Package 'M3'

June 14, 2024

Type Package

Title Reading M3 Files

Version 0.4

Date 2024-06-12

Author Jenise Swall <jswall@vcu.edu>

Maintainer Jenise Swall <jswall@vcu.edu>

Depends R (>= 2.10), ncdf4, sf, maps, mapdata

Imports stats

Description Provides functions to read in and manipulate air quality model output from Models3-formatted files. This format is used by the Community Multiscale Air Quality (CMAQ) model.

License Unlimited

NeedsCompilation no

Repository CRAN

Date/Publication 2024-06-14 10:00:06 UTC

Contents

`M3-package` *Functions to read in and manipulate air quality model output from files in Models3 format*

Description

Provides functions to read in and manipulate air quality model output from Models3-formatted files. This format is used by the Community Multiscale Air Quaility (CMAQ) model.

Note

The original version of this software program was reviewed in accordance with U.S. Environmental Protection Agency policy and approved for publication. Subsequent releases have not been reviewed by U.S. EPA, since author is no longer employed there. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

References

For detailed information about the format and conventions of the Models3 format, see https://www.cmascenter.org/ioapi/documentation/all_versions/html/AA.html.

`combine.date.and.time` *Combine date and time to obtain date-time in POSIX format*

Description

Combine date and time to obtain date-time in POSIX format

Usage

```
combine.date.and.time(date, time)
```

Arguments

`date`	Date in Date format or as character string in format "YYYY-MM-DD".
`time`	Time as list with hours (`hrs`), minutes (`mins`), and seconds (`secs`) components or as character string in format HH:MM:SS (with hours ranging from 00-23).

Value

A date-time in R's POSIX class.

Note

This function is called by `get.datetime.seq`, `get.M3.var`, and `var.subset`, but it will probably not be called by most users.

Author(s)

Jenise Swall

See Also

DateTimeClasses, strptime

Examples

```
## This function can accept dates as a character string:
combine.date.and.time(date="2011-05-03", time="16:15:30")

## Or, the dates can be in R's Date format.
combine.date.and.time(date=as.Date("2011-05-03"), time="16:15:30")

## The time can also be given as a list:
combine.date.and.time(date="2011-05-03", time=list(hrs=16, mins=15, secs=30))
```

---

decipher.M3.date *Decipher Models3 date format (YYYYDDD) into R's Date class.*

Description

Decipher Models3 date format (YYYYDDD) into R's Date class.

Usage

```
decipher.M3.date(M3.date)
```

Arguments

M3.date	Date (numeric) in the format YYYYDDD, where DDD is a Julian day (since the beginning of year YYYY).

Value

Date specified by YYYYDDD in R's Date class.

Note

For more information about M3 date-time conventions, see https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html. This function is called by function get.datetime.seq, but it will probably not be called by most users.

Author(s)

Jenise Swall

References

https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html

See Also

DateTimeClasses,
get.datetime.seq,
decipher.M3.time

Examples

```
## Returns 2011-03-10, which is the 69th day of 2011.
decipher.M3.date(2011069)

## Returns 2012-02-29.  This leap day is the 60th day of 2012.
decipher.M3.date(2012060)
```

`decipher.M3.time` *Decipher Models3 time format (HHMMSS) into hours, minutes, and seconds.*

Description

Decipher Models3 time format (HHMMSS) into hours, minutes, and seconds.

Usage

```
decipher.M3.time(M3.time)
```

Arguments

`M3.time`	Time (numeric) in the format HHMMSS (hours, minutes, seconds).

Value

List with the following components

`hrs`	hours
`mins`	minutes
`secs`	seconds

Note

For more information about Models3 date-time conventions, see https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html. This function is called by function `get.datetime.seq`, but it will probably not be called by most users.

Author(s)

Jenise Swall

References

https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html

See Also

DateTimeClasses,
get.datetime.seq,
decipher.M3.date

Examples

```
## Note that the function breaks up the (numeric) input,
## where hours are designated by 00-23, minutes by 00-59,
## seconds by 00-59.
decipher.M3.time(030105)
```

---

get.canusamex.bds *Obtain map boundaries for Canada, USA, and Mexico*

Description

Obtain map boundaries of Canada, USA, and Mexico, including state boundaries, in longitude and latitude coordinates.

Usage

```
get.canusamex.bds()
```

Details

Retrieves a data frame containing the coordinates (longitude and latitude) for drawing the boundaries of Canada, USA (including states), and Mexico. This function is intended to be called by the function get.map.lines.M3.proj; in practice, it should rarely be called directly by the user.

Value

Data frame specifying the polylines needed to plot the national (Canada, USA, Mexico) and state outlines. This matrix has two columns, with longitude in the first column and latitude in the second.

Author(s)

Jenise Swall

See Also

map, get.map.lines.M3.proj

Examples

```
## Set up a plotting region (in longitude/latitude) that includes an
## eastern portion of the Canada/USA border.
plot(c(-82,-67), c(39,49), type="n", xlab="Longitude", ylab="Latitude")
## Superimpose national boundaries from "world" database, which is
## fairly low-resolution (since it includes worldwide national boundaries).
map("world", regions="canada", add=TRUE)
## Now, if we try to superimpose the the USA state boundaries from the
## higher resolution "state" database, we have a conflict.  (See
## particularly the Maine border.)
map("state", add=TRUE, col="blue")
## The high-resolution national boundaries in database "worldHires" (in
## mapdata) also don't match up with the state lines.
map("worldHires", add=TRUE, col="magenta")

## Instead, we get the national boundaries (Canada, USA, Mexico) at
## high-resolution from database "worldHires" and the state boundaries
## (without the coastlines and national boundaries) from the "state"
## database.
dev.new()
plot(c(-82,-67), c(39,49), type="n", xlab="Longitude", ylab="Latitude")
lines(get.canusamex.bds())
```

---

get.coord.for.dimension

Get the grid coordinates for the grid rows or columns.

Description

For either the rows or the columns, return the coordinates of the centers or the edges of the grid cells.

Usage

```
get.coord.for.dimension(file, dimension, position = "ctr", units)
```

Arguments

file	Name of Models3-formatted file of interest.
dimension	If "column"/"col", will obtain coordinates for columns; if "row" will obtain coordinates for rows.
position	Choose whether to obtain coordinates of cell edges or centers for either grid rows or columns. If "ctr" (default), get the cell center. If "lower", get bottom or left cell edge. If "upper", get top or right cell edge.

`units` Units for coordinates of grid rows or columns. Must be one of "m", "km", or "deg". If unspecified, the default is "deg" if the file has a longitude/latitude grid, and "km" otherwise.

Value

A list containing two elements, `coords` and `units`. If `dimension` is "row", return as element `coords` a vector containing the y-coordinates of the centers ("ctr"), left ("lower"), or right ("upper") edges of each row, depending on the value of argument `position`. If `dimension` is "column" or "col", return as element `coords` a vector containing the x-coordinates of the centers ("ctr"), left ("lower"), or right ("upper") edges of each row, depending on the value of argument `position`. In both cases, return as element `units` the units of the coordinates (can be "km", "m", or "deg").

Note

Usually, the user will not call this function directly; instead, it will be called by other functions such as `get.matrix.all.grid.cell.ctrs` and `get.M3.var`.

Author(s)

Jenise Swall

See Also

`get.matrix.all.grid.cell.ctrs`, `get.M3.var`, `get.grid.info.M3`

Examples

```
## As mentioned in notes above, user will not typically call
## this function directly.

## Find the path to a demo file with lambert conic conformal projection.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")

## Get a list of the x-coordinates of the centers of the cells.
x.ctrs <- get.coord.for.dimension(lcc.file, dimension="col", units="km")
```

---

get.datetime.seq *Read in a sequence of date-time steps from a Models3-formatted file.*

Description

Read the date-time steps in the Models3-formatted file. Put these into R's datetime format.

Usage

```
get.datetime.seq(file)
```

Arguments

`file`	File name of Models3-formatted file which contains the date-time information of interest.

Details

This function relies on the R package **ncdf4** to read information from Models3-formatted files, since the Models3 format is built on netCDF (https://www.unidata.ucar.edu/software/netcdf/).

Value

Vector of sequence of datetimes included in the Models3-formatted file, in POSIXct format.

Warning

This code assumes that the time step is not negative. For instance, the Models3 I/OAPI does allow for negative time steps, but these negative time steps will NOT be handled properly by this function. For more information about Models3 date-time conventions, see https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html.

Note

This function is called by function get.M3.var, but it will probably not be called by most users.

Author(s)

Jenise Swall

References

Information about the Models3 date-time conventions is available at https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html.

See Also

DateTimeClasses, seq.POSIXt, get.M3.var

Examples

```
## As mentioned in notes above, user will not typically call
## this function directly.

## Find the path to a demo file with lambert conic conformal projection.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")

## Get vector containing date-times available in this file.
datetime.seq <- get.datetime.seq(lcc.file)
```

`get.grid.info.M3` *Get information about the grid used by the air quality model*

Description

Pull information about the grid used from the Models3-formatted file. This includes information such as the origin of the grid (lower left corner coordinates in grid units), cell spacing, etc.

Usage

```
get.grid.info.M3(file)
```

Arguments

`file`	File name of Models3-formatted file which contains information about the projection. Currently, this function can only handle files with Lambert conic conformal, polar stereographic, and longitude/latitude projections.

Details

This function assumes that the projection is either Lambert conic conformal or polar stereographic projection. Information about grid cell size, extent of grid, etc. is stored in the global attributes of the Models3-formatted file, which this function reads.

Value

List with the following components:

`x.orig`	X-coordinate of the origin point of the grid (lower left corner, coordinates in model projection units)
`y.orig`	Y-coordinate of the origin point of the grid (lower left corner, coordinates in model projection units)
`x.cell.width`	Width of grid cells in x-direction (in model projection units)
`y.cell.width`	Width of grid cells in y-direction (in model projection units)
`hz.units`	Units of the projection (“m”, “km”, or “deg”)
`ncols`	Number of columns of grid cells
`nrows`	Number of rows of grid cells
`nlays`	Number of vertical layers

Warning

Currently, this function can only handle files with Lambert conic conformal, polar stereographic, and longitude/latitude projections.

Note

This function relies on the R package **ncdf4** to read information from Models3-formatted files, since the Models3 format is built on netCDF
(https://www.unidata.ucar.edu/software/netcdf/).
Usually, the user will not call this function directly; instead, it will be called by the function get.coord.for.dimension.

Author(s)

Jenise Swall

See Also

get.proj.info.M3, get.coord.for.dimension

Examples

```
## As mentioned in notes above, user will not typically call
## this function directly.

## Find the path to a demo file with lambert conic conformal projection.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")
## Get a list containing information about the grid in this file.
grid.info <- get.grid.info.M3(lcc.file)


## Find the path to a demo file with polar stereographic projection.
polar.file <- system.file("extdata/surfinfo_polar.ncf", package="M3")
## Get a list containing information about the grid in this file.
grid.info <- get.grid.info.M3(polar.file)
```

---

get.M3.var *Read in variable values from Models3-formatted files*

Description

Read in variable values from Models3-formatted files.

Usage

```
get.M3.var(file, var, lcol, ucol, lrow, urow, llay, ulay,
                ldatetime, udatetime, hz.units)
```

Arguments

file	Name of Models3-formatted file to be read.
var	Name (character string) or number (positive integer) of variable whose values are to be read.
lcol	Lower column bound (positive integer) to be read. The default is to read all columns.
ucol	Upper column bound (positive integer) to be read. The default is to read all columns.
lrow	Lower row bound (positive integer) to be read. The default is to read all rows.
urow	Upper row bound (positive integer) to be read. The default is to read all rows.
llay	Lower layer bound (positive integer) to be read. The default is to read the first layer only.
ulay	Upper layer bound (positive integer) to be read. The default is to read the first layer only.
ldatetime	Starting date-time (Date or POSIX class) in GMT.
udatetime	Starting date-time (Date or POSIX class) in GMT.

The default is to read all date-times. If the file is time-independent, the one available time step will be read and user input for ldatetime and udatetime will be disregarded.

hz.units	Units associated with grid cell horizontal dimensions. Default is degrees ("deg") if the data is indexed according to longitude/latitude and kilometers ("km") otherwise. If the file is not indexed according to longitude/latitude, the user could choose meters ("m").

Details

This function assumes that the projection is either Lambert conic conformal or polar stereographic projection. It also assumes that the Models3-formatted file is either time-independent or time-stepped; it cannot be of type circular-buffer.

Value

List with the following components:

data	Array holding the specified variable values. Array is four-dimensional, unless the Models3-formatted file is time-independent (which we assume if TSTEP attribute is 0), in which case it is three-dimensional. Dimensions are columns, rows, layers, date-time steps.
data.units	Contains the units associated with data.
x.cell.ctr	X-coordinates of grid cell centers, in units given by hz.units.
y.cell.ctr	Y-coordinates of grid cell centers, in units given by hz.units.
hz.units	Contains the units associated with the horizontal dimensions of the grid cells (km, m, or deg). These are the units in which x.cell.ctr and y.cell.ctr are given.

rows	Row numbers extracted.
cols	Column numbers extracted.
layers	Layer numbers extracted.
datetime	Date-time steps (in POSIX format) associated with the variable, if the file is not time-independent. For time-independent files, element datetime is set to NULL.

Note

This function relies on the R package **ncdf4** to read information from Models3-formatted files, since the Models3 format is built on netCDF (https://www.unidata.ucar.edu/software/netcdf/).

Author(s)

Jenise Swall

References

https://www.cmascenter.org/ioapi/documentation/all_versions/html/VBLE.html,
https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html

See Also

ncvar_get, ncatt_get, get.coord.for.dimension, get.datetime.seq, combine.date.and.time

Examples

```
## Find the path to the first demo file (with polar
## stereographic projection).
polar.file <- system.file("extdata/surfinfo_polar.ncf", package="M3")

## Read in the terrain elevation variable.
elev <- get.M3.var(file=polar.file, var="HT")
## Make a plot.
image(elev$x.cell.ctr, elev$y.cell.ctr, elev$data[,,1],
      xlab="Projection x-coord (km)", ylab="Projection y-coord (km)",
      zlim=range(elev$data[,,1]), col=topo.colors(20))
## Find national boundaries on this projection, superimpose them on
## the plot.
world.bds <- get.map.lines.M3.proj(file=polar.file, database="world")$coords
lines(world.bds)

## Subset to a smaller geographic area in southwestern U.S.
subset.elev <- var.subset(elev, llx=-2100, urx=0, lly=-6500, ury=-4000)
## Make a plot of this subset.
image(subset.elev$x.cell.ctr, subset.elev$y.cell.ctr,
      subset.elev$data[,,1], xlab="Projection x-coord (km)",
      ylab="Projection y-coord (km)", zlim=range(subset.elev$data[,,1]),
      col=topo.colors(20))
## Find state boundaries on this projection, superimpose them on the plot.
```

```
state.bds <- get.map.lines.M3.proj(file=polar.file)$coords
lines(state.bds)



## Find the path to second demo file (with Lambert conic
## conformal projection).
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")

## Read in the ozone for July 4 for eastern U.S.
oz <- get.M3.var(file=lcc.file, var="O3", lcol=80, urow=95,
                     ldatetime=as.Date("2001-07-04"),
                     udatetime=as.Date("2001-07-04"))
## Make a plot.
image(oz$x.cell.ctr, oz$y.cell.ctr, oz$data[,,1,1],
      xlab="Projection x-coord (km)", ylab="Projection y-coord (km)",
      zlim=range(oz$data), col=heat.colors(15))

## Find map lines on this projection, superimpose them on the plot.
state.bds <- get.map.lines.M3.proj(file=lcc.file)$coords
lines(state.bds)
```

`get.map.lines.M3.proj` *Get map lines in the model projection units*

Description

Get map lines in the model projection units.

Usage

```
get.map.lines.M3.proj(file, database = "state", units, ...)
```

Arguments

`file`	File name of Models3-formatted file providing the model projection.
`database`	Geographical database to use. Choices include "state" (default), "world", "worldHires", "canusamex", etc. Use "canusamex" to get the national boundaries of the Canada, the USA, and Mexico, along with the boundaries of the states. The other choices ("state", "world", etc.) are the names of databases included with the **maps** and **mapdata** packages. (See the `map` function.) Default is "state".
`units`	Units for coordinates of grid rows or columns. Must be one of "m", "km", or "deg". If unspecified, the default is "deg" if the file has a longitude/latitude grid, and "km" otherwise.
`...`	Other arguments to pass to `get.proj.info.M3` function. In this case, the only relevant argument would be the earth radius to use for the projection in `file`.

Details

This function depends on the **maps** and **mapdata** packages to get the appropriate map boundary lines (for states, countries, etc.), **ncdf4** to read the projection information from the Models3-formatted file (using a call to function `get.proj.info.M3`), and **sf** to project the boundary lines to the specified projection.

Value

Map lines for the projection described in `file` in either kilometers or meters (depending on value of units.km). This is a matrix, with x-coordinates in the first column and y-coordinates in the second column.

Warning

This function will only work with Lambert conic conformal or polar stereographic projections.

Author(s)

Jenise Swall

See Also

`get.proj.info.M3`, `map`, `sf_project`

Examples

```
## Find the path to the demo file.
polar.file <- system.file("extdata/surfinfo_polar.ncf", package="M3")

## Read in the terrain elevation variable.
elev <- get.M3.var(file=polar.file, var="HT")
## Make a plot.
image(elev$x.cell.ctr, elev$y.cell.ctr, elev$data[,,1],
      xlab="Projection x-coord (km)", ylab="Projection y-coord (km)",
      zlim=range(elev$data[,,1]), col=heat.colors(15))

## Superimpose national boundaries on the plot
nat.bds <- get.map.lines.M3.proj(file=polar.file, database="world")$coords
lines(nat.bds)


## Subset to a smaller geographic area in southwestern U.S.
subset.elev <- var.subset(elev, llx=-2000, urx=0, lly=-6500, ury=-4000)
## Make a plot of this subset.
image(subset.elev$x.cell.ctr, subset.elev$y.cell.ctr,
      subset.elev$data[,,1], xlab="Projection x-coord (km)",
      ylab="Projection y-coord (km)", zlim=range(subset.elev$data[,,1]),
      col=heat.colors(15))

## Superimpose Mexico, US, and Candadian national borders on the plot,
## along with state borders.
```

```
canusamex.borders <- get.map.lines.M3.proj(file=polar.file, "canusamex")$coords
lines(canusamex.borders)
```

`get.matrix.all.grid.cell.ctrs` *Obtain a matrix giving the locations of the grid cell centers*

Description

Obtain a two-column matrix giving the locations of the grid cell centers in grid units.

Usage

```
get.matrix.all.grid.cell.ctrs(file, units)
```

Arguments

`file`	File name of Models3-formatted file which contains information about the projection. Currently, this function can only handle files with a Lambert conic conformal or polar stereographic projection.
`units`	Units for coordinates of grid rows and columns. Must be one of "m", "km", or "deg". If unspecified, the default is "deg" if the file has a longitude/latitude grid, and "km" otherwise.

Value

Matrix with number of rows equal to the number of grid cells and two columns. The first column contains the x-coordinate of the grid cell centers; the second column contains the y-coordinate of the grid cell centers. The rows are listed in order such that all cell centers with same y-coordinate are grouped together, with groups ordered by the y-coordinate, and ordered within these groups by the x-coordinate).

Warning

Currently, this function can only handle files with Lambert conic conformal, polar stereographic, and longitude/latitude projections.

Note

This function relies on calls `get.coord.for.dimensions`.

Author(s)

Jenise Swall

See Also

`get.coord.for.dimension`

Examples

```
## As mentioned in notes above, user will not typically call
## this function directly.

## Find the path to a demo file on lambert conic conformal projection.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")
## Get a list of the x- and y-coordinates of the centers of all
## grid cells.
ctrs <- get.matrix.all.grid.cell.ctrs(lcc.file, units="km")
```

`get.proj.info.M3` *Obtain information about the projection used in the Models3 file*

Description

Obtain information about the projection used by in the Models3-formatted file. Build a string describing the projection which can be used by the R package sf.

Usage

```
get.proj.info.M3(file, earth.radius=6370000)
```

Arguments

`file`	File name of Models3-formatted file which contains information about the projection. Currently, this function can only handle files with a Lambert conic conformal, polar stereographic, and longitude/latitude projections.
`earth.radius`	Radius of the earth (in meters), which is assumed to be spherical by the Models3 I/O API. Default value is 6 370 000 m. Note that the radius in some previous version of the Models3 I/O API was 6 370 997 m, which may be appropriate for some users. For instance, this latter value was used in previous packages supplied by Battelle.

Details

This function assumes that the file uses the Lambert conic conformal projection, polar stereographic projection, or longitude/latitude.
The Models3 I/O API assumes a spherical earth. The default value for `earth.radius` is 6 370 000 m (sometimes referred to as "sphere 20"), which is the current value used in the Models3 I/O API. Note that the radius in some previous versions of the Models3 I/O API was 6 370 997 m, and this value was also used in previous packages for reading Models3-formatted files, which were developed for EPA by Battelle.

Value

String describing model projection, which can be utilized by the **sf** package (for projections to and from longitude/latitude, for example).

Warning

Currently, this function can only handle files with Lambert conic conformal, polar stereographic, and longitude/latitude projections.

Note

This function relies on the R package **ncdf4** to read information from Models3-formatted files, since the Models3 format is built on netCDF
(https://www.unidata.ucar.edu/software/netcdf/).
The string that is returned by this function is appropriate for use with package **sf**. Usually, the user will not call this function directly; instead, it will be called by other functions in this package.

Author(s)

Jenise Swall

References

See information about the meaning of Models3 I/O API projection arguments at
https://www.cmascenter.org/ioapi/documentation/all_versions/html/GRIDS.html.

See Also

project.lonlat.to.M3, project.M3.to.lonlat,
project.M3.1.to.M3.2, get.map.lines.M3.proj

Examples

```
## Find the path to a demo file on lambert conic conformal projection.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")
## Get string with projection information, using previous value for
## the earth's radius.
get.proj.info.M3(lcc.file, earth.radius=6370997)

## Find the path to a demo file on polar stereographic projection.
polar.file <- system.file("extdata/surfinfo_polar.ncf", package="M3")
## Get string with projection information.
get.proj.info.M3(polar.file)
```

---

project.lonlat.to.M3 *Project coordinates from longitude/latitude to model units.*

Description

Project coordinates from longitude/latitude to model units as specified according to the projection given by a user-designated Models3-formatted file.

Usage

```
project.lonlat.to.M3(longitude, latitude, file, units, ...)
```

Arguments

longitude	Vector of longitudes for the points to be projected.
latitude	Vector of latitudes for the points to be projected.
file	File name of Models3-formatted file which contains information about the projection to which you want x and y to be projected.
units	Units in which to return x and y. Options are "km" or "m", with "km" as the default.
...	Other arguments to pass to get.proj.info.M3 function. In this case, the only relevant argument would be the earth radius to use for the projection in file.

Details

This function uses the function sf_project from the package **sf** to project from longitude/latitude to the projection defined by the Models3-formatted file.

Value

A list containing the elements coords and units. The element coords contains a matrix of coordinates using the projection in file. The element units contains the units of the coordinates, as specifed by units or "km" by default.

Author(s)

Jenise Swall

See Also

project.M3.to.lonlat, project.M3.1.to.M3.2, get.proj.info.M3

Examples

```
## List of state capital longitudes/latitudes
## (from http://www.xfront.com/us_states).
capitals <- data.frame(x=c(-84.39,-86.28,-81.04,-86.78,-78.64,-84.86),
                       y=c(33.76,32.36,34.00,36.17,35.77,38.20),
                       name=c("Atlanta", "Montgomery", "Columbia",
                                 "Nashville", "Raleigh", "Frankfort")
                        )
## Plot these on a map, with state lines.
plot(capitals$x, capitals$y)
map("state", add=TRUE)

## Now, put these on the same Lambert conic conformal projection used
## in the demo file below.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")
```

```
lcc.capitals <- project.lonlat.to.M3(capitals$x, capitals$y, lcc.file)

## Put these on a new plot.
dev.new()
plot(lcc.capitals$coords)
## Project state lines to this projection.
lcc.map <- get.map.lines.M3.proj(lcc.file)
lines(lcc.map$coords)
```

`project.M3.1.to.M3.2` *Project coordinates based on projection in the first file to the projection given in the second*

Description

Project coordinates based on projection in the first Models3-formatted file to the projection given in the second Models3-formatted file.

Usage

```
project.M3.1.to.M3.2(x, y, from.file, to.file, units, ...)
```

Arguments

`x`	x-coordinates in model units from projection in first file
`y`	y-coordinates in model units from projection in first file
`from.file`	Name of Models3-formatted file with the same model projection as `x` and `y`. These coordinates will be re-projected to the projection given in file2.
`to.file`	Name of Models3-formatted file with the model projection to which you want `x` and `y` to be projected.
`units`	Units of `x` and `y`. The coordinates returned will also be in these units.
`...`	Other arguments to pass to `get.proj.info.M3` function. In this case, the only relevant argument would be the earth radius to use when doing the projections.

Details

This function calls `get.proj.info.M3` which reads the projection information (using the package **ncdf4**) and transforms it to strings that are understood by functions in **sf**.

Value

A list containing the elements `coords` and `units`. The element `coords` contains a matrix of coordinates using projection in `to.file`. The element `units` contains the units of the coordinates, which are the same as those specified for input `x` and `y`.

Warning

This function assumes the projections in `from.file` and `to.file` are Lambert conic conformal or polar stereographic.

Author(s)

Jenise Swall

See Also

`project.lonlat.to.M3`, `project.M3.to.lonlat`, `get.proj.info.M3`

Examples

```
## Find the path to a demo file with lambert conic conformal projection.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")

## Read in the ozone for July 4 for eastern U.S.
lcc.oz <- get.M3.var(file=lcc.file, var="O3",
                     lcol=90, ucol=130, lrow=30, urow=80,
                     ldatetime=as.Date("2001-07-04"),
                     udatetime=as.Date("2001-07-04"))

## Get the cell centers for this subset.
east.ctrs <- expand.grid(lcc.oz$x.cell.ctr, lcc.oz$y.cell.ctr)
plot(east.ctrs, cex=0.3, xlab="x", ylab="y")

## Find map lines on this projection, superplot.
lcc.state.bds <- get.map.lines.M3.proj(file=lcc.file)$coords
lines(lcc.state.bds, col="darkblue")


## Find the path to a demo file with polar stereographic projection.
polar.file <- system.file("extdata/surfinfo_polar.ncf", package="M3")

## Put the cell centers from the subsetted ozone data on this polar
## stereographic projection.
polar.oz <- project.M3.1.to.M3.2(east.ctrs[,1], east.ctrs[,2],
                                 from.file=lcc.file,
                                 to.file=polar.file,
                                 units=lcc.oz$hz.units)

## Plot the cells centers and boundary lines on the polar
## stereographic projection.
dev.new()
plot(polar.oz$coords)
polar.state.bds <- get.map.lines.M3.proj(file=polar.file)$coords
lines(polar.state.bds, col="darkblue")
```

`project.M3.to.lonlat` *Project coordinates from model units to longitude/latitude*

Description

Project coordinates from model units (as specified according to the projection given by a user-designated Models3-formatted file) to longitude/latitude.

Usage

```
project.M3.to.lonlat(x, y, file, units, ...)
```

Arguments

`x`	x-coordinates of points in model units (meters or kilometers, depeding on the value of `units`
`y`	y-coordinates of points in model units (meters or kilometers, depeding on the value of `units`
`file`	Name of Models3-formatted file with the same projection as `x` and `y`.
`units`	Units of `x` and `y`.
`...`	Other arguments to pass to `get.proj.info.M3` function. In this case, the only relevant argument would be the earth radius to use for the projection in `file`.

Details

This function uses the function `sf_project` from the package **sf** to project to longitude/latitude given the projection defined by the Models3-formatted file.

Value

A list containing the elements `coords` and `units`. The element `coords` contains a matrix of coordinates in longitude/latitude. The element `units` contains the string "deg" to designate that `coords` is in degrees of longitude/latitude.

Author(s)

Jenise Swall

See Also

`project.lonlat.to.M3`, `project.M3.1.to.M3.2`, `get.proj.info.M3`

Examples

```
## List of state capital longitudes/latitudes
## (from http://www.xfront.com/us_states).
capitals <- data.frame(x=c(-84.39,-86.28,-81.04,-86.78,-78.64,-84.86),
                       y=c(33.76,32.36,34.00,36.17,35.77,38.20),
                       name=c("Atlanta", "Montgomery", "Columbia",
                                 "Nashville", "Raleigh", "Frankfort")
                        )
## Plot these on a map, with state lines.
plot(capitals$x, capitals$y)
map("state", add=TRUE)

## Now, put these on the same Lambert conic conformal projection used
## in the demo file below.
lcc.file <- system.file("extdata/ozone_lcc.ncf", package="M3")
lcc.capitals <- project.lonlat.to.M3(capitals$x, capitals$y, lcc.file)

## Now, project them back to longitude/latitude, make sure we get the
## same thing we started with.
chk.capitals <- project.M3.to.lonlat(lcc.capitals$coords[,"x"],
                                     lcc.capitals$coords[,"y"],
                                     lcc.file,
                                     units=lcc.capitals$units)
## These differences should be 0 or something very tiny.
summary(capitals[,c("x", "y")] - chk.capitals$coords)
```

`var.subset` *Subset the array resulting from a call to* `get.M3.var`.

Description

Subset the array resulting from a call to `get.M3.var` using projection units and/or human-readable dates and times.

Usage

```
var.subset(var.info, llx, urx, lly, ury, ldatetime, udatetime,
           hz.strict=TRUE)
```

Arguments

`var.info`	Variable list given by function `get.M3.var`.
`llx`	Lower x-coordinate bound for the subsetted grid using the same units given in element `hz.units` of `var.info`. Default is the left boundary of the grid.
`urx`	Upper x-coordinate bound for the subsetted grid using the same units given in element `hz.units` of `var.info`. Default is the right boundary of the grid.
`lly`	Lower y-coordinate bound for the subsetted grid using the same units given in element `hz.units` of `var.info`. Default is the lower boundary of the grid.

ury	Upper y-coordinate bound for the subsetted grid using the same units given in element hz.units of var.info. Default is the upper boundary of the grid.
ldatetime	Beginning date-time (either Date or POSIX class) bound for the subset. Default is earliest date-time.
udatetime	Ending date-time (either Date or POSIX class) bound for the subset. Default is latest date-time.
hz.strict	If TRUE (default), to be allowed in the subset, the whole grid cell must fit within the bounds given by llx, urx, lly, and ury. If FALSE, grid cells will be included in the subset if any portion of the grid cell's area falls within the given bounds.

Details

If the user wants to subset the variable by row, column, layer, or time step number, this can be accomplished easily using standard R methods for subsetting the array of variable values. This function was written to help the user who does not know the row, column, or time step numbers, but who wants to subset according to human-readable dates and times or according to projection units.

Value

Subsetted array of variable values. (The array's format is preserved.)

Author(s)

Jenise Swall

References

https://www.cmascenter.org/ioapi/documentation/all_versions/html/VBLE.html,
https://www.cmascenter.org/ioapi/documentation/all_versions/html/DATETIME.html

See Also

get.M3.var

Examples

```
## Find the path to the demo file.
polar.file <- system.file("extdata/surfinfo_polar.ncf", package="M3")

## Read in the terrain elevation variable.
elev <- get.M3.var(file=polar.file, var="HT")
## Make a plot.
image(elev$x.cell.ctr, elev$y.cell.ctr, elev$data[,,1],
      xlab="Projection x-coord (km)", ylab="Projection y-coord (km)",
      zlim=range(elev$data[,,1]), col=heat.colors(15))

## Subset to a smaller geographic area in southwestern U.S.
subset.elev <- var.subset(elev, llx=-2000, urx=0, lly=-6500, ury=-4000)
## Make a plot of this subset.
image(subset.elev$x.cell.ctr, subset.elev$y.cell.ctr,
```

```
      subset.elev$data[,,1], xlab="Projection x-coord (km)",
      ylab="Projection y-coord (km)", zlim=range(subset.elev$data[,,1]),
      col=heat.colors(15))

## Superimpose U.S. state boundaries on the plot.
state.bds <- get.map.lines.M3.proj(file=polar.file)$coords
lines(state.bds)
```

Index