

# Package ‘ebvcube’

April 6, 2023

**Title** Working with netCDF for Essential Biodiversity Variables

**Version** 0.1.3

**Date** 2023-01-04

**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,  
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 . . . . . | 2         |
| ebvcube . . . . .                     | 3         |
| ebv_add_data . . . . .                | 3         |
| ebv_analyse . . . . .                 | 5         |
| ebv_attribute . . . . .               | 6         |
| ebv_create . . . . .                  | 8         |
| ebv_datacubePaths . . . . .           | 9         |
| ebv_download . . . . .                | 10        |
| ebv_map . . . . .                     | 11        |
| ebv_properties . . . . .              | 12        |
| ebv_read . . . . .                    | 13        |
| ebv_read_bb . . . . .                 | 14        |
| ebv_read_shp . . . . .                | 16        |
| ebv_resample . . . . .                | 17        |
| ebv_trend . . . . .                   | 19        |
| ebv_write . . . . .                   | 20        |
| world_boundaries . . . . .            | 21        |
| <b>Index</b>                          | <b>22</b> |

---

|   |
|---|
| EBV netCDF properties-class             |
| <i>EBV netCDF properties class (S4)</i> |

---

**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

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.

---

 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. First, create a new EBV netCDF using [ebv\\_create\(\)](#).

## Usage

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

## Arguments

|               |  |
|---------------|--|
| filepath_nc   | Character. Path to the self-created netCDF file.   |
| datacubeopath | Character. Path to the datacube (use <a href="#">ebv_datacubeopaths()</a> ).   |
| 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 <a href="#">ebv_properties()</a> . Choose the entity you are interested in from the slot general and the list item entity_names. |
| 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.  |
| 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.  |
| 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", "martins_comcom_id1_20220208_v1.nc"), package="ebvcube")
#get all datacubeopaths of EBV netCDF
datacubeopaths <- ebv_datacubeopaths(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:
ebv_add_data(filepath_nc = file, datacubeopath = datacubeopath[1,1],
              entity = 1, timestep = 1:3, data = tif, band = 1:3)

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

## Arguments

|               |   |
|---------------|---|
| filepath      | Character. Path to the netCDF file.   |
| datacubeopath | Character. Path to the datacube (use <a href="#">ebv_datacubeopaths()</a> ).  |
| 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. |
| touches       | Logical. Optional. Default: TRUE. Only relevant if the subset is indicated by a shapefile. See <a href="#">ebv_read_shp()</a> .   |
| epsg          | Numeric. Optional. Only relevant if the subset is indicated by a bounding box and the coordinate reference system differs from WGS84. See <a href="#">ebv_read_bb()</a> .   |
| 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_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file, verbose=FALSE)
#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 = 1, timestep = 1:3)

#get measurements for germany only (using bounding box) and one timestep
data_bb_1900 <- ebv_analyse(filepath = file, datacube_path = datacubes[1,1],
                          entity = 1, timestep = 2, 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 = 1, timestep = 3, 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 = 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 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: 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","martins_comcom_id1_20220208_v1.nc"), package="ebvcube")

## Not run:
try({
  #change the standard_name of the metric
  attribute1 <- 'standard_name'
  value1 <- 'Relative change in the number of species (%)'
  level1 <- '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 <- '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)
}, 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 [Geobon Portal API](#). Data will be added afterwards. Use [ebv\\_add\\_data\(\)](#) to add the missing 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 <a href="#">Geobon Portal API</a> .   |
| 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 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\_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_id1_20220208_v1.nc"), package="ebvcube")

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

---

|              |                                    |
|--------------|------------------------------------|
| ebv_download | <i>Download an EBV netCDF file</i> |
|--------------|------------------------------------|

---

**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. Must be a single integer value or a character string representing the title of the data set. Both can be retrieved by running <a href="#">ebv_download()</a> without any arguments which returns the list of data sets available and their title and ID. |
| 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 (metadata) to the given output directory.

**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,
  datacube_path,
  entity = NULL,
  timestep = 1,
  countries = TRUE,
  col_rev = FALSE,
  classes = 5,
  all_data = FALSE,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

**Arguments**

|               |   |
|---------------|---|
| filepath      | Character. Path to the netCDF file.   |
| datacube_path | Character. Path to the datacube (use <a href="#">ebv_datacube_paths()</a> ).  |
| 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: 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).   |
| 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 visual 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_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file, verbose=FALSE)

#plot a map for the 3rd timestep, divide into 7 classes
ebv_map(filepath = file, datacube_path = datacubes[1,1], entity = 1,
        timestep = 3, 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 = TRUE)
```

**Arguments**

|               |  |
|---------------|--|
| filepath      | Character. Path to the netCDF file.  |
| datacube_path | Character. Optional. Path to the datacube (use <a href="#">ebv_datacube_paths()</a> ). |
| 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**

```
#set path to EBV netCDF
file <- system.file(file.path("extdata", "martins_comcom_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file, verbose=FALSE)

#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 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,
  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 <a href="#">ebv_datacube_paths()</a> ).  |
| 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 that a SpatRaster object from the terra package will be returned. The latter ('da') returns a DelayedArray or DelayedMatrix object. |
| 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_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file, verbose=FALSE)

#read data as DelayedArray
cSAR.delayedarray <- ebv_read(filepath = file, datacube_path = datacubes[1,1],
                             entity = 1, timestep = c(1,3), type='da',
                             sparse = TRUE)

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

#read data as Array
cSAR.array <- ebv_read(filepath = file, datacube_path = 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,
  datacube_path,
  entity = NULL,
  timestep = 1,
  bb,
  outputpath = NULL,
  epsg = 4326,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

## Arguments

|               |  |
|---------------|--|
| filepath      | Character. Path to the netCDF file.  |
| datacube_path | Character. Path to the datacube (use <a href="#">ebv_datacube_paths()</a> ).   |
| 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: TRUE. Turn off additional prints by setting it to FALSE.  |

### 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", "martins_comcom_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(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(5,15,47,55)
bb_utm32 <- c(271985, 941837, 5232640, 6101151)

#read bb (based on EPSG 4326) - return Raster
cSAR.germany <- ebv_read_bb(filepath = file, datacube_path = datacubes[1,1],
                           entity = 1, timestep = 1:3, bb = bb_wgs84)

## Not run:
#read bb (based on EPSG 4326) - write to GeoTiff
path <- ebv_read_bb(filepath = file, datacube_path = datacubes[1,1],
                   entity = 1, 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 = 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,
  entity = NULL,
  timestep = 1,
  shp,
  outputpath = NULL,
  touches = TRUE,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

## Arguments

|              |  |
|--------------|--|
| filepath     | Character. Path to the netCDF file.  |
| datacubepath | Character. Path to the datacube (use <a href="#">ebv_datacubepaths()</a> ).  |
| 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.  |
| 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).                      |
| 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 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**

```
#set path to EBV netCDF
file <- system.file(file.path("extdata","martins_comcom_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file, verbose=FALSE)

#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[1,1],
                             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,
  datacube_path_src,
  entity_src = NULL,
  timestep_src = 1,
  resolution,
  outputpath,
  method = "bilinear",
  return_raster = FALSE,
  overwrite = FALSE,
  ignore_RAM = FALSE,
  verbose = TRUE
)
```

**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 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" and "cubicspline". For categorical data, use 'near'. Based on <a href="#">terra::project()</a> .   |
| return_raster | Logical. Default: FALSE. Set to TRUE to directly get the corresponding SpatRast 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 outputpath of the GeoTiff with the new resolution. Optional: return the raster object with the new resolution.

### Examples

```
#set path to EBV netCDF
file <- system.file(file.path("extdata", "martins_comcom_id1_20220208_v1.nc"),
                    package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(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 Raster
data_raster <- ebv_resample(filepath_src = file, datacube_path_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, datacube_path_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,
  datacube_path,
  entity = NULL,
  method = "mean",
  subset = NULL,
  color = "dodgerblue4",
  touches = TRUE,
  verbose = TRUE
)
```

## Arguments

|               |  |
|---------------|--|
| filepath      | Character. Path to the NetCDF file.  |
| datacube_path | Character. Path to the datacube (use <a href="#">ebv_datacube_paths()</a> ).   |
| 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 <b>Note</b> for more information.  |
| 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 <a href="#">grDevices::colors()</a>  |
| touches       | Logical. Optional. Default: TRUE. Only relevant if the subset is indicated by a shapefile. See <a href="#">ebv_read_shp()</a> .  |
| 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_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file, verbose=FALSE)

#plot the change of the mean over time of the first datacube
ebv_trend(filepath = file, datacube_path = 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 DelayedMatrix (see return values of <a href="#">ebv_read()</a> )    |
| 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  | Logical. 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_id1_20220208_v1.nc"), package="ebvcube")
#get all datacube paths of EBV netCDF
datacubes <- ebv_datacube_paths(file, verbose=FALSE)

## Not run:
#read data
data <- ebv_read(filepath = file, datacube_path = 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, datacube_path = datacubes[1,1],
                      entity = 1, 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)

## 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

## \* datasets

world\_boundaries, [21](#)

DelayedArray::DelayedArray(), [13](#)

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

ebv\_add\_data, [3](#)

ebv\_add\_data(), [8](#)

ebv\_analyse, [5](#)

ebv\_analyse(), [4](#)

ebv\_attribute, [6](#)

ebv\_create, [8](#)

ebv\_create(), [3](#)

ebv\_datacube\_paths, [9](#)

ebv\_datacube\_paths(), [4](#), [5](#), [11–14](#), [16](#), [17](#), [19](#)

ebv\_download, [10](#)

ebv\_download(), [10](#)

ebv\_map, [11](#)

ebv\_map(), [4](#)

ebv\_properties, [12](#)

ebv\_properties(), [4](#), [7](#)

ebv\_read, [13](#)

ebv\_read(), [4](#), [20](#)

ebv\_read\_bb, [14](#)

ebv\_read\_bb(), [5](#), [6](#), [17](#)

ebv\_read\_shp, [16](#)

ebv\_read\_shp(), [5](#), [6](#), [15](#), [19](#)

ebv\_resample, [17](#)

ebv\_trend, [19](#)

ebv\_trend(), [4](#)

ebv\_write, [20](#)

ebvcube, [3](#)

grDevices::colors(), [19](#)

terra::project(), [18](#)

world\_boundaries, [21](#)