makeNcmlDataset

Description

Generates a NcML file from a collection of netCDF files.

Usage

```
makeNcmlDataset(source.dir, ncml.file)
```

Arguments

- `source.dir`: character string indicating a valid path of the directory containing the files
- `ncml.file`: character string indicating the NcML file name (and path, default to working directory), including the extension *.ncml*.

The output is a NcML file named as `file.name` which will be stored in the `output.dir`.

Details

- All files of the same dataset should be put together in the same directory, indicated by the `source.dir` argument.
- Currently the function works only with netCDF (*.nc*) file collections.
- A number of useful recommendations regarding dataset naming are provided ?here

Value

Creates a NcML file at the specified location

Notes

A NcML file is a ?XML representation of netCDF metadata. This is approximately the same information one gets when dumping the header of a netCDF file (e.g. by typing on the terminal the command `ncdump -h`). By means of NcML it is possible to create virtual datasets by modifying and aggregating other datasets, thus providing maximum flexibility and ease of access to data stored in collections of files containing data from different variables/time slices. The function `makeNcmlDataset` is intended to deal with reanalysis, forecasts and other climate data products, often consisting of collections of netCDF files corresponding to different variables and partitioned by years/decades or other time slices. It operates by applying to types of ?aggregation operations:

1. `Union`: Performs the union of all the dimensions, attributes, and variables in multiple NetCDF files
2. `JoinExisting`: Variables of the same name (in different files) are connected along their existing, outer dimension, called the aggregation dimension. In this case the aggregation dimension is *time*.

Examples

An example of this function is provided in the Examples section?

dataInventory

Description

Provides summary information about the main characteristics of a NcML dataset.

Usage

```
dataInventory(ncml.file)
```

Arguments

- `ncml.file`: a character string indicating the full path to the virtual dataset (the `NcML` file). This can be either a path containing the directory and name of the file, or an appropriate URL in case the dataset is remotely accessed (e.g., via the ?SPECS-EUPORIAS THREDDS).

Value

The output of the function consists of a list of variable length, depending on the number of variables contained in the dataset, following this structure:

- `Description`: Description of the variable
- `DataType`: Character string indicating data type (i.e. float ...)
- `Units`: Character string indicating the units of the variable
- `TimeStep`: A `difftime` class object representing the time interval between consecutive values in the time dimension axis

  `Dimensions`: A list of length *n*, containing the following information for each of the *n* dimensions:
  - `Type`: Character vector indicating the type of dimension (e.g. Time, Lon, Pressure ...)
  - `Units`: Character vector indicating the units of the dimension axis
  - `Values`: A vector containing all the dimension values. This might be a vector of `POSIXlt` class in case of a dimension of type *time*, or numeric in other cases.

Details

A common need prior to data analysis is to get an overview of all data available and their structure (variables, dimensions, units, geographical extent, time span ...). Note that the function provides an overview of the raw data as they are stored in the original data files. The units may change after loading the function if conversions are applied via dictionary.

Examples

An example of this function is provided in the Examples section?

loadObservations

Description

Loads observational station data from standard station datasets stored in *.csv* files.

Usage

```
loadObservations(source.dir, var, standard.vars=TRUE, stationID, startDate=NULL, endDate=NULL, season=NULL)
```

Arguments

- `source.dir`: Character string indicating the full path to the directory where the data are stored (see Details).
- `var`: Character string indicating the variable to load.
- `standard.vars`: Logical. Default to `FALSE`. If `TRUE` a dictionary must be available (see Details)
- `stationID`: Character vector indicating the identification code of the stations to load.
- `startDate`: Optional character string in the form "Year-month-day" (e.g. `"1950-01-01"`) indicating the starting day of the time series to retrieve. Default to first record available.
- `endDate`: Optional character string in the form "Year-month-day" (e.g. `"2000-12-31"`) indicating the last day of the time series to retrieve. Default to last record available.
- `season`: Optional. A vector of integers specifying the desired season (in months, January=1 ...). Options include one to several months. If `NULL` (the default), the function will return all records within the interval defined by `startDate` and `endDate`. For instance, `season = c(12,1,2)` will retrieve the data series for the standard boreal winter (DJF), period = 6:8 for summer (JJA) and so on. See details.

Details

This function works with standard *.csv* observational datasets. It allows loading data from one or several stations at a time. The dictionary is the table that translates the variable as stored in the dataset to the standard variables defined in the vocabulary. More details ?here

In the case of boreal winter selection (`season=c(12,1,2)`) the function will tie strictly to the time interval defined by the `startDate`and `endDate` arguments, and therefore will not retrieve data from the previous December, nor from the next January and February before/after the start/end years defined (this has a different behaviour than `loadSystem4`, which is more specifically oriented to seasonal forecast data.)

Value

A list with the containing the following elements:

- `StationID`: character vector with the identification codes of the stations loaded.
- `LatLonCoords`: 2D matrix with the LatLon? coordinates of the stations loaded (the order of the rows corresponds to the order othe 'StationID` field).
- `Altitude`: numeric vector with the altitude of the stations.
- `Dates`: A `POSIXlt` class vector of dates of the time series returned.
- `Data`: 2D matrix with the time series for each of the stations arranged by columns.

Examples

An example of this function is provided in the Examples section?

loadData

Description

Loads selected dimensional slices of a NcML dataset. The function is intended to deal with gridded data (interpolated surfaces, reanalysis, RCMs/GCMs ...)

Usage

```
loadData(dataset, var, standard.vars=FALSE, lonLim=NULL, latLim=NULL, level=NULL, startDate=NULL, endDate=NULL, season=NUL
```

Arguments

- `var`: Character string indicating the variable to load.
- `standard.vars`: Logical. Default to `FALSE`. If `TRUE` a dictionary must be available (see Details)
- `lonLim`: Vector of length = 2, with minimum and maximum longitude coordinates, in decimal degrees, of the bounding box selected. Alternatively, point selection is performed indicating the longitude as a single numeric value. If `NULL` (the default), the whole longitudinal range of the dataset is selected. See details.
- `latLim`: Vector of length = 2, with minimum and maximum latitude coordinates, in decimal degrees, of the bounding box selected. Alternatively, point selection is performed indicating the latitude as a single numeric value. If `NULL` (the default), the whole latitudinal range of the dataset is selected. See details.
- `level`: Vertical level, if defined for the variable. See details
- `startDate`: Optional character string in the form "Year-month-day" (e.g. "`1950-01-01`") indicating the starting day of the time series to retrieve. Default to first record available. See details.
- `endDate`: Optional character string in the form "Year-month-day" (e.g. "`2000-12-31`") indicating the last day of the time series to retrieve. Default to last record available. See details.
- `season`: Optional. A vector of integers specifying the desired season (in months, January=1 ...). Options include one to several months. If `NULL` (the default), the function will return all records within the interval defined by `startDate` and `endDate`. For instance, `season = c(12,1,2)` will retrieve the data series for the standard boreal winter (DJF), period = 6:8 for summer (JJA) and so on. See details.

Details

The function can select the whole spatial domain covered by the dataset, spatial windows defined by the minimum and maximum corner coordinates, and single grid-cell values. In the last two cases, the function operates by finding the closest grid-points to the coordinates introduced.

For variables with different vertical levels, only defined level values will be allowed, otherwise getting an error. The function does not look for the closest level to the value introduced, in order to avoid confusions. The function `dataInventory` is useful for finding the valid level values defined for a particular variable.

In the case of time slice selection in sub-daily datasets, the function will retrieve all records belonging to the days indicated. For instance, if `endDate=2000-12-31` and the dataset has a time resolution of 6h, the last date returned will be *"2000-12-31 18:00:00"*.

In the case of boreal winter selection (`season=c(12,1,2)`) the function will tie strictly to the time interval defined by the `startDate`and `endDate` arguments, and therefore will not retrieve data from the previous December, nor from the next January and February before/after the start/end years defined (this has a different behaviour than `loadSystem4`, which is more specifically oriented to seasonal forecast data.)

Value

A list with the following components:

- `VarName`: name of the variable returned.
- `Level`: Vertical level requested. `NULL` if the variable has no vertical levels defined.
- `Dates`: A `POSIXlt` time class vector of length *i*
- `LatLonCoords`: A 2-D matrix of *j* rows (where *j* = number of grid points selected) and two columns corresponding to the latitude and longitude coordinates respectively.
- `Data`: a 2-D matrix of *i* rows x *j* columns, of *i* times and *j* grid-points

Examples

An example of this function is provided in the Examples section?

loadSystem4

Description

Loads hindcast/forecast data from ECMWF's System4 model by remotely accessing the SPECS-EUPORIAS THREDDS Data Server.

Usage

```
loadSystem4(dataset, var, members, lonLim, latLim, season, years, leadMonth)
```

Arguments

- `dataset`: A character string indicating the full URL path to the OPeNDAP dataset. Currently, the accepted values correspond to the System4 ?available datasets at the SPECS-EUPORIAS THREDDS Data Server.
- `var`: Variable code (see Details).
- `members`: Optional. A vector of integers indicating the members to be loaded. Default to `NULL`, which loads all members available. For instance, `members=1:5` will retrieve the first five members.
- `lonLim`: Vector of length = 2, with minimum and maximum longitude coordinates, in decimal degrees, of the bounding box selected.
- `latLim`: Vector of length = 2, with minimum and maximum latitude coordinates, in decimal degrees, of the bounding box selected.
- `season`: A vector of integers specifying the desired season (in months, January=1 ...). Options include one to several months. If `NULL` (the default), the function will return a whole year forecast from January to December. For instance, `period = c(12,1,2)` will retrieve the forecast for the standard boreal winter (DJF), `period = 6:8` for summer (JJA) and so on.
- `years`: Optional vector of years to select. Default to all available years. Note that in the case of a year-crossing season for a particular year period (e.g. winter DJF, `season = c(12,1,2)` and `years = 1981:2000`), by convention the first season returned will be DJF 1980/81, if available (otherwise a warning message is given).
- `leadMonth`: Lead month forecast time corresponding to the first month of the specified season. Note that `leadMonth = 1` for `season = 1` (January) corresponds to the December initialization forecasts. The effect of the lead time in the forecast for a particular season can be analyzed by just changing this parameter.

Details

Currently, accepted values for the argument `var` are `tas`, `tasmin`, `tasmax`, `pr` or `mslp`, as internally defined in the vocabulary of System4 following the nomenclature displayed in the table below. However, note that new variables and datasets will be progressively included. Further details regarding the nature and temporal aggregation of these variables can be obtained through the `dataInventory` function.

Short Name	Long name	Units	Instantaneous
tasmax	**Maximum temperature at 2 metres**	**degC**	**No**
tasmin	**Minimum temperature at 2 metres**	**degC**	**No**
tas	**Mean temperature at 2 metres**	**degC**	**Yes**
pr	**Total precipitation accumulated**	**mm**	**No**
mslp	**Mean sea level pressure**	**Pa**	**Yes**

Value

The output returned by the function consists of a list with the following elements providing the necessary information for data representation and analysis:

- `VarName`: Character string indicating the variable long name, as defined in the vocabulary (see Table above)
- `VarUnits`: Character string. Units of the variable, as returned in `MemberData`
- `TimeStep`: A `difftime` class object. Indicates the time span of each forecast time
- `MemberData`: This is a list of length *n*, where *n* = number of members of the ensemble selected by the `member` argument. Each element of the dataset is a 2-D matrix of *i* rows x *j* columns, of *i* forecast times and *j* grid-points
- `LatLonCoords`: A 2-D matrix of *j* rows (where *j* = number of grid points selected) and two columns corresponding to the latitude and longitude coordinates respectively.
- `RunDates`: A `POSIXlt` time object corresponding to the initialization times selected. There is an initialization time associated to each forecast time.

  `ForecastDates`: A list with two `POSIXlt` time elements of length *i*, corresponding to the rows of each matrix in `MemberData`. The list contain tow elements:
  - `Start`: Starting times of the verification period of the variable
  - `End`: End time of the verification period of the variable

Note

A worked example describing a multi-model selection of a dataset is presented in the tutorial, which can be downloaded ?here,

Examples

An example of this function is provided in the Examples section?