R for Photobiology

A handbook

Pedro J. Aphalo

DRAFT

Contents

Preface

Use the template *preface.tex* together with the Springer document class SVMono (monograph-type books) or SVMult (edited books) to style your preface in the Springer layout.

A preface is a book's preliminary statement, usually written by the *author or editor* of a work, which states its origin, scope, purpose, plan, and intended audience, and which sometimes includes afterthoughts and acknowledgments of assistance.

When written by a person other than the author, it is called a foreword. The preface or foreword is distinct from the introduction, which deals with the subject of the work.

Customarily *acknowledgments* are included as last part of the preface.

Place(s),
month year

Firstname Surname
Firstname Surname

Part I

Getting ready

CHAPTER 1

Introduction

Abstract

In this chapter we explain the physical basis of optics and photochemisatry.

1.1 Radiation and molecules

CHAPTER 2

Optics

Abstract

In this chapter we explain how to .

2.1 Task:

CHAPTER 3

Photochemistry

Abstract

In this chapter we explain how to .

3.1 Task:

CHAPTER 4

Software

Abstract

In this chapter we explain how to .

4.1 Task:

CHAPTER 5

Photobiology R pacakges

Abstract

In this chapter we explain how to .

5.1 photoCRAN repo

I have created a small repository for the packages. This repository follows the CRAN folder structure, so now package installation can be done using just the normal R commands. This means that dependencies are installed automatically, and that automatic updates are possible. The build most suitable for the current system and R version is also picked automatically if available. It is normally recommended that you do installs and updates on a clean R session (just after starting R or RStudio).For easy installation and updates of packages, the photoCRAN repo can be added to the list of repos that R knows about.

Whether you use RStudio or not it is possible to add the photoCRAN repo to the current session as follows:

```
setRepositories(graphics = getOption("menu.graphics"), ind = NULL,
    addURLs = c(photoCRAN = "http://www.mv.helsinki.fi/aphalo/R"))
```

which will give you a menu of additional repos to activate.

If you know the indexes in the menu you can use this code, where 1 and 6 are the entries in the menu in the command above.

```
setRepositories(graphics = getOption("menu.graphics"), ind = c(1,
    6), addURLs = c(photoCRAN = "http://www.mv.helsinki.fi/aphalo/R"))
```

Be careful not to issue this command more than once per R session, otherwise the list of repos gets corrupted by having two repos with the same name.

After adding the repo to the session, it will appear in the menu when executing this command:

```
setRepositories()

## Error:  cannot set repositories non-interactively
```

and can be enabled and disabled.

In RStudio, after adding the photoCRAN repo as shown above, the photobiology packages can be installed and uninstalled through the normal RStudio menues and dialogues. For example when you type photob in the packages field, all the packages with names starting with photob will be listed. They can be also installed with:

```
install.packages(c("photobiologyAll", "photobiologygg"))

## Installing packages into 'C:/Users/aphalo/Documents/R/win-library/3.1'
## (as 'lib' is unspecified)
## Error:  trying to use CRAN without setting a mirror
```

and updated with:

```
update.packages()

## Error:  trying to use CRAN without setting a mirror
```

The added repo will persist only during the current R session. Adding it permanently requires editing the R configuration file.

5.2 How to install the pakages

The examples given in this page assume that photoCRAN is not in the list of repos known to the current R session. See the section 5.1 on the photoCRAN repo for an alternative to the approach given here.

To install the latest version of one package (photobiology used as example) you just need to indicate the repository. However this simple command will only install the dependencies bewteen the different photobiology packages.

```
install.packages("photobiology", repos = "http://www.mv.helsinki.fi/aphalo/R")

## Installing package into 'C:/Users/aphalo/Documents/R/win-library/3.1'
## (as 'lib' is unspecified)
## Warning:  package 'photobiology' is in use and will not be installed
```

To update what is already installed, this command is enough (even if the packages have been installed manually before):

```
update.packages(repos = "http://www.mv.helsinki.fi/aphalo/R")
```

The best way to install the packages is to specify both my repo and a normal CRAN repo, then all dependencies will be automatically installed. The new package photobiolgyAll just loads and imports all the packages in the suite, except for photobiolygg. Because of this dependency all the packages are installed unless already installed.

```
install.packages(c("photobiologyAll", "photobiologygg"), repos = c(photoCRAN = "http
    CRAN = "http://cran.rstudio.com"))
```

```
## Installing packages into 'C:/Users/aphalo/Documents/R/win-library/3.1'
## (as 'lib' is unspecified)
## Warning:  packages 'photobiologyAll', 'photobiologygg' are in use
and will not be installed
```

```
install.packages(c("photobiologyAll", "photobiologygg"), repos = c(photoCRAN = "http://www.mv
    CRAN = "http://cran.rstudio.com"))

## Installing packages into 'C:/Users/aphalo/Documents/R/win-library/3.1'
## (as 'lib' is unspecified)
## Warning:  packages 'photobiologyAll', 'photobiologygg' are in use
and will not be installed
```

This example also shows how one can use an array of package names (in this example all my currently available photobiology packages) in the call to the function install.packages, this is useful if you want to install only a subset of the files, or if you want to make sure that any older install of the packages is overwritten:

```
photobiology_packages <- c("photobiology", "photobiology", "photobiology",
    "photobiologyVIS", "photobiologyUV", "photobiologyCry", "photobiologyPhy",
    "photobiologyLamps", "photobiologyLEDs", "photobiologySun",
    "photobiologygg", "photobiologyFilters", "photobiologySensors")

install.packages(photobiology_packages, repos = c(photoCRAN = "http://www.mv.helsinki.fi/apha
    CRAN = "http://cran.rstudio.com"))

## Installing packages into 'C:/Users/aphalo/Documents/R/win-library/3.1'
## (as 'lib' is unspecified)
## Warning:  packages 'photobiology', 'photobiology', 'photobiology',
'photobiologyVIS', 'photobiologyUV', 'photobiologyCry', 'photobiologyPhy',
'photobiologyLamps', 'photobiologyLEDs', 'photobiologySun', 'photobiologygg',
'photobiologyFilters', 'photobiologySensors' are in use and will not
be installed
```

The commands above install all my packages and all their dependencies from CRAN if needed. The following command will update all the packages currently installed (if new versions are available) and install any new dependencies.

```
update.packages(repos = c(photoCRAN = "http://www.mv.helsinki.fi/aphalo/R",
    CRAN = "http://cran.rstudio.com"))

## Gmisc :
##  Version 0.6.2.0 installed in C:/Users/aphalo/Documents/R/win-library/3.1
##  Version 0.6.2.3 available at http://cran.rstudio.com
## Update (y/N/c)?
## markdown :
##  Version 0.6.5 installed in C:/Users/aphalo/Documents/R/win-library/3.1
##  Version 0.7 available at http://cran.rstudio.com
## Update (y/N/c)?
## mixtools :
##  Version 1.0.1 installed in C:/Users/aphalo/Documents/R/win-library/3.1
##  Version 1.0.2 available at http://cran.rstudio.com
## Update (y/N/c)?
## MASS :
##  Version 7.3-31 installed in C:/Program Files/R/R-3.1.0/library
```

```
##  Version 7.3-33 available at http://cran.rstudio.com
## Update (y/N/c)?
```

The instructions above should work under Windows as long as you have a supported version of R (3.0.0 or later) because I have built suitable binaries, under other OS you may need to add type="source" unless this is already the default. We will try to build OS X binaries for Mac so that installation is easier. Meanwhile if installation fails try adding type="source" to the commands given above. For example the first one would become:

```
install.packages("photobiology", repos = "http://www.mv.helsinki.fi/aphalo/R",
    type = "source")

## Installing package into 'C:/Users/aphalo/Documents/R/win-library/3.1'
## (as 'lib' is unspecified)

##
## The downloaded source packages are in
##  'C:\Users\aphalo\AppData\Local\Temp\RtmpWWGf0u\downloaded_packages'
```

When using type=source you may need to install some dependencies like the splus2R package beforehand from CRAN if building it from sources fails.

Part II

Cookbook

CHAPTER 6

Unweighted irradiance

Abstract

In this chapter we explain how to calculate unweighted energy and photon irradiances from spectral irradiance.

6.1 Task: (energy) irradiance from spectral irradiance

The task to be completed is to calculate the (energy) irradiance (E) in $\mathrm{W\,m^{-2}}$ from spectral (energy) irradiance ($Q(\lambda)$) in $\mathrm{W\,m^{-2}\,nm^{-1}}$ and the corresponding wavelengths (λ) in nm.

$$Q_{\lambda_1<\lambda<\lambda_2} = \int_{\lambda_1}^{\lambda_2} E(\lambda)\ \mathrm{d}\,\lambda \tag{6.1}$$

Let's assume that we want to calculate photosynthetically active radiation (PAR) energy irradiance, for which the most accepted limits are $\lambda_1 = 400$nm and $\lambda_1 = 700$nm. In this example we will use example data for sunlight to calculate $E_{400\,\mathrm{nm}<\lambda<700\,\mathrm{nm}}$:

```
with(sun.data, energy_irradiance(w.length, s.e.irrad,
                                 new_waveband(400, 700)))

## range.400.700
##         196.7
```

Function `PAR()` is predefined in package photobiologyVIS as a convenience function, so the code above can be replaced by:

```
with(sun.data, energy_irradiance(w.length, s.e.irrad, PAR()))

##   PAR
## 196.7
```

If no waveband is supplied as argument, then the whole range of wavelengths in the spectral data is used for the integration, and the 'name' attribute is generated accordingly:

```
with(sun.data, energy_irradiance(w.length, s.e.irrad))

## range.293.800
##         268.9
```

If a waveband that does not fully overlap with the data is supplied as argument, then spectral irradiance for wavelengths outside the range is assumed to be zero:

```
with(sun.data, energy_irradiance(w.length, s.e.irrad,
                                 new_waveband(700,1000)))

## range.700.1000
##           44.1
```

If a waveband that does not overlap with the data is supplied as argument, then spectral irradiance for wavelengths outside the range is assumed to be zero:

```
with(sun.data, energy_irradiance(w.length, s.e.irrad,
                                 new_waveband(100,200)))

## range.100.200
##             0
```

6.2 Task: photon irradiance from spectral irradiance

The task to be completed is to calculate the photon irradiance (Q) in $\mathrm{mol\,m^{-2}\,s^{-1}}$ from spectral (energy) irradiance ($E(\lambda)$) in $\mathrm{W\,m^{-2}\,nm^{-1}}$ and the corresponding wavelengths (λ) in nm.

The energy of a quantum of radiation in a vacuum, q, depends on the wavelength, λ, or frequency[1], ν,

$$q = h \cdot \nu = h \cdot \frac{c}{\lambda} \tag{6.2}$$

with the Planck constant $h = 6.626 \times 10^{-34}$ J s and speed of light in vacuum $c = 2.998 \times 10^{8}$ $\mathrm{m\,s^{-1}}$. When dealing with numbers of photons, the equation (6.2) can be extended by using Avogadro's number $N_A = 6.022 \times 10^{23}$ $\mathrm{mol^{-1}}$. Thus, the energy of one mole of photons, q', is

$$q' = h' \cdot \nu = h' \cdot \frac{c}{\lambda} \tag{6.3}$$

with $h' = h \cdot N_A = 3.990 \times 10^{-10}$ $\mathrm{J\,s\,mol^{-1}}$. Example 1: red light at 600 nm has about 200 $\mathrm{kJ\,mol^{-1}}$, therefore, 1 µmol photons has 0.2 J. Example 2: UV-B

[1] Wavelength and frequency are related to each other by the speed of light, according to $\nu = c/\lambda$ where c is speed of light in vacuum. Consequently there are two equivalent formulations for equation 6.2.

radiation at 300 nm has about 400 $\mathrm{kJ\,mol^{-1}}$, therefore, 1 µmol photons has 0.4 J. Equations 6.2 and 6.3 are valid for all kinds of electromagnetic waves.

Combining equations 6.1 and 6.3 we obtain:

$$Q_{\lambda_1<\lambda<\lambda_2} = \int_{\lambda_1}^{\lambda_2} E(\lambda)\ \frac{h' \cdot c}{\lambda} \mathrm{d}\,\lambda \tag{6.4}$$

Let's assume that we want to calculate photosynthetically active radiation (PAR) photon irradiance. In this example we will use example data for sunlight.

```
with(sun.data,
     photon_irradiance(w.length, s.e.irrad, PAR()))

##        PAR
## 0.0008938
```

If we want to have Q_{PAR} (PPFD) expressed in the usual units of $\mathrm{\mu mol\,m^{-2}\,s^{-1}}$, we need to multiply the result above by 10^6:

```
with(sun.data,
     photon_irradiance(w.length, s.e.irrad, PAR()) * 1e6

##    PAR
## 893.8
```

`PAR()` is predefined in package photobiologyVIS as a convenience function, see section 6.1 for an example with arbitrary values for λ_1 and λ_2.

6.3 Task: calculate energy and photon irradiances from spectral photon irradiance

In the case of the calculation of energy irradiance from spectral photon irradiance the calculation is:

$$I_{\lambda_1<\lambda<\lambda_2} = \int_{\lambda_1}^{\lambda_2} Q_\lambda\ \frac{\lambda}{h' \cdot c} \mathrm{d}\,\lambda \tag{6.5}$$

And the code[2]:

```
with(sun.data, energy_irradiance(w.length, s.q.irrad, PAR()),
    unit.in = "photon")

##        PAR
## 0.0008938
```

The calculation of photon irradiance from spectral photon irradiance, is a simple integration, analogous to that in equation 6.1, and the code is:

```
with(sun.data, photon_irradiance(w.length, s.q.irrad, PAR()),
    unit.in = "photon")

##        PAR
## 4.158e-09
```

[2]The dataframe `sun.data` contains both spectral energy irradiance vales in 'column' `s.e.irrad` and spectral photon irradiance in 'column' `s.q.irrad`

6.4 Task: irradiances for more than one waveband

It is possible to calculate the irradiances for several wavebands with a single function call by supplying a `list` of `wavebands` as argument:

```
with(sun.data, photon_irradiance(w.length, s.e.irrad, list(Red(),
    Green(), Blue()))) * 1e+06

##   Red.ISO Green.ISO  Blue.ISO
##     452.2     220.2     149.0

Q.RGB <- with(sun.data, photon_irradiance(w.length, s.e.irrad,
    list(Red(), Green(), Blue()))) * 1e+06
signif(Q.RGB, 3)

##   Red.ISO Green.ISO  Blue.ISO
##       452       220       149

Q.RGB[1]

## Red.ISO
##   452.2

Q.RGB["Green.ISO"]

## Green.ISO
##     220.2
```

A named list can be used to override the use as names for the output of the waveband names:

```
with(sun.data, photon_irradiance(w.length, s.e.irrad, list(R = Red(),
    G = Green(), B = Blue()))) * 1e+06

##     R     G     B
## 452.2 220.2 149.0
```

Even when using a single waveband:

```
with(sun.data,
     photon_irradiance(w.length, s.e.irrad,
                       list(UVB=UVB()))) * 1e6

##   UVB
## 1.527
```

6.5 Task: use simple wavebands

Please, consult the packages' documentation for a list of predefined functions for creating wavebands. Here we will present just a few examples of their use. We usually associate wavebands with colours, however, in many cases there are different definitions in use. For this reason, the functions provided accept an argument that can be used to select the definition to use. In general, the default, is to use the ISO standard whenever it is applicable. The case of the various definitions in use for the UV-B waveband are described on page 21

We can use a predefined function to create a new `waveband` object, which as any other R object can be assigned to a variable:

```
uvb <- UVB()
uvb

## UVB.ISO
## low (nm) 280
## high (nm) 315
```

As seen above, there is a specialized `print` function for `wavebands`. Functions available are `min`, `max`, `range`, `center_wl`, `labels`, and `color`.

```
red <- Red()
red

## Red.ISO
## low (nm) 610
## high (nm) 760

min(red)

## [1] 610

max(red)

## [1] 760

range(red)

## [1] 610 760

midpoint(red)

## [1] 685

labels(red)

## $label
## [1] "Red.ISO"

color(red)

## Red.ISO CMF  Red.ISO CC
##   "#900000"   "#FF0000"
```

Here we demonstrate the use of an argument to choose a certain definition:

```
UVB()

## UVB.ISO
## low (nm) 280
## high (nm) 315

UVB("ISO")

## UVB.ISO
## low (nm) 280
## high (nm) 315
```

```
UVB("CIE")

## UVB.CIE
## low (nm) 280
## high (nm) 315

UVB("medical")

## UVB.medical
## low (nm) 290
## high (nm) 320

UVB("none")

## UVB.none
## low (nm) 280
## high (nm) 320
```

Here we demonstrate the importance of complying with standards, and how much the photon irradiance calculated can depend on the definition used.

```
with(sun.data,
     photon_irradiance(w.length, s.e.irrad, UVB("ISO"))) * 1e6

## UVB.ISO
##   1.527

with(sun.data,
     photon_irradiance(w.length, s.e.irrad, UVB("none"))) * 1e6

## UVB.none
##    3.282
```

6.6 Task: define simple wavebands

Here we briefly introduce `new_waveband`, and only in chapter ?? we describe its use in full detail, including the use of spectral weighting functions (SWFs).

Defining a new `waveband` based on extreme wavelengths expressed in nm.

```
wb1 <- new_waveband(500,600)
wb1

## range.500.600
## low (nm) 500
## high (nm) 600

with(sun.data,
     photon_irradiance(w.length, s.e.irrad, wb1)) * 1e6

## range.500.600
##         314.1

wb2 <- new_waveband(500,600, wb.name="my.colour")
wb2
```

```
## my.colour
## low (nm) 500
## high (nm) 600

with(sun.data,
     photon_irradiance(w.length, s.e.irrad, wb2)) * 1e6

## my.colour
##     314.1
```

6.7 Task: photon ratios

In photobiology sometimes we are interested in calculation the photon ratio between two wavebands. It makes more sense to calculate such ratios if both numerator and denominator wavebands have the same 'width' or if the numerator waveband is fully nested in the denominator waveband. However, frequently used ratios like the UV-B to PAR photon ratio do not comply with this. For this reason, our functions do not enforce any such restrictions.

For example a ratio frequently used in plant photobiology is the read to far-red photon ratio (R:FR photon ratio or ζ). If we follow the wavelength ranges in the definition given by **Morgan1981a** using photon irradiance[3]:

$$\zeta = \frac{Q_{655\text{nm}<\lambda<665\text{nm}}}{Q_{725\text{nm}<\lambda<735\text{nm}}} \tag{6.6}$$

To calculate this for our example sunlight spectrum we can use the following code:

```
with(sun.data,
     photon_ratio(w.length, s.e.irrad, Red("Smith"), Far_red("Smith")))

## [1] 1.251
```

or using the predefined convenience function `R_FR_ratio`:

```
with(sun.data,
     R_FR_ratio(w.length, s.e.irrad))

## [1] 1.251
```

Using defaults for waveband definitions:

```
with(sun.data,
     energy_ratio(w.length, s.e.irrad, UVB(), PAR()))

## [1] 0.00299
```

[3] In the original text photon fluence rate is used but it not clear whether photon irradiance was meant instead.

6.8 Task: energy ratios

An energy ratio, equivalent to ζ can be calculated as follows:

```
with(sun.data,
     energy_ratio(w.length, s.e.irrad, Red("Smith"), Far_red("Smith")))

## [1] 1.384
```

For this infrequently used ratio, no pre-defined function is provided.

6.9 Task: calculate average number of photons per unit energy

When comparing photo-chemical and photo-biological responses under different light sources it is of interest to calculate the photons per energy in $\mathrm{mol\,W^{-1}}$. In this case only one waveband definition is used to calculate the quotient:

$$\bar{q}' = \frac{Q_{\lambda_1<\lambda<\lambda_2}}{E_{\lambda_1<\lambda<\lambda_2}} \tag{6.7}$$

```
with(sun.data,
     photons_energy_ratio(w.length, s.e.irrad, PAR()))

## [1] 4.544e-06
```

For obtaining the same quotient in $\mathrm{\mu mol\,W^{-1}}$ we just need to multiply by 10^6. We can use such a multiplier to convert E [$\mathrm{W\,m^{-2}}$] into Q [$\mathrm{\mu mol\,m^{-2}\,s^{-1}}$], or as a divisor to convert Q [$\mathrm{\mu mol\,m^{-2}\,s^{-1}}$] into E [$\mathrm{W\,m^{-2}}$], *for a given light source and waveband*:

```
with(sun.data, photons_energy_ratio(w.length, s.e.irrad, PAR())) *
    1e+06

## [1] 4.544
```

6.10 Task: calculate the contribution of different regions of a spectrum to energy irradiance

It can be of interest to split the total (energy) irradiance into adjacent regions delimited by arbitrary wavelengths. We can use the function `split_energy_irradiance` to obtain to energy of each of the regions delimited by the values in nm supplied in a numeric vector:

```
with(sun.data, split_energy_irradiance(w.length, s.e.irrad, c(400,
    500, 600, 700)))

## range.400.500 range.500.600 range.600.700
##         69.63         68.53         58.54
```

Here we demonstrate that the sum of the four 'split' irradiances add to the total for the range of wavelengths covered:

```
with(sun.data, sum(split_energy_irradiance(w.length, s.e.irrad,
    c(400, 500, 600, 700))))

## [1] 196.7

with(sun.data, energy_irradiance(w.length, s.e.irrad, PAR()))

##   PAR
## 196.7
```

It also possible to obtain the 'split' as a vector of fractions adding up to one,

```
with(sun.data, split_energy_irradiance(w.length, s.e.irrad, c(400,
    500, 600, 700), scale = "relative"))

## range.400.500 range.500.600 range.600.700
##        0.3540        0.3484        0.2976
```

or as percentages:

```
with(sun.data, split_energy_irradiance(w.length, s.e.irrad, c(400,
    500, 600, 700), scale = "percent"))

## range.400.500 range.500.600 range.600.700
##         35.40         34.84         29.76
```

A vector of two wavelengths is valid input, although not very useful for percentages:

```
with(sun.data, split_energy_irradiance(w.length, s.e.irrad, c(400,
    700), scale = "percent"))

## range.400.700
##           100
```

Although for `scale="absolute"`, the default, it can be used as a quick way of calculating an irradiance for a range of wavelengths without having to define a `waveband`:

```
with(sun.data, split_energy_irradiance(w.length, s.e.irrad, c(400,
    700)))

## range.400.700
##         196.7
```

6.11 Task: calculate the contribution of different regions of a spectrum to photon irradiance

The function `split_photon_irradiance` takes the same arguments as the equivalent function for photon irradiance, consequently only one code example is provided here (see section 6.10 for more details):

```
with(sun.data, split_photon_irradiance(w.length, s.e.irrad, c(400,
    500, 600, 700), scale = "percent"))

## range.400.500 range.500.600 range.600.700
##         29.41         35.14         35.45
```

CHAPTER 7

Weighted and effective irradiance

Abstract

In this chapter we explain how to .

7.1 Task:

CHAPTER 8

Colour

Abstract

In this chapter we explain how to .

8.1 Task:

CHAPTER 9

Photoreceptors

Abstract

In this chapter we explain how to .

9.1 Task:

CHAPTER 10

Radiation sources

Abstract

In this chapter we explain how to .

10.1 Task:

CHAPTER 11

Filters

Abstract

In this chapter we explain how to .

11.1 Task:

CHAPTER 12

Plotting spectra

Abstract

In this chapter we explain how to .

12.1 Task:

CHAPTER 13

Calibration

Abstract

In this chapter we explain how to .

13.1 Task:

CHAPTER 14

Simulation

Abstract

In this chapter we explain how to .

14.1 Task:

CHAPTER 15

Measurement

Abstract

In this chapter we explain how to .

15.1 Task:

CHAPTER 16

Optimizing performance

Abstract

In this chapter we explain how to .

16.1 Task: