Package 'ebvcube'

June 11, 2024

Title Working with netCDF for Essential Biodiversity Variables

Version 0.2.1

Date 2024-06-11

Author Luise Quoss [aut, cre] (<https://orcid.org/0000-0002-9910-1252>),
Nestor Fernandez [aut] (<https://orcid.org/0000-0002-9645-8571>),
Christian Langer [aut] (<https://orcid.org/0000-0003-1446-3527>),
Jose Valdez [aut] (<https://orcid.org/0000-0003-2690-9952>),
Henrique Miguel Pereira [aut] (<https://orcid.org/0000-0003-1043-1675>)

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

Description The concept of Essential Biodiversity Variables (EBV, <https://geobon.org/ebvs/what-are-ebvs/>)
comes with a data structure based on the Network Common Data Form (netCDF).
The 'ebvcube' 'R' package provides functionality to easily create, access and
visualise this data. The EBV netCDFs can be downloaded from the EBV Data
Portal: Christian Langer/ iDiv (2020) <https://portal.geobon.org/>.

URL 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 7.2.3

biocViews

Imports checkmate,
curl,
DelayedArray,
ggplot2,
HDF5Array,
httr,
jsonlite,
memuse,
methods,
ncdf4,
ncmeta,
reshape2,

rhdf5,
stringr,
terra,
tidyterra,
withr

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

SystemRequirements GDAL binaries

Depends R (>= 4.2.0)

Config/testthat/edition 3

R topics documented:

`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_name, project_url, 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, taxonomy, taxonomy_lsid

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

`temporal` Named list. Elements: resolution, units, timesteps, dates

`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. If the dataset does not encompass taxonomic info, the 'taxonomy' is NA. Besides, even if a dataset encompasses the taxonomy information, the 'taxonomy_lsid' can be NA.

`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 your EBV netCDF*

Description

Add data to your EBV netCDF from GeoTiffs or in-memory arrays. First, create a new EBV netCDF using `ebv_create()`.

Usage

```
ebv_add_data(
  filepath_nc,
  datacubepath = NULL,
  entity = NULL,
  timestep = 1,
  data,
  band = 1,
  scenario = NULL,
  metric = NULL,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

Arguments

`filepath_nc`	Character. Path to the self-created netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.
`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. The character string can be obtained using `ebv_properties()`. Choose the entity you are interested in from the slot general and the list item entity_names.
`timestep`	Integer or character. 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 and in order. Meaning c(4,5,6) is right but c(2,5,6) is wrong. Alternatively you can provide a date or list of dates in ISO format, such as '2015-01-01' (also in order).
`data`	Character or matrix or array. If character: Path to the GeoTiff file containing the data. Ending needs to be *.tif. If matrix or array: in-memory object holding the data.
`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.
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`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). Ignore this argument when you give an array or a matrix for 'data' (it will do nothing).
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE.

Value

Adds data to the EBV netCDF. Check your results using `ebv_read()` and/or `ebv_analyse()` and/or `ebv_map()` and/or `ebv_trend()`.

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 if needed.

Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","test.nc"), package="ebvcube")
#get all datacubepaths of EBV netCDF
datacubepaths <- ebv_datacubepaths(file, verbose=FALSE)
#set path to GeoTiff with data
tif <- system.file(file.path("extdata","entity1.tif"), package="ebvcube")

# add data to the timestep 1, 2 and 3 using the first three bands of the GeoTiff
## Not run:
#use datacubepath argument and define timestep by integer
ebv_add_data(filepath_nc = file, datacubepath = datacubepaths[1,1],
             entity = 1, timestep = 1:3, data = tif, band = 1:3)
#use metric argument and define timestep by ISO-format
ebv_add_data(filepath_nc = file, entity = 1,
             timestep = paste0(as.character(seq(1900,1920,10)), '-01-01'),
             metric = 1, data = tif, band = 1:3, verbose = FALSE)

## End(Not run)
```

`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 = NULL,
  entity = NULL,
  timestep = 1,
  subset = NULL,
  touches = TRUE,
  epsg = 4326,
  scenario = NULL,
  metric = NULL,
  numerical = TRUE,
  na_rm = TRUE,
  verbose = TRUE
)
```

Arguments

`filepath`	Character. Path to the netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.
`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 or character. Select one or several timestep(s). Either provide an integer value or list of values that refer(s) to the index of the timestep(s) (minimum value: 1) or provide a date or list of dates in ISO format, such as '2015-01-01'.
`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.
`touches`	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()`.
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`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: TRUE. Turn off additional prints by setting it to FALSE.

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

```
#get measurements for full extent and the first three timesteps
data_global <- ebv_analyse(filepath = file, datacubepath = datacubes[1,1],
                           entity = 1, timestep = 1:3, verbose = FALSE)

#get measurements for subset of Africa only (using bounding box) and one timestep
data_1910 <- ebv_analyse(filepath = file, datacubepath = datacubes[1,1],
                         entity = 1, timestep = "1900-01-01",
                         subset = c(-26, 64, 30, 38), verbose = FALSE)

#get measurements for cameroon only (using shp) and one timestep
data_1930 <- ebv_analyse(filepath = file, entity = 1,
                         timestep = "1930-01-01",
                         subset = shp_path, verbose = FALSE,
                         metric = 'Absolute change in the number of species',)
```

`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 = TRUE
)
```

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 datacubelevel just add the datacubepath, 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: TRUE. Turn off additional prints by setting it to FALSE.

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","baisero_spepop_id5_20220405_v1_empty.nc"),
package="ebvcube")

## Not run:
try({
#change the standard_name of the metric
attribute1 <- 'standard_name'
value1 <- 'habitat availability'
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 <- 'Land-use of 5,090 mammals calculated in sqkm'
level2 <- 'scenario_1/metric_1/ebv_cube' #equal to the datacubepath
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)
}, TRUE)

## End(Not run)
```

---

ebv_create *Create an EBV netCDF*

Description

Create the core structure of the EBV netCDF based on the json from the EBV Data Portal. Data will be added afterwards using `ebv_add_data()`.

Usage

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

Arguments

`jsonpath`	Character. Path to the json file downloaded from the EBV Data Portal. Login to the page and click on 'Uploads' and 'New Upload' to start the process.
`outputpath`	Character. Set path where the netCDF file should be created.
`entities`	Character string or vector of character strings. In case of single character string: Path to the csv table holding the entity names. Default: comma-separated delimiter, else change the `sep` argument accordingly. Should have only one column, each row is the name of one entity. In case of vector of character strings: Vector holding the entity names.
`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).
`timesteps`	Character. Vector of the timesteps in the dataset. Default: NULL - in this case the time will be calculated from the start-, endpoint and temporal resolution given in the metadata file (json). Else, the dates must be given in in ISO format 'YYYY-MM-DD' or shortened 'YYYY' in case of yearly timesteps.
`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'.
`sep`	Character. Default: ','. If the delimiter of the csv specifying the entity-names differs from the default, indicate here.
`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 output file defined by 'outputpath'.
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE.

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
## Not run:
ebv_create(jsonpath = json, outputpath = out, entities = entities,
           fillvalue=-3.4E38)

## End(Not run)
```

---

ebv_create_taxonomy *Create an EBV netCDF with taxonomy*

Description

Create the core structure of the EBV netCDF based on the json from the EBV Data Portal. Additionally, you can add the hierarchy of the taxonomy. This is not provided in the `ebv_create()` function. Use the `ebv_create()` function if your dataset holds no taxonomic information. Data will be added afterwards using `ebv_add_data()`.

Usage

```
ebv_create_taxonomy(
  jsonpath,
  outputpath,
  taxonomy,
  lsid = FALSE,
  epsg = 4326,
  extent = c(-180, 180, -90, 90),
  resolution = c(1, 1),
  timesteps = NULL,
  fillvalue = NULL,
  prec = "double",
  sep = ",",
  force_4D = TRUE,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`jsonpath`	Character. Path to the json file downloaded from the EBV Data Portal. Login to the page and click on 'Uploads' and 'New Upload' to start the process.
`outputpath`	Character. Set path where the netCDF file should be created.

`taxonomy`	Character. Path to the csv table holding the taxonomy. Default: comma-separated delimiter, else change the `sep` argument accordingly. The csv needs to have the following structure: The header displays the names of the different taxon levels ordered from the highest level to the lowest, e.g. "Order", "Family", "Genus", "Species". The last column (if `lsid`=FALSE) is equivalent to the `entity` argument in the `ebv_create()` function. Each row of the csv corresponds to a unique entity. In case the `lsid` argument (see below) is set to the TRUE, this table gets an additional last column which holds the lsid per entity - in this case the second last column contains the entity names, e.g. the following column order: "Order", "Family", "Genus", "Species", "lsid".
`lsid`	Logical. Default: FALSE. Set to TRUE if the last column in your taxonomy csv file defines the lsid for each entity. For more info check CF convention 1.8: Taxon Names and Identifiers.
`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).
`timesteps`	Character. Vector of the timesteps in the dataset. Default: NULL - in this case the time will be calculated from the start-, endpoint and temporal resolution given in the metadata file (json). Else, the dates must be given in in ISO format 'YYYY-MM-DD' or shortened 'YYYY' in case of yearly timesteps.
`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'.
`sep`	Character. Default: ','. If the delimiter of the csv specifying the entity-names differs from the default, indicate here.
`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 output file defined by 'outputpath'
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE. #' @note To check out the results take a look at your netCDF file with Panoply provided by the NASA.

Value

Creates the netCDF file at the 'outputpath' location including the taxonomy information.

Note

You can check the taxonomy info with `ebv_properties()` in the slot 'general' under the name 'taxonomy' and 'taxonomy_lsid'.

Examples

```
#set path to JSON file
json <- system.file(file.path("extdata/testdata","5.json"), package="ebvcube")
#set output path of the new EBV netCDF
out <-  tempfile(fileext='.nc')
#set path to the csv holding the taxonomy names
taxonomy <- file.path(system.file(package='ebvcube'),"extdata/testdata","id5_entities.csv")

#create new EBV netCDF with taxonomy
## Not run:
ebv_create_taxonomy(jsonpath = json, outputpath = out, taxonomy = taxonomy,
           fillvalue = -127, resolution = c(0.25, 0.25), verbose = FALSE)
#remove file
file.remove(out)

## End(Not run)
```

`ebv_datacubepaths` *Get datacubepaths of EBV netCDF*

Description

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

Usage

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

Arguments

`filepath`	Character. Path to the netCDF file.
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE.

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","martins_comcom_subset.nc"), package="ebvcube")

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

`ebv_download` *Download an EBV netCDF file*

Description

Returns the list of all available data sets at the EBV Portal if you no arguments are given. If an ID is given, the corresponding file (netCDF) and its metadata (json file) will be downloaded to the given output directory.

Usage

```
ebv_download(id = NULL, outputdir, overwrite = FALSE, verbose = TRUE)
```

Arguments

`id`	Integer or Character. There are three option to identify the dataset to be downloaded. (1) It can be a single integer value representing the ID of the dataset. (2) It can be a character string representing the title of the data set. (3) It can be a character string representing the DOI of the dataset in the format '10.25829/f2rdp4' (Dataset 'Habitat availability for African great apes' by Jessica Junker from the EBV Data Portal). All three identifier can be retrieved by running `ebv_download()` without any arguments which returns a data.frame of all available data sets and their ID, title and DOI.
`outputdir`	Character. Output directory of the downloaded files.
`overwrite`	Logical. Default: FALSE. Set to TRUE if you want to overwrite the netCDF and json.
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE.

Value

Downloads a netCDF and json file (ACDD metadata) to the given output directory. If run empty returns a data.frame of all available data sets and their ID, title and DOI.

Examples

```
#get all available datasets
datasets <- ebv_download()


ebv_download(id = datasets$id[1], outputdir =
tempdir(), overwrite=TRUE,
verbose=FALSE)
```

ebv_map *Map plot of an EBV netCDF*

Description

Map plot of the data of one timestep in one datacube of an EBV netCDF.

Usage

```
ebv_map(
  filepath,
  datacubepath = NULL,
  entity = NULL,
  timestep = 1,
  countries = TRUE,
  col_rev = FALSE,
  classes = 5,
  scenario = NULL,
  metric = NULL,
  all_data = FALSE,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

Arguments

`filepath`	Character. Path to the netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.
`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 or character. Select a timestep. Either provide an integer value that refers to the index of the timestep (minimum value: 1) or provide a date in ISO format, such as '2015-01-01'.
`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: FALSE Set to TRUE 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 11 classes (allowed maximum for palette RdYlBu is 11).
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.

`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`all_data`	Logical. Default: FALSE. The quantiles are based on the one timestep you chose (default). If you want include the full data of the datacube to produce several maps that are based on the same color scale, set this argument to TRUE (to allow for viusual comparison between entities or timesteps. Does not cover different datacubes.)
`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: TRUE. Turn off additional prints by setting it to FALSE.

Value

Plots a map.

Examples

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

#plot a map for the 3rd timestep, divide into 7 classes
ebv_map(filepath = file, datacubepath = datacubes[1,1], entity = 1,
        timestep = 3, classes = 7, verbose = FALSE)
ebv_map(filepath = file, entity = 'all bird species', timestep = "1950-01-01",
        metric = 'Relative change in the number of species (%)',
        classes = 7, verbose = FALSE)
```

`ebv_properties` *Read properties of EBV netCDF*

Description

Structured access to all attributes of the netCDF file.

Usage

```
ebv_properties(
  filepath,
  datacubepath = NULL,
  scenario = NULL,
  metric = NULL,
  verbose = TRUE
)
```

Arguments

`filepath`	Character. Path to the netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE.

Value

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

Examples

```
#define path to EBV netCDF
file <- system.file(file.path("extdata","martins_comcom_subset.nc"), package="ebvcube")
#get all datacubepaths of EBV netCDF
datacubes <- ebv_datacubepaths(file, verbose=FALSE)

#get properties only for the file
prop_file <- ebv_properties(file)
#get properties for the file and a specific datacube - use datacubepath
prop_dc <- ebv_properties(file, datacubepath = datacubes[1,1])
#get properties for the file and a specific datacube - use scenario & metric
#note: this dataset has no scenario -> only metric is defined
prop_dc <- ebv_properties(file, metric = 2)
```

`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 SpatRaster or an array-like object (DelayedMatrix) pointing to the on-disk netCDF file. The latter is useful for data that exceeds your memory.

Usage

```
ebv_read(
  filepath,
  datacubepath = NULL,
  entity = NULL,
  timestep = 1,
  type = "r",
  scenario = NULL,
  metric = NULL,
  sparse = FALSE,
  ignore_RAM = FALSE,
  verbose = FALSE
)
```

Arguments

`filepath`	Character. Path to the netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.
`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 or character. Select one or several timestep(s). Either provide an integer value or list of values that refer(s) to the index of the timestep(s) (minimum value: 1) or provide a date or list of dates in ISO format, such as '2015-01-01'.
`type`	Character. Choose between 'a', 'r' and 'da'. The first returns an array or matrix object. The 'r' indicates that a SpatRaster object from the terra package will be returned (default). The latter ('da') returns a DelayedArray or DelayedMatrix object.
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`sparse`	Logical. Default: FALSE. Set to TRUE if the data contains a lot empty raster cells. Only relevant for DelayedArray return value.
`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: TRUE. Turn off additional prints by setting it to FALSE.

Value

Array, SpatRaster or DelayedArray 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","martins_comcom_subset.nc"), package="ebvcube")
#get all datacubepaths of EBV netCDF
datacubes <- ebv_datacubepaths(file, verbose=FALSE)


#read data as DelayedArray
cSAR.delayedarray <- ebv_read(filepath = file, datacubepath = datacubes[1,1],
                              entity = 1, timestep = c(1,3), type='da',
                              sparse = TRUE)
#read data as SpatRaster
cSAR.raster <- ebv_read(filepath = file,entity = 1, timestep = "2000-01-01",
                        type='r', metric = 1)
#read data as Array
cSAR.array <- ebv_read(filepath = file, datacubepath = datacubes[1,1],
                              entity = 1, 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,
  datacubepath = NULL,
  entity = NULL,
  timestep = 1,
  bb,
  outputpath = NULL,
  epsg = 4326,
  scenario = NULL,
  metric = NULL,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

Arguments

`filepath`	Character. Path to the netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.

`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 or character. Select one or several timestep(s). Either provide an integer value or list of values that refer(s) to the index of the timestep(s) (minimum value: 1) or provide a date or list of dates in ISO format, such as '2015-01-01'.
`bb`	Integer Vector. Definition of subset by bounding box: c(xmin, xmax, ymin, ymax).
`outputpath`	Character. Default: NULL, returns the data as a SpatRaster 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.
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`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: TRUE. Turn off additional prints by setting it to FALSE.

Value

Returns a SpatRaster 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","martins_comcom_subset.nc"), package="ebvcube")
#get all datacubepaths of EBV netCDF
datacubes <- ebv_datacubepaths(file, verbose=FALSE)

#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(-26, 64, 30, 38)
bb_utm32 <- c(-2383703, 5532302, 3643854, 4564646)


## Not run:
#read bb (based on EPSG 4326) - return SpatRaster
cSAR_subset <- ebv_read_bb(filepath = file, datacubepath = datacubes[1,1],
                             entity = 1, timestep = 1:3, bb = bb_wgs84)

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

#read bb (based on ESRI 54009) - write to GeoTiff
path  <- ebv_read_bb(filepath = file, datacubepath = datacubes[1,1],
                     entity = 1, timestep = 1:2, bb = bb_utm32,
                     epsg = 32632, outputpath = out, overwrite = TRUE)

## End(Not run)
```

---

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.

Usage

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

Arguments

`filepath`	Character. Path to the netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.

`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 or character. Select one or several timestep(s). Either provide an integer value or list of values that refer(s) to the index of the timestep(s) (minimum value: 1) or provide a date or list of dates in ISO format, such as '2015-01-01'.
`shp`	Character. Path to the shapefile defining the subset. Ending needs to be *.shp.
`outputpath`	Character. Default: NULL, returns the data as a SpatRaster object in memory. Optional: set path to write subset as GeoTiff on disk.
`touches`	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).
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`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: TRUE. Turn off additional prints by setting it to FALSE.

Value

Returns a SpatRaster object if no output path is given. Otherwise the subset is written onto the disk and the ouput path is returned.

See Also

`ebv_read_bb()` for subsetting via bounding box.

Examples

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

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


#read subset - return SpatRaster
cSAR_cameroon <- ebv_read_shp(filepath = file, datacubepath = datacubes[1,1],
                              entity = 1, timestep = 1, shp = shp_path,
                              outputpath = NULL, ignore_RAM = TRUE)
```

`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.

Usage

```
ebv_resample(
  filepath_src,
  datacubepath_src = NULL,
  entity_src = NULL,
  timestep_src = 1,
  resolution,
  outputpath,
  method = "bilinear",
  scenario = NULL,
  metric = NULL,
  return_raster = FALSE,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

Arguments

`filepath_src`	Character. Path to the netCDF file whose resolution should be changed.
`datacubepath_src`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.
`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 or character. Select one or several timestep(s). Either provide an integer value or list of values that refer(s) to the index of the timestep(s) (minimum value: 1) or provide a date or list of dates in ISO format, such as '2015-01-01'.
`resolution`	Character or Numeric. 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 code, e.g. c(0.25, 0.25, 4326).
`outputpath`	Character. Set path to write data as GeoTiff on disk.
`method`	Character. Default: bilinear. Define resampling method. Choose from: "near","bilinear","cubic", "cubicspline", "lanczos", "sum", "min", "q1", "med", "q3", "max", "average", "mode" and "rms". For categorical data, use 'near'. Based on `terra::project()`.

`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`return_raster`	Logical. Default: FALSE. Set to TRUE to directly get the corresponding SpatRaster object.
`overwrite`	Logical. Default: FALSE. Set to TRUE to overwrite the output file 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: TRUE. Turn off additional prints by setting it to FALSE.

Value

Default: returns the output path of the GeoTiff with the new resolution. Optional: return the SpatRaster object with the new resolution.

Examples

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

#define different resolutions
res1 <- system.file(file.path("extdata",
        "baisero_spepop_id5_20220405_v1_empty.nc"), package="ebvcube")
res2 <- c(0.5,0.5,4326)
#define output path
out <- file.path(system.file(package='ebvcube'),"extdata","changeRes.tif")

## Not run:
#resample defining the resolution and EPSG code by hand - return SpatRaster
data_raster <- ebv_resample(filepath_src = file, datacubepath_src = datacubes[1,1],
                            entity_src=1, timestep_src = 1, resolution = res2,
                            outputpath = out, method='near', return_raster=TRUE,
                            overwrite=TRUE)
#resample using a netCDF file - return GeoTiff
ebv_resample(filepath_src = file, datacubepath_src = datacubes[1,1],
             entity_src=1, timestep_src = 1, resolution = res1,
             outputpath = out, overwrite=TRUE)


## End(Not run)
```

ebv_trend — *Plot the trend of an EBV netCDF*

Description

Plot the trend of one datacube of a EBV netCDF over time (x-axis). Different options can be chosen based on the `method` argument.

Usage

```
ebv_trend(
  filepath,
  datacubepath = NULL,
  entity = NULL,
  method = "mean",
  subset = NULL,
  color = "dodgerblue4",
  touches = TRUE,
  scenario = NULL,
  metric = NULL,
  verbose = TRUE
)
```

Arguments

`filepath`	Character. Path to the netCDF file.
`datacubepath`	Character. Optional. Default: NULL. Path to the datacube (use `ebv_datacubepaths()`). Alternatively, you can use the scenario and metric argument to define which cube you want to access.
`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.
`method`	Character. Default: mean. Choose one of the following options for different plots: mean, min, max, boxplot. See **Note** for more
`subset`	Character. Default: NULL. If you want to look at the trend for a spatial subset, define the path to the shapefile encompassing the area. Ending needs to be *.shp.
`color`	Character. Default: dodgerblue4. Change to any color known by R `grDevices::colors()`
`touches`	Logical. Optional. Default: TRUE. Only relevant if the subset is indicated by a shapefile. See `ebv_read_shp()`.
`scenario`	Character or integer. Optional. Default: NULL. Define the scenario you want to access. If the EBV netCDF has no scenarios, leave the default value (NULL). You can use an integer value defining the scenario or give the name of the scenario as a character string. To check the available scenarios and their name or number (integer), use `ebv_datacubepaths()`.
`metric`	Character or integer. Optional. Define the metric you want to access. You can use an integer value defining the metric or give the name of the scenario as a character string. To check the available metrics and their name or number (integer), use `ebv_datacubepaths()`.
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE.

Value

Returns plots and eventually values based on the `method` argument. See **Note** for more information

Note

More information on the `method` argument: using `mean` will result in a plot of the mean over time, additionally a vector of the mean values is returned. If the data encompasses only one timestep a single mean is returned. Corresponding behavior can be expected for `min` and `max`. The `boxplot` option results in boxplots over time (no values are returned).

Examples

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


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

`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.

Usage

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

Arguments

`data`	Your data object. May be SpatRaster, array, DelayedMatrix or list of Delayed-Matrix (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: INT1S, INT2S, INT2U, INT4S, INT4U, FLT4S, FLT8S.
`overwrite`	Locigal. Default: FALSE. Set to TRUE to overwrite the outputfile defined by 'outputpath'.
`verbose`	Logical. Default: TRUE. Turn off additional prints by setting it to FALSE.

Value

Returns the outputpath.

Note

If the nodata value of your data is not detected correctly, this could be due to the wrong choice of the datatype (type argument).

Examples

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

## Not run:
#read data
data <- ebv_read(filepath = file, datacubepath = datacubes[1,1], timestep = 1, entity = 1)
# HERE YOU CAN WORK WITH YOUR DATA

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

#read a subset
data_bb <- ebv_read_bb(filepath = file, datacubepath = datacubes[1,1],
                       entity = 1, timestep = 1:3, bb = c(-26, 64, 30, 38))

#write subset to disk as GeoTiff
ebv_write(data = data_bb, outputpath = out, extent = c(-26, 64, 30, 38), overwrite = TRUE)

## End(Not run)
```

`world_boundaries` *Simple outlines of world countries*

Description

Simple outlines of world countries

Usage

```
world_boundaries
```

Format

A data.frame with 177 elements and geometry as WKT

Source

Data downloaded from Natural Earth. Used version 4.0.0 and reduced attributes.

Index