

Package ‘ebvcube’

December 14, 2021

Title Working with netCDF for Essential Biodiversity Variables

Version 0.0.1

Date 2021-12-01

Author Luise Quöß [aut, cre],
Christian Langer[aut],
Henrique Pereira [aut],
Néstor Fernández [aut],
José Valdez [aut]

Maintainer Luise Quöß <luise.quoss@idiv.de>

Description Functions to easily access the data of the EBV netCDFs which can be downloaded here: <portal.geobon.org>. Also some basic visualization of the data is provided. Advanced users can build their own netCDFs with the EBV standard using this package.

URL <<https://portal.geobon.org/>> <<https://geobon.org/ebvs/what-are-ebvs/>> <<https://github.com/LuiseQuoss/ebvcube>>

BugReports <<https://github.com/LuiseQuoss/ebvcube/issues>>

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports checkmate,
colorspace,
gdalUtils,
graphics,
HDF5Array,
jsonlite,
lattice,
memuse,
methods,
ncdf4,
raster,
rhdf5,
rgdal,
sp,
stats,
stringr,

utils,
withr,
ncmeta

Suggests knitr,
rmarkdown

SystemRequirements GDAL binaries

Depends R (>= 3.5.0)

VignetteBuilder knitr, rmarkdown

R topics documented:

EBV netCDF properties-class	2
ebvcube	3
ebv_add_data	3
ebv_analyse	4
ebv_attribute	6
ebv_create	7
ebv_datacubePaths	8
ebv_indicator	9
ebv_map	10
ebv_properties	11
ebv_read	12
ebv_read_bb	13
ebv_read_shp	15
ebv_resample	16
ebv_write	18
world_boundaries	19
Index	20

EBV netCDF properties-class
EBV netCDF properties class (S4)

Description

EBV netCDF properties class (S4)

Value

S4 class containing the EBV netCDF properties

Slots

general Named list. Elements: title, description, ebv_class, ebv_name, ebv_domain, references, source, project, creator_name, creator_institution, creator_email, contributor_name, publisher_name, publisher_institution, publisher_email, comment, keywords, id, history, licence, conventions, naming_authority, date_created, date_issued, entity_names, entity_type, entity_scope, entity_classification_name, entity_classification_url

spatial Named list. Elements: wkt2, epsg, extent, resolution, crs_units, dimensions, scope, description

temporal Named list. Elements: resolution, units, timesteps, timesteps_natural

metric Named list. Elements: name, description

scenario Named list. Elements: name, description

ebv_cube Named list. Elements: units, coverage_content_type, fillvalue, type

Note

If the properties class holds e.g. no scenario information this is indicated with an element called status in the list.

If you read an EBV netCDF based on an older standard, the properties will differ from the definition above.

ebvcube

Working with netCDF for Essential Biodiversity Variables

Description

This package can be used to easily access the data of the EBV netCDFs which can be downloaded from the [Geobon Portal](#). It also provides some basic visualization of the data. Advanced users can build their own netCDFs with the EBV standard using this package.

Details

This package contains three main usecases: accessing the data, visualising it and creating your own data in the EBV netCDF standard.

ebv_add_data

Add data to a self-created EBV netCDF

Description

Add data to the self-created EBV netCDF from GeoTiffs.

Usage

```
ebv_add_data(
  filepath_nc,
  datacube_path,
  entity = NULL,
  timestep = 1,
  filepath_tif,
  band = 1,
  ignore_RAM = FALSE,
  verbose = FALSE
)
```

Arguments

filepath_nc	Character. Path to the self-created netCDF file.
datacubeopath	Character. Path to the datacube (use ebv_datacubeopaths()).
entity	Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.
timestep	Integer. Default: 1. Define to which timestep or timesteps the data should be added. If several timesteps are given they have to be in a continuous order. Meaning c(4,5,6) is right but c(2,5,6) is wrong.
filepath_tif	Character. Path to the GeoTiff file containing the data. Ending needs to be *.tif.
band	Integer. Default: 1. Define which band(s) to read from GeoTiff. Can be several. Don't have to be in order as the timesteps definition requires.
ignore_RAM	Logical. Default: FALSE. Checks if there is enough space in your memory to read the data. Can be switched off (set to TRUE).
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Adds data to the EBV netCDF. Check your results using [ebv_read\(\)](#) and/or [ebv_analyse\(\)](#).

Note

If the data exceeds your memory the RAM check will throw an error. No block-processing or other method implemented so far. Move to a machine with more capacities for the moment if needed.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacubeopaths of EBV netCDF
datacubes <- ebv_datacubeopaths(file)
#set path to GeoTiff with data
tif <- system.file(file.path("extdata","cSAR_write_ts234.tif"), package="ebvcube")

# add data to the timestep 2, 3 and 4 using the first three bands of the GeoTiff
#ebv_add_data(filepath_nc = file, datacubeopath = datacubeopaths[1,1],
#             entity = NULL, timestep = 2:4, filepath_tif = tif, band = 1:3)
```

ebv_analyse

Get a simple explorative analysis of an EBV netCDF datacube

Description

Get basic measurements of the data, including min, max, mean, sd, n, #NAs, q25, q50, q75 (no mean for categorical data).

Usage

```
ebv_analyse(
  filepath,
  datacubepath,
  entity = NULL,
  timestep = 1,
  subset = NULL,
  at = TRUE,
  epsg = 4326,
  numerical = TRUE,
  na_rm = TRUE,
  verbose = FALSE
)
```

Arguments

filepath	Character. Path to the netCDF file.
datacubepath	Character. Path to the datacube (use ebv_datacubepaths()).
entity	Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.
timestep	Integer. Choose one or several timesteps (vector).
subset	Optional if you want measurements on a smaller subset. Possible via the path to a shapefile (character) or the indication of a bounding box (vector of four numeric values) defining the subset. Else the whole area is analysed.
at	Logical. Optional. Default: TRUE. Only relevant if the subset is indicated by a shapefile. See ebv_read_shp() .
epsg	Numeric. Optional. Only relevant if the subset is indicated by a bounding box and the coordinate reference system differs from WGS84. See ebv_read_bb() .
numerical	Logical. Default: TRUE. Change to FALSE if the data covered by the netCDF contains categorical data.
na_rm	Logical. Default: TRUE. NA values are removed in the analysis. Change to FALSE to include NAs.
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Returns a named list containing the measurements.

See Also

[ebv_read_bb\(\)](#) and [ebv_read_shp\(\)](#) for the usage of subsets.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacubepaths of EBV netCDF
datacubes <- ebv_datacubepaths(file)
#set path to shp file
```

```

shp_path <- system.file(file.path("extdata","subset_germany.shp"), package="ebvcube")

#get measurements for full extent and all timesteps
data_global <- ebv_analyse(filepath = file, datacube_path = datacubes[1,1],
                           entity = NULL, timestep = 1:12)

#get measurements for germany only (using bounding box) and one timestep
# data_bb_1900 <- ebv_analyse(filepath = file, datacube_path = datacubes[1,1],
#                               entity = NULL, timestep = 1:12, subset = c(5,15,47,55))

#get measurements for germany only (using shp) and one timestep
# data_shp_1900 <- ebv_analyse(filepath = file, datacube_path = datacubes[1,1],
#                               entity = NULL, timestep = 1:12, subset = shp_path)

```

ebv_attribute

Write a new attribute value to an EBV netCDF

Description

Write a new attribute value to an EBV netCDF. Not all attributes can be changed. Some are always created automatically, e.g. the attributes belonging to the crs, time and var_entity datasets. In this case you have to re-create the netCDF file.

Usage

```

ebv_attribute(
  filepath,
  attribute_name,
  value,
  levelpath = NULL,
  verbose = FALSE
)

```

Arguments

filepath	Character. Path to the netCDF file.
attribute_name	Character. Name of the attribute that should be changed.
value	New value that should be assigned to the attribute.
levelpath	Character. Default: NULL. Indicates the location of the attribute. The default means that the attribute is located at a global level. If the attribute is located at the datacube level just add the datacube_path, e.g. metric_1/ebv_cube. For the metric level the value may be 'metric_1' or 'scenario_1/metric_1'. This path depends on whether the netCDF hierarchy has scenarios or not.
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Adds the new value to the attribute. Check your results using [ebv_properties\(\)](#).

Note

You can change the ebv_class and the ebv_name. In this case you need to change the ebv_class first. Don't forget to change the ebv_name accordingly!

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")

#change the standard_name of the metric
attribute1 <- 'standard_name'
value1 <- 'new fake metric name'
level1 <- 'scenario_1/metric_1'
# ebv_attribute(filepath = file, attribute_name = attribute1,
#               value = value1, level = level1)

#change the units of the ebv_cube
attribute2 <- 'units'
value2 <- 'mean'
level2 <- 'scenario_1/metric_1/ebv_cube' #equal to the datacube path
# ebv_attribute(filepath = file, attribute_name = attribute2,
#               value = value2, level = level2)

#change the name of the creator at the global level
attribute3 <- 'creator_name'
value3 <- 'Jane Doe'
# ebv_attribute(filepath = file, attribute_name = attribute3,
#               value = value3)
```

ebv_create

Create an EBV netCDF

Description

Create the core structure of the EBV NetCDF based on the json from the [Geobon Portal API](#). Data and attributes will be added afterwards. Use [ebv_add_data\(\)](#) to add the missing attributes.

Usage

```
ebv_create(
  jsonpath,
  outputpath,
  entities,
  epsg = 4326,
  extent = c(-180, 180, -90, 90),
  resolution = c(1, 1),
  fillvalue = NULL,
  prec = "double",
  force_4D = TRUE,
  overwrite = FALSE,
  verbose = FALSE
)
```

Arguments

jsonpath	Character. Path to the json file downloaded from the Geobon Portal API .
outputpath	Character. Set path where the NetCDF file should be created.

entities	Character. Csv table holding the entity names. Should have only one column, each row is the name of one entity.
epsg	Integer. Default: 4326 (WGS84). Defines the coordinate reference system via the corresponding epsg code.
extent	Numeric. Default: c(-180,180,-90,90). Defines the extent of the data: c(xmin, xmax, ymin, ymax).
resolution	Numerical. Vector of two numerical values defining the longitudinal and latitudinal resolution of the pixel: c(lon,lat).
fillvalue	Numeric. Value of the missing data in the array. Not mandatory but should be defined!
prec	Character. Default: 'double'. Precision of the data set. Valid options: 'short' 'integer' 'float' 'double' 'char' 'byte'.
force_4D	Logical. Default is TRUE. If the argument is TRUE, there will be 4D cubes (lon, lat, time, entity) per metric. If this argument is changed to FALSE, there will be 3D cubes (lon, lat, time) per entity (per metric). So the latter yields a higher amount of cubes and does not bundle all information per metric. In the future the standard will be restricted to the 4D version. Recommendation: go with the 4D cubes!
overwrite	Logical. Default: FALSE. Set to TRUE to overwrite the outputfile defined by 'outputpath'.
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Creates the NetCDF file at the 'outputpath' location.

Note

To check out the results take a look at your netCDF file with [Panoply](#) provided by the NASA.

Examples

```
#set path to JSON file
json <- system.file(file.path("extdata","metadata.json"), package="ebvcube")
#set output path of the new EBV netCDF
out <- file.path(system.file(package='ebvcube'),"extdata","sCAR_new.nc")
#set path to the csv holding the entity names
entities <- file.path(system.file(package='ebvcube'),"extdata","entities.csv")

#create new EBV netCDF
# ebv_create(jsonpath = json, outputpath = out, entities = entities,
#           fillvalue=-3.4E38)
```

ebv_datacubepaths	<i>Get datacubepaths of EBV netCDF</i>
-------------------	--

Description

Get the paths to the datacubes of the EBV netCDF to access the data.

Usage

```
ebv_datacubepaths(filepath, verbose = FALSE)
```

Arguments

filepath Character. Path to the netCDF file.

verbose Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Dataframe containing the paths to access the datacubes and descriptions of scenario, metric and entity if existing.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")

#get all datacubepaths of EBV netCDF
datacubes <- ebv_datacubepaths(file)
```

ebv_indicator	<i>Plot the average over time of one datacube of an EBV NetCDF</i>
---------------	--

Description

Plot the average (y-axis) of one datacube of a EBV NetCDF over time (x-axis). If the datacube has only one timestep a single mean value is returned.

Usage

```
ebv_indicator(
  filepath,
  datacubepath,
  entity = NULL,
  color = "dodgerblue4",
  verbose = FALSE
)
```

Arguments

filepath Character. Path to the NetCDF file.

datacubepath Character. Path to the datacube (use [ebv_datacubepaths\(\)](#)).

entity Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.

color Character. Default: dodgerblue4. Change to any color known by R [grDevices::colors\(\)](#)

verbose Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Plots a line plot and returns a vector of the average. If the data encompasses only one timestep a single mean is returned.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#plot the change of the mean over time of the first datacube
ebv_indicator(filepath = file, datacube_path = datacubes[1,1], entity = NULL)
```

ebv_map	<i>Map plot of an EBV NetCDF</i>
---------	----------------------------------

Description

Map plot of the data of one timestep in one datacube of an EBV NetCDF. This function sometimes writes temporary files on your disk. Specify a directory for these settings via `options('ebv_temp'='/path/to/temp/directory')`.

Usage

```
ebv_map(
  filepath,
  datacube_path,
  entity = NULL,
  timestep = 1,
  countries = TRUE,
  col_rev = TRUE,
  classes = 5,
  ignore_RAM = FALSE,
  verbose = FALSE
)
```

Arguments

filepath	Character. Path to the NetCDF file.
datacube_path	Character. Path to the datacube (use ebv_datacube_paths()).
entity	Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.
timestep	Integer. Choose one timestep.
countries	Logical. Default: TRUE. Simple country outlines will be plotted on top of the raster data. Disable by setting this option to FALSE.
col_rev	Logical. Default: TRUE. Set to FALSE if you want the color ramp to be the other way around.
classes	Integer. Default: 5. Define the amount of classes (quantiles) for the symbology. Currently restricted to maximum 15 classes.

ignore_RAM	Logical. Default: FALSE. Checks if there is enough space in your memory to read the data. Can be switched off (set to TRUE).
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Plots a map.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#plot a map for the 9th timestep, divide into 7 classes
ebv_map(filepath = file, datacube_path = datacubes[1,1], entity = NULL,
        timestep = 9, classes = 7)
```

ebv_properties	<i>Read properties of EBV netCDF</i>
----------------	--------------------------------------

Description

Structured access to all attributes of the netCDF file.

Usage

```
ebv_properties(filepath, datacube_path = NULL, verbose = FALSE)
```

Arguments

filepath	Character. Path to the netCDF file.
datacube_path	Character. Optional. Path to the datacube (use ebv_datacube_paths()).
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

S4 class containing information about file or file and datacube depending on input.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#get properties only for the file
prop_file <- ebv_properties(file)
#get properties for the file and a specific datacube
prop_dc <- ebv_properties(file, datacubes[1,1])
```

ebv_read

Read data from an EBV netCDF

Description

Read one or more layers from one datacube of the netCDF file. Decide between in-memory array, in-memory raster or an array-like object (DelayedMatrix) pointing to the on-disk netCDF file. Latter is useful for data that exceeds your memory.

Usage

```
ebv_read(
    filepath,
    datacube_path,
    entity = NULL,
    timestep = 1,
    type = "a",
    sparse = FALSE,
    ignore_RAM = FALSE,
    verbose = FALSE
)
```

Arguments

filepath	Character. Path to the netCDF file.
datacube_path	Character. Path to the datacube (use ebv_datacube_paths()).
entity	Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.
timestep	Integer. Choose one or several timesteps (vector).
type	Character. Choose between 'a', 'r' and 'da'. The first returns an array or matrix object. The 'r' indicates raster as return class. The latter returns a DelayedArray object.
sparse	Logical. Default: FALSE. Set to TRUE if the data contains a lot empty raster cells. Only relevant for DelayedMatrix. No further implementation by now.
ignore_RAM	Logical. Default: FALSE. Checks if there is enough space in your memory to read the data. Can be switched off (set to TRUE).
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Array, Raster or DelayedMatrix object containing the data of the corresponding datacube and timestep(s).

Note

For working with the DelayedMatrix take a look at [DelayedArray::DelayedArray\(\)](#) and the [DelayedArray-utils](#).

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#read data as DelayedArray
cSAR.delayedarray <- ebv_read(filepath = file, datacube_path = datacubes[1,1],
                             entity = NULL, timestep = c(1,6), type='da',
                             sparse=TRUE)

#read data as Raster
cSAR.raster <- ebv_read(filepath = file, datacube_path = datacubes[1,1],
                        entity = NULL, timestep = 1:3, type='r')

#read data as Array
cSAR.array <- ebv_read(filepath = file, datacube_path = datacubes[1,1],
                       entity = NULL, timestep = 1, type='r')
```

ebv_read_bb

Read subset (bounding box) of one datacube of an EBV netCDF

Description

Read a subset of one or more layers from one datacube of the NetCDF file. Subset definition by a bounding box.

Usage

```
ebv_read_bb(
  filepath,
  datacube_path,
  entity = NULL,
  timestep = 1,
  bb,
  outputpath = NULL,
  epsg = 4326,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = FALSE
)
```

Arguments

filepath	Character. Path to the netCDF file.
datacube_path	Character. Path to the datacube (use ebv_datacube_paths()).
entity	Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.
timestep	Integer. Choose one or several timesteps.
bb	Integer Vector. Definition of subset by bounding box: c(xmin, xmax, ymin, ymax).

outputpath	Character. Default: NULL, returns the data as a raster object in memory. Optional: set path to write subset as GeoTiff on disk.
epsg	Integer. Default: 4326 (WGS84). Change accordingly if your bounding box coordinates are based on a different coordinate reference system.
overwrite	Logical. Default: FALSE. Set to TRUE to overwrite the outputfile defined by 'outputpath'.
ignore_RAM	Logical. Default: FALSE. Checks if there is enough space in your memory to read the data. Can be switched off (set to TRUE).
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Returns a raster object if no outputpath is given. Otherwise the subset is written onto the disk and the outputpath is returned.

Note

In case the epsg of the Bounding Box and the netCDF differ, the data is returned based on the epsg of the netCDF Dataset.

See Also

[ebv_read_shp\(\)](#) for subsetting via shapefile.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#set outputpath
out <- file.path(system.file(package='ebvcube'), "extdata", "subset_bb.tif")
#define two different bounding boxes based on different EPSG codes
bb_wgs84 <- c(5,15,47,55)
bb_utm32 <- c(271985, 941837, 5232640, 6101151)

print(gdalUtils::gdalSrsInfo(paste0("EPSG:", 32632)))

#read bb (based on EPSG 4326) - return Raster
# cSAR.germany <- ebv_read_bb(filepath = file, datacube_path = datacubes[1,1],
#                               entity = NULL, timestep = c(1,4,12), bb = bb_wgs84)

#read bb (based on EPSG 4326) - write to GeoTiff
# path <- ebv_read_bb(filepath = file, datacube_path = datacubes[1,1],
#                      entity = NULL, timestep = 1, bb = bb_wgs84,
#                      outputpath = out, overwrite = TRUE)

#read bb (based on EPSG 32632) - write to GeoTiff
# path <- ebv_read_bb(filepath = file, datacube_path = datacubes[1,1],
#                      entity = NULL, timestep = 1:3, bb = bb_utm32,
#                      epsg = 32632, outputpath = out, overwrite = TRUE)
```

ebv_read_shp

*Read subset (shapefile) of one datacube of an EBV netCDF***Description**

Read a subset of one or more layers from one datacube of the netCDF file. Subset definition by a shapefile. This functions writes temporary files on your disk. Specify a directory for these setting via options('ebv_temp'='/path/to/temp/directory').

Usage

```
ebv_read_shp(
  filepath,
  datacubepath,
  entity = NULL,
  timestep = 1,
  shp,
  outputpath = NULL,
  at = TRUE,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = FALSE
)
```

Arguments

filepath	Character. Path to the netCDF file.
datacubepath	Character. Path to the datacube (use ebv_datacubepaths()).
entity	Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.
timestep	Integer. Choose one or several timesteps (vector).
shp	Character. Path to the shapefile defining the subset. Ending needs to be *.shp.
outputpath	Character. Default: NULL, returns the data as a raster object in memory. Optional: set path to write subset as GeoTiff on disk.
at	Logical. Default: TRUE, all pixels touched by the polygon(s) will be updated. Set to FALSE to only include pixels that are on the line render path or have center points inside the polygon(s).
overwrite	Logical. Default: FALSE. Set to TRUE to overwrite the outputfile defined by 'outputpath'.
ignore_RAM	Logical. Default: FALSE. Checks if there is enough space in your memory to read the data. Can be switched off (set to TRUE).
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Returns a raster object if no outputpath is given. Otherwise the subset is written onto the disk and the outputpath is returned.

See Also

[ebv_read_bb\(\)](#) for subsetting via bounding box.

Examples

```
#define temp directory
options('ebv_temp'=system.file("extdata/", package="ebvcube"))
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#set path to shp file
shp_path <- system.file(file.path("extdata","subset_germany.shp"), package="ebvcube")

#read subset - return Raster
# cSAR.germany <- ebv_read_shp(filepath = file, datacube_path = datacubes[1],
#                               entity = NULL, timestep = 1, shp = shp_path,
#                               outputpath = NULL)
```

ebv_resample

Change the resolution of the data of an EBV NetCDF

Description

Change the resolution of one datacube of a EBV NetCDF based on another EBV NetCDF or a given resolution. This function writes temporary files on your disk. Specify a directory for these settings via `options('ebv_temp'='/path/to/temp/directory')`.

Usage

```
ebv_resample(
  filepath_src,
  datacube_path_src,
  entity_src = NULL,
  timestep_src = 1,
  resolution,
  outputpath,
  method = "average",
  return_raster = FALSE,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = FALSE
)
```

Arguments

`filepath_src` Character. Path to the NetCDF file whose resolution should be changed.

`datacube_path_src` Character. Path to the datacube (use [ebv_datacube_paths\(\)](#)) whose resolution should be changed.

entity_src	Character or Integer. Default is NULL. If the structure is 3D, the entity argument is set to NULL. Else, a character string or single integer value must indicate the entity of the 4D structure of the EBV netCDFs.
timestep_src	Integer. Choose one or several timesteps (vector).
resolution	Either the path to an EBV NetCDF file that determines the resolution (character) or the resolution defined directly (numeric). The vector defining the resolution directly must contain three elements: the x-resolution, the y-resolution and the corresponding epsg.
outputpath	Character. Set path to write data as GeoTiff on disk.
method	Character. Default: Average. Define resampling method. Choose from: "near", "bilinear", "cubic", "cubic_spline" and "q3". For detailed information see: gdalwarp .
return_raster	Logical. Default: FALSE. Set to TRUE to directly get the corresponding raster object.
overwrite	Logical. Default: FALSE. Set to TRUE to overwrite the outputfile defined by 'outputpath'.
ignore_RAM	Logical. Default: FALSE. Checks if there is enough space in your memory to read the data. Can be switched off (set to TRUE).
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Default: returns the outputpath of the GeoTiff with the new resolution. Optional: return the raster object with the new resolution.

Examples

```
#define temp directory
options('ebv_temp'=system.file("extdata/", package="ebvcube"))
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#define different resolutions
res1 <- system.file(file.path("extdata","rodinini_001.nc"), package="ebvcube")
res2 <- c(1,1,4326)
#define output path
out <- file.path(system.file(package='ebvcube'), "extdata", "changeRes.tif")

#resample using a netCDF file - return GeoTiff
# ebv_resample(filepath_src = file, datacube_path_src = datacubes[1,1],
#               entity_src=NULL, timestep_src = 1, resolution = res1,
#               outputpath = out)

#resample defining the resolution and EPSG code by hand - return Raster
# data_raster <- ebv_resample(filepath_src = file, datacube_path_src = datacubes[1,1],
#                             entity_src=NULL, timestep_src = 1, resolution = res1,
#                             outputpath = out, method='max', return_raster=TRUE)
```

ebv_write

Write the extracted data on your disk as a GeoTiff

Description

After you extracted data from the EBV netCDF and worked with it this function gives you the possibility to write it to disk as a GeoTiff. This functions writes temporary files on your disk. Specify a directory for these setting via options('ebv_temp'='/path/to/temp/directory').

Usage

```
ebv_write(
  data,
  outputpath,
  epsg = 4326,
  extent = c(-180, 180, -90, 90),
  type = "FLT8S",
  overwrite = FALSE,
  verbose = FALSE
)
```

Arguments

data	Your data object. May be raster, array, DelayedMatrix or list of DelayedMatrix (see return values of ebv_read())
outputpath	Character. Set the path where you want to write the data to disk as a GeoTiff. Ending needs to be *.tif.
epsg	Integer. Default: 4326 (WGS84). Defines the coordinate reference system via the corresponding epsg code.
extent	Numeric. Default: c(-180,180,-90,90). Defines the extent of the data: c(xmin, xmax, ymin, ymax).
type	Character. Default is FLT8S Indicate the datatype of the GeoTiff file. Possible values: LOG1S, INT1S, INT1S, INT2S, INT2U, INT4S, INT4U, FLT4S, FLT8S.
overwrite	Logical. Default: FALSE. Set to TRUE to overwrite the outputfile defined by 'outputpath'.
verbose	Logical. Default: FALSE. Turn on all warnings by setting it to TRUE.

Value

Returns the outputpath.

Note

Not yet implemented for subsets of the data (only whole spatial coverage of the corresponding EBV netCDF).

For more info on the datatype definition see [raster::dataType\(\)](#).

Examples

```
#define temp directory
options('ebv_temp'=system.file("extdata/", package="ebvcube"))
#set path to EBV netCDF
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file)

#read data
data <- ebv_read(filepath = file, datacube_path = datacubes[1,1], timestep = 1)
# HERE YOU CAN WORK WITH YOUR DATA

#write data to disk as GeoTiff
out <- system.file(file.path("extdata","write_data.tif"), package="ebvcube")
# ebv_write(data = data, outputpath = out)

#read a subset
# data_bb <- ebv_read_bb(filepath = file, datacube_path = datacubes[1,1],
#                        entity = NULL, timestep = 1:3, bb = c(5,15,47,55))

#write subset to disk as GeoTiff
# ebv_write(data = data_bb, outputpath = out, extent = c(5,15,47,55), overwrite = TRUE)
```

world_boundaries

Simple outlines of world countries

Description

Simple outlines of world countries

Usage

```
world_boundaries
```

Format

A Spatial Polygons Data Frame with 177 elements

Source

Data downloaded from [Natural Earth](#). Used version 4.0.0 and reduced attributes.

Index

* datasets

world_boundaries, [19](#)

DelayedArray::DelayedArray(), [12](#)

EBV netCDF properties-class, [2](#)

ebv_add_data, [3](#)

ebv_add_data(), [7](#)

ebv_analyse, [4](#)

ebv_analyse(), [4](#)

ebv_attribute, [6](#)

ebv_create, [7](#)

ebv_datacubePaths, [8](#)

ebv_datacubePaths(), [4](#), [5](#), [9–13](#), [15](#), [16](#)

ebv_indicator, [9](#)

ebv_map, [10](#)

ebv_properties, [11](#)

ebv_properties(), [6](#)

ebv_read, [12](#)

ebv_read(), [4](#), [18](#)

ebv_read_bb, [13](#)

ebv_read_bb(), [5](#), [16](#)

ebv_read_shp, [15](#)

ebv_read_shp(), [5](#), [14](#)

ebv_resample, [16](#)

ebv_write, [18](#)

ebvcube, [3](#)

grDevices::colors(), [9](#)

raster::dataType(), [18](#)

world_boundaries, [19](#)