Package 'CoastalLight'

November 18, 2020

Version 0.7

Date 2020-11-09

Title Available Light on the Coastal Ocean [0-200m] Floor

Author Bernard Gentili <bernard.gentili@orange.fr>, Jean-Pierre Gattuso <gattuso@obs-vlfr.fr>

Maintainer Bernard Gentili <bernard.gentili@orange.fr>

Depends R (>= 3.3.0), ncdf4, raster, sp, viridis

Description Satellite data collected between 1998 and 2018 (SeaWiFS, MODIS, MERIS, VIIRS), in conjonction with GEBCO gridded bathymetric data (15 arc seconds), are used to estimate, at a nearly global scale, the irradiance reaching the bottom of the coastal ocean.

License GPL (>= 2)

URL https://doi.pangaea.de/10.1594/PANGAEA.910898, http://obs-vlfr.fr/Pfunction/, https://obs-vlfr.fr

RoxygenNote 6.1.1

R topics documented:

`CoastalLight-package` *CoastalLight: Available Light on the Coastal Ocean [0-200m] Floor*

Description

Satellite data collected between 1998 and 2018 (SeaWiFS, MODIS, MERIS, VIIRS), in conjonction with GEBCO gridded bathymetric data (15 arc seconds), are used to estimate, at a nearly global scale, the irradiance reaching the bottom of the coastal ocean.

Details

This package has **two goals** :

1. calculate the surface area of the sea floor which receives more than a given threshold of irradiance.
2. get various geographic and optical data from a database consisting of files to download

Goal 1 :

1. function `cl_surface()` is the main function; it returns the surface areas receiving irradiance above given thresholds, or the P-functions (see Gattuso et al. - reference below) for three standard regions; **NonPolar**, **Arctic**, and **Antarctic**; It is the only function to use if you are interested in one of these standard regions; P-functions for these regions have been tabulated and the tables included in the package.
2. function `cl_subregion()` is used to produce the P-functions of a **subregion of one of the three standard regions**; it connects to a server that does the calculations and offers to download a file containing the P-functions for this subregion; this file is ready to be used by the function `cl_surface()`.

Goal 2 :

1. function `cl_DownloadData()` downloads the data;
2. function `cl_GetData()` gets the data on a geographic zone defined by the user;
3. function `cl_PlotData()` plots the data, as returned by function `cl_GetData()`.
4. function `cl_Transect()` complements function `cl_GetData()` for the extraction of data along a transect.

Data and units :

- P-functions : $\%$
- longitude : $decimal\,degree$ $[-180; 180]$
- latitude : $decimal\,degree$ $[-90; 90]$
- depth : m
- area : km^2
- par, parbottom : $mol.photons\ m^{-2}\ d^{-1}$
- kdpar : m^{-1}

References

Gattuso J.-P., Gentili B., Antoine D. & Doxaran D., 2020. Global distribution of photosynthetically available radiation on the seafloor. Earth System Science Data 12:1697-1709. http://dx.doi.org/10.5194/essd-12-1697-2020

See Also

Useful links:

- https://doi.pangaea.de/10.1594/PANGAEA.910898
- http://obs-vlfr.fr/Pfunction/
- https://obs-vlfr.fr

`cl_DownloadData` *Download files from pangaea :*

`https://doi.pangaea.de/10.1594/PANGAEA.910898`

otherwise from

`http://obs-vlfr.fr/Pfunction/`

Description

Download files used by `cl_GetData()` function.

Usage

```
cl_DownloadData(month = 0, DownloadGeo = TRUE,
  dirdata = "CoastalLight.d", alt = 0)
```

Arguments

`month`	: [integer] decimal month (0-12). Default = 0. • 0 indicates the climatology over 21 years (1998-2018). • 1 to 12 indicates the decimal value for a single month (1 = January, ...);
`DownloadGeo`	: [logical] download the "geographic" file containing longitude, latitude, depth and surface area of the pixels, mandatory for retrieving pixel locations (default = TRUE)
`dirdata`	: [character] The directory where the files are stored (default = "./CoastalLight.d");
`alt`	: [integer] the way of downloading (0, 1 or 2). Default = 0 • 0 automatically connect to Pangaea database and download • 1 launch your default browser and connect for manual download to URL `http://obs-vlfr.fr/Pfunction/`

- 2 launch your default browser and connect for manual download to URL
  `https://doi.pangaea.de/10.1594/PANGAEA.910898`

N.B. : if you choose a manual download (alt = 1 or 2), you have to download files in the appropriate directory, i.e. the argument `dirdata` has no effect.

Value

: none

Examples

```
cl_DownloadData() # annual climatology over 21 years (1998-2018)
cl_DownloadData(month = 1) # monthly climatology (January) over 21 years (1998-2018)
cl_DownloadData(month = 8) # monthly climatology (August) over 21 years (1998-2018)
```

`cl_GetData` *Get Data*

Description

This function gets (geographic and optical) data for a geographic zone. Such a zone may be a single geographic point, a longitudinal transect, a latitudinal transect, or an area. The type of geographic zone depends the arguments `lon` and `lat` passed to the function (see details)

Usage

```
cl_GetData(lon, lat, what = c("depth", "area", "par", "kdpar",
  "parbottom"), dirdata = "CoastalLight.d", month = 0)
```

Arguments

`lon`	: [numeric] longitude, vector of length 1 or 2, (interval [-180; 180], and see details) ; unit : $decimal\ degree$
`lat`	: [numeric] latitude, vector of length 1 or 2, (interval [-90; 90], and see details) ; unit : $decimal\ degree$
`what`	: [character] a vector of the variables to extract among "depth", "area", "par", "kdpar" and "parbottom". Several variables can be listed. Default is all of them. Note that longitude and latitude are automatically added to the variables. Units : depth (m), area (km^2), par, parbottom ($mol.photons\ m^{-2}\ d^{-1}$), kdpar ($m^{-1}$).
`dirdata`	: [character] The directory where the data files (previouly downloaded with the function `cl_DownloadData()`) are stored. (default = "./CoastalLight.d");
`month`	: [integer] : the decimal value of the month ([0-12] of interest. 0 indicate the climatology over 21 years (1998-2018), 1 is January, ...

Details

There are 4 options for parameters `lon` and `lat`:

- `lon` is of length 1, `lat` is of length 1 : the type is "Point"
- `lon` is of length 2, `lat` is of length 1 : the type is "LonTransect"
- `lon` is of length 1, `lat` is of length 2 : the type is "LatTransect"
- `lon` is of length 2, `lat` is of length 2 : the type is "Area"

To get data along an ordinary transect (i.e. a polygonal line, given by its vertices) you have, in a first time, to get data in an "Area" containing this polygonal line, and, in a second time, to use function `cl_Transect()` in order to extract data along the transect (see example of this function).

Value

: [list] a list with 4 components :

- `type` : the type of geographical zone : "Point", "LonTransect", "LatTransect" or "Area"
- `lon` : the lon argument as passed to the function
- `lat` : the lat argument as passed to the function
- `data` : a matrix of data with columns names "longitude", "latitude", and variables requested.

Examples

```
## All examples assume that data have been downloaded in directory "CoastalLight.d"
## with function cl_DownloadData()

## Area
gabes <- cl_GetData(lon = c(10, 14), lat = c(32.5, 36), dir = "./CoastalLight.d")
par(mfrow = c(3,2))
cl_PlotData(gabes)

## a longitudinal transect in January
long.transect <- cl_GetData(lon = c(10, 14), lat = c(34), dir = "./CoastalLight.d", month = 1)
par(mfrow = c(1,1))
cl_PlotData(long.transect)

## a latitudinal transect in August
lat.transect <- cl_GetData(lon = c(12), lat = c(32.5, 36), dir = "./CoastalLight.d", month = 8)
par(mfrow = c(1,1))
cl_PlotData(lat.transect)
```

`cl_PlotData` *Plots the data returned by function* `cl_GetData()` *or function* `cl_Transect()`

Description

Functions `cl_GetData()` or `cl_Transect()` return a list; plot differ depending on the field `type` of this list :

"Point" no plot

"LonTransect" variables are plotted vs longitude

"LatTransect" variables are plotted vs latitude

"Area" geographic maps of the variables are produced

"Transect" variables are plotted vs the distance traveled from the first vertex of the polygonal line

Usage

```
cl_PlotData(x, vertices = NULL)
```

Arguments

`x`	: [list] a list returned by function `cl_GetData()`
`vertices`	: [list] a polygonal line that represents a transect; each element of this list is a geographic point, i.e. a vector of length 2 (longitude, latitude); default is NULL; this argument is only used if type is "Area" in order to add a transect to the maps. N.B. : function `cl_Transect()` has an optional argument (`plt`) to plot the transect on the maps.

Value

- a rasterstack with all variables if type is "Area"
- NULL otherwise

Examples

```
## See function \code{cl_GetData()} examples
```

cl_subregion *Download a P-function for a subregion*

Description

Connects to a server in order to calculate and download a P-function for a subregion of one of the three standard regions

Usage

```
cl_subregion(browse = TRUE)
```

Arguments

browse	: [logical] do you want to connect to the server with your default browser ? (default = TRUE). If FALSE no connection, but the url is returned (you may connect to this url, independently of the package).

Value

: [character] the url ("http://obs-vlfr.fr/Pfunction/")

Examples

```
cl_subregion()
```

cl_surface *Returns surfaces and P-function values*

Description

This function is the main function of the package. It operates on the three standard regions "NonPolar", "Arctic", and "Antarctic" or on a subregion (see user's guide and function cl_subregion())

Usage

```
cl_surface(region, month = 0, E = 1.6, type = "s", dir = ".")
```

Arguments

region	: [character] choose among : "NonPolar", "Arctic" and "Antarctic" or give the name of a subregion of one of these three regions. For example : suppose you have calculated the Pfunction of a subregion called "med" from the main region "NonPolar". You have downloaded a file called "NonPolar.med.Pfunctions.dat" (see function cl_subregion()). In this case pass "NonPolar.med".

month	: [integer] the month (-1, 0, 1-12); if -1 gives the values for global and all months; if 0 gives the global value; if 1 < month < 12 gives the values for the month;
E	: [numeric] $mol.photons\ m^{-2}\ d^{-1}$
type	: [character] choose among : "Sg" "s" "P" "E" (default "s")
dir	: [character] only used if region is not one of the three standard regions but a subregion; in this case, it is the path to the directory where is stored the Pfunction file (example : "NonPolar.med.Pfunctions.dat") of this subregion (default = ".", that is the current directory)

Details

This function uses the Pfunction file :

- included in the package if region is "NonPolar" "Arctic" or "Antarctic"
- previously calculated and downloaded if region is the name of a subregion.

Value

if type == "Sg" [numeric] : an unique value (surface area of the region in km^2)

if type == "s" [numeric vector] : surface areas in km^2 receiving more than E $mol.photons\ m^{-2}\ d^{-1}$

if type == "P" [numeric array] : values of the P-function in % for the E values

if type == "E" [numeric vector] : discrete values of E used for tabulated P-functions

Examples

```
## surface of the "NonPolar" region
cl_surface("NonPolar", type = "Sg")

## global P-function
cl_surface("NonPolar", month = 0, E = c(0.01,0.02,0.05,0.1,0.2,0.5,1), type = "P")

## get E values in "Arctic" region then computes and plot global and monthly P-functions
region <- "Arctic"
E <- cl_surface(region, type = "E")
pc <- cl_surface(region, month = -1, E = E, type = "P")
matplot(E, pc, type = "l", log = "x", xlim = rev(range(E)),
        lty = 1:5, col = 1:6, lwd = 2, ylab = "%", main = region)
legend("topleft", legend = colnames(pc),
        lty = 1:5, col = 1:6, lwd = 2, bty = "n")

## example for a subregion
region <- "NonPolar.Gabes"
## for this example the directory where to find the P-function file
##   "NonPolar.Gabes.Pfunctions.dat" is in the package itself;
##   you can obtain your own P-function files and save them in
##   the directory of your choice - see function cl_subregion()
dir <- system.file("extdata", package = "CoastalLight")
E <- cl_surface(region, type = "E", dir = dir)
```

```
pc <- cl_surface(region, month = -1, E = E, type = "P", dir = dir)
matplot(E, pc, type = "l", log = "x", xlim = rev(range(E)),
        lty = 1:5, col = 1:6, lwd = 2, ylab = "%", main = region)
legend("topleft", legend = colnames(pc),
        lty = 1:5, col = 1:6, lwd = 2, bty = "n")
```

`cl_Transect` *Extracts a transect from data returned by function* `cl_GetData()`

Description

When it is operated with type "Area" function `cl_GetData()` returns data at all available points of a geographic area; function `cl_Transect()` extract a transect inside this area in 2 steps;

- rasterization of the data
- extraction of the transect

output of this function has the same structure as output of function `cl_GetData()` and may be plotted by function `cl_PlotData()`

Usage

```
cl_Transect(x, vertices, plt = FALSE)
```

Arguments

`x`	: [list] a list returned by function `cl_GetData()` whose field `type` has value "Area"
`vertices`	: [list] a polygonal line that represents a transect; each element of this list is a geographic point, i.e. a vector of length 2 (longitude, latitude);
`plt`	: [logical] make a plot of the transect on the maps of the variables (default is FALSE)

Value

: [list] a list with 5 components :

- the same 4 components as returned by function `cl_GetData()` :
  - `type` : the type of geographical zone; its value is "Transect"
  - `lon` : identical to component `lon` of x
  - `lat` : identical to component `lat` of x
  - `data` : a matrix of data along the transect with columns names "longitude", "latitude", and variables requested.
- a fifth component `distances` which represents the distance traveled from the first vertex of the polygonal line

Examples

```
gabes <- cl_GetData(lon = c(10, 14), lat = c(32.5, 36), dir = "./CoastalLight.d")
vertices <- list(c(10.1,33), c(13.5,35), c(13,34))
#X11()
par(mfrow = c(3, 2))
tr <- cl_Transect(gabes, vertices, plt = TRUE)
cl_PlotData(tr)
```

Index