```
jaz.spct <- fshift(jaz0.spct, range = c(255, 290), f = "mean")

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032
## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.00035
## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.00035

jaz.spct <- trim_wl(jaz.spct, range = c(290, 800))

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032

plot(jaz.spct)

## Warning:  Removed 13 rows containing non-finite values
## (stat_peaks).
## Warning:  Removed 13 rows containing non-finite values
## (stat_wb_irrad).
## Warning:  Removed 9 rows containing missing values (geom_path).
```



We can next try to smooth the spectrum as it is very noisy outside the visible region.

```
jaz.spct <- smooth_spct(jaz.spct)
plot(jaz.spct)
```

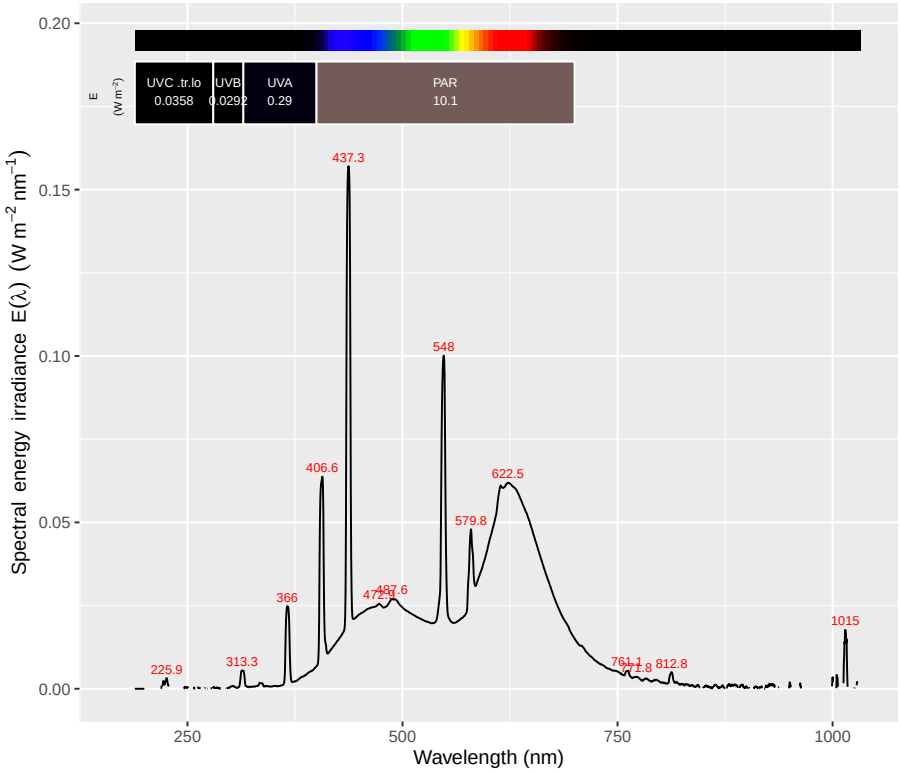


Photon and energy irradiances.

```
e_irrad(jaz.spct, PAR())       # W m-2

##      PAR
## 10.10459
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "energy irradiance total"
```

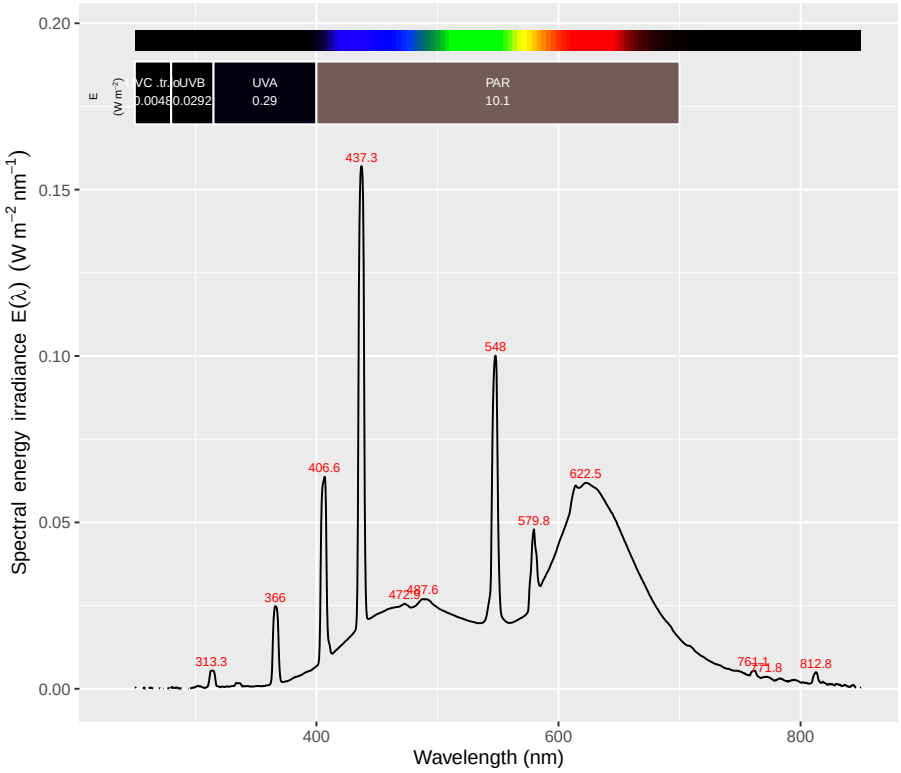
All in one statement.

```
plot(read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad"))

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032
## Warning:  Removed 378 rows containing non-finite values
## (stat_peaks).
## Warning:  Removed 378 rows containing non-finite values
## (stat_wb_irrad).
## Warning:  Removed 1 rows containing missing values (geom_path).
```



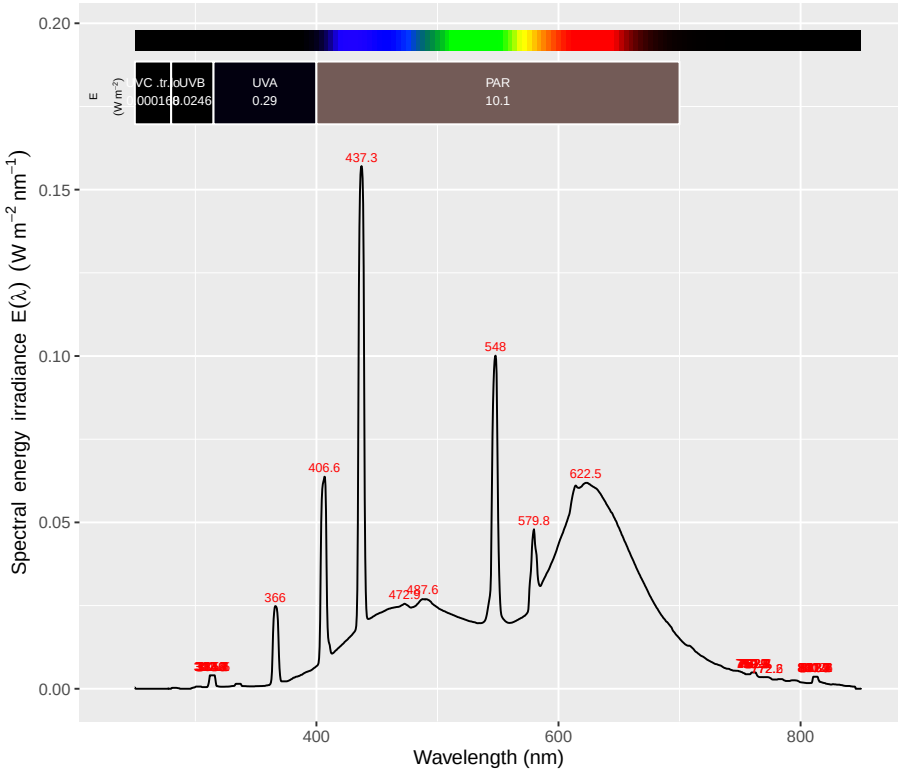
As above but limiting the wavelength range plotted.

```
plot(read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad"),
     range = c(250,850))

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032
## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032
## Warning:  Removed 52 rows containing non-finite values
## (stat_peaks).
## Warning:  Removed 52 rows containing non-finite values
## (stat_wb_irrad).
```



Adding our custom "adaptive" smoothing.

```
plot(smooth_spct(read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad")),
     range = c(250,850))

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032
## Warning in smooth_spct.source_spct(read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad")):
89 'bad' estimates in spectral irradiance
## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.027
## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.027
```



2.2 Other modular spectrometers from Ocean Optics

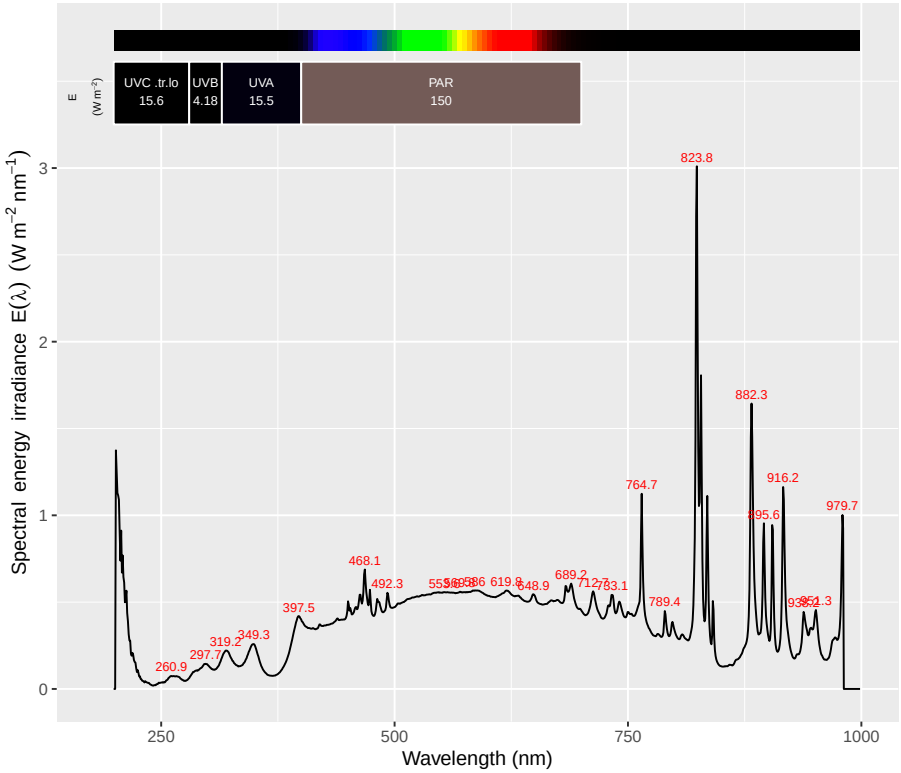
Now a file from an Ocean Optics' Q6500 spectrometer, with data processed with the Spectra Suite software.

Format of the header is similar, but not identical. The first few lines of the file look like this:

```
SpectraSuite Data File
++++++++++++++++++++++++++++++++++
Date: Mon May 06 15:13:40 CEST 2013
User: User
Dark Spectrum Present: Yes
Reference Spectrum Present: No
Number of Sampled Component Spectra: 1
Spectrometers: QEB1523
Integration Time (usec): 100000 (QEB1523)
Spectra Averaged: 1 (QEB1523)
Boxcar Smoothing: 0 (QEB1523)
Correct for Electrical Dark: No (QEB1523)
Strobe/Lamp Enabled: No (QEB1523)
Correct for Detector Non-linearity: No (QEB1523)
Correct for Stray Light: Yes (QEB1523)
Number of Pixels in Processed Spectrum: 1044
>>>>>Begin Processed Spectral Data<<<<<
199.08 0.0000E00
```

```
199.89 0.0000E00
200.70 0.0000E00
...
```

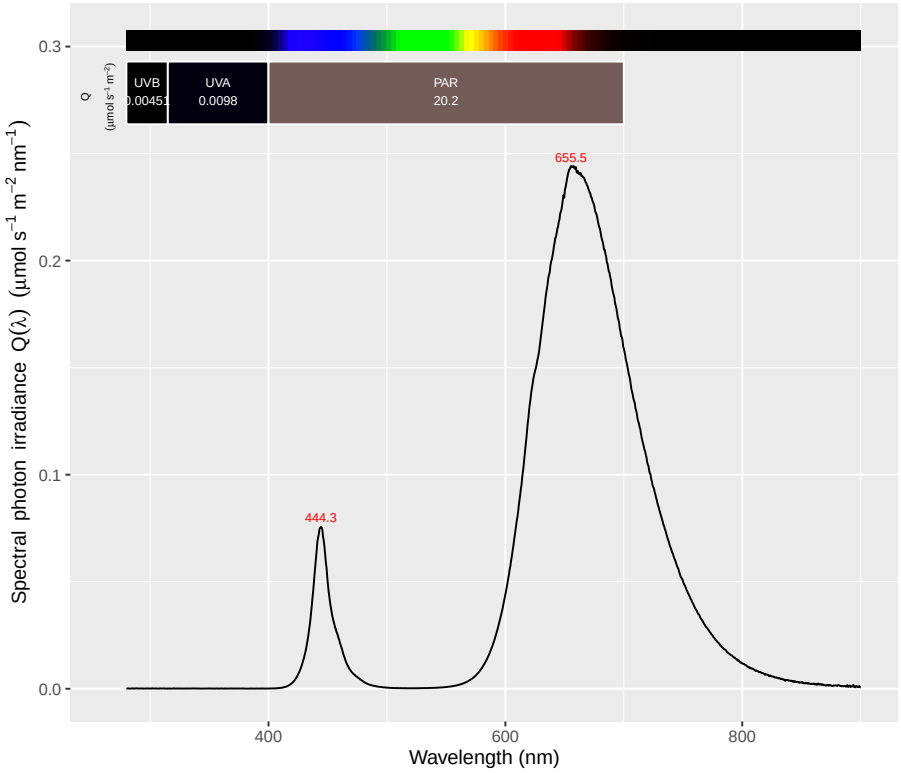
```
plot(read_oo_ssirrad(file = "data-vignettes/spectrum.SSIrrad"))
```



2.3 Modular spectrometers from Avantes

Avantes' two column .csv files can also be imported.

```
plot(read_avaspec_csv(file = "data-vignettes/spectrum-avaspec.csv"),
     range = c(280, 900), unit.out = "photon")

## Warning in range_check(x, strict.range = strict.range): Negative spectral photon
irradiance values; minimun s.q.irrad = -1.5e-08
```



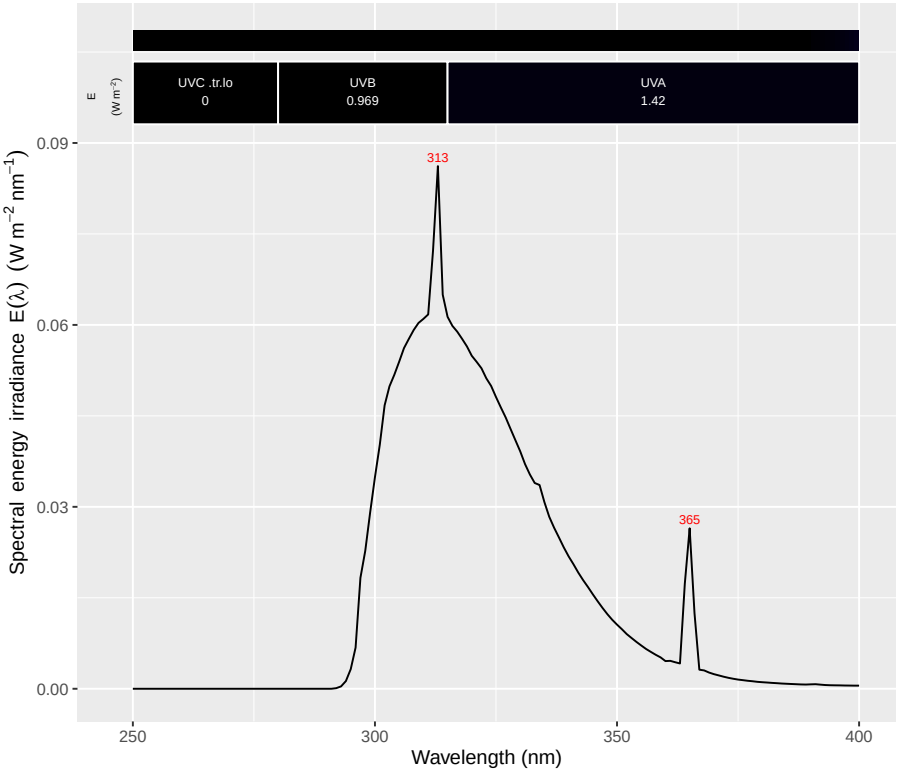
2.4 Scanning spectrometer from Macam

Macam's single column DTA files can also be imported.

The first few lines of the file look like this with all data in a single column with alternate rows for wavelengths (in nm) and irradiances, and a very terse header:

```
@19/5/1997
@17:44:58
#No Title
 2.5000000000E+02
 0.0000000000E+00
 2.5100000000E+02
 0.0000000000E+00
 2.5200000000E+02
 0.0000000000E+00
...
```

```
plot(read_macam_dta(file = "data-vignettes/spectrum.DTA"))
```



2.5 LI-1800 scanning spectrometer from LI-COR

And a file generated by LI-COR's PC1800 program for the LI-1800 spectroradiometer.

The output has a relatively detailed header, but it lacks year information. Files can contain either energy or photon based spectral irradiances, and this is signalled in the header. In this example photon (= quantum) spectral irradiance is returned. The first few lines of the file look like this:

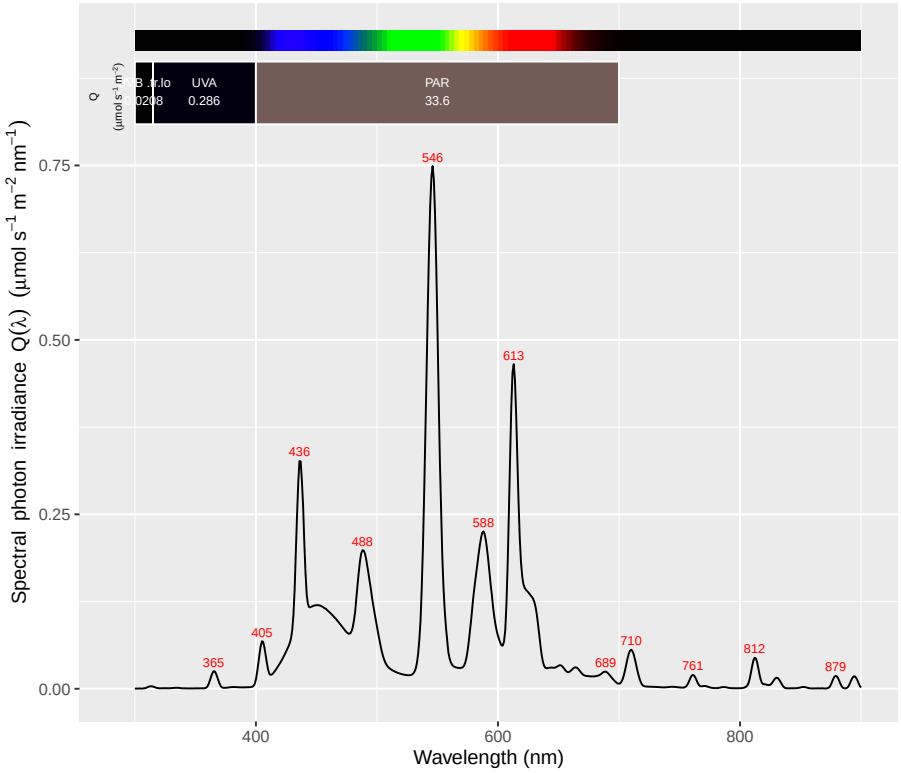
```
"FILE:FL2"
"REM: TLD 36W/865       (QNTM)"
"LIMS: 300- 900NM"
"INT:  1NM"
"DATE:08/23 16:32"
"MIN:  300NM  1.518E-04"
"MAX:  546NM  7.491E-01"
 300  1.518E-04
 301  3.355E-04
 302  2.197E-04
 303  3.240E-04
...
```

Function `read_licor_prn` will automatically detect whether the data is energy or photon based.

```
licor.spct <- read_licor_prn(file = "data-vignettes/spectrum.PRN")
```

In all cases as much information as possible is decoded, and the data file headers are preserved as comments in the source.spct objects.

```
licor.spct

## Object: source_spct [601 x 2]
## Wavelength range 300 to 900 nm, step 1 nm
## Label: File: data-vignettes/spectrum.PRN
## Measured on 0000-08-23 16:32:00 UTC
## Time unit 1s
##
## # A tibble: 601 x 2
##   w.length s.q.irrad
##      <dbl>     <dbl>
## 1      300 1.518e-10
## 2      301 3.355e-10
## 3      302 2.197e-10
## # ... with 598 more rows

cat(comment(licor.spct))

## LICOR LI-1800 file 'data-vignettes/spectrum.PRN' imported on 2016-07-19 18:49:20 UTC
## "FILE:FL2"
## "REM: TLD 36W/865       (QNTM)"
## "LIMS: 300- 900NM"
## "INT:  1NM"
## "DATE:08/23 16:32"
## "MIN:  300NM  1.518E-04"
## "MAX:  546NM  7.491E-01"

plot(licor.spct, unit.out = "photon")
```



3 Output from simulation models

3.1 TUV

The output from the TUV model can be imported either by editing it before import, or by making a simple edit to the output routine of TUV. This function is known to work with TUV version 5.0 output. The output from TUV can contain a variable number of spectra in "parallel" columns, which are *melted* into a single column, with a factor with letter as levels, a numeric variable with the zenith angle and a POSIXct column with times. A date needs to be always supplied as the output file from TUV has only time of day information.

```
tuv.spct <- read_tuv_usrout(file = "data-vignettes/usrout.txt",
                            date = ymd("2014-03-21"))
summary(subset(tuv.spct, spct.idx == "A"))

## Summary of object: source_spct [482 x 5]
## containg  8  spectra in long form
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit: 1s
##
##     w.length        spct.idx     s.e.irrad
```

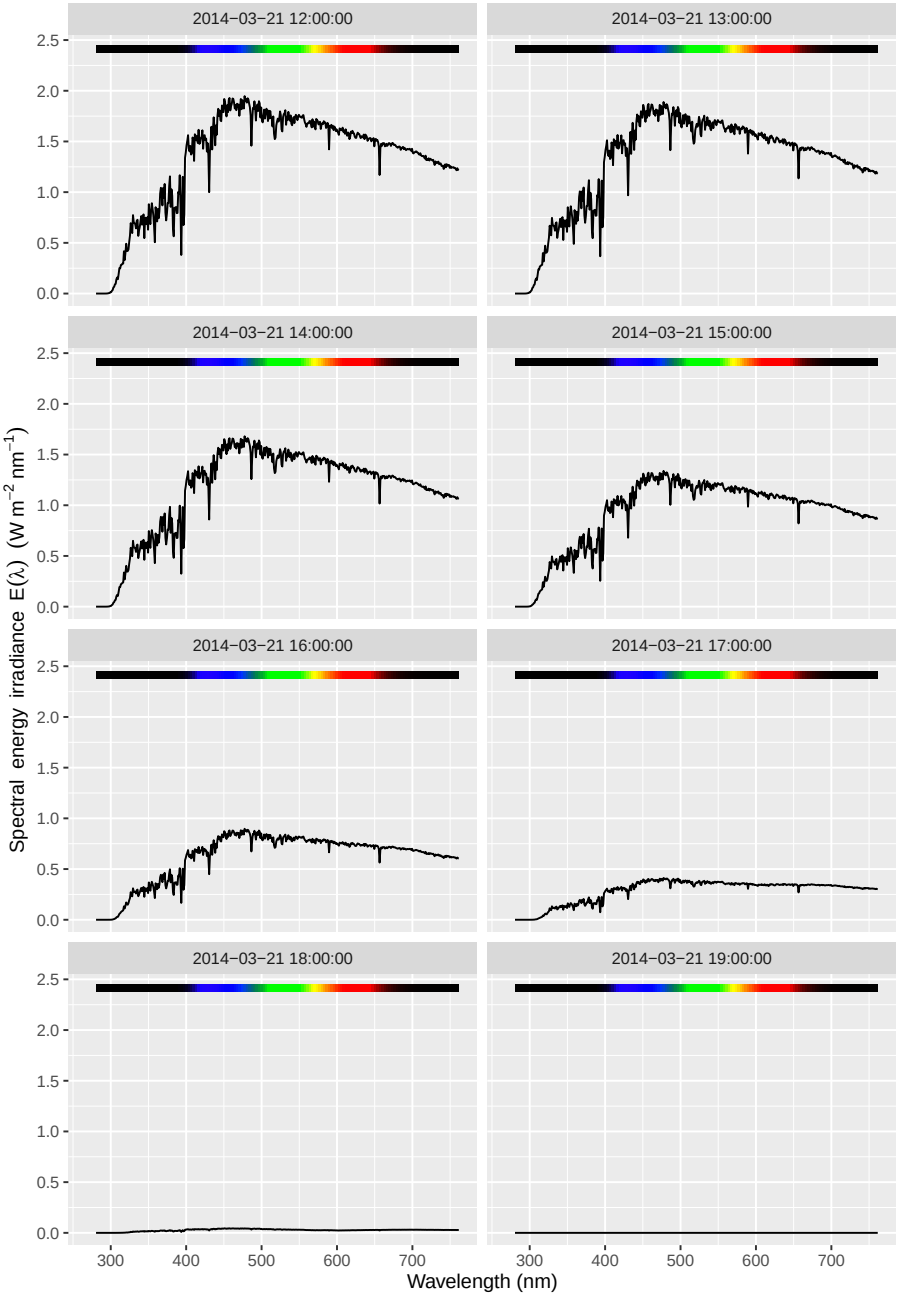
```
##  Min.   :280.5   A      :482   Min.   :0.000
##  1st Qu.:400.8   B      :  0   1st Qu.:1.216
##  Median :521.0   C      :  0   Median :1.483
##  Mean   :521.0   D      :  0   Mean   :1.322
##  3rd Qu.:641.2   E      :  0   3rd Qu.:1.680
##  Max.   :761.5   F      :  0   Max.   :1.947
##                  (Other):  0
##      angle            date
##  Min.   :1.829   Min.   :2014-03-21 12:00:00
##  1st Qu.:1.829   1st Qu.:2014-03-21 12:00:00
##  Median :1.829   Median :2014-03-21 12:00:00
##  Mean   :1.829   Mean   :2014-03-21 12:00:00
##  3rd Qu.:1.829   3rd Qu.:2014-03-21 12:00:00
##  Max.   :1.829   Max.   :2014-03-21 12:00:00
##
```

```
tuv.spct
```

```
## Object: source_spct [3,856 x 5]
## containing 8 spectra in long form
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Label: TUV spectral simulation File: data-vignettes/usrout.txt
## Measured between 2014-03-21 12:00:00 and 2014-03-21 19:00:00 UTC
## Time unit 1s
##
## # A tibble: 3,856 x 5
##   w.length spct.idx s.e.irrad angle
##      <dbl>   <fctr>     <dbl> <dbl>
## 1    280.5        A 3.041e-15 1.829
## 2    281.5        A 1.164e-13 1.829
## 3    282.5        A 1.824e-12 1.829
## # ... with 3,853 more rows, and 1 more variables:
## #   date <time>
```

It is possible to extract individual spectra with subset, or as done here plot them in different panels.

```
plot(tuv.spct, annotations = c("colour.guide")) +
  facet_wrap(~date, ncol = 2)
```



The output is a single `source_spct` object that can be easily converted into a `source_mspct` object containing the individual spectra as members of the collection.

```
tuv.mspct <- subset2mspct(tuv.spct)
tuv.mspct
```

```
## Object: source_mspct [8 x 1]
## --- Member: A ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
## # A tibble: 482 x 4
##   w.length s.e.irrad angle               date
## *    <dbl>     <dbl> <dbl>             <time>
## 1    280.5 3.041e-15 1.829 2014-03-21 12:00:00
## 2    281.5 1.164e-13 1.829 2014-03-21 12:00:00
## 3    282.5 1.824e-12 1.829 2014-03-21 12:00:00
## # ... with 479 more rows
## --- Member: B ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
## # A tibble: 482 x 4
##   w.length s.e.irrad  angle               date
## *    <dbl>     <dbl>  <dbl>             <time>
## 1    280.5 1.314e-15 13.198 2014-03-21 13:00:00
## 2    281.5 5.415e-14 13.198 2014-03-21 13:00:00
## 3    282.5 9.039e-13 13.198 2014-03-21 13:00:00
## # ... with 479 more rows
## --- Member: C ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
## # A tibble: 482 x 4
##   w.length s.e.irrad angle               date
## *    <dbl>     <dbl> <dbl>             <time>
## 1    280.5 4.521e-17  28.2 2014-03-21 14:00:00
## 2    281.5 2.510e-15  28.2 2014-03-21 14:00:00
## 3    282.5 5.413e-14  28.2 2014-03-21 14:00:00
## # ... with 479 more rows
## --- Member: D ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
## # A tibble: 482 x 4
##   w.length s.e.irrad  angle               date
## *    <dbl>     <dbl>  <dbl>             <time>
## 1    280.5 3.075e-20 43.202 2014-03-21 15:00:00
## 2    281.5 3.273e-18 43.202 2014-03-21 15:00:00
## 3    282.5 1.234e-16 43.202 2014-03-21 15:00:00
## # ... with 479 more rows
## --- Member: E ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
```

```
## # A tibble: 482 x 4
##   w.length s.e.irrad  angle               date
## *    <dbl>     <dbl>  <dbl>             <time>
## 1    280.5 2.253e-26 58.205 2014-03-21 16:00:00
## 2    281.5 7.751e-24 58.205 2014-03-21 16:00:00
## 3    282.5 8.148e-22 58.205 2014-03-21 16:00:00
## # ... with 479 more rows
## --- Member: F ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
## # A tibble: 482 x 4
##   w.length s.e.irrad  angle               date
## *    <dbl>     <dbl>  <dbl>             <time>
## 1    280.5 1.929e-27 73.208 2014-03-21 17:00:00
## 2    281.5 5.710e-25 73.208 2014-03-21 17:00:00
## 3    282.5 5.202e-23 73.208 2014-03-21 17:00:00
## # ... with 479 more rows
## --- Member: G ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
## # A tibble: 482 x 4
##   w.length s.e.irrad  angle               date
## *    <dbl>     <dbl>  <dbl>             <time>
## 1    280.5 4.721e-28 88.211 2014-03-21 18:00:00
## 2    281.5 1.385e-25 88.211 2014-03-21 18:00:00
## 3    282.5 1.250e-23 88.211 2014-03-21 18:00:00
## # ... with 479 more rows
## --- Member: H ---
## Object: source_spct [482 x 4]
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Time unit 1s
##
## # A tibble: 482 x 4
##   w.length s.e.irrad   angle               date
## *    <dbl>     <dbl>   <dbl>             <time>
## 1    280.5         0 103.213 2014-03-21 19:00:00
## 2    281.5         0 103.213 2014-03-21 19:00:00
## 3    282.5         0 103.213 2014-03-21 19:00:00
## # ... with 479 more rows
##
## --- END ---
```

With the default of `lubridate::today()` for date times are 'mapped' to the current local date using the time zone of the computer as visible to R.

```
tuv_nd.spct <- read_tuv_usrout(file = "data-vignettes/usrout.txt")
tuv_nd.spct

## Object: source_spct [3,856 x 5]
## containing 8 spectra in long form
## Wavelength range 280.5 to 761.5 nm, step 1 nm
## Label: TUV spectral simulation File: data-vignettes/usrout.txt
## Measured between 2016-07-19 12:00:00 and 2016-07-19 19:00:00 UTC
```

```
## Time unit 1s
##
## # A tibble: 3,856 x 5
##   w.length spct.idx s.e.irrad angle
##      <dbl>   <fctr>     <dbl> <dbl>
## 1    280.5        A 3.041e-15 1.829
## 2    281.5        A 1.164e-13 1.829
## 3    282.5        A 1.824e-12 1.829
## # ... with 3,853 more rows, and 1 more variables:
## #   date <time>
```

3.2 libRadtran

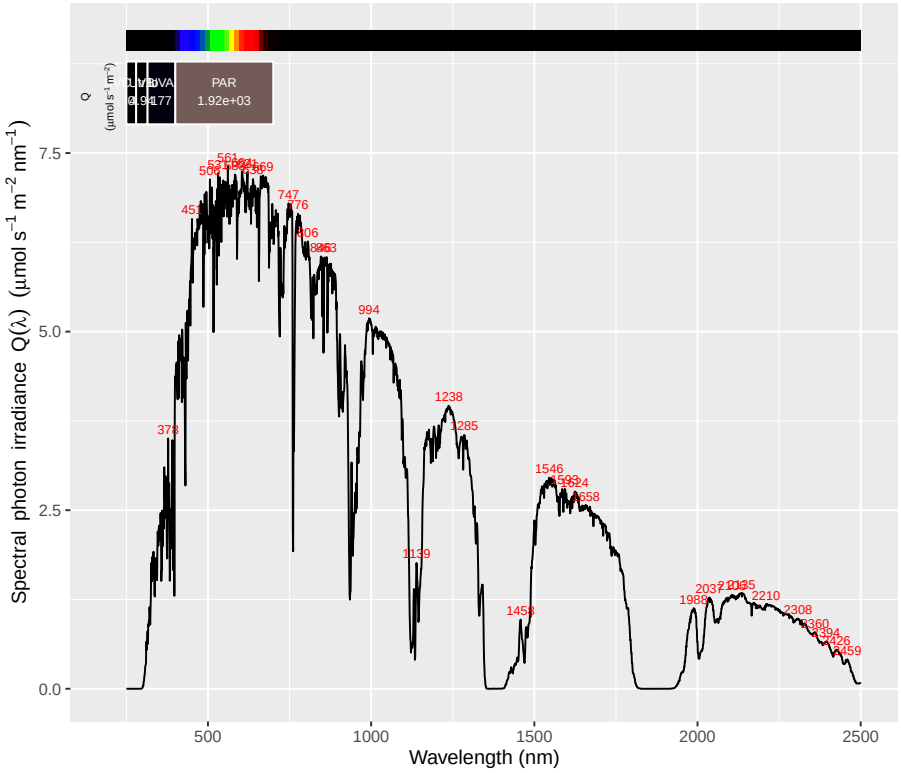
By default 'libRadtran' writes only spectral irradiances to a text file as output. This is different from 'TUV' which by default includes an extensive header with the parameter settings used for the simulation.

For reading this simple output, no special function is needed. We can use either `read.table` from base R. Here we read a file with two columns with wavelengths and global spectral energy irradiance (named "eglo" in libRadtran) in $\mathrm{mW\,m^{-2}\,nm^{-1}}$. The file was created with one of the 'uvspec' examples included with libRadtran, but reducing the output to two columns.

The first few lines of the file look like this:

```
  250.000  0.000000e+00
  251.000  0.000000e+00
  252.000  0.000000e+00
  253.000  0.000000e+00
...
```

```
lrt.df <- read.table(file = "data-vignettes/libradtran-plain-2col.dat",
                     col.names = c("w.length", "s.e.irrad"))
summary(lrt.df)

##     w.length      s.e.irrad
##  Min.   : 250   Min.   :   0.000
##  1st Qu.:1188   1st Qu.:   3.808
##  Median :2125   Median :  23.999
##  Mean   :2125   Mean   : 244.546
##  3rd Qu.:3062   3rd Qu.: 264.351
##  Max.   :4000   Max.   :1744.596

libradtran.spct <- source_spct(w.length = lrt.df$w.length,
                               s.e.irrad = lrt.df$s.e.irrad * 1e-3)
plot(libradtran.spct, range = c(250, 2500), unit.out = "photon")
```



We give two additional examples, which will most likely need some adjustment by users, as these are for output from libRadtran post-processed to add additional information. These are included in the package because myself and collaborators use these formats heavily. In fact users could develop shell scripts or Perl scripts using the same output format.

3.3 Output enriched with time and date data

In this case the file to be read is similar as above, but including separate columns for direct and diffuse components of the spectral energy irradiance. In addition two columns, one with date strings in ISO format and one with times have been added. The file instead of containing a single spectrum, contains several spectra in long form.

The first few lines of the file look like this:

```
290.000 2015-05-19 11_00_00 0.000000e+00 0.000000e+00
291.000 2015-05-19 11_00_00 0.000000e+00 0.000000e+00
292.000 2015-05-19 11_00_00 0.000000e+00 0.000000e+00
293.000 2015-05-19 11_00_00 1.893645e-05 3.439497e-05
294.000 2015-05-19 11_00_00 1.648530e-04 2.764368e-04
...
```

A function is included for reading data saved in a text file in this format. It also automatically converts $\mathrm{mW\,m^{-2}\,nm^{-1}}$ into $\mathrm{W\,m^{-2}\,nm^{-1}}$.

```
lbr.multi.spct <- read_libradtran_vesa("data-vignettes/libradtran-multi.dat")
print(lbr.multi.spct, n = 5)

## Object: source_spct [3,055 x 5]
## containing 5 spectra in long form
## Wavelength range 290 to 900 nm, step 1 nm
## Label: libRadtran spectral simulation File: data-vignettes/libradtran-multi.dat
## Measured between 2015-05-19 11:00:00 and 2015-05-19 11:04:00 UTC
## Time unit 1s
##
## # A tibble: 3,055 x 5
##   w.length            datetime s.e.irrad.dir
##      <dbl>              <time>         <dbl>
## 1      290 2015-05-19 11:00:00  0.000000e+00
## 2      291 2015-05-19 11:00:00  0.000000e+00
## 3      292 2015-05-19 11:00:00  0.000000e+00
## 4      293 2015-05-19 11:00:00  1.893645e-08
## 5      294 2015-05-19 11:00:00  1.648530e-07
## # ... with 3,050 more rows, and 2 more variables:
## #   s.e.irrad.diff <dbl>, s.e.irrad <dbl>
```

3.4 Scripts developed by Anders Lindfors

Functions `read_fmi_cum` and `read_m_fmi_cum` can be used to read text files output by a simulation model of solar spectral irradiance. The model was developed at the Finnish Meteorological Institute (FMI) by Dr. Anders Lindfors and collaborators and uses functions from 'libRadtran' as its engine, but saves some additional metadata to the output file.

The first few lines of the file look like this:

```
# date number_of_scans start_scan stop_scan max_time_gap max_sza_gap warnings
# 20140821 15 3:30:00 17:30:00 60 7.4
# wavelength exposure(J/m2/nm)
2900 0.00000000e+00
2910 2.93132235e-05
2920 7.23526379e-04
...
```

We can read an individual file into a `source_spct` object while adding some metadata read from the file header. In this case values are for daily global spectral energy exposures rather than irradiances. Wavelengths are expressed in Angstroms instead of nanometres.

```
z.spct <- read_fmi_cum("data-vignettes/2014-08-21_cum.hel")
class_spct(z.spct)

## [1] "source_spct"  "generic_spct"

getWhenMeasured(z.spct)

## [1] "2014-08-21 UTC"

z.spct
```

```
## Object: source_spct [511 x 2]
## Wavelength range 290 to 800 nm, step 1 nm
## Label: File: data-vignettes/2014-08-21_cum.hel
## Measured on 2014-08-21 UTC
## Time unit 86400s (~1 days)
##
## # A tibble: 511 x 2
##   w.length    s.e.irrad
##      <dbl>        <dbl>
## 1      290 0.000000e+00
## 2      291 2.931322e-05
## 3      292 7.235264e-04
## # ... with 508 more rows
```

With function `read_m_fmi_cum` with an "m" in the name we can read several files each containing a single spectrum. The returned object is a collection of source spectra.

```
z.mspct <- read_m_fmi_cum(c("data-vignettes/2014-08-21_cum.hel",
                            "data-vignettes/2014-08-22_cum.hel"))
class(z.mspct)

## [1] "source_mspct"  "generic_mspct" "list"

getWhenMeasured(z.mspct)

## # A tibble: 2 x 2
##              spct.idx when.measured
##                 <fctr>        <time>
## 1 2014_08_21_cum.hel    2014-08-21
## 2 2014_08_22_cum.hel    2014-08-22

z.mspct

## Object: source_mspct [2 x 1]
## --- Member: 2014_08_21_cum.hel ---
## Object: source_spct [511 x 2]
## Wavelength range 290 to 800 nm, step 1 nm
## Label: File: data-vignettes/2014-08-21_cum.hel
## Measured on 2014-08-21 UTC
## Time unit 86400s (~1 days)
##
## # A tibble: 511 x 2
##   w.length    s.e.irrad
##      <dbl>        <dbl>
## 1      290 0.000000e+00
## 2      291 2.931322e-05
## 3      292 7.235264e-04
## # ... with 508 more rows
## --- Member: 2014_08_22_cum.hel ---
## Object: source_spct [511 x 2]
## Wavelength range 290 to 800 nm, step 1 nm
## Label: File: data-vignettes/2014-08-22_cum.hel
## Measured on 2014-08-22 UTC
## Time unit 86400s (~1 days)
##
```

```
## # A tibble: 511 x 2
##   w.length    s.e.irrad
##      <dbl>        <dbl>
## 1      290 0.000000e+00
## 2      291 2.926634e-05
## 3      292 7.213022e-04
## # ... with 508 more rows
##
## --- END ---
```

Above we gave the names of the files explicitly, but as we show here, one can build on-the-fly a list of file names matching some pattern.

```
files <- system("ls ./data-vignettes/*cum.hel", intern = TRUE)
z1.mspct <- read_m_fmi_cum(files)
class(z1.mspct)

## [1] "source_mspct"  "generic_mspct" "list"

getWhenMeasured(z1.mspct)

## # A tibble: 2 x 2
##              spct.idx when.measured
##                 <fctr>        <time>
## 1 2014_08_21_cum.hel    2014-08-21
## 2 2014_08_22_cum.hel    2014-08-22

z1.mspct

## Object: source_mspct [2 x 1]
## --- Member: 2014_08_21_cum.hel ---
## Object: source_spct [511 x 2]
## Wavelength range 290 to 800 nm, step 1 nm
## Label: File: ./data-vignettes/2014-08-21_cum.hel
## Measured on 2014-08-21 UTC
## Time unit 86400s (~1 days)
##
## # A tibble: 511 x 2
##   w.length    s.e.irrad
##      <dbl>        <dbl>
## 1      290 0.000000e+00
## 2      291 2.931322e-05
## 3      292 7.235264e-04
## # ... with 508 more rows
## --- Member: 2014_08_22_cum.hel ---
## Object: source_spct [511 x 2]
## Wavelength range 290 to 800 nm, step 1 nm
## Label: File: ./data-vignettes/2014-08-22_cum.hel
## Measured on 2014-08-22 UTC
## Time unit 86400s (~1 days)
##
## # A tibble: 511 x 2
##   w.length    s.e.irrad
##      <dbl>        <dbl>
## 1      290 0.000000e+00
## 2      291 2.926634e-05
```

```
## 3      292 7.213022e-04
## # ... with 508 more rows
##
## --- END ---
```

One also add a geocode at the time of import (or later).

```
z2.mspct <-
  read_m_fmi_cum(files,
                 geocode = geocode("Kumpula, Helsinki, Finland",
                                   source = "google"))
class(z2.mspct)

## [1] "source_mspct"  "generic_mspct" "list"

getWhenMeasured(z2.mspct)

## # A tibble: 2 x 2
##             spct.idx when.measured
##               <fctr>        <time>
## 1 2014_08_21_cum.hel    2014-08-21
## 2 2014_08_22_cum.hel    2014-08-22

getWhereMeasured(z2.mspct)

## # A tibble: 2 x 3
##             spct.idx      lon      lat
##               <fctr>    <dbl>    <dbl>
## 1 2014_08_21_cum.hel 24.96474 60.20911
## 2 2014_08_22_cum.hel 24.96474 60.20911

z2.mspct

## Object: source_mspct [2 x 1]
## --- Member: 2014_08_21_cum.hel ---
## Object: source_spct [511 x 2]
## Wavelength range 290 to 800 nm, step 1 nm
## Label: File: ./data-vignettes/2014-08-21_cum.hel
## Measured on 2014-08-21 UTC
## Measured at 60.20911 N, 24.96474 E
## Time unit 86400s (~1 days)
##
## # A tibble: 511 x 2
##   w.length    s.e.irrad
##      <dbl>        <dbl>
## 1      290 0.000000e+00
## 2      291 2.931322e-05
## 3      292 7.235264e-04
## # ... with 508 more rows
## --- Member: 2014_08_22_cum.hel ---
## Object: source_spct [511 x 2]
## Wavelength range 290 to 800 nm, step 1 nm
## Label: File: ./data-vignettes/2014-08-22_cum.hel
## Measured on 2014-08-22 UTC
## Measured at 60.20911 N, 24.96474 E
## Time unit 86400s (~1 days)
##
```

```
## # A tibble: 511 x 2
##   w.length    s.e.irrad
##      <dbl>        <dbl>
## 1      290 0.000000e+00
## 2      291 2.926634e-05
## 3      292 7.213022e-04
## # ... with 508 more rows
##
## --- END ---
```

4 Other R packages

A general way of exchanging data with other R packages or for use with base R functions is to create a matrix from a collection of spectra, or a collection of spectra from a matrix. However, a matrix is only guaranteed to contain numeric data and a `"dim"` attribute.

4.1 From R matrix

Spectral data can be stored in a matrix either by row or by column, and the user must supply this information explicitly. We also assume, that wavelengths are not stored as part of the matrix, and so the user needs to supply these values as a separate vector.

We here use artificial data, in this first example with spectra saved by column.

```
x <- matrix(1:100, ncol = 2)
wl <- 501:550 # in nanometres
z <- mat2mspct(x, wl, "filter_spct", "Tpc")
z

## Object: filter_mspct [1 x 2]
## --- Member: spct_1 ---
## Object: filter_spct [50 x 2]
## Wavelength range 501 to 550 nm, step 1 nm
##
## # A tibble: 50 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      501  0.01
## 2      502  0.02
## 3      503  0.03
## # ... with 47 more rows
## --- Member: spct_2 ---
## Object: filter_spct [50 x 2]
## Wavelength range 501 to 550 nm, step 1 nm
##
## # A tibble: 50 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      501  0.51
```

```
## 2      502  0.52
## 3      503  0.53
## # ... with 47 more rows
##
## --- END ---
```

In this second example we supply explicit names for the spectra.

```
z <- mat2mspct(x, wl, "filter_spct", "Tpc", spct.names = c("A", "B"))
z

## Object: filter_mspct [1 x 2]
## --- Member: A ---
## Object: filter_spct [50 x 2]
## Wavelength range 501 to 550 nm, step 1 nm
##
## # A tibble: 50 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      501  0.01
## 2      502  0.02
## 3      503  0.03
## # ... with 47 more rows
## --- Member: B ---
## Object: filter_spct [50 x 2]
## Wavelength range 501 to 550 nm, step 1 nm
##
## # A tibble: 50 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      501  0.51
## 2      502  0.52
## 3      503  0.53
## # ... with 47 more rows
##
## --- END ---
```

There is no change in the call for data stored by row. To demonstrate this, we create a new matrix, with the same data as above, but stored by row, as some other packages do.

```
xrow <- matrix(1:100, nrow = 2, byrow = TRUE)
z1 <- mat2mspct(xrow, wl, "filter_spct", "Tpc")
z1

## Object: filter_mspct [1 x 2]
## --- Member: spct_1 ---
## Object: filter_spct [50 x 2]
## Wavelength range 501 to 550 nm, step 1 nm
##
## # A tibble: 50 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      501  0.01
## 2      502  0.02
## 3      503  0.03
```

```
## # ... with 47 more rows
## --- Member: spct_2 ---
## Object: filter_spct [50 x 2]
## Wavelength range 501 to 550 nm, step 1 nm
##
## # A tibble: 50 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      501  0.51
## 2      502  0.52
## 3      503  0.53
## # ... with 47 more rows
##
## --- END ---
```

There is only one case when an explict argument for `byrow` is needed: square matrices (same number of spectra as of wavelength values in each spectrum).

4.2 To R matrix

In this case, the metadata contained in the individual spectral objects is discarded, and only the `"comment"` attribute of the collection of spectra copied to the returned object. The wavelength values are preserved in an attribute named "w.length", but are not included as part of the matrix.

```
z2c.mat <- mspct2mat(z2.mspct, "s.e.irrad")
class(z2c.mat)

## [1] "matrix"

dim(z2c.mat)

## [1] 511   2

head(dimnames(z2c.mat)$spct)

## [1] "2014_08_21_cum.hel" "2014_08_22_cum.hel"

head(dimnames(z2c.mat)$w.length)

## [1] "290" "291" "292" "293" "294" "295"

head(attr(z2c.mat, "w.length"))

## [1] 290 291 292 293 294 295
```

Collections of spectra have a `spct.byrow` attribute but it is discarded as it describes dimensions would require spectral data to be stored in a three dimensional array and cannot be mapped to a matrix. The argument byrow in the conversion function has the same meaning as in the `matrix` constructor function.

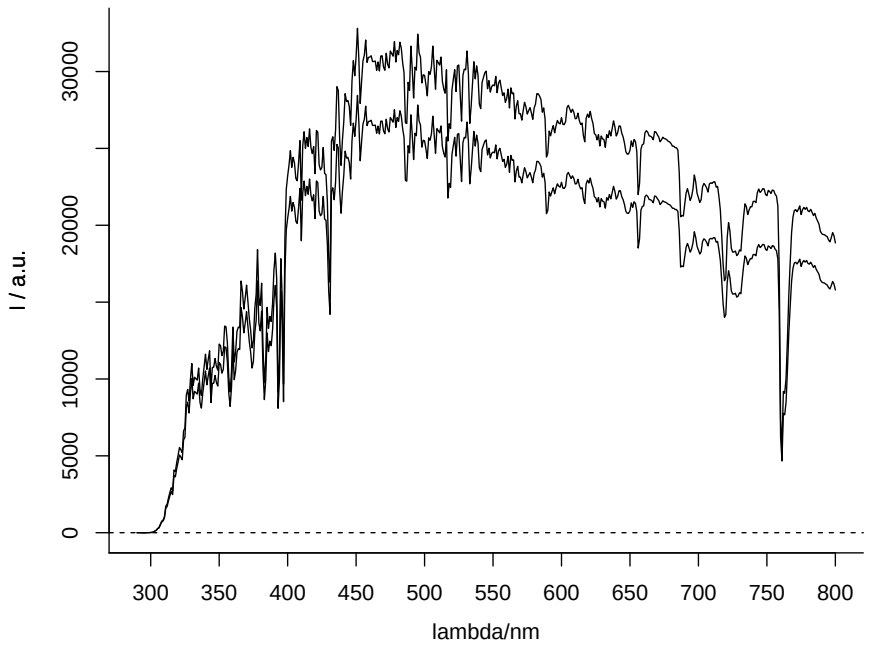
```
z2r.mat <- mspct2mat(z2.mspct, "s.e.irrad", byrow = TRUE)
class(z2r.mat)

## [1] "matrix"

dim(z2r.mat)

## [1]   2 511

head(dimnames(z2r.mat)$spct)

## [1] "2014_08_21_cum.hel" "2014_08_22_cum.hel"

head(dimnames(z2r.mat)$w.length)

## [1] "290" "291" "292" "293" "294" "295"

head(attr(z2r.mat, "w.length"))

## [1] 290 291 292 293 294 295
```

These functions are not fully automatic, the user needs to provide the name of the variable to extract from each spectrum. If the wavelength values are not consistent among spectra, only those with the same values as the first spectrum in the collection are retained and the remaining ones dropped with a warning.

4.3 To 'hyperSpec'

Can export to "hyperSpec" objects only collections of spectra where all members have identical `w.length` vectors, as objects of class `hyperSpec` store a single vector of wavelengths for the whole collection of spectra.

```
z2.hspct <- mspct2hyperSpec(z2.mspct, "s.e.irrad")

## Warning in .local(.Object, ...):  Spectra in data are overwritten by argument spc.

class(z2.hspct)

## [1] "hyperSpec"
## attr(,"package")
## [1] "hyperSpec"

plot(z2.hspct)
```

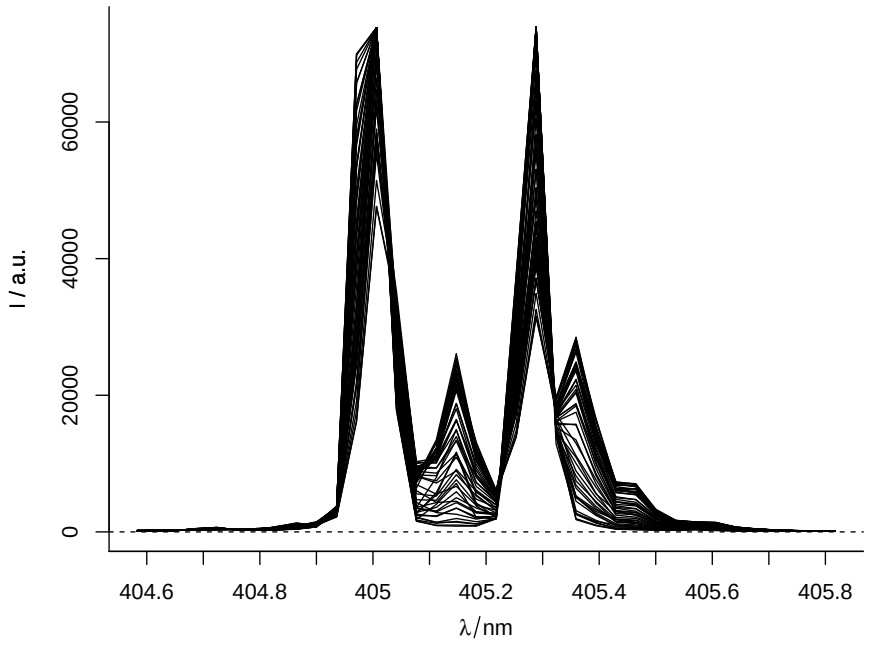


4.4 From 'hyperSpec'

Can import only data with wavelength in nanometres. Other quantities and units are not supported by the 'photobiology' classes for spectral data. See package 'hyperSpec' vignette "laser" for details on the data and the conversion of the original wavelength units into nanometres.

```
class(laser)

## [1] "hyperSpec"
## attr(,"package")
## [1] "hyperSpec"

laser

## hyperSpec object
##    84 spectra
##    2 data columns
##    36 data points / spectrum
## wavelength: lambda/nm [numeric] 404.5828 404.6181 ... 405.8176
## data:  (84 rows x 2 columns)
##    1. t: t / s [numeric] 0 2 ... 5722
##    2. spc: I / a.u. [matrix36] 164.650 179.724 ... 112.086
```

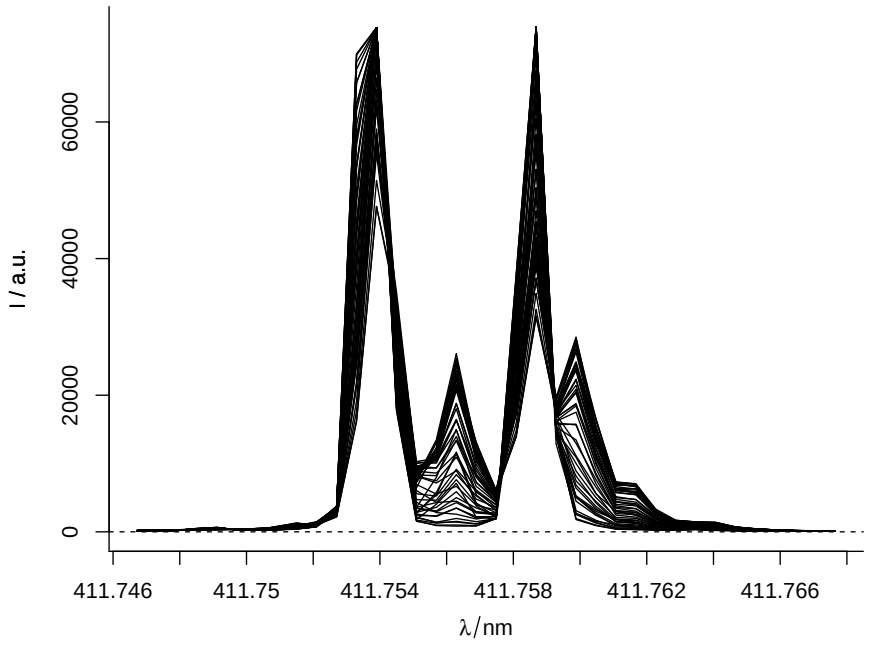
```
plot(laser)

## Warning in plotspc(x, ...):  Number of spectra exceeds spc.nmax.  Only the first
50 are plotted.
```

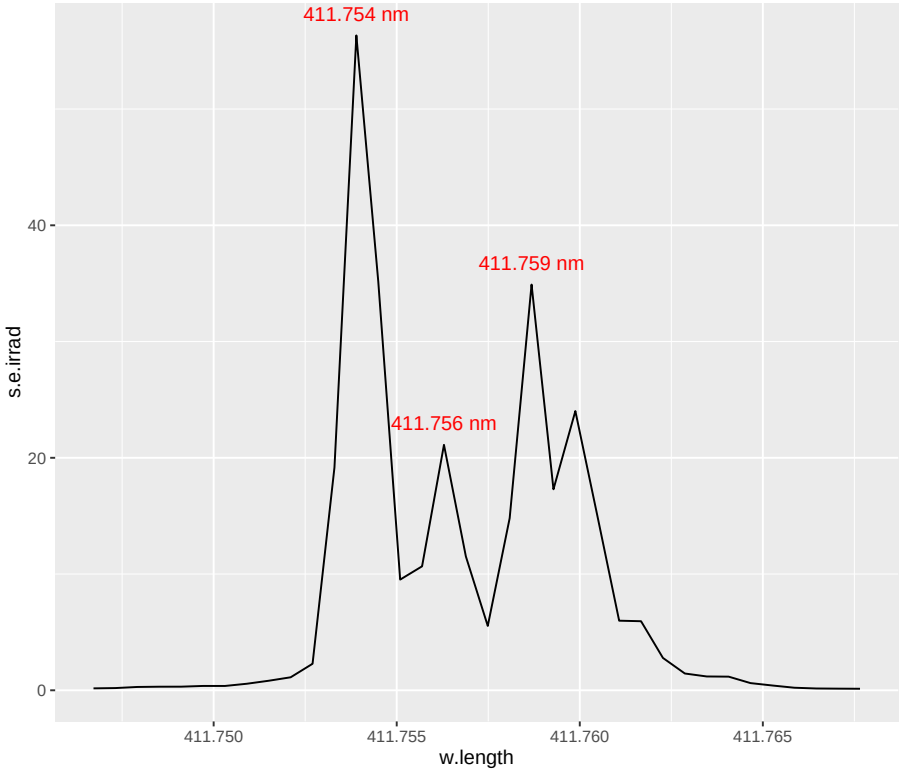


We assume here, that the quantity for the spectral emission of the laser is spectral *energy* irradiance, expressed in $\mathrm{mW\,m^{-2}\,nm^{-1}}$. This is likely to be wrong but for the sake of showing how the conversion takes place is irrelevant. The parameter `multiplier` can be passed a numeric argument to rescale the original data. The default multiplier is 1.

```
wl(laser) <- list (
  wl = 1e7 / (1/405e-7 - wl (laser)),
  label = expression (lambda / nm)
)
laser

## hyperSpec object
##    84 spectra
##    2 data columns
##    36 data points / spectrum
## wavelength: lambda/nm [numeric] 411.7467 411.7473 ... 411.7677
## data:  (84 rows x 2 columns)
##    1. t: t / s [numeric] 0 2 ... 5722
##    2. spc: I / a.u. [matrix36] 164.650 179.724 ... 112.086
```

```r
plot(laser)

## Warning in plotspc(x, ...):  Number of spectra exceeds spc.nmax.  Only the first
50 are plotted.

laser.mspct <-
  hyperSpec2mspct(laser, "source_spct", "s.e.irrad", multiplier = 1e-3)
ggplot(laser.mspct[[1]]) +
  geom_line() +
  stat_peaks(geom = "text", vjust = -1, label.fmt = "%.6g nm", color = "red")
```





4.5 From 'colorSpec'

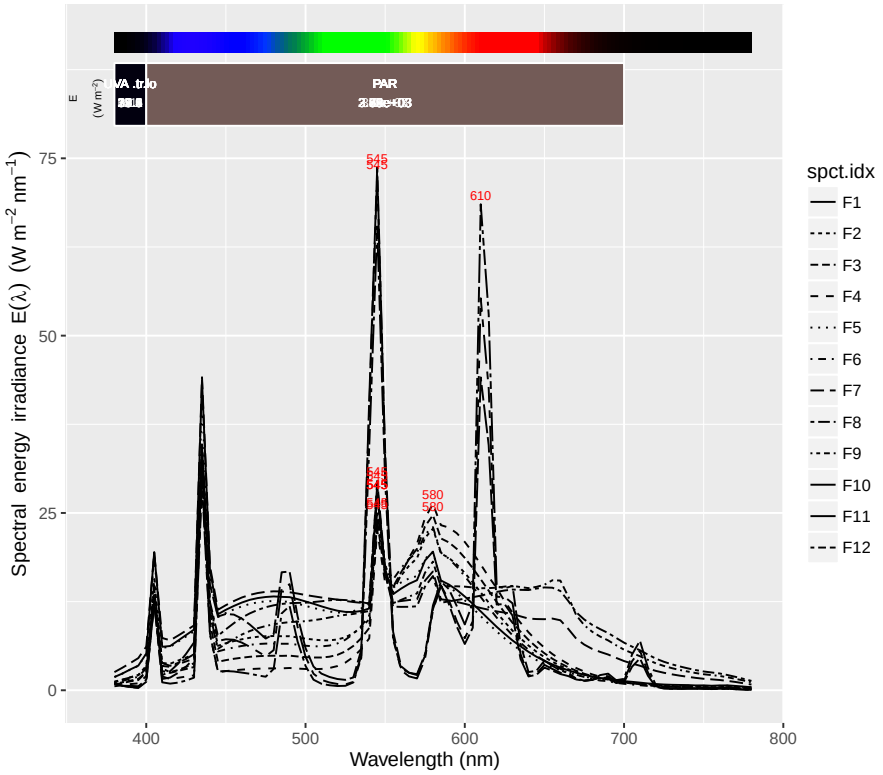
```
fluorescent.mspct <- colorSpec2mspct(Fs.5nm)
print(fluorescent.mspct, n = 3, n.members = 3)

## Object: source_mspct [12 x 1]
## --- Member: F1 ---
## Object: source_spct [81 x 2]
## Wavelength range 380 to 780 nm, step 5 nm
## Time unit 1s
##
## # A tibble: 81 x 2
##   w.length s.e.irrad
##      <dbl>     <dbl>
## 1      380      1.87
## 2      385      2.36
## 3      390      2.94
## # ... with 78 more rows
## --- Member: F2 ---
## Object: source_spct [81 x 2]
## Wavelength range 380 to 780 nm, step 5 nm
## Time unit 1s
##
## # A tibble: 81 x 2
```

```
##   w.length s.e.irrad
##      <dbl>     <dbl>
## 1      380      1.18
## 2      385      1.48
## 3      390      1.84
## # ... with 78 more rows
## --- Member: F3 ---
## Object: source_spct [81 x 2]
## Wavelength range 380 to 780 nm, step 5 nm
## Time unit 1s
##
## # A tibble: 81 x 2
##   w.length s.e.irrad
##      <dbl>     <dbl>
## 1      380      0.82
## 2      385      1.02
## 3      390      1.26
## # ... with 78 more rows
## ......................
## 9 other member spectra not shown
##
## --- END ---
```

```
colorSpec2mspct(Hoya)

## Object: filter_mspct [4 x 1]
## --- Member: R-60 ---
## Object: filter_spct [46 x 2]
## Wavelength range 300 to 750 nm, step 10 nm
##
## # A tibble: 46 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      300     0
## 2      310     0
## 3      320     0
## # ... with 43 more rows
## --- Member: G-533 ---
## Object: filter_spct [46 x 2]
## Wavelength range 300 to 750 nm, step 10 nm
##
## # A tibble: 46 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      300     0
## 2      310     0
## 3      320     0
## # ... with 43 more rows
## --- Member: B-440 ---
## Object: filter_spct [46 x 2]
## Wavelength range 300 to 750 nm, step 10 nm
##
## # A tibble: 46 x 2
##   w.length   Tfr
##      <dbl> <dbl>
## 1      300     0
```

```
## 2      310     0
## 3      320     0
## # ... with 43 more rows
## --- Member: LB-120 ---
## Object: filter_spct [46 x 2]
## Wavelength range 300 to 750 nm, step 10 nm
##
## # A tibble: 46 x 2
##   w.length     Tfr
##      <dbl>   <dbl>
## 1      300 0.00003
## 2      310 0.00580
## 3      320 0.08100
## # ... with 43 more rows
##
## --- END ---
```

```
fluorescent.spct <- colorSpec2spct(Fs.5nm)
plot(fluorescent.spct) + aes(linetype = spct.idx)
```

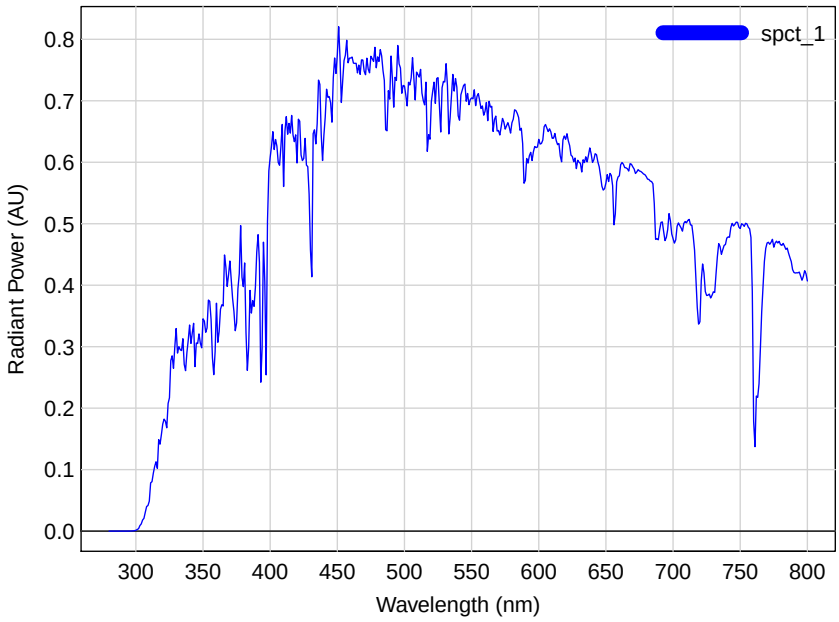



```
colorSpec2chroma_spct(xyz1931.5nm)

## Object: chroma_spct [81 x 4]
```

```
## Wavelength range 380 to 780 nm, step 5 nm
##
## # A tibble: 81 x 4
##        x     y      z w.length
## *  <dbl> <dbl>  <dbl>    <dbl>
## 1 0.0014 0e+00 0.0065      380
## 2 0.0022 1e-04 0.0105      385
## 3 0.0042 1e-04 0.0201      390
## # ... with 78 more rows
```

4.6 To 'colorSpec'

```
sun.cspec <- spct2colorSpec(sun.spct)
plot(sun.cspec, col = "blue")
```




```
spct2colorSpec(yellow_gel.spct)

##
## colorSpec object.   The organization is 'vector'.  Object size is 11488 bytes.
## the object describes 1 transparent materials, and the quantity is 'transmittance'.
## Wavelength range: 190 to 800 nm.  Step size is 1 nm.
##
```

```
## 1 spectra
## 611 data points / spectrum
##
##   Material   Min    Max LambdaMax Integral
## 1   spct_1 1e-05 0.9025     768.5 260.3194
```

```
chroma_spct2colorSpec(beesxyzCMF.spct)

##
## colorSpec object.   The organization is 'matrix'.  Object size is 4128 bytes.
## the object describes a responder to light with 3 output channels, and the quantity is 'power->neural'.
## Wavelength range: 300 to 700 nm.  Step size is 5 nm.
##
## 3 spectra
## 81 data points / spectrum
##
##   Channel   Min Max LambdaMax E.response
## 1       x 0.000   1       340     68.965
## 2       y 0.000   1       435    103.850
## 3       z 0.006   1       560    135.620
```

4.7 From 'pavo'

In this example we convert an `rspec` object from package 'pavo' into a collection of spectra and then we plot it with `ggplot` methods from package 'ggspectra' (an extension to 'ggplot2'). The data are the spectral reflectance of the plumage from seven different individual birds of the same species, measured in three different body parts.

```
data(sicalis)
class(sicalis)

## [1] "rspec"      "data.frame"

names(sicalis)

##  [1] "wl"     "ind1.C" "ind1.T" "ind1.B" "ind2.C"
##  [6] "ind2.T" "ind2.B" "ind3.C" "ind3.T" "ind3.B"
## [11] "ind4.C" "ind4.T" "ind4.B" "ind5.C" "ind5.T"
## [16] "ind5.B" "ind6.C" "ind6.T" "ind6.B" "ind7.C"
## [21] "ind7.T" "ind7.B"
```

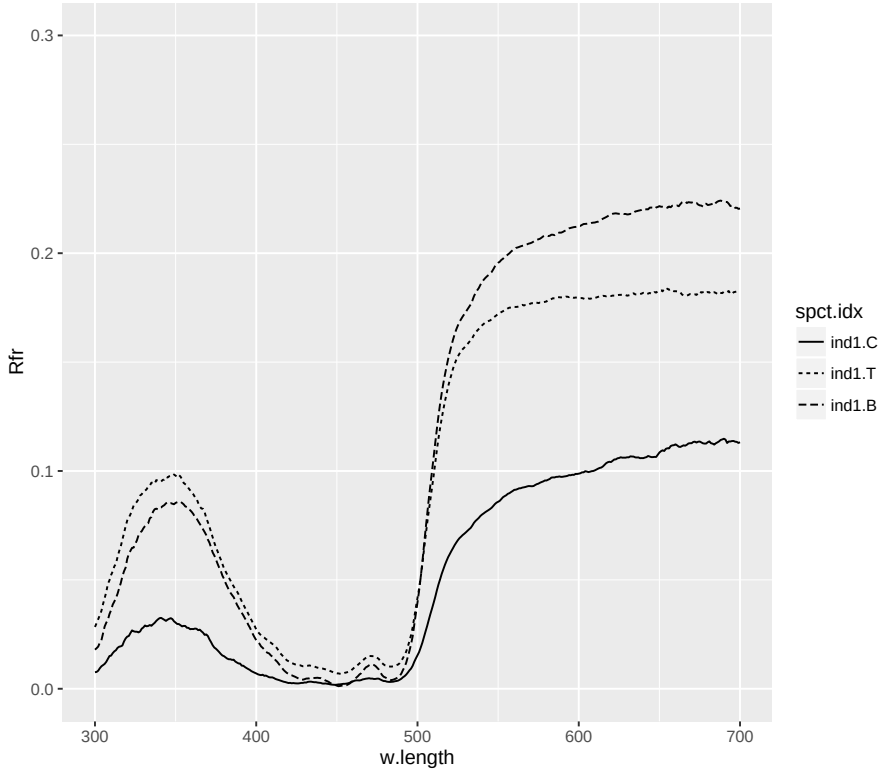
We convert the data into a collection of spectra, and calculate summaries for three spectra.

```
sicalis.mspct <- rspec2mspct(sicalis, "reflector_spct", "Rpc")
summary(sicalis.mspct[[1]])

## Summary of object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
##     w.length        Rfr
```

```
##  Min.   :300   Min.   :0.001798
##  1st Qu.:400   1st Qu.:0.008288
##  Median :500   Median :0.031709
##  Mean   :500   Mean   :0.052848
##  3rd Qu.:600   3rd Qu.:0.098775
##  Max.   :700   Max.   :0.114807

summary(sicalis.mspct[[2]])

## Summary of object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
##     w.length        Rfr
##  Min.   :300   Min.   :0.006783
##  1st Qu.:400   1st Qu.:0.030112
##  Median :500   Median :0.096994
##  Mean   :500   Mean   :0.105449
##  3rd Qu.:600   3rd Qu.:0.179691
##  Max.   :700   Max.   :0.183823

summary(sicalis.mspct[[3]])

## Summary of object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
##     w.length        Rfr
##  Min.   :300   Min.   :0.001191
##  1st Qu.:400   1st Qu.:0.022293
##  Median :500   Median :0.085235
##  Mean   :500   Mean   :0.116253
##  3rd Qu.:600   3rd Qu.:0.212554
##  Max.   :700   Max.   :0.224162
```

We convert the subset of the collection corresponding to the first individual into a single spectra object for plotting with `ggplot`.

```
ggplot(rbindspct(sicalis.mspct[1:3])) +
  aes(linetype = spct.idx) +
  ylim(0,0.3) +
  geom_line()
```

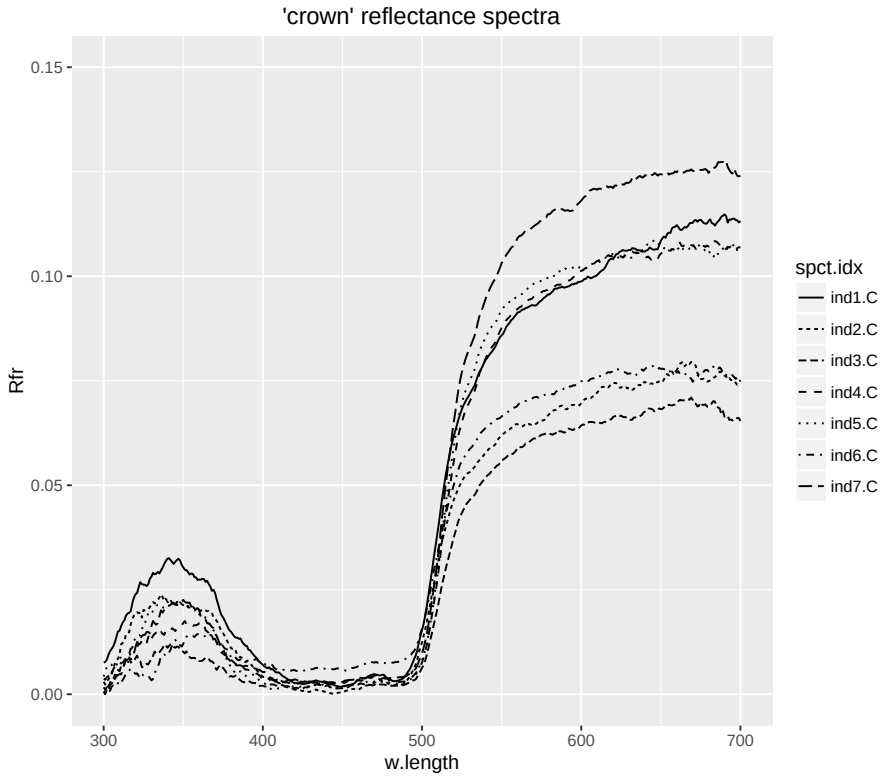


Here we extract the "crown" data from all individuals and plot these spectra in a single plot.

```
print(sicalis.mspct[c(TRUE, FALSE, FALSE)])

## Object: reflector_mspct [7 x 1]
## --- Member: ind1.C ---
## Object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
## # A tibble: 401 x 2
##   w.length         Rfr
##      <dbl>       <dbl>
## 1      300 0.007594984
## 2      301 0.007727841
## 3      302 0.008288000
## # ... with 398 more rows
## --- Member: ind2.C ---
## Object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
## # A tibble: 401 x 2
##   w.length         Rfr
##      <dbl>       <dbl>
## 1      300 0.002971167
## 2      301 0.002326722
```

```
## 3      302 0.003227833
## # ... with 398 more rows
## --- Member: ind3.C ---
## Object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
## # A tibble: 401 x 2
##   w.length          Rfr
##      <dbl>        <dbl>
## 1      300 0.0005947619
## 2      301 0.0000000000
## 3      302 0.0011853968
## # ... with 398 more rows
## --- Member: ind4.C ---
## Object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
## # A tibble: 401 x 2
##   w.length         Rfr
##      <dbl>       <dbl>
## 1      300 0.003750159
## 2      301 0.003465714
## 3      302 0.004133333
## # ... with 398 more rows
## --- Member: ind5.C ---
## Object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
## # A tibble: 401 x 2
##   w.length         Rfr
##      <dbl>       <dbl>
## 1      300 0.004225349
## 2      301 0.005361222
## 3      302 0.006554556
## # ... with 398 more rows
## --- Member: ind6.C ---
## Object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
## # A tibble: 401 x 2
##   w.length          Rfr
##      <dbl>        <dbl>
## 1      300 0.0006326984
## 2      301 0.0006136508
## 3      302 0.0001934921
## # ... with 398 more rows
## --- Member: ind7.C ---
## Object: reflector_spct [401 x 2]
## Wavelength range 300 to 700 nm, step 1 nm
##
## # A tibble: 401 x 2
##   w.length         Rfr
##      <dbl>       <dbl>
## 1      300 0.001680730
## 2      301 0.001043270
## 3      302 0.001702476
## # ... with 398 more rows
```

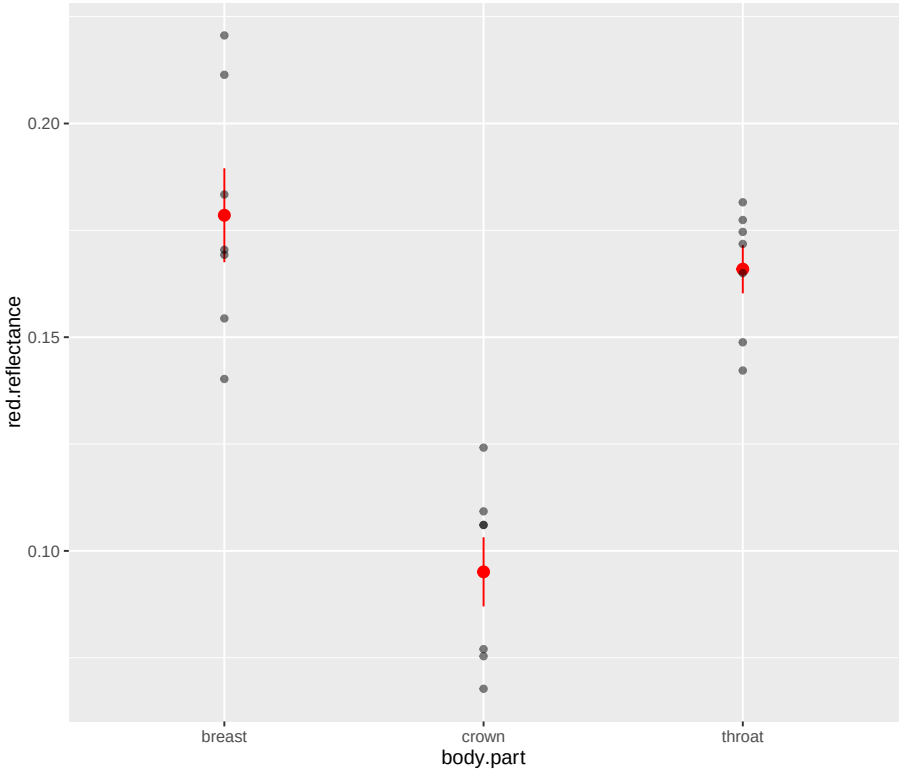
```
##
## --- END ---

ggplot(rbindspct(sicalis.mspct[c(TRUE, FALSE, FALSE)])) +
  aes(linetype = spct.idx) +
  ylim(0,0.15) +
  geom_line() +
  ggtitle("'crown' reflectance spectra")
```



We calculate the mean reflectance in wavebands corresponding to ISO colors obtaining a data frame. We then add to this returned data frame a factor indicating the body parts.

```
refl.by.band <- reflectance(sicalis.mspct, w.band = list(Red(), Green(), Blue(), UVA()))
refl.by.band$body.part <- c("crown", "throat", "breast")
```

```
refl.red <- reflectance(sicalis.mspct, w.band = Red())
names(refl.red)[2] <- "red.reflectance"
refl.red$body.part <- c("crown", "throat", "breast")
ggplot(refl.red, aes(x = body.part, y = red.reflectance)) +
  stat_summary(fun.data = "mean_se", color = "red") +
  geom_point(alpha = 0.5)
```



5 Dealing with odd and bad data

5.1 Using locales

Most functions in this package have a parameter `locale`, that accepts `readr::locale` objects as arguments. At the moment only the time zone and decimal mark are respected. This allows files using comma for decimal marker be easily imported, or the dates and times **in the input file** be interpreted in a given time zone. Setting the correct time zone is very important to avoid errors. Time coordinates are always stored in the created objects using universal time coordinates ("UTC").

```
my.locale <- locale(decimal_mark = ",", tz = "EET")
read_oo_jazirrad(file = "data-vignettes/spectrum-comma.JazIrrad",
                 locale = my.locale)

## Warning in range_check(x, strict.range = strict.range):  Negative spectral energy
irradiance values; minimun s.e.irrad = -0.032

## Object: source_spct [2,048 x 2]
## Wavelength range 188.82523 to 1033.1483 nm, step 0.357056 to 0.459625 nm
## Label: File: data-vignettes/spectrum-comma.JazIrrad
```

```
## Measured on 2015-02-03 07:44:41 UTC
## Time unit 1s
##
## # A tibble: 2,048 x 2
##   w.length s.e.irrad
##      <dbl>     <dbl>
## 1 188.8252         0
## 2 189.2849         0
## 3 189.7444         0
## # ... with 2,045 more rows
```

5.2 Overriding default metadata

We revisit now the Jaz irradiance data to show how the metadata can be changed by the user if needed (e.g. clock settings at the time of data acquisition were wrong).

A variable with the user supplied date and time data, or the date read from the header (the text itself) not the file date as the file date may not reflect the creation date and time.

```
jaz01.spct <- read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad",
                               date = NULL)
getWhenMeasured(jaz01.spct)

## [1] "2015-02-03 09:44:41 UTC"
```

```
jaz02.spct <- read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad",
                               date = ymd_hms("2015-11-15 12:00:00"))
getWhenMeasured(jaz02.spct)

## [1] "2015-11-15 12:00:00 UTC"
```

```
jaz03.spct <- read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad",
                               date = now())
getWhenMeasured(jaz03.spct)

## [1] "2016-07-19 18:49:34 UTC"
```

5.3 Adding additional metadata

When can add a geocode, either directly by giving latitude and longitude coordinates or by generating it from a Google maps search using function `ggmap::geocode()` as shown here.

```
jaz04.spct <- read_oo_jazirrad(file = "data-vignettes/spectrum.JazIrrad",
                               geocode = geocode("Vikki, 00790 Helsinki, Finland",
                                                 source = "google"))
jaz04.spct
```

```
## Object: source_spct [2,048 x 2]
## Wavelength range 188.82523 to 1033.1483 nm, step 0.357056 to 0.459625 nm
## Label: File: data-vignettes/spectrum.JazIrrad
## Measured on 2015-02-03 09:44:41 UTC
## Measured at 60.22515 N, 25.02006 E
## Time unit 1s
##
## # A tibble: 2,048 x 2
##   w.length s.e.irrad
##      <dbl>     <dbl>
## 1 188.8252         0
## 2 189.2849         0
## 3 189.7444         0
## # ... with 2,045 more rows

getWhereMeasured(jaz04.spct)

##        lon      lat
## 1 25.02006 60.22515
```