# Package 'ebvnetcdf'

May 7, 2021

**Title** Working with EVB NetCDFs

**Version** 0.0.1

**Description** This package can be used to easily access the data of the EBV NetCDFs which can be downloaded here: portal.geobon.org. It also provides some basic visualization of the data. Advanced users can build their own NetCDFs with the EBV standard using this package.

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.1

**BiocViews**

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

**Depends** R (>= 2.10)

## R topics documented:


EBV NetCDF properties-class . . . . . 2
ebvnetcdf . . . . . 3
ebv_datacubepaths . . . . . 3

ebv_data_analyse . . . . . 4
ebv_data_change_res . . . . . 5
ebv_data_read . . . . . 6
ebv_data_read_bb . . . . . 7
ebv_data_read_shp . . . . . 8
ebv_data_write . . . . . 10
ebv_ncdf_add_data . . . . . 11
ebv_ncdf_create . . . . . 12
ebv_ncdf_entity_attributes . . . . . 13
ebv_plot_indicator . . . . . 14
ebv_plot_map . . . . . 14
ebv_properties . . . . . 15
wrld_simpl . . . . . 16

**Index** **17**

---

`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_subgroups, creator

`spatial` Named list. Elements: srs, epsg, resolution, extent, dimensions

`temporal` Named list. Elements: units, t_delta, timesteps, timesteps_natural

`metric` Named list. Elements: standard_name, description

`scenario` Named list. Elements: standard_name, description

`entity` Named list. Elements: standard_name, description, unit, type, fillvalue, value_range

## Note

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

## ebvnetcdf *Working with EVB NetCDFs*

### 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 and visualising the data from the portal and creating your own data in the EBV NetCDF standard. All function have a corresponding naming pattern: ebv_data_ for data reading, ebv_plot_ for visualisation and ebv_netcdf_ for creating a NetCDF.

## 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 = 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

```
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
```

`ebv_data_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_data_analyse(
  filepath,
  datacubepath,
  subset = NULL,
  timestep = 1,
  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()`). |
| `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. |
| `timestep` | Integer. Choose one or several timesteps (vector). |
| `at` | Logical. Optional. Default: TRUE. Only relevant if the subset is indicated by a shapefile. See `ebv_data_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_data_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_data_read_bb()` and `ebv_data_read_shp()` for the usage of subsets.

## Examples

```
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
data.global.year <- ebv_data_analyse(file, datacubes[1,1], timestep=c(1:12))
data.germany.1900 <- ebv_data_analyse(file, datacubes[1,1], c(5,15,47,55), timestep=1)
```

---

`ebv_data_change_res` *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 functions writes temporary files on your disk. Speficy a directory for these setting via options('temp_directory'='/path/to/temp/directory').

## Usage

```
ebv_data_change_res(
  filepath_src,
  datacubepath_src,
  resolution,
  outputpath,
  timestep = 1,
  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. |
| `datacubepath_src` | Character. Path to the datacube (use `ebv_datacubepaths()`) whose resolution should be changed.. |
| `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. |
| `timestep` | Integer. Choose one or several timesteps (vector). |
| `method` | Character. Default: Average. Define resampling method. Choose from: "near","bilinear","cubic","cul 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('temp_directory'=system.file("extdata/", package="ebvnetcdf"))
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
res1 <- system.file(file.path("extdata","rodinini_001.nc"), package="ebvnetcdf")
res2 <- c(1,1,4326)
out <- file.path(system.file(package='ebvnetcdf'),"extdata","changeRes.tif")
#ebv_data_change_res(file, datacubes[1,1], res1,  out, c(1,6))
#d <- ebv_data_change_res(file, datacubes[1,1], res2, NULL, 3, method='max', return.raster=TRUE)
```

---

# ebv_data_read *Read datasube 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_data_read(
  filepath,
  datacubepath,
  timestep,
  delayed = TRUE,
  sparse = FALSE,
  raster = FALSE,
  ignore.RAM = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| filepath | Character. Path to the NetCDF file. |
| datacubepath | Character. Path to the datacube (use ebv_datacubepaths()). |
| timestep | Integer. Choose one or several timesteps (vector). |
| delayed | Logical. Default: TRUE. Returns data as DelayedMatrix object. More timesteps are not returned as a 3D array but as a list of the DelayedMatrix (one matrix per band). |
| 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. |
| raster | Logical. Default: FALSE. Set to TRUE and 'delayed' to FALSE to get a raster. If both arguments are set to FALSE the function returns an array. |

| | |
|---|---|
| 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

```
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
#cSAR.delayedarray <- ebv_data_read(file, datacubes[1,1], c(1,6), delayed=T, sparse=T)
#cSAR.raster <- ebv_data_read(file, datacubes[1,1], 1, delayed = F, raster = T)
#cSAR.array <- ebv_data_read(file, datacubes[1,1], c(1,1,3), delayed = F, raster = F)
```

---

# ebv_data_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_data_read_bb(
  filepath,
  datacubepath,
  bb,
  outputpath = NULL,
  timestep = 1,
  epsg = 4326,
  overwrite = FALSE,
  ignore.RAM = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| filepath | Character. Path to the NetCDF file. |
| datacubepath | Character. Path to the datacube (use ebv_datacubepaths()). |
| bb | Integer Vector. Definition of subsset 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. |
| timestep | Integer. Choose one or several timesteps. |
| 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 ouputpath 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_data_read_shp() for subsetting via shapefile.

## Examples

```
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
out <- file.path(system.file(package='ebvnetcdf'),"extdata","subset_bb.tif")
bb_wgs84 <- c(5,15,47,55)
bb_utm32 <- c(271985, 941837, 5232640, 6101151)
#cSAR.germany <- ebv_data_read_bb(file, datacubes[1], bb_wgs84, timestep = c(1,4,12))
#path <- ebv_data_read_bb(file, datacubes[1], bb_wgs84, out, timestep = c(2,3))
#path <- ebv_data_read_bb(file, datacubes[1], bb_utm32, out, timestep=1, epsg=32632, overwrite=T)
```

---

# ebv_data_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('temp_directory'='/path/to/temp/directory').

## Usage

```
ebv_data_read_shp(
  filepath,
  datacubepath,
  shp,
  outputpath = NULL,
  timestep = 1,
  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()). |
| shp | Character. Path to the shapefile defining the subset. |
| outputpath | Character. Default: NULL, returns the data as a raster object in memory. Optional: set path to write subset as GeoTiff on disk. |
| timestep | Integer. Choose one or several timesteps (vector). |
| 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 ouputpath is returned.

## See Also

ebv_data_read_bb() for subsetting via bounding box.

## Examples

```
#define temp directory
options('temp_directory'=system.file("extdata/", package="ebvnetcdf"))
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
shp <- system.file(file.path("extdata","subset_germany.shp"), package="ebvnetcdf")
#cSAR.germany <- ebv_data_read_bb(file, datacubes[1], shp)
```

ebv_data_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('temp_directory'='/path/to/temp/directory').

## Usage

```
ebv_data_write(
  data,
  filepath,
  datacubepath,
  outputpath,
  overwrite = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| data | Your data object. May be raster, array, DelayedMatrix or list of DelayedMatrix (see return values of ebv_data_read()) |
| filepath | Character. Path to the NetCDF file you read the data from. Used for the detection of properties as spatial extent and epsg. |
| datacubepath | Character. Path to the datacube you got the data from. Used for the detection of properties as data type and nodata value. |
| outputpath | Character. Set the path where you want to write the data to disk as a GeoTiff. |
| overwrite | Locigal. 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).

## Examples

```
#define temp directory
options('temp_directory'=system.file("extdata/", package="ebvnetcdf"))
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
data <- ebv_data_read(file, datacubes[1,1], 1)
# WORK WITH YOUR DATA
out <- system.file(file.path("extdata","write_data.tif"), package="ebvnetcdf")
#ebv_data_write(data, file, datacubes[1,1], out)
```

ebv_ncdf_add_data *Add data to a self-created EBV NetCDF*

## Description

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

## Usage

```
ebv_ncdf_add_data(
  filepath_nc,
  filepath_tif,
  datacubepath,
  timestep = 1,
  band = 1,
  ignore.RAM = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| filepath_nc | Character. Path to the self-created NetCDF file. |
| filepath_tif | Character. Path to the GeoTiff file containing the data. |
| datacubepath | Character. Path to the datacube (use ebv_datacubepaths()). |
| 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. |
| 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_data_read() and/or ebv_data_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

```
file <- system.file(file.path("extdata","cSAR_new.nc"), package="ebvnetcdf")
tif <- system.file(file.path("extdata","cSAR_write_ts234.tif"), package="ebvnetcdf")
# datacubes <- ebv_datacubepaths(file)
ts <- c(2:4)
band <- c(1:3)
#ebv_ncdf_add_data(file, tif, datacubepaths[1,1], ts, band)
```

`ebv_ncdf_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. See `ebv_ncdf_add_data()` and `ebv_ncdf_entity_attributes()` for the next steps.

## Usage

```
ebv_ncdf_create(
  jsonpath,
  outputpath,
  entities.no = 0,
  epsg = 4326,
  extent = c(-180, 180, -90, 90),
  fillvalue = NULL,
  prec = "double",
  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.no` | Integer. Default: 0. Indicates how many entities there are per metric. |
| `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). |
| `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'. |
| `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

```
json <- system.file(file.path("extdata","1.json"), package="ebvnetcdf")
out <- file.path(system.file(package='ebvnetcdf'),"extdata","sCAR_new.nc")
#ebv_ncdf_create(json, out, 3, fillvalue=-3.4E38)
```

---

`ebv_ncdf_entity_attributes`

*Add datacube attributes to an EBV NetCDF*

---

## Description

Add standard_name, description and fillvalue to an entitiy/datacube of a self-created EBV NetCDF. First use `ebv_ncdf_create()` to create a NetCDF afterwards use this function to add attributes.

## Usage

```
ebv_ncdf_entity_attributes(
  filepath,
  datacubepath,
  standard_name,
  description,
  fillvalue = NULL,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `filepath` | Character. Path to the self-created NetCDF file. |
| `datacubepath` | Character. Path to the datacube (use `ebv_datacubepaths()`). |
| `standard_name` | Value of the standard_name attribute. |
| `description` | Character. Value of the description attribute. |
| `fillvalue` | Numeric. Value of the fillvalue attribute. |
| `verbose` | Logical. Default: FALSE. Turn on all warnings by setting it to TRUE. |

## Value

Adds attributes to NetCDF at datacube level. Check results using `ebv_properties()`

## Examples

```
file <- system.file(file.path("extdata","cSAR_new.nc"), package="ebvnetcdf")
# datacubes <- ebv_datacubepaths(file)
sn <- 'non forest birds species'
desc <- 'Changes in bird diversity at the grid cell level caused by land-use'
fv <- -3.4E38
#ebv_ncdf_entity_attributes(file, datacubes[1,1], sn, desc, fv)
```

## ebv_plot_indicator *Plot the average over time of one datacube of an EBV NetCDF*

### 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_plot_indicator(
  filepath,
  datacubepath,
  color = "dodgerblue4",
  verbose = FALSE
)
```

### Arguments

| | |
|---|---|
| filepath | Character. Path to the NetCDF file. |
| datacubepath | Character. Path to the datacube (use ebv_datacubepaths()). |
| 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

```
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
ebv_plot_indicator(file, datacubes[1,1])
```

## ebv_plot_map *Map plot of an EBV NetCDF*

### Description

Map plot of the data of one timestep in one datacube of an EBV NetCDF. This functions sometimes writes temporary files on your disk. Speficy a directory for these setting via options('temp_directory'='/path/to/temp/direc

## Usage

```
ebv_plot_map(
  filepath,
  datacubepath,
  timestep = 1,
  countries = TRUE,
  col.rev = TRUE,
  classes = 5,
  ignore.RAM = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| filepath | Character. Path to the NetCDF file. |
| datacubepath | Character. Path to the datacube (use ebv_datacubepaths()). |
| 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.

## Note

Uses the country outlines data from the maptools package.

## Examples

```
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
ebv_plot_map(file, datacubes[1,1], timestep=9, classes=7)
```

---

# ebv_properties *Read properties of EBV NetCDF*

---

## Description

Structured access to all attributes of the NetCDF file.

## Usage

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

## Arguments

| | |
|---|---|
| filepath | Character. Path to the NetCDF file. |
| datacubepath | Character. Optional. Path to the datacube (use ebv_datacubepaths()). |
| 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

```
file <- system.file(file.path("extdata","cSAR_idiv_v1.nc"), package="ebvnetcdf")
datacubes <- ebv_datacubepaths(file)
prop_file <- ebv_properties(file)
prop_dc <- ebv_properties(file, datacubes[1,1])
```

---

# wrld_simpl *Simple outlines of world countries*

---

## Description

Simple outlines of world countries

## Usage

```
wrld_simpl
```

## Format

A Spatial Polygons Data Frame with 246 elements

## Source

Data imported from the maptools package: data(wrld_simpl, package='maptools')

# Index

∗ **datasets**
wrld_simpl, 16

DelayedArray::DelayedArray(), *7*

EBV NetCDF properties-class, 2
ebv_data_analyse, 4
ebv_data_analyse(), *11*
ebv_data_change_res, 5
ebv_data_read, 6
ebv_data_read(), *10, 11*
ebv_data_read_bb, 7
ebv_data_read_bb(), *4*, *9*
ebv_data_read_shp, 8
ebv_data_read_shp(), *4*, *8*
ebv_data_write, 10
ebv_datacubepaths, 3
ebv_datacubepaths(), *4–7*, *9*, *11*, *13–16*
ebv_ncdf_add_data, 11
ebv_ncdf_add_data(), *12*
ebv_ncdf_create, 12
ebv_ncdf_create(), *13*
ebv_ncdf_entity_attributes, 13
ebv_ncdf_entity_attributes(), *12*
ebv_plot_indicator, 14
ebv_plot_map, 14
ebv_properties, 15
ebv_properties(), *13*
ebvnetcdf, 3

grDevices::colors(), *14*

wrld_simpl, 16