UV and VIS radiation calculations

Theory and calculations (Using R)

Pedro J. Aphalo

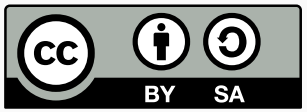
2020-10-06

OEB, ViPS, University of Helsinki

and

Viikki Plant Science Center, University of Helsinki

Department of Biosciences, University of Helsinki, Finland.
http://blogs.helsinki.fi/senpep-blog/



Outline

The starting point

Spectra

The r4photobiology R packages

Equations compared to r4photobiology code

Workflow examples

Metadata

The starting point

Photobiology and beyond

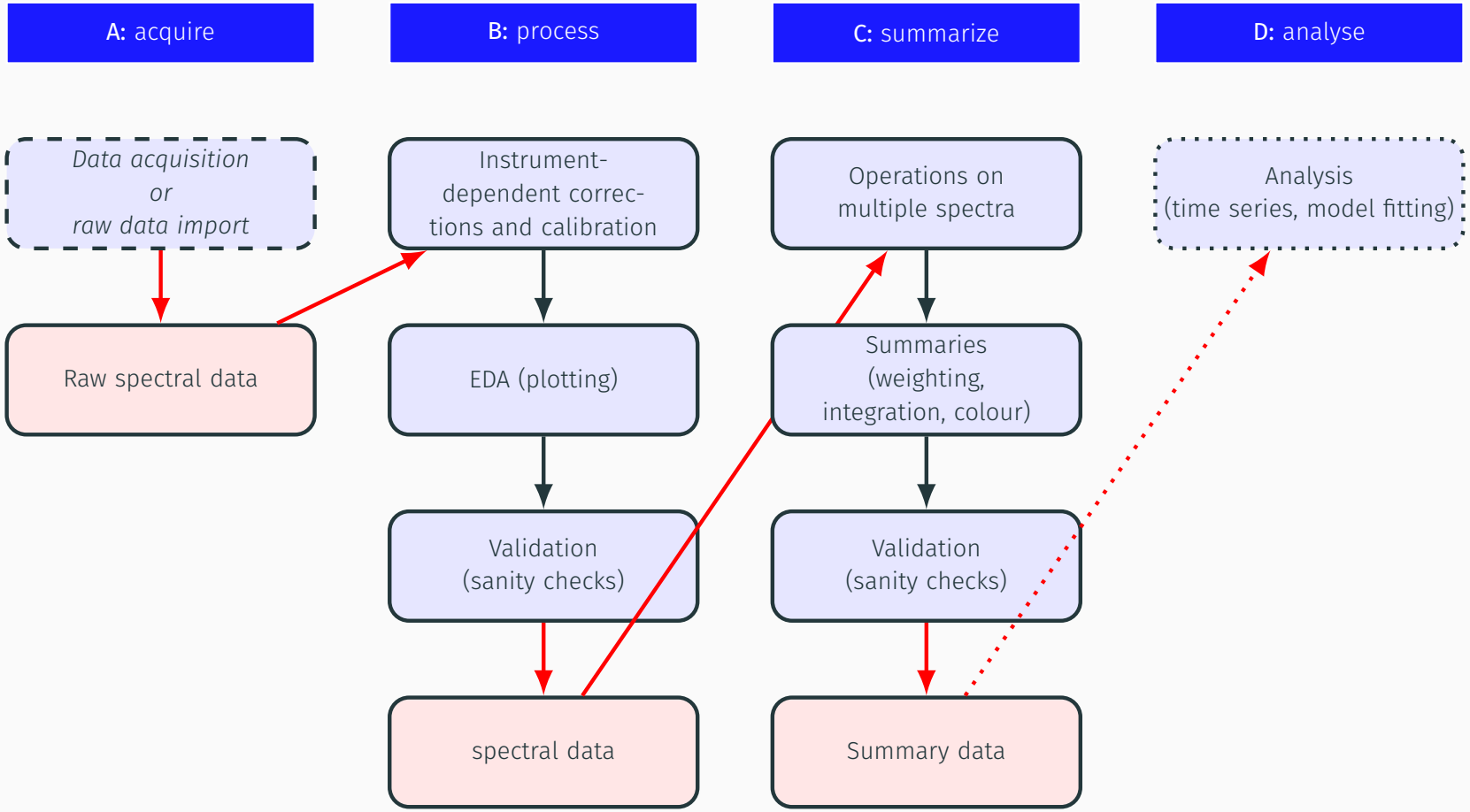
- Photobiology studies interactions of organisms with “light”.
- Ultraviolet (UV), visible (VIS), and near infrared (NIR).
- Same quantities and calculations used in other disciplines...
- e.g. meteorology, climatology and photochemistry.
- We quantify radiation, optical properties of objects and responses to radiation.

Radiation quantification: some of the difficulties

- Sanity checks/validation of data rarely done (errors creep in).
- Instrument manufacturers supply general purpose software (algorithms fixed).
- Many calculations are done in ad-hoc spreadsheets (error prone and slow).
- The amount of data can be large (scripting and reproducibility needed).
- Measuring UV radiation is difficult.
- **Result**: Some publications contain gross errors in UV-quantification (e.g. by orders of magnitude).
- **Main(?) cause**: Lack of easy to use, but flexible, software and reference data.

Spectra

Data flow: from acquisition to report



The r4photobiology R packages

Overall aims

- Facilitate data validation $\rightarrow$ prevent mistakes.
- Facilitate data manipulation $\rightarrow$ algorithms and standard-based definitions for consistency.
- Support reproducibility $\rightarrow$ open-source code can be reviewed.
- Support reproducibility $\rightarrow$ scripts to avoid ambiguities.
- Support reproducibility $\rightarrow$ automatic metadata handling.

Pedro's rules

- Plot spectral data before any calculations!
- Plot spectral data after any processing!
- If possible also plot summaries that have computed!
- Compare results of to what is expected!
  (literature, manufacturer's catalogues, your own earlier measurements)

Radiation-related spectra

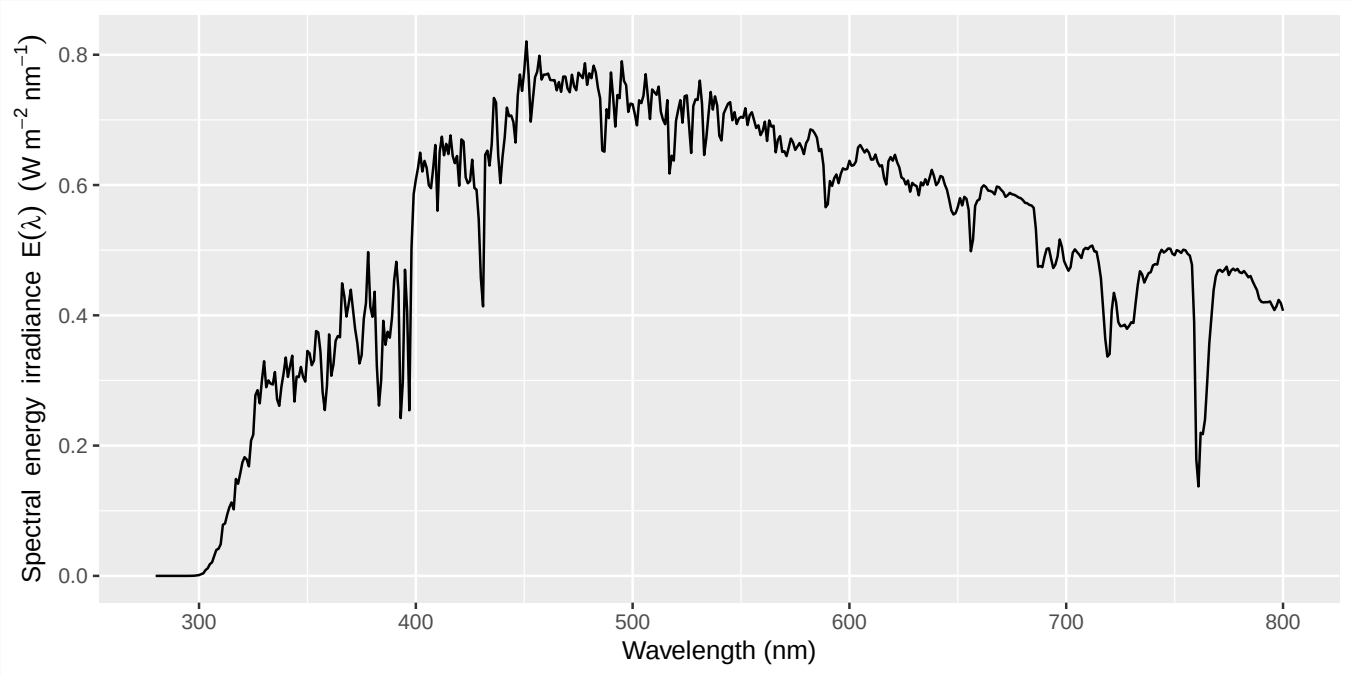
- A variable **wavelength**, λ, (or equivalent quantity) is always present.
- Another physical quantity either *expressed per unit wavelength* or as a ratio is also present.

- Wavelength is a continuous variable, but measurements are usually taken at discrete positions.
- Physical spectral quantities are also continuous, however, raw detector signals are usually digitized as discrete “counts”.

Plotting 1

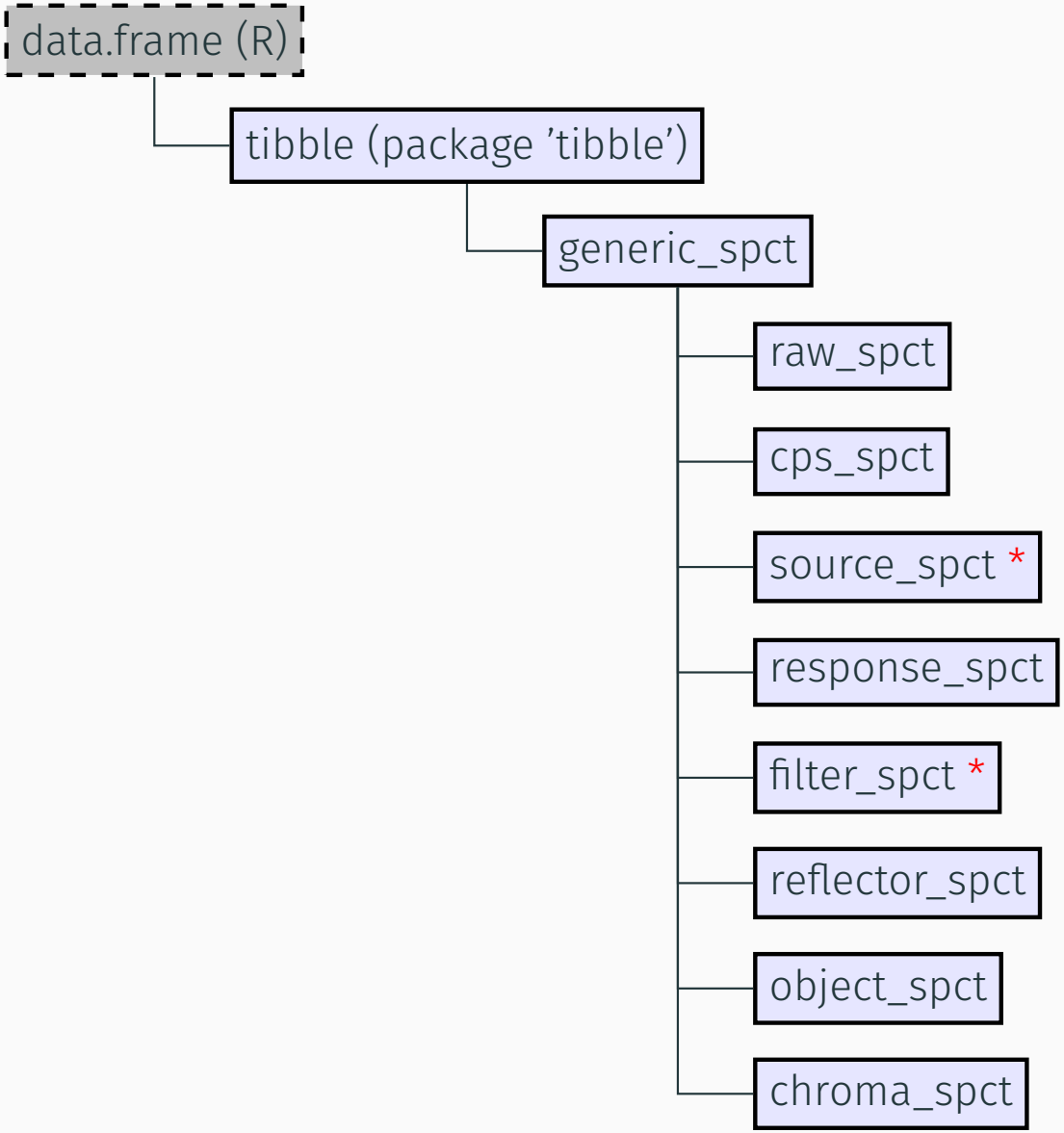
In our first example `autoplot()` is a *method* and `sun.spct` is an *object*.

```
autoplot(sun.spct, annotations = "")
```



`sun.spct` is example data for spectral irradiance, and `autoplot()` is a special version of R's `autoplot()` method tailored for spectra. Thanks to the metadata stored in `sun.spct` labels and annotations automatically match the data.

Classes of spectra



I will use only objects of two classes, marked with *, in the examples.

Objects and methods

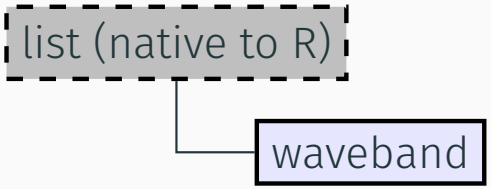
In our second example `e_irrad()` is a *method* and `sun.spct` and `UVB` are *objects*. `UVB` is an object of class `waveband`.

```
UVB <- UVB()
e_irrad(sun.spct, UVB)

## E_UVB.ISO
## 0.6445105
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

Here we compute the (non-weighted) UVB irradiance according to the ISO standard definition. `UVB()` is a constructor function. It accepts an additional argument for choosing other defintinitions for UVB that are in frequent use.

The waveband class



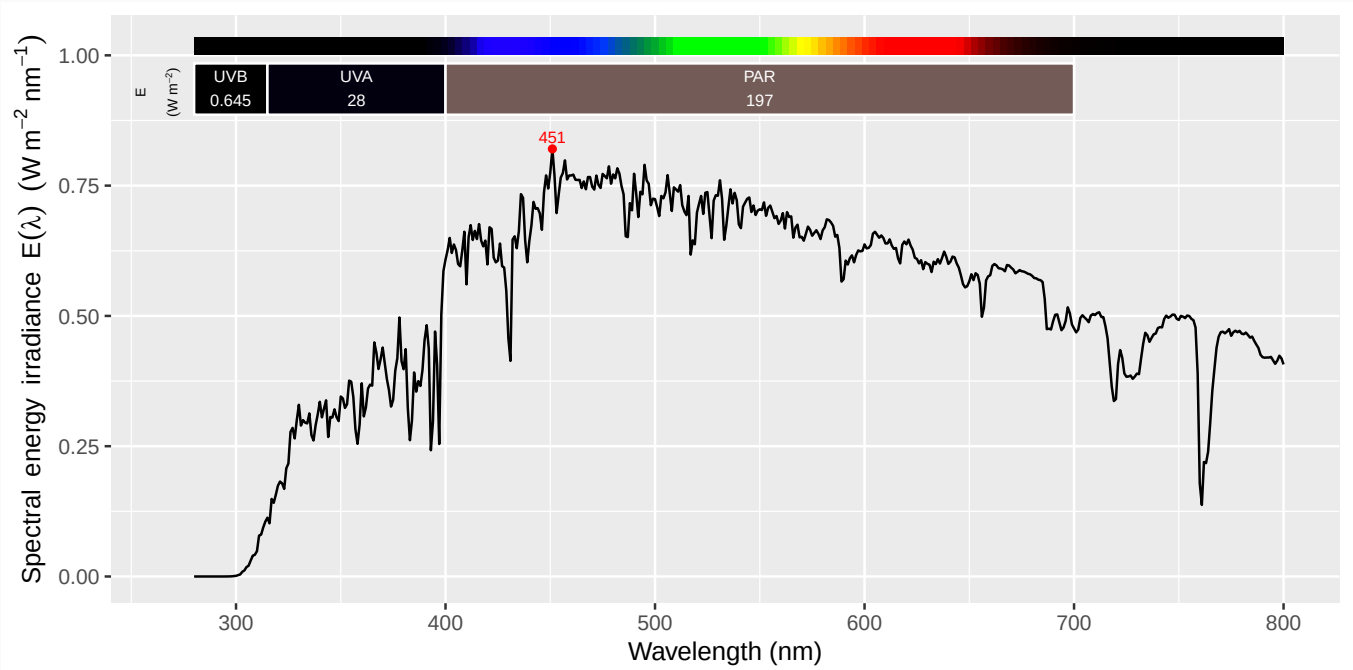
Objects of class `waveband` can contain either simple definitions of wavelength ranges or in addition such ranges plus spectral weighting function definitions.

The package `photobiologyWavebands` defines several constructors of `waveband` objects: standard and non-standard ranges of wavelengths for colours, and spectral weighting functions (SWFs), such as CIE's erythemal and Martyn Caldwell's generalized plant action spectrum (including different formulations).

Plotting 2

In the first example `autoplot()` was called suppressing automatic annotations. Here we replot using defaults.

```
autoplot(sun.spct)
```

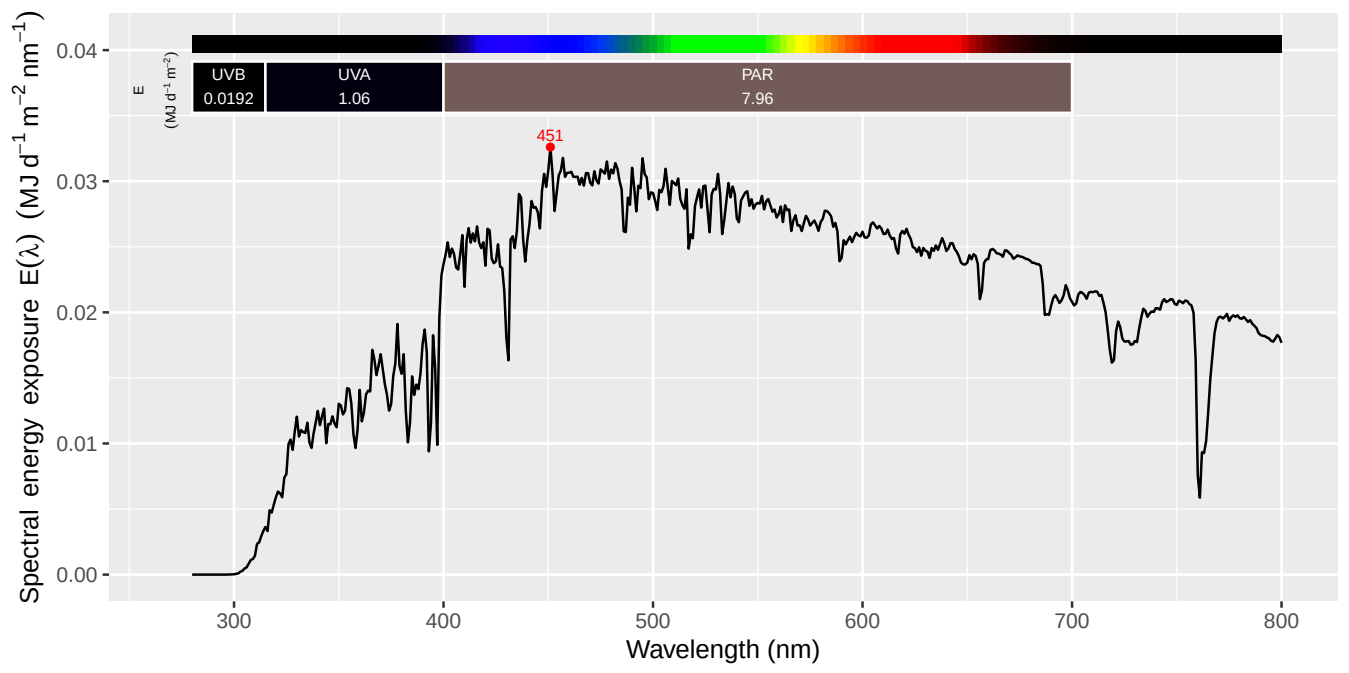


The new plot displays additional information, including annotations with energy irradiance for different wavebands, their names, and the units of expression for these summary quantities.

Plotting 3

To exemplify the role of metadata we plot an object containing a daily integral of the solar spectrum.

```
autoplot(sun.daily.spct)
```



Notice how axis labels and summary units have changed with respect to the previous plot.

Sun position

To further exemplify the use of metadata, we calculate the solar elevation and day length for two spectra.

```
sun_elevation(time = getWhenMeasured(sun.spct),
              geocode = getWhereMeasured(sun.spct))
```

```
## [1] 52.82721
```

```
day_length(date = getWhenMeasured(sun.daily.spct),
           geocode = getWhereMeasured(sun.daily.spct),
           unit.out = "hours")
```

```
## [1] 18.39215
```

Equations compared to r4photobiology code

C: Summaries of spectra

$$I_{\lambda_{\text{min}}\cdots\lambda_{\text{max}}} = \int_{\lambda=\lambda_{\text{min}}}^{\lambda=\lambda_{\text{max}}} I(\lambda)\mathrm{d}\lambda \tag{1}$$

```
e_irrad(sun.spct)
```

```
##  E_Total
## 269.1249
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

C: Summaries of spectra

$$I_{\text{UV-A}} = \int_{\lambda=\text{start}(\text{UV-A})}^{\lambda=\text{end}(\text{UV-A})} I(\lambda)\mathrm{d}\lambda \tag{2}$$

```
e_irrad(sun.spct, UVA())
```

```
## E_UVA.ISO
##  27.98421
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

C: Weighted irradiance

$$I_{\text{CIE}} = \int_{\lambda=\text{start}(w_{\text{CIE}})}^{\lambda=\text{end}(w_{\text{CIE}})} I(\lambda) \times w_{\text{CIE}}(\lambda) \mathrm{d}\lambda \tag{3}$$

```
e_irrad(sun.spct, CIE())
```

```
## E_]CIE98.298
##   0.08181583
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

C: Weighted spectral irradiance

$$I(\lambda) \times w_{\mathrm{CIE}}(\lambda)\mathrm{d}\lambda \tag{4}$$

```
sun.spct * CIE()
```

```
## Object: source_spct [122 x 2]
## Wavelength range 280 to 400 nm, step 0.9230769 to 1 nm
## Label: sunlight, simulated
## Measured on 2010-06-22 09:51:00 UTC
## Measured at 60.20911 N, 24.96474 E; Kumpula, Helsinki, FI
## Time unit 1s
## Data weighted using 'CIE98.298' BSWF
##
## # A tibble: 122 x 2
##   w.length s.e.irrad
##      <dbl>     <dbl>
## 1     280          0
## 2     281.         0
## 3     282.         0
## 4     283.         0
## # ... with 118 more rows
```

C: Weighted irradiance

$$I_{\text{CIE}} = \int I(\lambda) \times w_{\text{CIE}}(\lambda) \, \mathrm{d}\lambda \tag{5}$$

```
e_irrad(sun.spct * CIE())
```

```
## E_CIE98.298*Total
##        0.08181583
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "energy irradiance total effective: CIE98.298"
```

Workflow examples

Introduction to the examples

Letters such as **B:** at the start of the title of slides indicates the column in the data flow diagram the example applies to.

The R4P suite includes many data sets, including a spectrum of sunlight at ground level stored as `sun.spct`, which we will use in examples.

The examples I will give today are simple.

> There are more than 200 method specializations and functions in the *r4photobiology* packages and among the example data included, more than 100 different filter spectra, nearly 30 lamp spectra, nearly 30 LED spectra, some plant related reflectance and transmittance spectra, photoreceptor absorbance spectra, reflectance spectra for some metals, extraterrestrial and ground level solar spectra, broadband sensor and detector arrays response spectra, etc. All these can be combined in many different ways, and with data from other sources including your own measurements. Titta Kotilainen has been good at showing me many possible calculations and plots that I had not thought about myself when writing the code!

A: Spectral data acquisition

Simplest possible example for an Ocean Optics spectrometer.

```
w <- start_session()
list_instruments(w)
descriptor <- get_oo_descriptor(w)
settings <- acq_settings(descriptor, HDR.mult = 1,
                         integ.time = 100e-3, num.scans = 5)
spct_0 <- acq_raw_spct(descriptor, settings)
end_session(w)
```

B: Example data: Spectral irradiance of sunlight

We can print the spectrum implicitly as below, or by explicitly using the `print()` method.

```
sun.spct
```

```
## Object: source_spct [522 x 3]
## Wavelength range 280 to 800 nm, step 0.9230769 to 1 nm
## Label: sunlight, simulated
## Measured on 2010-06-22 09:51:00 UTC
## Measured at 60.20911 N, 24.96474 E; Kumpula, Helsinki, FI
## Time unit 1s
##
## # A tibble: 522 x 3
##   w.length s.e.irrad s.q.irrad
##      <dbl>     <dbl>     <dbl>
## 1     280          0         0
## 2     281.         0         0
## 3     282.         0         0
## 4     283.         0         0
## # ... with 518 more rows
```

Quantities are always stored using the same units (e.g.):

`w.length` = wavelength in nanometres (10^{-9} m, 1 nm = 10 Å)

`s.e.irrad` = spectral energy irradiance in $\mathrm{W\,m^{-2}\,nm^{-1}}$

B: Example data: a glass filter from Schott

Another big set of spectral data are spectra for all Schott filters currently in production stored in a collection of spectra object. In this case we use `$` to select a member from the collection.

```
filters.mspct$Schott_GG400
```

```
## Object: filter_spct [1,001 x 2]
## Wavelength range 200 to 5200 nm, step 1 to 50 nm
## Label: SCHOTT GG4000.003
## Transmittance of type 'internal'
## Rfr (/1): 0.082, thickness (mm): 3, attenuation mode: absorption.
##
## # A tibble: 1,001 x 2
##   w.length     Tfr
## *    <dbl>   <dbl>
## 1      200 0.00001
## 2      201 0.00001
## 3      202 0.00001
## 4      203 0.00001
## # ... with 997 more rows
```

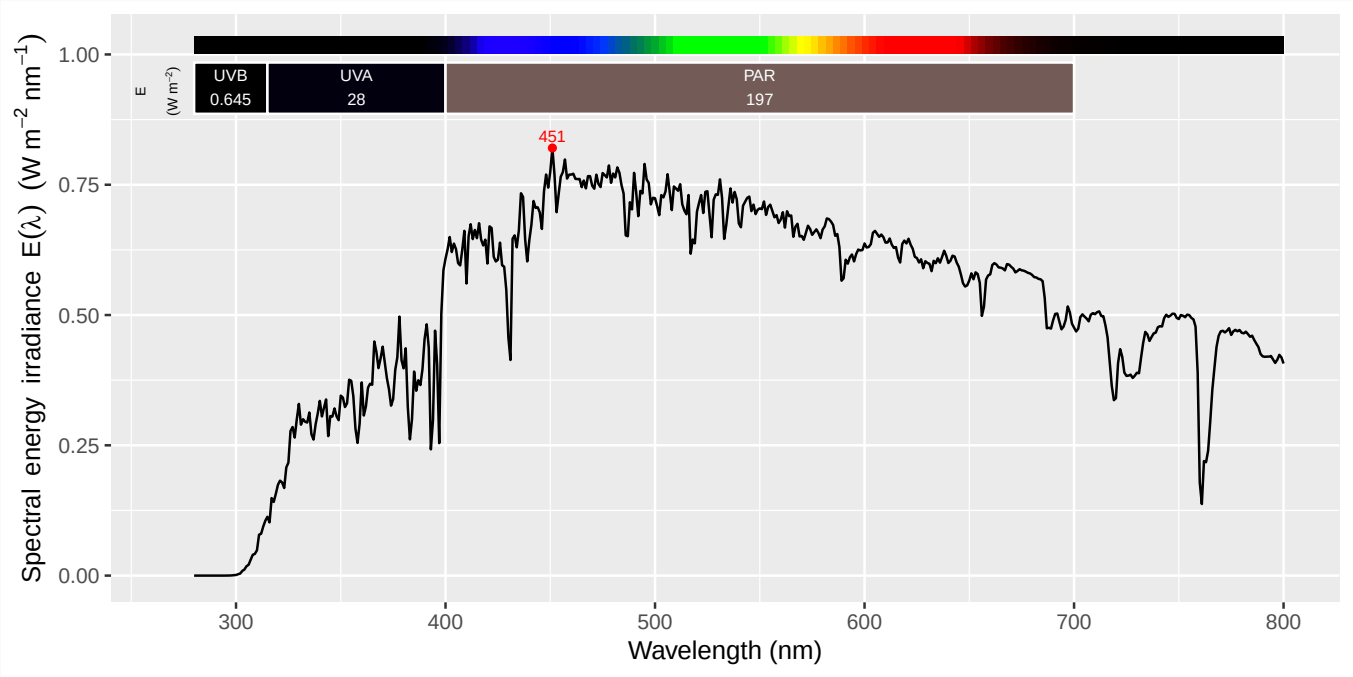
`w.length` = wavelength in nanometres (10^{-9} m, 1 nm = 10 Å)

`Tfr` = spectral transmittance as a fraction of one

B: plot methods for spectra

As the spectral data is stored using known units, plus contains metadata as attributes, the plot method can automatically produce suitable axis labels.

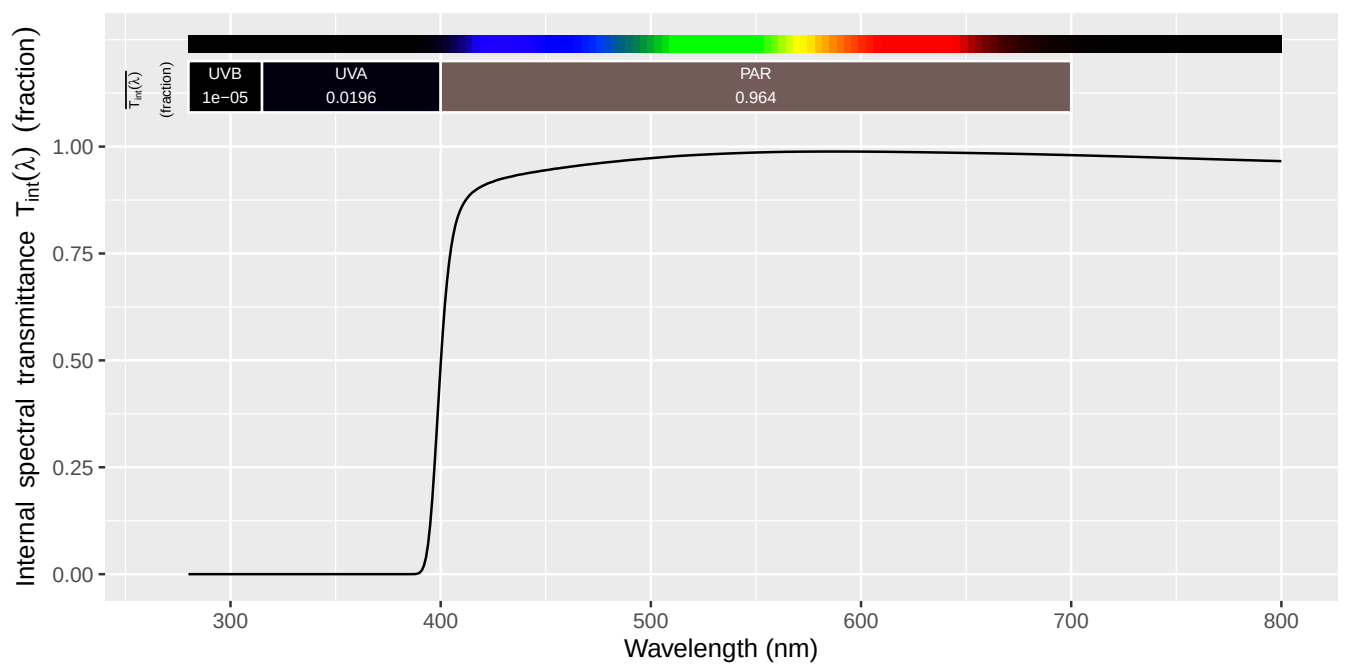
```
autoplot(sun.spct)
```



B: plot methods for spectra

As the spectral data is stored using known units, plus contains metadata as attributes, the plot method can automatically produce suitable axis labels. In this example we select part of the spectral data.

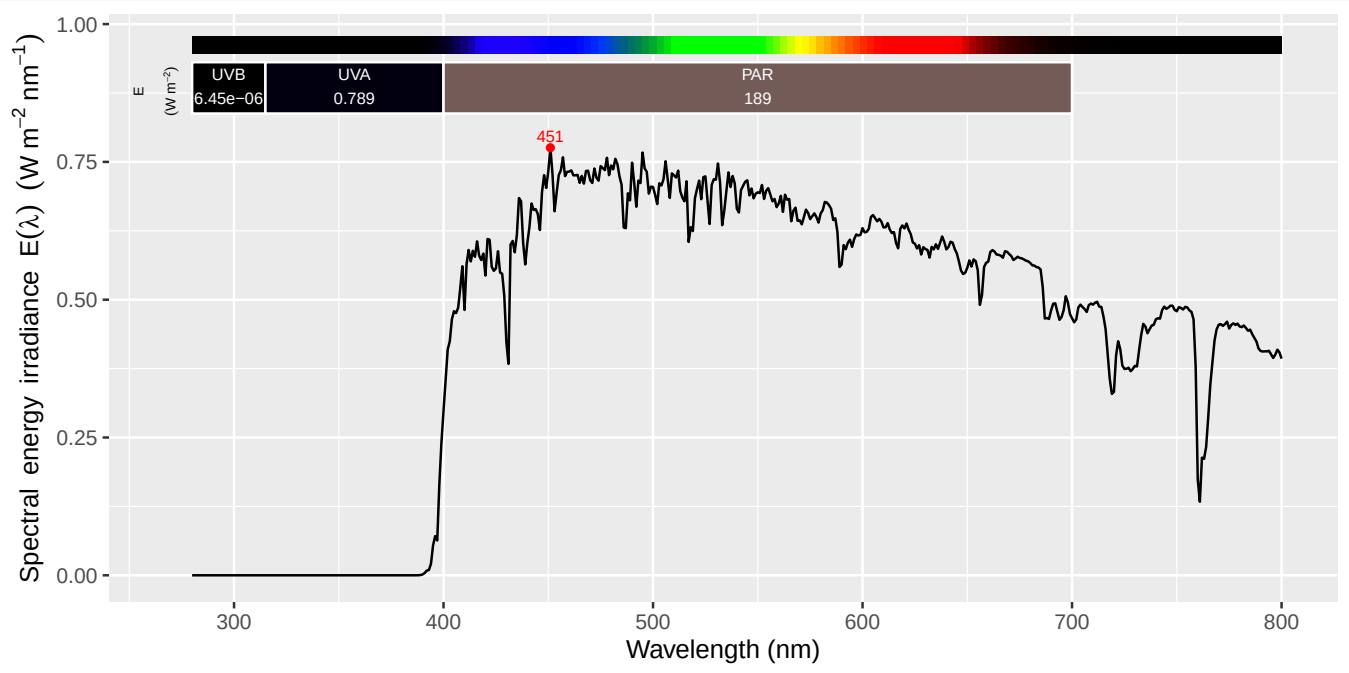
```
autoplot(filters.mspct$Schott_GG400, range = sun.spct)
```



C: Operators: e.g. spectral irradiance under a filter

We can simulate the spectral irradiance of sunlight filtered with the long-pass UV filter shown earlier by multiplying the two spectra.

```
autoplot(sun.spct * filters.mspct$Schott_GG400)
```

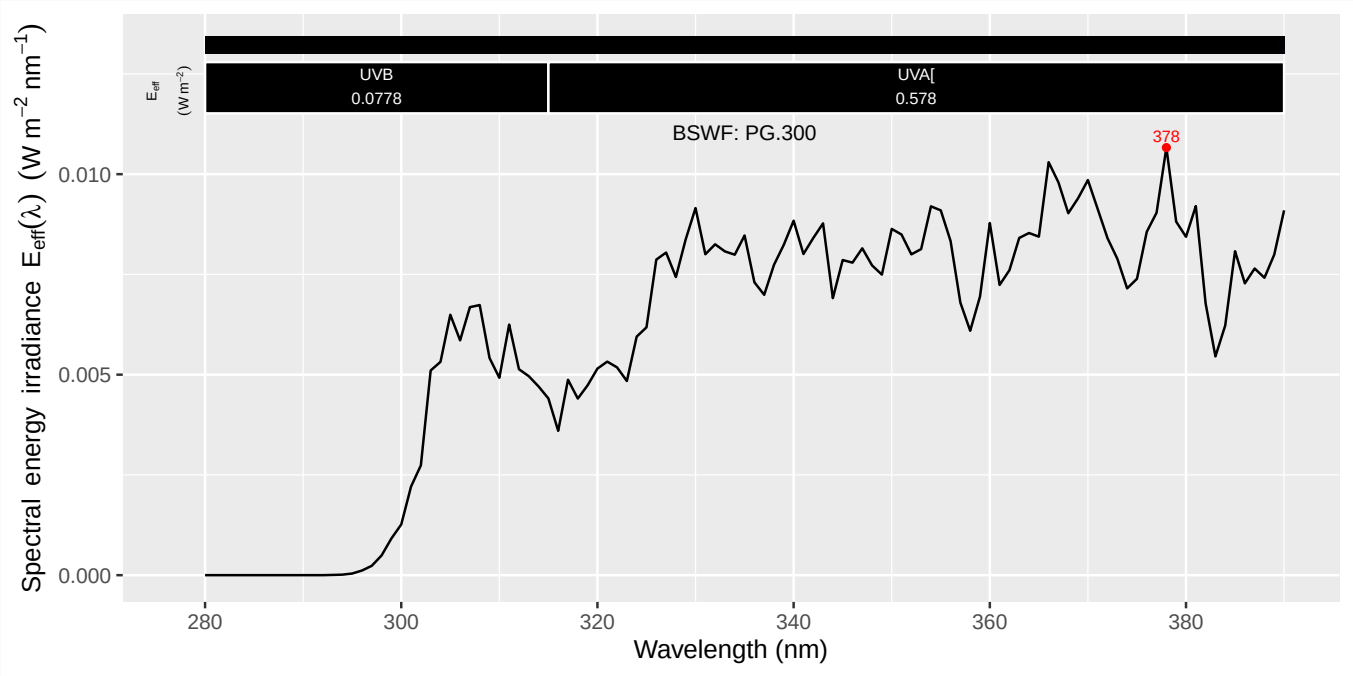


What is not obvious here is that as the two spectra have been measured at different wavelength values, interpolation was needed before the convolution.

C: Operators for spectra: effective spectral irradiance

Effective spectral irradiance according to the Plant Growth BSWF.

```
autoplot(sun.spct * PG(), range = c(280, 500))
```



C: Summaries of spectra

```
summary(sun.spct)
```

```
## Summary of source_spct [522 x 3] object: sun.spct
## Wavelength range 280 to 800 nm, step 0.9230769 to 1 nm
## Label: sunlight, simulated
## Measured on 2010-06-22 09:51:00 UTC
## Measured at 60.20911 N, 24.96474 E; Kumpula, Helsinki, FI
## Time unit 1s
##
##     w.length       s.e.irrad        s.q.irrad
##  Min.   :280.0   Min.   :0.0000   Min.   :0.000e+00
##  1st Qu.:409.2   1st Qu.:0.4115   1st Qu.:1.980e-06
##  Median :539.5   Median :0.5799   Median :2.929e-06
##  Mean   :539.5   Mean   :0.5160   Mean   :2.407e-06
##  3rd Qu.:669.8   3rd Qu.:0.6664   3rd Qu.:3.154e-06
##  Max.   :800.0   Max.   :0.8205   Max.   :3.375e-06
```

C: Irradiance for the band(s) defined according to wavelengths

```
e_irrad(sun.spct, waveband(c(400, 700)) )
```

```
## E_range.400.700
##        196.6343
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

```
e_irrad(sun.spct, split_bands(c(400, 500, 600, 700)) )
```

```
##    E_wb1    E_wb2    E_wb3
## 69.69043 68.48950 58.45434
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

C: photon (quantum) irradiance for two bands and photon ratio

```
q_irrad(sun.spct,
        w.band = list(UVB(), UVA(), PAR())) * 1e6
```

```
##  Q_UVB.ISO  Q_UVA.ISO      Q_PAR
##   1.675362  84.819697 894.135224
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total photon irradiance"
```

```
q_ratio(sun.spct,
        w.band.num = list(UVB(), UVA()),
        w.band.denom = PAR())
```

```
## UVB:PAR[q:q] UVA:PAR[q:q]
##  0.001873724  0.094862270
## attr(,"radiation.unit")
## [1] "q:q ratio"
```

To `q_irrad()` we pass two arguments, and to `q_ratio()` three.

C: Simulating effect of a filter on irradiance

As shown abobe for plots, we can convolve spectra on-the-fly when calculating summaries.

```
e_irrad(sun.spct, UVA())
```

```
## E_UVA.ISO
##  27.98421
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

```
e_irrad(sun.spct * filters.mspct$Schott_GG400, UVA())
```

```
## E_UVA.ISO
## 0.7887421
## attr(,"time.unit")
## [1] "second"
## attr(,"radiation.unit")
## [1] "total energy irradiance"
```

Metadata

Attributes: time unit

```
getTimeUnit(sun.spct)
```

```
## [1] "second"
```

```
getTimeUnit(sun.daily.spct)
```

```
## [1] "day"
```

Attributes: measurement metadata

```
getWhenMeasured(white_led.source_spct)
```

```
## [1] "2016-12-19 16:19:57 UTC"
```

```
getInstrDesc(white_led.source_spct)
```

```
## Data acquired with 'MayaPro2000' s.n. MAYP11278
## grating 'HC1', slit '010s'
```

```
getInstrSettings(white_led.source_spct)
```

```
## integ. time (s): 0.233, 1.43, 5
## total time (s): 10, 10, 10
## counts @ peak (% of max): 94.2
```

C: Collections of spectra 1

Container classes for spectra provided. For example the object `filters.mspct` used above is one such collection. It contains more than 100 transmittance spectra. We can retrieve individual spectra by name as shown above using the `$` operator. With a vector of names between square brackets we can retrieve a new collection containing a subset.

```
length(filters.mspct)
```

```
## [1] 339
```

```
length(filters.mspct[c("Schott_UG1", "Schott_GG400")])
```

```
## [1] 2
```

```
names(filters.mspct[c("Schott_UG1", "Schott_GG400")])
```

```
## [1] "Schott_UG1"   "Schott_GG400"
```

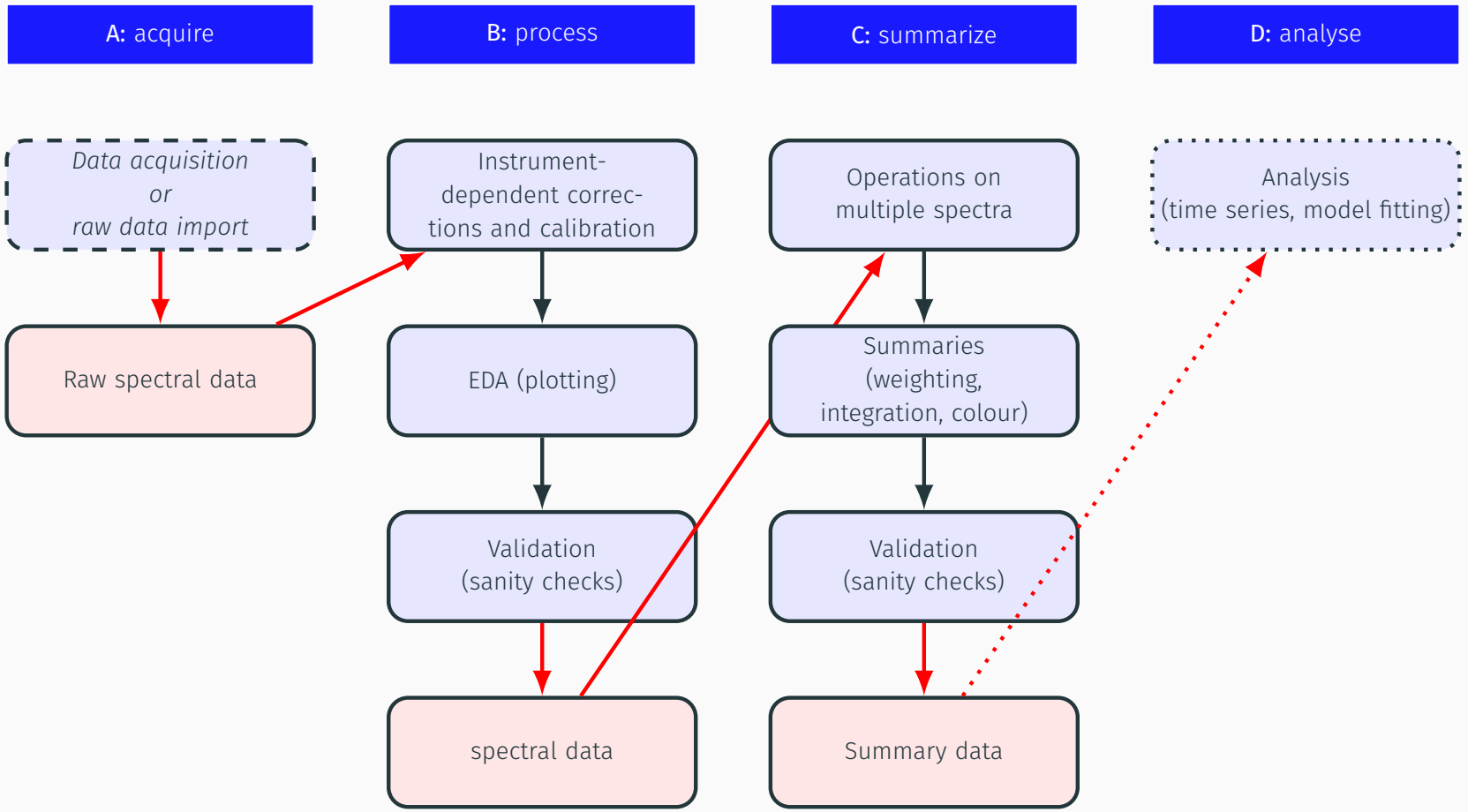
C: Collections of spectra 2

As shown above, we can convolve individual spectra when calculating summaries. We can also convolve collections of spectra.

```
e_irrad(convolve_each(sun.spct,
                      filters.mspct[c("Schott_UG1", "Schott_GG400")]),
        w.band = list(UVB(), UVA(), PAR()))
```

```
## # A tibble: 2 x 4
##   spct.idx      E_UVB.ISO E_UVA.ISO   E_PAR
##   <fct>             <dbl>     <dbl>   <dbl>
## 1 Schott_UG1   0.228         18.4     0.491
## 2 Schott_GG400 0.00000645     0.789 189.
```

Data flow


A: acquire
B: process
C: summarize
D: analyse
Data acquisition or raw data import
Raw spectral data
Instrument-dependent corrections and calibration
EDA (plotting)
Validation (sanity checks)
spectral data
Operations on multiple spectra
Summaries (weighting, integration, colour)
Validation (sanity checks)
Summary data
Analysis (time series, model fitting)

Conclusion: R's 'programmatic' interface...

- Allows scripting to be used at all steps in the workflow.
- Allows users to extend (and modify) functionality as needed.
- Can be used together with other R packages.
- Data can be imported and exported at different steps of the workflow.
- Being open-source every step in the data processing is open to scrutiny.
- Reproducibility is drastically improved.