# Package 'easyNCDF'

March 4, 2019

**Title** Tools to Easily Read/Write NetCDF Files into/from Multidimensional R Arrays

**Version** 0.0.7

**Description** Set of wrappers for the 'ncdf4' package to simplify and extend its reading/writing capabilities into/from multidimensional R arrays.

**Depends** R (>= 2.14.1)

**Imports** ncdf4, abind

**License** LGPL-3

**URL** https://earth.bsc.es/gitlab/es/easyNCDF/wikis/home

**BugReports** https://earth.bsc.es/gitlab/es/easyNCDF/issues

**LazyData** true

**SystemRequirements** netcdf development libraries

**NeedsCompilation** no

**Author** BSC-CNS [aut, cph],
Nicolau Manubens [aut, cre]

**Maintainer** Nicolau Manubens <nicolau.manubens@bsc.es>

## R topics documented:

ArrayToNc . . . . . 2
NcClose . . . . . 6
NcOpen . . . . . 7
NcReadDims . . . . . 8
NcReadVarNames . . . . . 9
NcToArray . . . . . 10
Subset . . . . . 12

**Index** **14**

ArrayToNc *Save multidimensional R arrays into NetCDF files*

## Description

This function takes as input one or a list of multidimensional R arrays and stores them in a NetCDF file, using the ncdf4 package. The full path and name of the resulting file must be specified. Metadata can be attached to the arrays and propagated into the NetCDF file in 3 possible ways:

- Via the list names if a list of arrays is provided:Each name in the input list, corresponding to one multidimensional array, will be interpreted as the name of the variable it contains.
  E.g:
  ```
  ArrayToNc(arrays = list(temperature = array(1:9, c(3, 3))),                fil
  ```

- Via the dimension names of each provided array:The dimension names of each of the provided arrays will be interpreted as names for the dimensions of the NetCDF files. Read further for special dimension names that will trigger special behaviours, such as 'time' and 'var'.
  E.g:
  ```
  temperature <- array(rnorm(100 * 50 * 10), dim = c(100, 50, 10))
  names(dim(temperature)) <- c('longitude', 'latitude', 'time') ArrayToNc(list
  ```

- Via the attribute 'variables' of each provided array:The arrays can be provided with metadata in an attribute named 'variables', which is expected to be a named list of named lists, where the names of the container list are the names of the variables present in the provided array, and where each sub-list contains metadata for each of the variables. The attribute names and values supported in the sub-lists must follow the same format the package ncdf4 uses to represent the NetCDF file headers.
  E.g:
  ```
  a <- array(1:400, dim = c(5, 10, 4, 2)) metadata <- list(                tos
  ) attr(a, 'variables') <- metadata names(dim(a)) <- c('lat', 'lon', 'time',
  ArrayToNc(a, 'tmp.nc')
  ```

The special dimension names are 'var'/'variable' and 'time'.
If a dimension is named 'var' or 'variable', ArrayToNc will interpret each array entry along such dimension corresponds to a separate new variable, hence will create a new variable inside the NetCDF file and will use it to store all the data in the provided array for the corresponding entry along the 'var'/'variable' dimension.
If a dimension is named 'time', by default it will be interpreted and built as an unlimited dimension. The 'time' dimension must be the last dimension of the array (the right-most). If a 'var'/'variable' dimension is present, the 'time' dimension can be also placed on its left (i.e. the one before the last dimension). The default behaviour of creating the 'time' as unlimited dimension can be disabled by setting manually the attribute unlim = FALSE, as shown in the previous example.

a2nc is an alias of ArrayToNc.

## Usage

```
ArrayToNc(arrays, file_path)
a2nc(arrays, file_path)
```

## Arguments

| | |
|---|---|
| arrays | One or a list of multidimensional data arrays. The list can be provided with names, which will be interpreted as variable names. The arrays can be provided with dimension names. The arrays can be provided with metadata in the attribute 'variables' (read section Description for details). |
| file_path | Path and name of the NetCDF file to be created. |

## Value

This function returns NULL

## Author(s)

History:
0.0 - 2017-01 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code.

## Examples

```
  ## Not run:
# Minimal use case
ArrayToNc(array(1:9, c(3, 3)), 'tmp.nc')

# Works with arrays of any number of dimensions
ArrayToNc(array(1:27, c(3, 3, 3)), 'tmp.nc')

# Arrays can also be provided in [named] lists
ArrayToNc(list(tos = array(1:27, c(3, 3, 3))), 'tmp.nc')

# Or with dimension names
# 'var' dimension name will generate multiple variables in the
# resulting NetCDF file
a <- array(1:27, dim = c(3, 3, 3))
names(dim(a)) <- c('lon', 'lat', 'var')
ArrayToNc(a, 'tmp.nc')

# 'variable' as dimension name will do the same
a <- array(1:27, dim = c(3, 3, 3))
names(dim(a)) <- c('lon', 'lat', 'variable')
ArrayToNc(a, 'tmp.nc')

# The 'time' dimension will be built as unlimited dimension, by default
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(a, 'tmp.nc')

# Putting the 'time' dimension in a position which is not the last, or the one
# right before 'var'/'variable' will crash. Unlimited dimension must be in the
# last position
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('time', 'lat', 'lon', 'var')
ArrayToNc(a, 'tmp.nc')
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('lat', 'time', 'lon', 'var')
ArrayToNc(a, 'tmp.nc')
```

```
# The dimension 'var'/'variable' can be in any position and can have any length
a <- array(1:1600, dim = c(10, 20, 4, 2))
names(dim(a)) <- c('lat', 'var', 'lon', 'time')
ArrayToNc(a, 'tmp.nc')

# Multiple arrays can be provided in a list
a <- array(1:400, dim = c(5, 10, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(list(a, a), 'tmp.nc')

# If no dimension names are given to an array, new names will be automatically
# generated
a <- array(1:400, dim = c(5, 10, 4, 2))
b <- array(1:400, dim = c(5, 11, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(list(a, b), 'tmp.nc')

# If two arrays use a same dimension but their lengths differ, the function
# will crash
a <- array(1:400, dim = c(5, 10, 4, 2))
b <- array(1:400, dim = c(5, 11, 4, 2))
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
names(dim(b)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(list(a, b), 'tmp.nc')

# Metadata can be provided for each variable in each array, via the
# attribute 'variables'. In this example the metadata is empty.
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(),
              tas = list()
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(a, 'tmp.nc')

# Variable names can be manually specified
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(name = 'name1'),
              tas = list(name = 'name2')
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(a, 'tmp.nc')

# Units can be specified
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(units = 'K'),
              tas = list(units = 'K')
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(a, 'tmp.nc')

# addOffset and scaleFactor can be specified
```

```
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(addOffset = 100,
                         scaleFact = 10),
              tas = list(addOffset = 100,
                         scaleFact = 10)
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(a, 'tmp.nc')

# Global attributes can be specified
a <- array(rnorm(10), dim = c(a = 5, b = 2))
attrs <- list(variables =
           list(tas = list(var_attr_1 = 'test_1_var',
                           var_attr_2 = 2)),
           global_attrs = list(global_attr_name_1 = 'test_1_global',
                               global_attr_name_2 = 2)
         )
attributes(a) <- c(attributes(a), attrs)
ArrayToNc(a, 'tmp.nc')

# Unlimited dimensions can be manually created
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(addOffset = 100,
                         scaleFact = 10,
                         dim = list(list(name = 'unlimited',
                                         unlim = TRUE))),
              tas = list(addOffset = 100,
                         scaleFact = 10,
                         dim = list(list(name = 'unlimited',
                                         unlim = TRUE)))
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'unlimited', 'var')
ArrayToNc(a, 'tmp.nc')

# A 'time' dimension can be built without it necessarily being unlimited
a <- array(1:400, dim = c(5, 10, 4, 2))
metadata <- list(
              tos = list(addOffset = 100,
                         scaleFact = 10,
                         dim = list(list(name = 'time',
                                         unlim = FALSE))),
              tas = list(addOffset = 100,
                         scaleFact = 10,
                         dim = list(list(name = 'time',
                                         unlim = FALSE)))
            )
attr(a, 'variables') <- metadata
names(dim(a)) <- c('lat', 'lon', 'time', 'var')
ArrayToNc(a, 'tmp.nc')

# Multiple arrays with data for multiple variables can be saved into a
# NetCDF file at once.
tos <- array(1:400, dim = c(5, 10, 4))
```

```
metadata <- list(tos = list(units = 'K'))
attr(tos, 'variables') <- metadata
names(dim(tos)) <- c('lat', 'lon', 'time')
lon <- seq(0, 360 - 360 / 10, length.out = 10)
dim(lon) <- length(lon)
metadata <- list(lon = list(units = 'degrees_east'))
attr(lon, 'variables') <- metadata
names(dim(lon)) <- 'lon'
lat <- seq(-90, 90, length.out = 5)
dim(lat) <- length(lat)
metadata <- list(lat = list(units = 'degrees_north'))
attr(lat, 'variables') <- metadata
names(dim(lat)) <- 'lat'
ArrayToNc(list(tos, lon, lat), 'tmp.nc')

## End(Not run)
```

---

`NcClose` *Close a NEtCDF File*

---

## Description

Close a `ncdf4` open connection to a file.

## Usage

```
NcClose(file_object)
```

## Arguments

| | |
|---|---|
| `file_object` | NetCDF object as returned by `ncdf4::nc_open`. |

## Value

The result of `ncdf4::nc_close`.

## Author(s)

History:
0.0 - 2017-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code

## Examples

```
# Create an array from R
file_path <- tempfile(fileext = '.nc')
a <- array(1:9, dim = c(member = 3, time = 3))
# Store into a NetCDF twice, as two different variables
ArrayToNc(list(var_1 = a, var_2 = a + 1), file_path)
# Read the dimensions and variables in the created file
fnc <- NcOpen(file_path)
fnc_dims <- NcReadDims(fnc)
var_names <- NcReadVarNames(fnc)
# Read the two variables from the file into an R array
a_from_file <- NcToArray(fnc, vars_to_read = var_names)
```

```
NcClose(fnc)
# Check the obtained array matches the original array
print(a)
print(a_from_file[1, , ])
```

---

NcOpen *Open a NetCDF File*

---

## Description

Silently opens a NetCDF file with `ncdf4::nc_open`. Returns NULL on failure.

## Usage

```
NcOpen(file_path)
```

## Arguments

| | |
|---|---|
| `file_path` | Character string with the path to the file to be opened. |

## Value

A NetCDF object as returned by `ncdf4::nc_open` or NULL on failure.

## Author(s)

History:
0.0 - 2017-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code

## Examples

```
# Create an array from R
file_path <- tempfile(fileext = '.nc')
a <- array(1:9, dim = c(member = 3, time = 3))
# Store into a NetCDF twice, as two different variables
ArrayToNc(list(var_1 = a, var_2 = a + 1), file_path)
# Read the dimensions and variables in the created file
fnc <- NcOpen(file_path)
fnc_dims <- NcReadDims(fnc)
var_names <- NcReadVarNames(fnc)
# Read the two variables from the file into an R array
a_from_file <- NcToArray(fnc, vars_to_read = var_names)
NcClose(fnc)
# Check the obtained array matches the original array
print(a)
print(a_from_file[1, , ])
```

NcReadDims *Read Dimensions of a NetCDF File*

## Description

Reads the dimension names and sizes of a set of variables in a NetCDF file, using the package ncdf4. The different variables in the file are considered to be stored along a dimension called 'var', so reading the dimensions of a variable 'foo' with dimensions 'lat' and 'lon' would result in a vector with the format c('var' = 1, 'lat' = n_lats, 'lon' = n_lons).

## Usage

```
NcReadDims(file_to_read, var_names = NULL)
```

## Arguments

| | |
|---|---|
| file_to_read | Path to the file to be read or a NetCDF object as returned by easyNCDF::NcOpen or ncdf4::nc_open. |
| var_names | Vector of character strings with the names of the variables which to read the dimensions for. If multiple variables are requested, their dimensions will be merged and returned in a single vector. |

## Value

Named numeric vector with the names and sizes of the dimensions for the requested variables.

## Author(s)

History:
0.0 - 2017-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code

## Examples

```
# Create an array from R
file_path <- tempfile(fileext = '.nc')
a <- array(1:9, dim = c(member = 3, time = 3))
# Store into a NetCDF twice, as two different variables
ArrayToNc(list(var_1 = a, var_2 = a + 1), file_path)
# Read the dimensions and variables in the created file
fnc <- NcOpen(file_path)
fnc_dims <- NcReadDims(fnc)
var_names <- NcReadVarNames(fnc)
# Read the two variables from the file into an R array
a_from_file <- NcToArray(fnc, vars_to_read = var_names)
NcClose(fnc)
# Check the obtained array matches the original array
print(a)
print(a_from_file[1, , ])
```

NcReadVarNames *Read Names of Variables in a NetCDF File*

## Description

Reads the names of the variables in a NetCDF file and returns them as a vector of character strings.

## Usage

```
NcReadVarNames(file_to_read)
```

## Arguments

| | |
|---|---|
| file_to_read | Path to the file to be read or a NetCDF object as returned by easyNCDF::NcOpen or ncdf4::nc_open. |

## Value

Vector of character strings with the names of the variables in the NetCDF file.

## Author(s)

History:
0.0 - 2017-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code

## Examples

```
# Create an array from R
file_path <- tempfile(fileext = '.nc')
a <- array(1:9, dim = c(member = 3, time = 3))
# Store into a NetCDF twice, as two different variables
ArrayToNc(list(var_1 = a, var_2 = a + 1), file_path)
# Read the dimensions and variables in the created file
fnc <- NcOpen(file_path)
fnc_dims <- NcReadDims(fnc)
var_names <- NcReadVarNames(fnc)
# Read the two variables from the file into an R array
a_from_file <- NcToArray(fnc, vars_to_read = var_names)
NcClose(fnc)
# Check the obtained array matches the original array
print(a)
print(a_from_file[1, , ])
```

NcToArray *Read a NetCDF File Into an R Array*

## Description

Reads one or a set of variables together with metadata items from a single NetCDF file into an R array (see package 'startR' to read data from multiple files/data sets). Indices to retrieve (not necessarily consecutive) can be specified for each of the dimensions. Depending on the format of the request, the variables will be merged in into a single extended array or returned in a list with an array for each variable. The different variables in the file are considered to be stored along a dimension called 'var', so reading a variable 'foo' with dimensions 'lat' and 'lon' would result in an array with the dimensions c('var' = 1, 'lat' = n_lats, 'lon' = n_lons).

## Usage

```
NcToArray(file_to_read, dim_indices = NULL, vars_to_read = NULL,
          drop_var_dim = FALSE, unlist = TRUE,
          expect_all_indices = FALSE, allow_out_of_range = TRUE)
nc2a(file_to_read, dim_indices = NULL, vars_to_read = NULL,
     drop_var_dim = FALSE, unlist = TRUE,
     expect_all_indices = FALSE, allow_out_of_range = TRUE)
```

## Arguments

`file_to_read` Path to the file to be read or a NetCDF object as returned by `easyNCDF::NcOpen` or `ncdf4::nc_open`. See package 'startR' if need to read data from multiple files/data sets.

`dim_indices` Named list with numeric vectors of indices to take for each dimension. The names should correspond to the dimension names which to take the indices for. Non-consecutive indices can be specified. If `expect_all_indices = FALSE` (default), it is not mandatory to specify the indices for all (or even any of) the dimensions. In that case all the indices along such dimensions will be read in. If `expect_all_indices = TRUE`, then indices for all the dimensions have to be specified for the function to return a data array. In that case, `NA` can be used to request all indices for a dimension if desired.

Since this function considers the variables in a NetCDF file are stored along a 'var' dimension, indices for the (actually non-existing) 'var'/'variable' dimension can be specified. They can be specified in 3 ways:
- A vector of numeric indices: e.g. `list(var = c(1, 3, 5))` to take the 1st, 3rd and 5th found variables.
- A vector of character strings with variable names: e.g. `list(var = c('foo', 'bar'))`.
- A list of vectors with numeric indices or character strings: e.g. `list(var = list(c(1, 3, '`
Vectors with combined numeric indices and character strings are accepted.
Whereas the first two options will return a single extended array with the merged variables, the second option will return a list with an array for each requested variable.

`vars_to_read` This parameter is a shortcut to (and has less priority than) specifying the requested variable names via `dim_indices = list(var = ...)`. It is useful when all the indices for all the requested variables have to be taken, so

the parameter dim_indices can be skipped, but still only a specific variable or set of variables have to be taken. Check the documentation for the parameter dim_indices to see the three possible ways to specify this parameter.

drop_var_dim Whether to drop the 'var' dimension this function assumes (read description). If multiple variables are requested in a vector and unlist = TRUE, the drop won't be performed (not possible).

unlist Whether to merge the resulting array variables into a single array if possible (default) or not. Otherwise a list with as many arrays as requested variables is returned.

expect_all_indices
Whether the function should stop if indices are not provided for all the dimensions of any of the requested variables (TRUE) rather than assuming that all the indices are requested for the unspecified dimensions (FALSE). By default the later is done (FALSE).

allow_out_of_range
Whether to allow indices out of range (simply disregard them) or to stop if indices out of range are found.

## Value

Array or list of arrays with the data for one or more than one of the requested variables (depending on the parameters). The dimensions are named. The arrays contain the attribute 'variables' with the metadata items found in the NetCDF file.

## Author(s)

History:
0.0 - 2017-03 (N. Manubens, <nicolau.manubens at bsc.es>) - Original code

## Examples

```
# Create an array from R
file_path <- tempfile(fileext = '.nc')
a <- array(1:9, dim = c(member = 3, time = 3))
# Store into a NetCDF twice, as two different variables
ArrayToNc(list(var_1 = a, var_2 = a + 1), file_path)
# Read the dimensions and variables in the created file
fnc <- NcOpen(file_path)
fnc_dims <- NcReadDims(fnc)
var_names <- NcReadVarNames(fnc)
# Read the two variables from the file into an R array
a_from_file <- NcToArray(fnc, vars_to_read = var_names)
NcClose(fnc)
# Check the obtained array matches the original array
print(a)
print(a_from_file[1, , ])

# Example with extra dimensions of length 1
# Creating sample data with singleton dimensions. Only dimensions 'a', 'b' and
# 'c' are of length > 1.
test_var <- array(1:24, dim = c(1, 1, 2, 1, 1, 3, 1, 4, 1, 1))
names(dim(test_var)) <- c('x', 'y', 'a', 'z', 't', 'b', 'u', 'c', 'v', 'w')
# Storing the data into a NetCDF file
a2nc(list(test_var = test_var), file_path)
```

```
# Reading the data back
fff <- nc2a(file_path, list(a = NA, b = NA, c = NA), vars_to_read = 'test_var')
# By default, if no indices are provided for the singleton dimensions, they are
# automatically read in and preserved
dim(fff)
# Reading the data back with expect_all_indices = TRUE
fff <- nc2a(file_path, list(a = NA, b = NA, c = NA), vars_to_read = 'test_var',
            expect_all_indices = TRUE)
# If indices for all dimensions are not provided and expect_all_indices = TRUE,
# the function crashes, except if those dimensions are of length 1. In that
# case, the function ignores those (those are dropped)
dim(fff)
```

---

`Subset` *Subset a Data Array*

---

## Description

This function allows to subset (i.e. slice, take a chunk of) an array, in a similar way as done in the function `take()` in the package plyr. There are two main inprovements:

The input array can have dimension names, either in `names(dim(x))` or in the attribute 'dimensions', and the dimensions to subset along can be specified via the parameter `along` either with integer indices or either by their name.

There are additional ways to adjust which dimensions are dropped in the resulting array: either to drop all, to drop none, to drop only the ones that have been sliced or to drop only the ones that have not been sliced.

If an array is provided without dimension names, dimension names taken from the parameter `dim_names` will be added to the array.

## Usage

```
Subset(x, along, indices, drop = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | A multidimensional array to be sliced. It can have dimension names either in `names(dim(x))` or either in the attribute 'dimensions'. |
| `along` | Vector with references to the dimensions to take the subset from: either integers or dimension names. |
| `indices` | List of indices to take from each dimension specified in 'along'. If a single dimension is specified in 'along' the indices can be directly provided as a single integer or as a vector. |
| `drop` | Whether to drop all the dimensions of length 1 in the resulting array, none, only those that are specified in 'along', or only those that are not specified in 'along'. The possible values are, respectively: 'all' or TRUE, 'none' or FALSE, 'selected', and 'non-selected'. |

## Examples

```
# Create an array from R with data for 3 'var', 3 'member' and 3 'time'
a <- array(1:27, dim = c(var = 3, member = 3, time = 3))
# Take a subset with all 'member' and 'time' for the 1st 'var'
b <- Subset(a, 'var', 1)
```

# Index

∗Topic **datagen**
ArrayToNc, 2
NcClose, 6
NcOpen, 7
NcReadDims, 8
NcReadVarNames, 9
NcToArray, 10
∗Topic **dplot**
Subset, 12

a2nc *(ArrayToNc)*, 2
ArrayToNc, 2

nc2a *(NcToArray)*, 10
NcClose, 6
NcOpen, 7
NcReadDims, 8
NcReadVarNames, 9
NcToArray, 10

Subset, 12