EPA/ 600/C-11/005

June, 2011

Quality Assurance Information for R Packages "aqfig" and "M3"

Data/Software

Jenise L. Swall

Atmospheric Modeling and Analysis Division

National Exposure Research laboratory

Office of Research and Development

U.S. Environmental Protection Agency

Research Triangle Park, NC

Quality assurance information for R packages "aqfig" and "M3"

Jenise L. Swall

June 6, 2011

1 What are these packages?

R packages "aqfig" and "M3" are optional modules for use with R statistical software (`http://www.r-project.org`). Package "aqfig" contains functions to aid users in the preparation of publication-quality figures for the display of air quality and other environmental data (e.g., legends of color schemes, preserving the aspect ratio of a plot, color scatterplots for spatial data, etc.). The functions in package "M3" allow users to read in and subset output from Models3-formatted files (e.g., CMAQ output files) and to transform data from one map projection to another. The functions in "m3" replace the most-often used function in the old "RMET" packages, which were prepared by Battelle under contract several years ago and which are now deprecated.

2 Potential users

Most users of this package will probably be personnel in AMAD or in OAQPS divisions with whom AMAD works closely. However, the goal is to get this package cleared so that it can be contributed to the Comprehensive R Archive Network (CRAN). This would make the package easily available to users all over the world, including collaborators and colleagues outside EPA. CRAN also makes contributed packages available and easier to install for R users running operating systems like Windows and Mac OS X.

3 System requirements

Package "aqfig" makes use of functionality in package "geoR", which is available for download from the Comprehensive R Archive Network (CRAN) at `http://www.r-project.org`. Package "M3" requires packages "ncdf4", "rgdal", and "maps", which are also available from CRAN. Both "aqfig" and "M3" were developed and tested using R version 2.12.1 on Linux, but can reasonably be expected to work with R version 2.10.0 and above. Since the packages consist of R code only, they can be used on any operating system for which R and the required packages can be installed. (Note: As of this writing, a binary version of package "ncdf4" is not availabe from CRAN for Windows operation systems, and a binary for "rgdal" is not available for computers running Mac OS X.)

4 Documentation

When a package is built, R performs automated checks to determine whether documentation is present and whether examples included as part of the documentation execute properly. Packages "aqfig" and "M3" have undergone this process successfully. R also requires that each function is documented separately, so that the R help system can provide documentation to the user for each individual function. Such documentation includes explanations of the inputs to the function, the function's outputs, a list of other related functions, and any relevant notes. For most functions (except those that are intended to be called by other functions, rather than directly by users), examples of typical usage are included as well. In addition, for each package, R builds a file which compiles the documentation for each function. These documentation files (PDF format) are included with this clearance package.

5 Disclaimer

These software programs have been reviewed in accordance with U.S. Environmental Protection Agency policy and approved for publication. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

Package 'aqfig'

June 15, 2011

Type Package

Title Functions to help display air quality model output and monitoring data

Version 0.1

Date 2011-02-25

Author Jenise Swall <Swall.Jenise@epa.gov>

Maintainer Jenise Swall <Swall.Jenise@epa.gov>

Depends geoR

Suggests maps

Description This package contains functions to help display air quality model output and monitoring data, such as creating color scatterplots, maps preserving aspect ratio, and color legends.

License Unlimited

LazyLoad yes

R topics documented:

`aqfig-package` *Functions to help draw figures displaying air quality data and model output*

Description

This package contains several functions to help users draw figures to display air pollution measurements and air quality model output. These include functions to generate figures with appropriate dimensions and aspect ratio, to place a legend to the right of a plot, to draw a scatterplot with the points' colors or sizes reflecting the value of a response variable.

Note

This software program has been reviewed in accordance with U.S. Environmental Protection Agency policy and approved for publication. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

`aspect.ratio.plot` *Makes/initializes a plot with proper aspect ratio and of a specified width/height.*

Description

This code allows the user to make/initialize a plot with a proper aspect ratio and of a specified width/height. (Note that this size includes the actual plot and the margins around it.) The function has default specifications for the margins, intended to minimize the margin space for many commonly generated plots, but these can be changed. of a specified size. The user should not use this function if more than one plot is desired on the device (e.g., if using mfrow, mfcol, layout, etc.) It may be used to generate plots in encapsulated PostScript, PDF, Windows metafile, and TIFF formats.

Usage

```
aspect.ratio.plot(x, y, file, dev = "pdf", type = "p", xlim,
   ylim, width, height, mai = c(0.6, 0.6, 0.1, 0.1),
   mgp = c(1.8, 0.5, 0), ...)
```

Arguments

`x`	x-coordinates to plot.
`y`	y-coordinates to plot. If either `x` or `y` is missing, `xlim` and `ylim` are used to initialize an empty plot.
`file`	Name of file to which to save the plot.

dev	Type of file to create. Must be one of "eps" (encapsulated PostScript), "pdf" (portable document format), "wmf" (Windows metafile, available on Windows operating systems only), or "tif" (tagged image format).
type	Default is "p" for points if x and y are given. If they are not, type is set to "n", so that the plot region is only initialized. Information about other choices can be found in the help for the plot function.
xlim	Extent of the x-axis.
ylim	Extent of the y-axis. If either xlim or ylim is missing, it is set based on the range of values in x and y.
width	Width of the figure in inches.
height	Height of the figure in inches. For more information about what happens if either of these is missing, see 'Details' section below.
mai	Settings for margins in inches, as defined for parameter mai in the help for the par() function.
mgp	Settings determining how close axis titles, labels, and lines are to the plot, as defined for parameter mgp in the help for the par function.
...	Any other parameters the user adds will be passed to the plot function, and may include options for color and type ("l" for lines, etc.), among many others.

Details

Usually only one of height and width will be specified. In this case, a figure will be initialized with the given dimension, and with the other dimension determined according to the relative limits of y and x (to preserve the aspect ratio). If both are missing, a figure will be generated of width 3.5in, with the height determined to preserve the aspect ratio. If both are present, the plot with be made with the specified dimensions, disregarding the aspect ratio and generating a warning.

By default, if the user specifies x and y, a scatterplot with an aspect ratio based on the relative range of x and y will be generated. If the user wishes to make another kind of plot (e.g., an image plot), s/he can specify 'type="n"'. If x and y are missing, then a plot will be set up based on 'xlim' and 'ylim' and with 'type="n"'.

Device "wmf" (Windows metafile) is only available when running R on the Windows operating system. An error message will be returned if the user chooses this device on a non-Windows platform.

Value

By default, a scatterplot is written to the device 'dev' with margins and overall figure dimensions as specified. If type="n", sets up a figure and plotting region with the specified dimensions and margins on the device 'dev'.

Author(s)

Jenise Swall

See Also

par, plot, image, Devices

Examples

```
## Load dataset containing longitude and latitude of some ozone
## concentration measurements.
data(ozone1)
## Make plot showing monitoring locations.
aspect.ratio.plot(ozone1$longitude, ozone1$latitude,
                  file="aspect_ratio_tst.pdf", xlab="longitude",
                  ylab="latitude")
## If maps package is available, put on state lines.
if (require("maps"))  map("state", add=TRUE, col="lightgray")
dev.off()
```

---

init.fig.dimen *Initializes a device for a figure with specified dimensions.*

Description

This function allows the user to initialize a device for a figure of a specified size, where this size includes the actual plot and the margins around it. It may be used to generate plots in encapsulated PostScript, PDF, Windows metafile, and TIFF formats.

Usage

```
init.fig.dimen(file, dev = "pdf", width, height,
   mai = c(0.6, 0.6, 0.1, 0.1), mgp = c(1.8, 0.5, 0), ...)
```

Arguments

file	File name to which to save figure.
dev	Type of file to create. Must be one of "eps" (encapsulated PostScript), "pdf" (portable document format), "wmf" (Windows metafile, available on Windows operating systems only), or "tif" (tagged image format).
width	Width of the figure in inches.
height	Height of the figure in inches.
mai	Settings for margins in inches, as defined for parameter mai in the help for the par function.
mgp	Settings determining how close axis titles, labels, and lines are to the plot, as defined for parameter mgp in the help for the par function.
...	Any other parameters the user adds will be passed to the par function, and may include options to control axes, symbol sizes, etc.

Details

The function has default specifications for the margins, intended to minimize the margin space for many commonly generated plots, but these can be changed to use other `mai` options or other arguments to `par`. The user should not use this function if more than one plot is desired on the device (e.g. if using mfrow, mfcol, layout, etc.)

This function is used to initialize figures inside the function `aspect.ratio.plot`.

Value

Sets up a figure and plotting region with the specified dimensions and margins on the device `dev`.

Note

This function is called by the functin aspect.ratio.plot (also included in this package).

Author(s)

Jenise Swall

See Also

`par`

---

ozone1 *Daily maximum ozone concentrations*

Description

This data set gives the daily maximum ozone concentrations (in ppb) observed at a subset of 23 Clean Air Status and Trends Network (CASTNET) monitoring sites on August 9, 2004.

Usage

```
ozone
```

Format

A file in CSV format containing 23 records.

Source

Hourly ozone data at CASTNET sites can be obtained from EPA's Clean Air Status and Trends Network program at http://www.epa.gov/castnet. Thanks to Steven Howard for his help collating and summarizing these hourly data.

ozone2 *Daily maximum ozone concentrations*

Description

This data set gives the daily maximum ozone concentrations (in ppb) observed at a subset of 23 Clean Air Status and Trends Network (CASTNET) monitoring sites on August 10, 2004.

Usage

```
ozone
```

Format

A file in CSV format containing 23 records.

Source

Hourly ozone data at CASTNET sites can be obtained from EPA's Clean Air Status and Trends Network program at http://www.epa.gov/castnet. Thanks to Steven Howard for his help collating and summarizing these hourly data.

plot3d.points *Make 3-D scatterplot (using colored or differently-sized points)*

Description

Given a list of points' coordinates and the values observed at those points, return a scatterplot with points located as specified by the coordinates and coded by color and/or size to represnt the observed value at the location. This code is basically a wrapper for a call to the function `points.geodata` in the geoR package.

Usage

```
plot3d.points(x, y, z, zlim = range(z, na.rm = TRUE),
   add = FALSE, col = heat.colors(12), pch = 21,
   cex.min = 1, cex.max = 1,
   symbol.border.col = "black", ...)
```

Arguments

x	x-coordinates of locations at which response values z are recorded.
y	y-coordinates of locations at which response values z are recorded.
z	Response values z observed at locations whose coordinates are given by (x, y).
zlim	Vector of minimum and maximum values of z to which to assign the two most extreme colors in the col argument (col[1] and col[length(col)]). Default is to use the range of z. This is very much like the zlim argument to the image function.
add	If FALSE (default), the function will begin a new plot. If TRUE, adds scatterplot to a pre-existing plot.
col	Color range to use for the scatterplot, with the first color assigned to zlim[1] and last color assigned to zlim[2]. Default is "heat.colors(12)", as it is for image.
pch	The point symbol to use. Possible values are 21, 22, 23, 24, and 25. This is because points.geodata requires these points, which have outlines around them. Default is a circle ('pch=21').
cex.min	Minimum amount to shrink/expand the point symbols.
cex.max	Maximum amount to shrink/expand the point symbols. Parameters cex.min and cex.max control the minimum and maximum amounts to shrink/expland the points, based on the value of z. By default, these are both set to one, which makes all the points the same size. For more information, see the help page for points.geodata.
symbol.border.col	This controls the color of the border around the plotting symbol. By default, it is black. If a border is not desired, use 'symbol.border.col="transparent"'.
...	Any other parameters the user adds will be passed to the plot function if 'add=FALSE', and may include options for axis labels, titles, etc.

Details

This function is a wrapper to the points.geodata function in the geoR package.

Value

A scatterplot with points at (x,y). These points are colored according to the correspoinding value of z and the colors specified in col. They are sized according to the corresponding value of z and the minimum and maximum sizes specified by cex.min and cex.max.

Author(s)

Jenise Swall

See Also

points.geodata, points

Examples

```
## Plot ozone at each location using colors from rainbow.colors
## and differently-sized points.  Add a legend using function
## vertical.image.legend (included in this package).
data(ozone1)
col.rng <- rev(rainbow(n=12, start=0, end=4/6))
plot3d.points(x=ozone1$longitude, y=ozone1$latitude, z=ozone1$daily.max,
              xlab="longitude", ylab="latitude", col=col.rng,
              cex.min=0.5, cex.max=1.5)
## To verify, label the points with their concentrations.
text(ozone1$longitude, ozone1$latitude+0.15, ozone1$daily.max, cex=0.7)
## If maps package is available, put on state lines.
if (require("maps"))  map("state", add=TRUE, col="lightgray")
## Put on legend.
vertical.image.legend(col=col.rng, zlim=range(ozone1$daily.max))


## Plot second day of ozone data.  Note that day 2 experienced a smaller
## range of concentrations, so we plot day 2 on same scale as day 1.
data(ozone1)
data(ozone2)
z.rng <- range(ozone1$daily.max)
col.rng <- rev(rainbow(n=12, start=0, end=4/6))
X11()
plot3d.points(x=ozone2$longitude, y=ozone2$latitude, z=ozone2$daily.max,
              xlab="longitude", ylab="latitude", col=col.rng,
              zlim=z.rng, cex.min=0.5, cex.max=1.5)
## To verify, label the points with their concentrations.
text(ozone2$longitude, ozone2$latitude+0.15, ozone2$daily.max, cex=0.7)
## If maps package is available, put on state lines.
if (require("maps"))  map("state", add=TRUE, col="lightgray")
vertical.image.legend(col=col.rng, zlim=z.rng)
```

`ragged.image` *Produces a "ragged" image plot*

Description

This code produces an image plot in the case in which there is not a known response value `z` for every possible combination of `x` and `y`. This ragged image plot is a variant of an image plot which is not complete across the entire rectangle of the gridded area.

Usage

```
ragged.image(x, y, z, zlim = range(z, na.rm = TRUE), add = FALSE,
   col = heat.colors(12), ...)
```

Arguments

x	x-coordinates of grid cell centers at which response values z are available.
y	y-coordinates of grid cell centers at which response values z are available.
z	Response values recorded at the grid cell centers whose coordinates are given by (x, y).
zlim	Vector of minimum and maximum values of z to which to assign the two most extreme colors in the 'col' argument (col[1] and col[length(col)]). Default is to use the range of z. This is very much like the 'zlim' argument to the image function.
add	If FALSE (default), the ragged image will begin a new plot. If TRUE, adds ragged image to a pre-existing plot.
col	Color range to use for the ragged image, with the first color assigned to zlim[1] and last color assigned to zlim[2]. Default is "heat.colors(12)", as it is for image.
...	Any additional parameters to be passed to the image function, if add=FALSE.

Details

This code produces a ragged image plot. This is in contrast to the standard image function, which assumes that there is a known response value z for every combination of the elements of x and y, i.e. that there is a complete rectangular grid, or image. A ragged image plot is a variant of the regular image plot which is not complete across the entire rectangle. The user specifies vectors x, y, and z, such that x and y identify a portion of the grid. This function maps the vector z onto a matrix of the type needed for the image function, but has NA entries for combinations of x and y that are not listed. The NA values are not plotted by image, so a ragged image will appear.

Value

A ragged image, i.e. a portion of an image for which we have specified grid cell centers x and y.

Note

This function is slow if x, y, and z are long vectors.

Author(s)

Jenise Swall

See Also

image, heat.colors

Examples

```
## Build x, y, and z.
x <- c(1, 2, 3, 1, 2, 3)
y <- c(1, 1, 1, 2, 2, 2)
z <- 1:6
```

```
z.mat <- matrix(c(1:6), ncol=2)

par(mfrow=c(1,2))

## Show complete matrix.
image(x=unique(x), y=unique(y), z.mat, xlab="x", ylab="y")

## Plot only part of this as a ragged image.  Set z range so that this
## image will use colors consistent with the previous one.
ragged.image(x=x[1:4], y=y[1:4], z=z[1:4], zlim=range(z),
             xlab="x", ylab="y")
```

`vertical.image.legend`

Put color bar legend in the right plot margin.

Description

Put color bar legend in the right-hand side margin of an existing plot.

Usage

```
vertical.image.legend(zlim, col)
```

Arguments

`zlim`	Gives the range of z values to which the colors specified in `col` are assigned.
`col`	Gives the range of colors to use. To keep multiple plots consistent in terms of the colors assigned to various values, keep `zlim` and `col` the same for each of the plots and the legend.

Details

This function makes two important assumptions. The first is that the user has already finished making the main portion of the plot; i.e., user should **add the legend last**. The user should not try to add secondary information onto the plot after using this legend command. This is because this funciton alters the `par` settings to draw the legend, and trying to reset them sometimes causes errors that are mysterious (at least to the writer of this function). Secondly, this function works best when there is only one plot on the device, in which case the margin space is straightforward (no confusion between `oma`/`omi` and `mar`/`mai`, etc.

Value

Puts vertical color bar legend to the right of the plot.

Note

As noted in the help for `image.plot` in the fields package, putting a legend on a plot is harder than it might seem. The user may have to experiment with this function a bit to get it to work well for a specific application. The user may also want to try the previously mentioned `image.plot` function in the fields package (to just add the legend, use "legend.only=TRUE").

Author(s)

Jenise Swall

See Also

`image.plot`

Examples

```
## Plot ozone at each location using colors from rainbow.colors
## and differently-sized points.  Add a legend using function
## vertical.image.legend (included in this package).
data(ozone1)
col.rng <- rev(rainbow(n=12, start=0, end=4/6))
plot3d.points(x=ozone1$longitude, y=ozone1$latitude, z=ozone1$daily.max,
              xlab="longitude", ylab="latitude", col=col.rng,
              cex.min=0.5, cex.max=1.5)
## To verify, label the points with their concentrations.
text(ozone1$longitude, ozone1$latitude+0.15, ozone1$daily.max, cex=0.7)
## If maps package is available, put on state lines.
if (require("maps"))  map("state", add=TRUE, col="lightgray")
## Put on legend.
vertical.image.legend(col=col.rng, zlim=range(ozone1$daily.max))
```

Package 'M3'

June 15, 2011

Type Package

Title Reading M3 files

Version 0.1

Date 2011-03-03

Author Jenise Swall <Swall.Jenise@epa.gov>

Maintainer Jenise Swall <Swall.Jenise@epa.gov>

Depends ncdf4,rgdal,maps

Description This package contains 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.

License Unlimited

LazyLoad yes

R topics documented:

M3-package

Functions to read in and manipulate air quality model output from files in Models3 format

Description

This package contains 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

This software program has been reviewed in accordance with U.S. Environmental Protection Agency policy and approved for publication. 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 `http://www.baronams.com/products/ioapi`.

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
`http://www.baronams.com/products/ioapi/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

http://www.baronams.com/products/ioapi/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
`http://www.baronams.com/products/ioapi/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

`http://www.baronams.com/products/ioapi/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.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 (http://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 http://www.baronams.com/products/ioapi/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 http://www.baronams.com/products/ioapi/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
(`http://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 (http://www.unidata.ucar.edu/software/netcdf).
It also relies on the **rgdal** interface with GDAL (Geospatial Data Abstraction Library, http://www.gdal.org) to obtain the x,y-coordinates of the grid cell centers on the projection.

Author(s)

Jenise Swall

References

http://www.baronams.com/products/ioapi/VBLE.html,
http://www.baronams.com/products/ioapi/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, region="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, region = "state", units, ...)
```

Arguments

`file`	File name of Models3-formatted file providing the model projection.
`region`	Geogrpaphical database to use. Choices are "state" (default), "world", "USA", "county", and any other databases which are included with the **maps** package and which can be passed to 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** package to get the appropriate map boundary lines (for states, counties, etc.), **ncdf4** to read the projection information from the Models3-formatted file (using a call to function `get.proj.info.M3`), and **rgdal** (which is an interface to GDAL (Geospatial Data Abstraction Library, `http://www.gdal.org`) 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

References

`http://www.gdal.org`

See Also

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

`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 rgdal.

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 **rgdal** 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
(`http://www.unidata.ucar.edu/software/netcdf`).
The string that is returned by this function is appropriate for interface with GDAL (Geospatial Data Abstraction Library, `http://www.gdal.org`) through package **rgdal**. 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
`http://www.baronams.com/products/ioapi/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 `project` from the package **rgdal** to project from longitude/latitude to the projection defined by the Models3-formatted file. Package **rgdal** provides the R interface to GDAL (Geospatial Data Abstraction Library, `http://www.gdal.org`).

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

References

`http://www.gdal.org`

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.
X11()
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 **rgdal**. Package **rgdal** provides the R interface to GDAL (Geospatial Data Abstraction Library, http://www.gdal.org).

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

References

http://www.gdal.org

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.
X11()
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 `project` from the package **rgdal** to project to longitude/latitude given the projection defined by the Models3-formatted file. Package **rgdal** provides the R interface to GDAL (Geospatial Data Abstraction Library, `http://www.gdal.org`).

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

References

`http://www.gdal.org`

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

http://www.baronams.com/products/ioapi/VBLE.html,
http://www.baronams.com/products/ioapi/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)
```