Package 'efts'

April 26, 2018

Type Package

Title High-Level Functions to Read and Write Ensemble Forecast Time Series in netCDF

Description The binary file format 'netCDF' is developed primarily for climate, ocean and meteorological data, and 'efts' is a package to read and write Ensemble Forecast Time Series data in 'netCDF'. 'netCDF' has traditionally been used to store time slices of gridded data, rather than complete time series of point data.
'efts' facilitates data handling stored in 'netCDF' files that follow a convention devised in the domain of ensemble hydrologic forecasting, but possibly applicable in other domains. 'efts' uses reference class objects to provide a high level interface to read and write such data, wrapping lower level operations performed using 'ncdf4'.

Version 0.9-0

Date 2018-04-22

URL https://github.com/jmp75/efts

BugReports https://github.com/jmp75/efts/issues

VignetteBuilder knitr

Depends R (>= 3.1)

Imports methods, stringr (>= 1.2), lubridate (>= 1.7), magrittr, xts (>= 0.10), plyr, ncdf4 (>= 1.16), udunits2

Suggests testthat, zoo, knitr

License GPL-2

RoxygenNote 6.0.1

NeedsCompilation no

Author Jean-Michel Perraud [aut, cre],
David Robertson [aut],
James Bennett [aut]

Maintainer Jean-Michel Perraud <jean-michel.perraud@csiro.au>

Repository CRAN

Date/Publication 2018-04-26 11:30:39 UTC

R topics documented:

`efts-package` *Accessing ensemble forecast time series (EFTS) data set stored in netCDF formats*

Description

Accessing ensemble forecast time series (EFTS) data set stored in netCDF formats

Details

Package:	efts
Type:	Package
Version:	0.9-0
Date:	2018-04-22
Release Notes:	Facilities and stricter checks for compliance with netCDF EFTS conventions. Fixes for addressing feedback
License:	GPL-2

Accessing ensemble forecast time series (EFTS) data set stored in netCDF formats, without the need for lower-level ncdf4 operations. See `open_efts` `create_efts` for code examples to read/write files using this package.

This work was carried out in the CSIRO Water for Healthy Country National Research Flagship and was supported by the Water Information Research and Development Alliance between CSIRO and the Australian Bureau of Meteorology.

Version	Date	Notes
0.8.0	2018-04-08	Improve and test subsetting and reordering of multidimensional arrays retrieved via ncdf4. Remove is
0.7.0	2018-02-14	Initial commit to github
0.6.x	2017	Non public releases of a package originally used only for unit test purposes

Author(s)

Jean-Michel Perraud <jean-michel.perraud_at_csiro.au>

`check_is_utc` *Check that a date-time is in the UTC time zone, and return the date time offset 'zero'*

Description

Check that a date-time is in the UTC time zone, and return the date time offset 'zero'

Usage

```
check_is_utc(d)
```

Arguments

d	an object coercible to a POSIXct

Value

a character, the time zone offset string '+0000'

Examples

```
start_time <- ISOdate(year=2010, month=08, day=01, hour = 12, min = 0, sec = 0, tz = 'UTC')
check_is_utc(d=start_time)
```

create_efts *Creates a EftsDataSet for write access to a netCDF EFTS data set*

Description

Creates a EftsDataSet for write access to a netCDF EFTS data set

Usage

```
create_efts(fname, time_dim_info, data_var_definitions, stations_ids,
  station_names = NULL, nc_attributes = NULL, optional_vars = NULL,
  lead_length = 48, ensemble_length = 50, lead_time_tstep = "hours")
```

Arguments

fname	file name to create to. The file must not exist already.
time_dim_info	a list with the units and values defining the time dimension of the data set
data_var_definitions	a data frame, acceptable by create_variable_definitions, or list of netCDF variable definitions, e.g. list(rain_sim=list(name='rain_sim', longname='ECMWF Rainfall ense
stations_ids	station identifiers, coercible to an integer vector (note: may change to be a more flexible character storage)
station_names	optional; names of the stations
nc_attributes	a named list of characters, attributes for the whole file, including mandatory ones: title, institution, source, catchment, comment. You may use create_global_attributes as a starting template.
optional_vars	a data frame defining optional netCDF variables. For a templated default see default_optional_variable_definitions_v2_0 and https://github.com/jmp75/efts/blob/107c553045a37e6ef36b2eababf6a299e7883d50/docs/netcdf_for_water_forecasting.md#optional-variables
lead_length	length of the lead forecasting time series.
ensemble_length	number of ensembles, i.e. number of forecasts for each point on the main time axis of the data set
lead_time_tstep	string specifying the time step of the forecast lead length.

Value

A EftsDataSet object

Examples

```
# NOTE
# The sample code below is purposely generic; to produce
# a data set conforming with the conventions devised for
# ensemble streamflow forecast you will need to
# follow the additional guidelines at
# https://github.com/jmp75/efts/blob/master/docs/netcdf_for_water_forecasting.md

fname <- tempfile()

stations_ids <- c(123,456)
nEns <- 3
nLead <- 4
nTimeSteps <- 12

timeAxisStart <- ISOdate(year=2010, month=08, day=01, hour = 14, min = 0, sec = 0, tz = 'UTC')
time_dim_info <- create_time_info(from=timeAxisStart,
  n=nTimeSteps, time_step = "hours since")

# It is possible to define variables for three combinations of dimensions.
# dimensions '4' ==> [lead_time,station,ens_member,time]
# dimensions '3' ==> [station,ens_member,time]
# dimensions '2' ==> [station,time]

variable_names <- c('var1_fcast_ens','var2_fcast_ens', 'var1_obs',
  'var2_obs', 'var1_ens','var2_ens')

va <- create_var_attribute_definition(
  type = 2L,
  type_description = "accumulated over the preceding interval",
  dat_type = "der",
  dat_type_description = paste(rep(c("var1", "var2"), 3), "synthetic test data"),
  location_type = "Point")


(varDef <- create_variable_definition_dataframe(
  variable_names=variable_names,
  long_names = paste(variable_names, 'synthetic data'),
  dimensions = c(4L,4L,2L,2L,3L,3L),
  var_attributes = va))

glob_attr <- create_global_attributes(
  title="data set title",
  institution="my org",
  catchment="Upper_Murray",
  source="A journal reference, URL",
  comment="example for vignette")

(opt_metadatavars <- default_optional_variable_definitions_v2_0())

snc <- create_efts(
```

```
  fname=fname,
  time_dim_info=time_dim_info,
  data_var_definitions=varDef,
  stations_ids=stations_ids,
  nc_attributes=glob_attr,
  optional_vars = opt_metadatavars,
  lead_length=nLead,
  ensemble_length=nEns,
  lead_time_tstep = "hours")

# Following is code that was used to create unit tests for EFTS.
# This is kept in this example to provide sample on now to write data of various dimension.
td <- snc$get_time_dim()
m <- matrix(ncol=nEns, nrow=nLead)
for (rnum in 1:nLead) {
    for (cnum in 1:nEns) {
      m[rnum,cnum] = rnum*0.01 + cnum*0.1
  }
}
#      [,1] [,2] [,3]
# [1,] 0.11 0.21 0.31
# [2,] 0.12 0.22 0.32
# [3,] 0.13 0.23 0.33
# [4,] 0.14 0.24 0.34
for (i in 1:length(td)) {
  for (j in 1:length(stations_ids)) {
    station <- stations_ids[j]
    var1Values <- i + 0.1*j + m
    var2Values <- 2*var1Values
    dtime = td[i]
    snc$put_ensemble_forecasts(var1Values, variable_name = variable_names[1],
      identifier = station, start_time = dtime)
    snc$put_ensemble_forecasts(var2Values, variable_name = variable_names[2],
      identifier = station, start_time = dtime)
  }
}

timeSteps <- 1:length(td)
for (j in 1:length(stations_ids)) {
  var3Values <- timeSteps + 0.1*j
  var4Values <- var3Values + 0.01*timeSteps + 0.001*j

  station <- stations_ids[j]
  snc$put_single_series(var3Values, variable_name = variable_names[3], identifier = station)
  snc$put_single_series(var4Values, variable_name = variable_names[4], identifier = station)
}

for (j in 1:length(stations_ids)) {

  var5Xts <- matrix(rep(1:nEns, each=nTimeSteps) + timeSteps + 0.1*j, ncol=nEns)

  # [time,ens_member] to [ens_member,time], as expected by put_ensemble_series
  var5Values <- t(var5Xts)
```

```
  var6Values <- 0.25 * var5Values

  station <- stations_ids[j]
  snc$put_ensemble_series(var5Values, variable_name = variable_names[5], identifier = station)
  snc$put_ensemble_series(var6Values, variable_name = variable_names[6], identifier = station)
}

# We can get/put values for some metadata variables:
snc$get_values("x")
snc$put_values(c(1.1, 2.2), "x")
snc$put_values(letters[1:2], "station_name")

# Direct get/set access to data variables, however, is prevented;
#  the following would thus cause an error:
# snc$get_values("var1_fcast_ens")

snc$close()
# Cleaning up temp file:
if (file.exists(fname))
  file.remove(fname)
```

`create_efts_variables` *Create netCDF variables according to the definition*

Description

Create netCDF variables according to the definition

Usage

```
create_efts_variables(data_var_def, time_dim_info, num_stations, lead_length,
  ensemble_length, optional_vars, lead_time_tstep)
```

Arguments

`data_var_def`	a list, with each item itself a list suitable as a variable definition argument to create_data_variable
`time_dim_info`	a list with the units and values defining the time dimension of the data set
`num_stations`	number of (gauging) stations identifying points in the data set
`lead_length`	length of the lead forecasting time series.
`ensemble_length`	number of ensembles, i.e. number of forecasts for each point on the main time axis of the data set

optional_vars — a data frame defining optional netCDF variables. For a templated default see default_optional_variable_definitions_v2_0 and https://github.com/jmp75/efts/blob/107c553045a37e6ef36b2eababf6a299e7883d50/docs/netcdf_for_water_forecasting.md#optional-variables

lead_time_tstep — string specifying the time step of the forecast lead length.

See Also

See create_efts for examples

---

create_global_attributes

Define a set of global attributes for netCDF files.

Description

The conventions require a set of global attributes to be present, see https://github.com/jmp75/efts/blob/master/docs/netcdf_for_water_forecasting.md#global-attributes. This function is recommended to define these attributes.

Usage

```
create_global_attributes(title, institution, source, catchment, comment,
  strict = FALSE)
```

Arguments

title	text, a succinct description of what is in the dataset
institution	text, Where the original data was produced
source	text, published or web-based references that describe the data or methods used to produce it
catchment	text, the catchment for which the data is created. White spaces are replaced with underscores
comment	text, miscellaneous information
strict	logical, if true perform extra checks on the input information

`create_nc_dims` *Creates dimensions for a netCDF EFTS data set*

Description

Creates dimensions for a netCDF EFTS data set. Note that end users are unlikely to need to use this function directly, hence this is not exported

Usage

```
create_nc_dims(time_dim_info, str_len = 30, lead_length = 1,
  ensemble_length = 1, num_stations = 1)
```

Arguments

`time_dim_info`	a list with the units and values defining the time dimension of the data set
`str_len`	maximum length of the character for the station identifiers.
`lead_length`	length of the lead time.
`ensemble_length`	number of ensembles, i.e. number of forecasts for each point on the main time axis of the data set
`num_stations`	number of stations

Value

A list of ncdf4 dimensions

See Also

See `create_efts` for examples

`create_netcdf_time_axis` *Create a time axis unit known to work for netCDF*

Description

Create a time axis unit known to work for netCDF

Usage

```
create_netcdf_time_axis(d, time_step = "hours since", tzoffset)
```

Arguments

d	an object coercible to a POSIXct
time_step	the character prefix to put before the date, in the netCDF time axis unit definition.
tzoffset	an optional character, the time offset from UTC, e.g. '+1000' for 10 hours ahead of UTC. Can be missing, in which case it must be explicitly a UTC time. Note that the tzoffset completely supersedes the time zone if present.

Value

a character, the axis units to use for the netCDF 'time' dimension

Examples

```
start_time <- ISOdate(year=2010, month=08, day=01, hour = 12, min = 0, sec = 0, tz = 'UTC')
create_netcdf_time_axis(d=start_time)
start_time <- ISOdate(year=2015, month=10, day=04, hour = 01,
  min = 0, sec = 0, tz = 'Australia/Sydney')
create_netcdf_time_axis(d=start_time, tzoffset='+1000')
```

---

create_time_info — *Helper function to create the definition of the time dimension for use in a netCDF file*

Description

Helper function to create the definition of the time dimension for use in a netCDF file. Defaults to create an hourly axis.

Usage

```
create_time_info(from, n, time_step = "hours since", time_step_delta = 1L,
  tzoffset)
```

Arguments

from	the start date of the time axis
n	length of the time dimension
time_step	unit prefix in the time dimension units
time_step_delta	integer, length of time units between each steps
tzoffset	an optional character, the time offset from UTC, e.g. '+1000' for 10 hours ahead of UTC. Can be missing, in which case 'from' must be explicitly a UTC time. Note that the tzoffset completely supersedes the time zone if present.

Value

A list with keys units and values

See Also

See `create_efts` for examples

Examples

```
timeAxisStart <- ISOdate(2015, 10, 4, 0, 0, 0, tz = "Australia/Canberra")
(time_dim_info <- create_time_info(from = timeAxisStart, n = 24L,
  time_step = "hours since", time_step_delta = 3L, tzoffset = "+1000"))

# Note that the time zone information of thes sart date is NOT
# used by create_time_info; the tzoffset argument takes precedence
timeAxisStart <- ISOdate(2015, 10, 4, 0, 0, 0, tz = "Australia/Perth")
(time_dim_info <- create_time_info(from = timeAxisStart, n = 24L,
  time_step = "hours since", time_step_delta = 3L, tzoffset = "+1000"))
```

`create_variable_definition` *Create a variable definition*

Description

Create a variable definition usable by the function `create_efts_variables` to create netCDF variables.

Usage

```
create_variable_definition(name, longname = "", units = "mm",
  missval = -9999, precision = "double", dim_type = "4",
  var_attribute = create_var_attribute_definition())
```

Arguments

`name`	variable name
`longname`	variable long name
`units`	variable units
`missval`	value code for missing data
`precision`	precision
`dim_type`	dimension type (EFTS integer code)
`var_attribute`	list of attributes for the netCDF variable to create

Value

a list

Examples

```
var_def <- create_variable_definition(name='rain_der',
  longname='Rainfall ensemble forecast derived from some prediction', units='mm',
  missval=-9999.0, precision='double', var_attribute=list(type=2L,
    description="accumulated over the preceding interval",
    dat_type = "der", dat_type_description="AWAP data interpolated from observations",
    location_type = "Point"))
```

---

create_variable_definitions

Create variable definitions from a data frame

Description

Given a data frame as input, create a list of variable definitions usable by the function create_efts_variables to create netCDF variables.

Usage

```
create_variable_definitions(dframe)
```

Arguments

dframe	a data frame, one line is one variable definition. Must have at least the following column names: 'name', 'longname', 'units', 'missval', 'precision', 'type', 'type_description', 'location_type'

Value

a list of length equal to the number of rows in the input data frame

See Also

See create_efts for examples

Examples

```
varsDef = data.frame(name=letters[1:3], stringsAsFactors=FALSE)
varsDef$longname=paste('long name for', varsDef$name)
varsDef$units='mm'
varsDef$missval=-999.0
varsDef$precision='double'
varsDef$type=2
varsDef$type_description='accumulated over the previous time step'
varsDef$location_type='Point'
```

```
str(create_variable_definitions(varsDef))
```

`create_variable_definition_dataframe`
Create a variables definition data frame

Description

Create a variable definition usable by the function `create_variable_definitions` to create netCDF variables. The use of this function is not compulsory to create a EFTS netCDF schema, just offered as a convenience.

Usage

```
create_variable_definition_dataframe(variable_names,
  long_names = variable_names, standard_names = variable_names,
  units = "mm", missval = -9999, precision = "double", dimensions = 4L,
  var_attributes = create_var_attribute_definition())
```

Arguments

`variable_names`	character vector, names of the variables
`long_names`	character vector, long names of the variables (defaults to variable_names if missing)
`standard_names`	character vector, standard names of the variables (optional, defaults to variable_names)
`units`	character vector, units for the variable(s)
`missval`	numeric vector, missing value code(s) for the variable(s)
`precision`	character vector, precision of the variables
`dimensions`	character or integer vector, number of dimensions each variable (2, 3 or 4)
`var_attributes`	a list of named attributes. See `create_var_attribute_definition`

Value

a data frame suitable for `create_variable_definition`

See Also

See `create_variable_definition` and `create_efts` for examples

`create_var_attribute_definition` *Create variable attribute definition*

Description

Create variable attribute definition

Usage

```
create_var_attribute_definition(type = 2L,
  type_description = "accumulated over the preceding interval",
  dat_type = "der",
  dat_type_description = "AWAP data interpolated from observations",
  location_type = "Point")
```

Arguments

`type`	A data type identifier, as a coded description.
`type_description`	description of this data type identifier.
`dat_type`	a character, the type of data stored in this variable
`dat_type_description`	a character, human readable description of the data stored in this variable
`location_type`	a character, type of location, e.g. 'Point'

Value

a list of attributes, describing the type of variable stored

Examples

```
va <- create_var_attribute_definition(type=2L,
  type_description='accumulated over the preceding interval', location_type='Point')
vdef <- create_variable_definition(name='rain_sim',
  longname='Rainfall ensemble forecast derived from some prediction',
  units='mm', missval=-9999.0, precision='double',
  var_attribute=va)
```

default_optional_variable_definitions_v2_0 *Provide a template definition of optional geolocation variables*

Description

Provide a template definition of optional geolocation and geographic variables x, y, area and elevation. See https://github.com/jmp75/efts/blob/107c553045a37e6ef36b2eababf6a299e7883d50/docs/netcdf_for_water_forecasting.md#optional-variables.

Usage

```
default_optional_variable_definitions_v2_0()
```

Value

a data frame

See Also

See create_variable_definition and create_efts for examples

EftsDataSet-class *Reference class convenient for access to a Ensemble Forecast Time Series in netCDF file.*

Description

Reference class convenient for access to a Ensemble Forecast Time Series in netCDF file.

Fields

`time_dim` a cached POSIXct vector, the values for the time dimension of the data set.

`time_zone` the time zone for the time dimensions of this data set.

`identifiers_dimensions` a cache, list of values of the primary data identifiers; e.g. station_name or station_id

`stations_varname` name of the variable that stores the names of the stations for this data set.

Methods

`get_all_series(variable_name = "rain_obs", dimension_id = get_stations_varname())` Return a multivariate time series, where each column is the series for one of the identifiers (e.g. rainfall station identifiers)

`get_dim_names()` Gets the name of all dimensions in the data set

`get_ensemble_for_stations(variable_name = "rain_sim", identifier, dimension_id = "ens_member", star` Return a time series, representing a single ensemble member forecast for all stations over the lead time

`get_ensemble_forecasts(variable_name = "rain_sim", identifier, dimension_id = get_stations_varname(` Return a time series, ensemble of forecasts over the lead time

`get_ensemble_forecasts_for_station(variable_name = "rain_sim", identifier, dimension_id = get_stati` Return an array, representing all ensemble member forecasts for a single stations over all lead times

`get_ensemble_series(variable_name = "rain_ens", identifier, dimension_id = get_stations_varname())` Return an ensemble of point time series for a station identifier

`get_ensemble_size()` Length of the ensemble size dimension

`get_lead_time_count()` Length of the lead time dimension

`get_single_series(variable_name = "rain_obs", identifier, dimension_id = get_stations_varname())` Return a single point time series for a station identifier. Falls back on get_all_series if the argument "identifier" is missing

`get_station_count()` Length of the lead time dimension

`get_stations_varname()` Gets the name of the variable that has the station identifiers

`get_time_dim()` Gets the time dimension variable as a vector of date-time stamps

`get_time_unit()` Gets the time units of a read time series, i.e. "hours since 2015-10-04 00:00:00 +1030". Returns the string "hours"

`get_time_zone()` Gets the time zone to use for the read time series

`get_utc_offset(as_string = TRUE)` Gets the time zone to use for the read time series, i.e. "hours since 2015-10-04 00:00:00 +1030". Returns the string "+1030" or "-0845" if as_string is TRUE, or a lubridate Duration object if FALSE

`get_values(variable_name)` Gets (and cache in memory) all the values in a variable. Should be used only for dimension variables

`get_variable_dim_names(variable_name)` Gets the names of the dimensions that define the geometry of a given variable

`get_variable_names()` Gets the name of all variables in the data set

`index_for_identifier(identifier, dimension_idifier = get_stations_varname())` Gets the index at which an identifier is found in a dimension variable

`index_for_time(dateTime)` Gets the index at which a date-time is found in the main time axis of this data set

`initialize(nc = NULL)` Create an object wrapping an ncdf4 object

`put_ensemble_forecasts(x, variable_name = "rain_sim", identifier, dimension_id = get_stations_varna` Puts one or more ensemble forecast into a netCDF file

`put_ensemble_forecasts_for_station(x, variable_name = "rain_sim", identifier, dimension_id = "ens_m`
Puts a single ensemble member forecasts for all stations into a netCDF file

`put_ensemble_series(x, variable_name = "rain_ens", identifier, dimension_id = get_stations_varname(`
Puts an ensemble of time series, e.g. replicate rainfall series

`put_single_series(x, variable_name = "rain_obs", identifier, dimension_id = get_stations_varname(),`
Puts a time series, or part thereof

`put_values(x, variable_name)` Puts all the values in a variable. Should be used only for dimension variables

`set_time_zone(tzone_id)` Sets the time zone to use for the read time series

`summary()` Print a summary of this EFTS netCDF file

See Also

See `create_efts` and `open_efts` for examples on how to read or write EFTS netCDF files using this dataset.

`eftsdotDollarNames` *method for dollar-tab-completion in R consoles.*

Description

method for dollar-tab-completion in R consoles. It may be unnecessary in the future but such a method was required at some time to at least avoid some issues in RStudio. We may also want to customise matches compared to default reference classes.

Usage

```
## S3 method for class 'EftsDataSet'
.DollarNames(x, pattern = "")
```

Arguments

`x`	A reference class object
`pattern`	the regex pattern to search for in potential methods. Default value is possibly required by some versions of RStudio

find_utc_offset *Finds the UTC offset in a date-time string*

Description

Finds the UTC offset in a date-time or time axis specification string such as 'hours since 2015-10-04 00:00:00 +1030'

Usage

```
find_utc_offset(time_units, as_string = TRUE)
```

Arguments

time_units	the string to process
as_string	a boolean. If true, return the time offset as a character, otherwise return a difftime object.

Value

the time offset as a character, or as a difftime object.

Examples

```
x <- "hours since 2015-10-04 00:00:00 +1023"
find_utc_offset(x)
find_utc_offset(x, FALSE)
x <- "hours since 2015-10-04 00:00:00 -0837"
find_utc_offset(x)
find_utc_offset(x, FALSE)
x <- "hours since 2015-10-04 00:00:00"
find_utc_offset(x)
find_utc_offset(x, FALSE)
```

get_start_date *Retrieves the first date of the time dimension from a netCDF file*

Description

Retrieves the first date of the time dimension from a netCDF file of daily data, given the units found in the netCDF attribute for the time dimension

Usage

```
get_start_date(time_units, time_zone = "UTC")
```

Arguments

time_units	The string description of the units of the time dimension e.g. 'days since 1980-01-01' or 'hours since 2010-08-01 13:00:00 +0000'
time_zone	the time zone to use for the returned value.

Value

A POSIXct object, origin of the time dimension as defined

Examples

```
x <- "hours since 2015-10-04 00:00:00 +1023"
get_start_date(x)
get_start_date(x,time_zone = 'UTC')
get_start_date(x,time_zone = 'Australia/Perth')
get_start_date(x,time_zone = 'Australia/Canberra')
```

---

get_time_dimension *Retrieves the time dimension from a netCDF file*

Description

Retrieves the time dimension from a netCDF file

Usage

```
get_time_dimension(ncfile, time_dim_name = "time", time_zone = "UTC")
```

Arguments

ncfile	an object of class ncdf4
time_dim_name	The name of the time dimension, by default 'time' as per the CF conventions.
time_zone	the time zone to use for the returned value.

Value

A vector of Dates

get_time_step_function *Detect the unit of the time step in the time axis unit*

Description

Detect the unit of the time step in the time axis unit

Usage

```
get_time_step_function(time_units)
```

Arguments

time_units	The string description of the units of the time dimension e.g. 'days since 1980-01-01' or 'hours since 2010-08-01 13:00:00 +0000'

Value

A duration function from lubridate

get_time_units *Retrieves the unit string of the time dimension from a netCDF file*

Description

Retrieves the unit string of the time dimension from a netCDF file

Usage

```
get_time_units(ncfile, time_dim_name = "time")
```

Arguments

ncfile	an object of class ncdf4
time_dim_name	The name of the time dimension, by default 'time' as per the CF conventions.

Value

a character

NetCdfDataSet-class *Reference class convenient for access to a netCDF file.*

Description

Reference class convenient for access to a netCDF file. Note for internal implementation that ncdf4 objects are basically lists with a class attribute. This class [NetCdfDataSet-class] is used as a parent class to the [EftsDataSet-class] class but may be reused and expanded for other types of netCDF data.

Fields

`ncfile` an object of class ncdf4

Methods

`initialize(nc = NULL)` Create an object wrapping an ncdf4 object

open_efts *Creates a EftsDataSet for access to a netCDF EFTS data set*

Description

Creates a EftsDataSet for access to a netCDF EFTS data set

Usage

```
open_efts(ncfile, writein = FALSE)
```

Arguments

`ncfile`	name of the netCDF file, or an object of class 'ncdf4'
`writein`	if TRUE the data set is opened in write mode

Value

A EftsDataSet object

Examples

```
library(efts)
ext_data <- system.file('extdata', package='efts')
ens_fcast_file <- file.path(ext_data, 'Upper_Murray_sample_ensemble_rain_fcast.nc')
stopifnot(file.exists(ens_fcast_file))
snc <- open_efts(ens_fcast_file)
(variable_names <- snc$get_variable_names())
(stations_ids <- snc$get_values('station_id'))
nEns <- snc$get_ensemble_size()
nLead <- snc$get_lead_time_count()
td <- snc$get_time_dim()
stopifnot('rain_fcast_ens' %in% variable_names)

ens_fcast_rainfall <- snc$get_ensemble_forecasts('rain_fcast_ens',
  stations_ids[1], start_time=td[2])
names(ens_fcast_rainfall) <- as.character(1:ncol(ens_fcast_rainfall))
plot(ens_fcast_rainfall, legend.loc='right')

snc$close()
```

---

pad_global_attribute *Add a value to a global attribute of a netCDF file*

Description

Add a value to a global attribute of a netCDF file

Usage

```
pad_global_attribute(nc, attribute_name, attribute_value, sep = "\n")
```

Arguments

nc	an object 'ncdf4'
attribute_name	the name of the global attribute to add to
attribute_value	the value to pad
sep	separator to add between the existing value and the padded value.

Index