Package 'marmap'

February 20, 2015

Type Package

Title Import, Plot and Analyze Bathymetric and Topographic Data

Version 0.9.2

Date 2015-01-28

Author Eric Pante, Benoit Simon-Bouhet, and Jean-Olivier Irisson

Maintainer Eric Pante <pante.eric@gmail.com>

Depends R (>= 2.10)

Imports DBI, RSQLite, gdistance, geosphere, sp, raster, ncdf, plotrix, shape, reshape2, adehabitatMA, ggplot2

Suggests maps, mapdata, lattice, mapproj, R.rsp

Description Import xyz data from the NOAA, GEBCO and other sources, plot xyz data to prepare publication-ready figures, analyze xyz data to extract transects, get depth / altitude based on geographical coordinates, or calculate z-constrained least-cost paths.

License GPL (>= 3)

VignetteBuilder R.rsp

NeedsCompilation no

Repository CRAN

Date/Publication 2015-01-28 23:21:11

R topics documented:

`aleutians` *Bathymetric data for the Aleutians (Alaska)*

Description

Bathymetric matrix of class `bathy` created from NOAA GEODAS data.

Usage

```
data(aleutians)
```

Details

Data imported from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html) and transformed into an object of class bathy by as.bathy.

Value

A text file.

Author(s)

see http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html

See Also

as.bathy, read.bathy, antimeridian.box

Examples

```
# load celt data
data(aleutians)

# class "bathy"
class(aleutians)
summary(aleutians)

# test plot.bathy
plot(aleutians,image = TRUE,
     bpal = list(c(0,max(aleutians),"grey"),
                 c(min(aleutians),0,"darkblue","lightblue")),
     land = TRUE, lwd = 0.1, axes = FALSE)
antimeridian.box(aleutians, 10)
```

---

antimeridian.box *Adds a box to maps including antimeridian*

Description

Adds a box on maps including the antimeridian (180)

Usage

```
antimeridian.box(object, tick.spacing)
```

Arguments

object	matrix of class bathy
tick.spacing	spacing between tick marks (in degrees, default=20)

Value

The function adds a box and tick marks to an existing plot which contains the antimeridian line (180 degrees).

Author(s)

Eric Pante & Benoit Simon-Bouhet

See Also

plot.bathy

Examples

```
data(aleutians)

# default plot:
plot(aleutians,n=1)

# plot with corrected box and labels:
plot(aleutians,n=1,axes=FALSE)
antimeridian.box(aleutians, 10)
```

---

as.bathy *Convert to bathymetric data in an object of class bathy*

Description

Reads either an object of class RasterLayer, SpatialGridDataFrame or a three-column data.frame containing longitude (x), latitude (y) and depth (z) data and converts it to a matrix of class bathy.

Usage

```
as.bathy(x)
```

Arguments

x	Object of RasterLayer or SpatialGridDataFrame, or a three-column data.frame with longitude (x), latitude (y) and depth (z) (no default)

Details

x can contain data downloaded from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd
in the form of an xyz table. The function as.bathy can also be used to transform objects of class raster (see package raster) and SpatialGridDataFrame (see package sp).

Value

The output of as.bathy is a matrix of class bathy, which dimensions and resolution are identical to the original object. The class bathy has its own methods for summarizing and ploting the data.

Author(s)

Benoit Simon-Bouhet

See Also

summary.bathy, plot.bathy, read.bathy, as.xyz, as.raster, as.SpatialGridDataFrame.

Examples

```
# load NW Atlantic data
data(nw.atlantic)

# use as.bathy
atl <- as.bathy(nw.atlantic)

# class "bathy"
class(atl)

# summarize data of class "bathy"
summary(atl)
```

---

as.raster *Convert bathymetric data to a raster layer*

Description

Transforms an object of class bathy to a raster layer.

Usage

```
as.raster(bathy)
```

Arguments

bathy	an object of class bathy

Details

as.raster transforms bathy objects into objects of class RasterLayer as defined in the raster package. All methods from the raster package are thus available for bathymetric data (e.g. rotations, projections...).

Value

An object of class RasterLayer with the same characteristics as the bathy object (same longitudinal and latitudinal ranges, same resolution).

Author(s)

Benoit Simon-Bouhet

See Also

as.xyz, as.bathy, as.SpatialGridDataFrame

Examples

```
# load Hawaii bathymetric data
data(hawaii)

# use as.raster
r.hawaii <- as.raster(hawaii)

# class "RasterLayer"
class(r.hawaii)

# Summaries
summary(hawaii)
summary(r.hawaii)

# structure of the RasterLayer object
str(r.hawaii)

## Not run:
# Plots
#require(raster)
plot(hawaii,image=TRUE,lwd=.2)
plot(r.hawaii)

## End(Not run)
```

as.SpatialGridDataFrame *Convert bathymetric data to a spatial grid*

Description

Transforms an object of class bathy to a SpatialGridDataFrame object.

Usage

```
as.SpatialGridDataFrame(bathy)
```

Arguments

bathy	an object of class bathy

Details

as.SpatialGridDataFrame transforms bathy objects into objects of class SpatialGridDataFrame as defined in the sp package. All methods from the sp package are thus available for bathymetric data (e.g. rotations, projections...).

Value

An object of class SpatialGridDataFrame with the same characteristics as the bathy object (same longitudinal and latitudinal ranges, same resolution).

Author(s)

Benoit Simon-Bouhet

See Also

as.xyz, as.bathy, as.raster

Examples

```
# load Hawaii bathymetric data
data(hawaii)

# use as.SpatialGridDataFrame
sp.hawaii <- as.SpatialGridDataFrame(hawaii)

# Summaries
summary(hawaii)
summary(sp.hawaii)

# structure of the SpatialGridDataFrame object
str(sp.hawaii)
```

```
# Plots
plot(hawaii,image=TRUE,lwd=.2)
image(sp.hawaii)
```

`as.xyz` *Convert to xyz format*

Description

Converts a matrix of class `bathy` into a three-column data.frame containing longitude, latitude and depth data.

Usage

```
as.xyz(bathy)
```

Arguments

`bathy`	matrix of class `bathy`.

Details

The use of `as.bathy` and `as.xyz` allows to swicth back and forth between the long format (xyz) and the wide format of class `bathy` suitable for plotting bathymetric maps, computing least cost distances, etc. `as.xyz` is especially usefull for exporting xyz files when data are obtained using `subsetSQL`, i.e. bathymetric matrices of class `bathy`.

Value

Three-column data.frame with a format similar to xyz files downloaded from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html). The first column contains longitude data, the second contains latitude data and the third contains depth/elevation data.

Author(s)

Benoit Simon-Bouhet

See Also

`as.bathy`, `summary.bathy`

Examples

```
# load celt data
data(celt)
dim(celt)
class(celt)
summary(celt)
plot(celt,deep= -300,shallow= -25,step=25)

# use as.xyz
celt2 <- as.xyz(celt)
dim(celt2)
class(celt2)
summary(celt2)
```

`autoplot.bathy` *Ploting bathymetric data with ggplot*

Description

Plots contour or image map from bathymetric data matrix of class `bathy` with ggplot2

Usage

```
## S3 method for class 'bathy'
autoplot(x, geom="contour", mapping=NULL, coast=TRUE, ...)
```

Arguments

`x`	bathymetric data matrix of class `bathy`, imported using `read.bathy`
`geom`	geometry to use for the plot, i.e. type of plot; can be 'contour', 'tile' or 'raster'. contour does a contour plot. tile and raster produce an image plot. tile allows true geographical projection through `coord_map`. raster only allows approximate projection but is faster to plot. Names can be abbreviated. Geometries can be combined by specifying several in a vector.
`mapping`	additional mappings between the data obtained from calling `fortify.bathy` on x and the aesthetics for all geoms. When not NULL, this is a call to aes().
`coast`	boolean; wether to highlight the coast (isobath 0 m) as a black line
`...`	passed to the chosen geom(s)

Details

`fortify.bathy` is called with argument `x` to produce a data.frame compatible with ggplot2. Then layers are added to the plot based om the argument `geom`. Finally, the whole plot is projected geographically using `coord_map` (for `geom="contour"`) or an approximation thereof.

Author(s)

Jean-Olivier Irisson

See Also

fortify.bathy, plot.bathy, read.bathy, summary.bathy

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

  # basic plot
## Not run:
    library("ggplot2")
autoplot(atl)

  # plot images
autoplot(atl, geom=c("tile"))
autoplot(atl, geom=c("raster")) # faster but not resolution independant

  # plot both!
autoplot(atl, geom=c("raster", "contour"))

  # geom names can be abbreviated
autoplot(atl, geom=c("r", "c"))

  # do not highlight the coastline
autoplot(atl, coast=FALSE)

  # better colour scale
   autoplot(atl, geom=c("r", "c")) +
    scale_fill_gradient2(low="dodgerblue4", mid="gainsboro", high="darkgreen")

  # set aesthetics
autoplot(atl, geom=c("r", "c"), colour="white", size=0.1)

  # topographical colour scale, see ?scale_fill_etopo
autoplot(atl, geom=c("r", "c"), colour="white", size=0.1) + scale_fill_etopo()

# add sampling locations
data(metallo)
  last_plot() + geom_point(aes(x=lon, y=lat), data=metallo, alpha=0.5)

  # an alternative contour map making use of additional mappings
  # see ?stat_contour in ggplot2 to understand the ..level.. argument
autoplot(atl, geom="contour", mapping=aes(colour=..level..))

## End(Not run)
```

celt *Bathymetric data for the North Est Atlantic*

Description

Bathymetric matrix of class bathy created from NOAA GEODAS data.

Usage

```
data(celt)
```

Details

Data imported from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html) and transformed into an object of class bathy by as.bathy.

Value

A text file.

Author(s)

see http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html

See Also

as.bathy, read.bathy

Examples

```
# load celt data
data(celt)

# class "bathy"
class(celt)
summary(celt)

# test plot.bathy
plot(celt, deep=-300, shallow=-50, step=25)
```

check.bathy *Sort bathymetric data matrix by increasing latitude and longitude*

Description

Reads a bathymetric data matrix and orders its rows and columns by increasing latitude and longitude.

Usage

```
check.bathy(x)
```

Arguments

x	a matrix

Details

check.bathy allows to sort rows and columns by increasing latitude and longitude, which is necessary for ploting with the function image (package graphics). check.bathy is used within the marmap functions read.bathy and as.bathy (it is also used in getNOAA.bathy through as.bathy).

Value

The output of check.bathy is an ordered matrix.

Author(s)

Eric Pante

See Also

read.bathy, as.bathy, getNOAA.bathy

Examples

```
matrix(1:100, ncol=5, dimnames=list(20:1, c(3,2,4,1,5))) -> a
check.bathy(a)
```

col2alpha *Adds alpha transparency to a (vector of) color(s)*

Description

Adds transparency to a color or a vector of colors by specifying one or several alpha values.

Usage

```
col2alpha(color,alpha = 0.5)
```

Arguments

color	a (vector of) color codes or names
alpha	a value (or vector of values) between 0 (full transparency) and 1 (no transparency).

Details

When the size of color and alpha vectors are different, alpha values are recycled.

Value

A (vector) of color code(s).

Author(s)

Benoit Simon-Bouhet

Examples

```
# Generate random data
dat <- rnorm(4000)

# plot with plain color for points
plot(dat,pch=19,col="red")

# Add some transparency to get a better idea of density
plot(dat,pch=19,col=col2alpha("red",.3))

# Same color for all points but with increasing alpha (decreasing transparency)
plot(dat,pch=19,col=col2alpha(rep("red",4000),seq(0,1,len=4000)))

# Two colors, same alpha
plot(dat,pch=19,col=col2alpha(rep(c("red","purple"),each=2000),.2))

# Four colors, gradient of transparency for each color
plot(dat,pch=19,col=col2alpha(rep(c("blue","purple","red","orange"),each=1000),seq(.1,.6,len=1000)))
```

```
# Alpha transparency applied to a gradient of colors
plot(dat,pch=19,col=col2alpha(rainbow(4000),.5))
```

`collate.bathy` *Collates two bathy matrices with data from either sides of the antimeridian*

Description

Collates two bathy matrices, one with longitude 0 to 180 degrees East, and the other with longitude 0 to 180 degrees West

Usage

```
collate.bathy(east,west)
```

Arguments

`east`	matrix of class `bathy` with eastern data (West of antimeridian)
`west`	matrix of class `bathy` with western data (East of antimeridian)

Details

This function is meant to be used with `read.bathy()` or `readGEBCO.bathy()`, when data is downloaded from either sides of the antimeridian line (180 degrees longitude). If, for example, data is downloaded from GEBCO for longitudes of 170E-180 and 180-170W, `collate.bathy()` will create a single matrix of class `bathy` with a coordinate system going from 170 to 190 degrees longitude.

`getNOAA.bathy()` deals with data from both sides of the antimeridian and does not need further processing with `collate.bathy()`.

Value

A single matrix of class `bathy` that can be interpreted by `plot.bathy`. When plotting collated data (with longitudes 0 to 180 and 180 to 360 degrees), plots can be modified to display the conventional coordinate system (with longitudes 0 to 180 and -180 to 0 degrees) using function `antimeridian.box()`.

Author(s)

Eric Pante

See Also

`getNOAA.bathy`, `summary.bathy`, `plot.bathy`, `antimeridian.box`

Examples

```
## faking two datasets using aleutians, for this example
## "a" and "b" simulate two datasets downloaded from GEBCO, for ex.
data(aleutians)
aleutians[1:181,] -> a ; "bathy" -> class(a)
aleutians[182:601,] -> b ; "bathy" -> class(b)
-(360-as.numeric(rownames(b))) -> rownames(b)

## check these objects with summary(): pay attention of the Longitudinal range
summary(aleutians)
summary(a)
summary(b)

## merge datasets:
collate.bathy(a,b) -> collated
summary(collated) # should be identical to summary(aleutians)
```

---

combine.buffers *Create a new, composite buffer from a list of existing buffers.*

Description

Create a new buffer object from a list of existing buffers of compatible dimensions.

Usage

```
combine.buffers(buf)
```

Arguments

buf	a list of 2 or more buffer objects as produced by create.buffer. All buffer objects in the list must have the same number of rows and columns.

Value

An object of class bathy of the same dimension as the original buffers containing only NAs outside of the combined buffer and values of depth/altitude (taken from the list of buffers buf) within the combined buffer.

Author(s)

Benoit Simon-Bouhet

See Also

create.buffer, outline.buffer, plot.bathy

Examples

```
# load and plot a bathymetry
data(florida)
plot(florida, lwd = 0.2)
plot(florida, n = 1, lwd = 0.7, add = TRUE)

# add points around which a buffer will be computed
loc <- data.frame(c(-80,-82), c(26,24))
points(loc, pch = 19, col = "red")

# create 2 distinct buffer objects with different radii
buf1 <- create.buffer(florida, loc[1,], radius=1.9)
buf2 <- create.buffer(florida, loc[2,], radius=1.2)

# combine both buffers
buf <- combine.buffers(list(buf1,buf2))

## Not run:
# Add outline of the resulting buffer in red
# and the outline of the original buffers in blue
plot(outline.buffer(buf), add = TRUE, lwd = 3, col = 2)
plot(outline.buffer(buf1), add = TRUE, lwd = 0.5, col="blue")
plot(outline.buffer(buf2), add = TRUE, lwd = 0.5, col="blue")

## End(Not run)
```

`create.buffer` *Create a buffer of specified radius around one or several points*

Description

Create a circular buffer of user-defined radius around one or several points defined by their longitudes and latitudes.

Usage

```
create.buffer(mat, loc, radius, km = FALSE)
```

Arguments

`mat`	an object of class `bathy`
`loc`	a 2-column `data.frame` of longitudes and latitudes for points around which the buffer is to be created.
`radius`	`numeric`. Radius of the buffer in the same unit as the `bathy` object (i.e. usually decimal degrees) when `km=FALSE` (default) or in kilometers when `radius=TRUE`.
`km`	`logical`. If `TRUE`, the `radius` should be provided in kilometers. When `FALSE` (default) the radius is in the same unit as the `bathy` object (i.e. usually decimal degrees).

Details

This function takes advantage of the buffer function from package adehabitatMA and several functions from packages sp to define the buffer around the points provided by the user.

Value

An object of class bathy of the same size as mat containing only NAs outside of the buffer and values of depth/altitude (taken from mat) within the buffer.

Author(s)

Benoit Simon-Bouhet

See Also

outline.buffer, combine.buffers, plot.bathy

Examples

```
# load and plot a bathymetry
data(florida)
plot(florida, lwd = 0.2)
plot(florida, n = 1, lwd = 0.7, add = TRUE)

# add points around which a buffer will be computed
loc <- data.frame(c(-80,-82), c(26,24))
points(loc, pch = 19, col = "red")

# compute buffer
buf <- create.buffer(florida, loc, radius=1.5)

# highlight isobath with the buffer and add outline
plot(buf, add = TRUE, n = 10, col = 2, lwd=.4)
plot(outline.buffer(buf), add = TRUE, lwd = 0.7, col = 2)
```

---

diag.bathy *Finds matrix diagonal for non-square matrices*

Description

Finds either the values of the coordinates of the non-linear diagonal of non-square matrices.

Usage

```
diag.bathy(mat,coord=FALSE)
```

Arguments

mat	a data matrix
coord	whether of not to output the coordinates of the diagonal (default is FALSE)

Details

diag.bathy gets the values or coordinates from the first element of a matrix to its last elements. If the matrix is non-square, that is, its number of rows and columns differ, diag.bathy computes an approximate diagonal.

Value

A vector of diagonal values is coord is FALSE, or a table of diagonal coordinates ifcoord is FALSE

Author(s)

Eric Pante

See Also

get.transect, diag

Examples

```
# a square matrix: diag.bathy behaves as diag
matrix(1:25, 5, 5) -> a ; a
diag(a)
diag.bathy(a)

# a non-square matrix: diag.bathy does not behaves as diag
matrix(1:15, 3, 5) -> b ; b
diag(b)
diag.bathy(b)

# output the diagonal or its coordinates:
rownames(b) <- seq(32,35, length.out=3)
colnames(b) <- seq(-100,-95, length.out=5)
diag.bathy(b, coord=FALSE)
diag.bathy(b, coord=TRUE)
```

`dist2isobath` *Computes the shortest great circle distance between any point and a given isobath*

Description

Computes the shortest (great circle) distance between a set of points and an isoline of depth or altitude. Points can be selected interactively by clicking on a map.

Usage

```
dist2isobath(mat, x, y=NULL, isobath=0, locator=FALSE, ...)
```

Arguments

`mat`	Bathymetric data matrix of class `bathy`, as imported with `read.bathy`.
`x`	Either a list of two elements (numeric vectors of longitude and latitude), a 2-column matrix or data.frame of longitudes and latitudes, or a numeric vector of longitudes.
`y`	Either `NULL` (default) or a numerical vector of latitudes. Ignored if `x` is not a numeric vector.
`isobath`	A single numerical value indicating the isobath to which the shortest distance is to be computed (default is set to 0, *i.e.* the coastline).
`locator`	Logical. Whether to choose data points interactively with a map or not. If `TRUE`, a bathymetric map must have been plotted and both `x` and `y` are both ignored.
`...`	Further arguments to be passed to `locator` when the interactive mode is used (`locator=TRUE`).

Details

`dist2isobath` allows the user to compute the shortest great circle distance between a set of points (selected interactively on a map or not) and a user-defined isobath. `dist2isobath` takes advantage of functions from packages `sp` (`Lines()` and `SpatialLines()`) and `geosphere` (`dist2Line`) to compute the coordinates of the nearest location along a given isobath for each point provided by the user.

Value

A 5-column data.frame. The first column contains the distance in meters between each point and the nearest point located on the given `isobath`. Columns 2 and 3 indicate the longitude and latitude of starting points (*i.e.* either coordinates provided as `x` and `y` or coordinates of points selected interactively on a map when `locator=TRUE`) and columns 4 and 5 contains coordinates (longitudes and latitudes) arrival points *i.e.* the nearest points on the `isobath`.

Author(s)

Benoit Simon-Bouhet

See Also

linesGC, lc.dist

Examples

```
# Load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

# Create vectors of latitude and longitude
lon <- c(-70, -65, -63, -55, -48)
lat <- c(33, 35, 40, 37, 33)

# Compute distances between each point and the -200m isobath
d <- dist2isobath(atl, lon, lat, isobath = -200)
d

# Visualize the great circle distances
blues <- c("lightsteelblue4","lightsteelblue3","lightsteelblue2","lightsteelblue1")
plot(atl, image=TRUE, lwd=0.1, land=TRUE, bpal = list(c(0,max(atl),"grey"), c(min(atl),0,blues)))
plot(atl, deep=-200, shallow=-200, step=0, lwd=0.6, add=TRUE)
points(lon,lat, pch=21, col="orange4", bg="orange2", cex=.8)
linesGC(d[2:3],d[4:5])
```

---

etopo *Etopo colours*

Description

Various ways to access the colours or on the etopo colour scale

Usage

```
etopo.colors(n)

scale_fill_etopo(...)
scale_colour_etopo(...)
```

Arguments

n	number of colours to get from the scale. Those are evenly spaced within the scale.
...	passed to scale_fill_gradientn or scale_colour_gradientn

Details

etopo.colors is equivalent to all usual colours in R (heat.colors, cm.colors).

scale_fill/contour_etopo are meant to be used with ggplot2. They allow consistent plots in various subregions by setting the limits of the scale explicitly.

Author(s)

Jean-Olivier Irisson

See Also

autoplot.bathy, palette.bathy

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

  library("ggplot2")
  head(fortify(atl))

  # one can now use bathy objects with ggplot directly
  ggplot(atl) + geom_contour(aes(x=x, y=y, z=z))

  # which allows complete plot configuration
  atl.df <- fortify(atl)
  ggplot(atl.df, aes(x=x, y=y)) +
    geom_raster(aes(fill=z), data=atl.df[atl.df$z <= 0,]) +
    geom_contour(aes(z=z),
      breaks=c(-100, -200, -500, -1000, -2000, -4000),
      colour="white", size=0.1
    )
```

---

florida *Bathymetric data around Florida, USA*

Description

Bathymetric object of class bathy created from NOAA GEODAS data.

Usage

```
data(florida)
```

Details

Data imported from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html) and transformed into an object of class bathy by read.bathy.

Value

A bathymetric object of class bathy with 539 rows and 659 columns.

Author(s)

see http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html

See Also

plot.bathy, summary.bathy

Examples

```
# load florida data
data(florida)

# class "bathy"
class(florida)
summary(florida)

# test plot.bathy
plot(florida,asp=1)
plot(florida,asp=1,image=TRUE,drawlabels=TRUE,land=TRUE,n=40)
```

---

fortify.bathy *Extract bathymetry data in a data.frame*

Description

Extract bathymetry data in a data.frame

Usage

```
## S3 method for class 'bathy'
fortify(x, ...)
```

Arguments

x	bathymetric data matrix of class bathy, imported using read.bathy
...	ignored

Details

fortify.bathy is really just calling as.xyz and ensuring consistent names for the columns. It then allows to use ggplot2 functions directly.

Author(s)

Jean-Olivier Irisson

See Also

autoplot.bathy, as.xyz

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

  library("ggplot2")
  head(fortify(atl))

  # one can now use bathy objects with ggplot directly
  ggplot(atl) + geom_contour(aes(x=x, y=y, z=z))

  # which allows complete plot configuration
  atl.df <- fortify(atl)
  ggplot(atl.df, aes(x=x, y=y)) +
    geom_raster(aes(fill=z), data=atl.df[atl.df$z <= 0,]) +
    geom_contour(aes(z=z),
      breaks=c(-100, -200, -500, -1000, -2000, -4000),
      colour="white", size=0.1
    )
```

---

get.area *Get projected surface area*

Description

Get projected surface area for specific depth layers

Usage

```
get.area(mat, level.inf, level.sup=0, xlim=NULL, ylim=NULL)
```

Arguments

mat	bathymetric data matrix of class bathy, imported using read.bathy (no default)
level.inf	lower depth limit for calculation of projected surface area (no default)
level.sup	upper depth limit for calculation of projected surface area (default is zero)
xlim	longitudinal range of the area of interest (default is NULL)
ylim	latitudinal range of the area of interest (default is NULL)

Details

get.area calculates the projected surface area of specific depth layers (e.g. upper bathyal, lower bathyal), the projected plane being the ocean surface. The resolution of get.area depends on the resolution of the input bathymetric data. xlim and ylim can be used to restrict the area of interest. Area calculation is based on areaPolygon of package geosphere (using an average Earth radius of 6,371 km).

Value

A list of four objects: the projeced surface area in squared kilometers, a matrix with the cells used for calculating the projected surface area, the longitude and latitude of the matrix used for the calculations.

Author(s)

Benoit Simon-Bouhet and Eric Pante

See Also

plotArea, plot.bathy, contour, areaPolygon

Examples

```
## get area for the entire hawaii dataset:
data(hawaii)
plot(hawaii, lwd=0.2)

mesopelagic <- get.area(hawaii, level.inf=-1000, level.sup=-200)
bathyal <- get.area(hawaii, level.inf=-4000, level.sup=-1000)
abyssal <- get.area(hawaii, level.inf=min(hawaii), level.sup=-4000)

col.meso <- rgb(0.3, 0, 0.7, 0.3)
col.bath <- rgb(0.7, 0, 0, 0.3)
col.abys <- rgb(0.7, 0.7, 0.3, 0.3)

plotArea(mesopelagic, col = col.meso)
plotArea(bathyal, col = col.bath)
plotArea(abyssal, col = col.abys)

me <- round(mesopelagic$Square.Km, 0)
ba <- round(bathyal$Square.Km, 0)
ab <- round(abyssal$Square.Km, 0)

legend(x="bottomleft",
legend=c(paste("mesopelagic:",me,"km2"),
         paste("bathyal:",ba,"km2"),
         paste("abyssal:",ab,"km2")),
col="black", pch=21,
pt.bg=c(col.meso,col.bath,col.abys))

# Use of xlim and ylim
data(hawaii)
plot(hawaii, lwd=0.2)

mesopelagic <- get.area(hawaii, xlim=c(-161.4,-159), ylim=c(21,23),
                        level.inf=-1000, level.sup=-200)
bathyal <- get.area(hawaii, xlim=c(-161.4,-159), ylim=c(21,23),
                        level.inf=-4000, level.sup=-1000)
abyssal <- get.area(hawaii, xlim=c(-161.4,-159), ylim=c(21,23),
```

```
                        level.inf=min(hawaii), level.sup=-4000)

col.meso <- rgb(0.3, 0, 0.7, 0.3)
col.bath <- rgb(0.7, 0, 0, 0.3)
col.abys <- rgb(0.7, 0.7, 0.3, 0.3)

plotArea(mesopelagic, col = col.meso)
plotArea(bathyal, col = col.bath)
plotArea(abyssal, col = col.abys)

me <- round(mesopelagic$Square.Km, 0)
ba <- round(bathyal$Square.Km, 0)
ab <- round(abyssal$Square.Km, 0)

legend(x="bottomleft",
legend=c(paste("mesopelagic:",me,"km2"),
         paste("bathyal:",ba,"km2"),
         paste("abyssal:",ab,"km2")),
col="black", pch=21,
pt.bg=c(col.meso,col.bath,col.abys))
```

`get.box` *Get bathymetric information of a belt transect*

Description

`get.box` gets depth information of a belt transect of width `width` around a transect defined by two points on a bathymetric map.

Usage

```
get.box(bathy,x1,x2,y1,y2,width,locator=FALSE,ratio=FALSE, ...)
```

Arguments

`bathy`	Bathymetric data matrix of class `bathy`.
`x1`	Numeric. Start longitude of the transect. Requested when `locator=FALSE`.
`x2`	Numeric. Stop longitude of the transect. Requested when `locator=FALSE`.
`y1`	Numeric. Start latitude of the transect. Requested when `locator=FALSE`.
`y2`	Numeric. Stop latitude of the transect. Requested when `locator=FALSE`.
`width`	Numeric. Width of the belt transect in degrees.
`locator`	Logical. Whether to choose transect bounds interactively with a map or not. When `FALSE` (default), a bathymetric map (`plot.bathy(bathy,image=TRUE)`) is automatically plotted and the position of the belt transect is added.
`ratio`	Logical. Should aspect ratio for the `wireframe` plotting function (package `lattice`) be computed (default is `FALSE`).
`...`	Other arguments to be passed to `locator` and `lines` to specify the characteristics of the points and lines to draw on the bathymetric map for both the transect and the bounding box of belt transect.

Details

get.box allows the user to get depth data for a rectangle area of the map around an approximate linear transect (belt transect). Both the position and size of the belt transect are user defined. The position of the transect can be specified either by inputing start and stop coordinates, or by clicking on a map created with plot.bathy. In its interactive mode, this function uses the locator function (graphics package) to retrieve and plot the coordinates of the selected transect. The argument width allows the user to specify the width of the belt transect in degrees.

Value

A matrix containing depth values for the belt transect. rownames indicate the kilometric distance from the start of the transect and colnames indicate the distance form the central transect in degrees. If ratio=TRUE, a list of two elements: depth, a matrix containing depth values for the belt transect similar to the description above and ratios a vector of length two specifying the ratio between (i) the width and length of the belt transect and (ii) the depth range and the length of the belt transect. These ratios can be used by the function wireframe to produce realistic 3D bathymetric plots of the selected belt transect.

Author(s)

Benoit Simon-Bouhet and Eric Pante

See Also

plot.bathy, get.transect, get.depth

Examples

```
# load and plot bathymetry
data(hawaii)
plot(hawaii,im=TRUE)

# get the depth matrix for a belt transect
depth <- get.box(hawaii,x1=-157,y1=20,x2=-155.5,y2=21,width=0.5,col=2)

# plotting a 3D bathymetric map of the belt transect
require(lattice)
wireframe(depth,shade=TRUE)

# get the depth matrix for a belt transect with realistic aspect ratios
depth <- get.box(hawaii,x1=-157,y1=20,x2=-155.5,y2=21,width=0.5,col=2,ratio=TRUE)

# plotting a 3D bathymetric map of the belt transect with realistic aspect ratios
require(lattice)
wireframe(depth[[1]],shade=TRUE,aspect=depth[[2]])
```

`get.depth` *Get depth data by clicking on a map*

Description

Outputs depth information based on points selected by clicking on a map

Usage

```
get.depth(mat, x, y=NULL, locator=TRUE, distance=FALSE, ...)
```

Arguments

`mat`	Bathymetric data matrix of class `bathy`, as imported with `read.bathy`.
`x`	Either a list of two elements (numeric vectors of longitude and latitude), a 2-column matrix or data.frame of longitudes and latitudes, or a numeric vector of longitudes.
`y`	Either `NULL` (default) or a numerical vector of latitudes. Ignored if `x` is not a numeric vector.
`locator`	Logical. Whether to choose data points interactively with a map or not. If `TRUE` (default), a bathymetric map must have been plotted and both `x` and `y` are both ignored.
`distance`	whether to compute the haversine distance (in km) from the first data point on (default is `FALSE`). Only available when at least two points are provided.
`...`	Further arguments to be passed to `locator` when the interactive mode is used (`locator=TRUE`).

Details

`get.depth` allows the user to get depth data by clicking on a map created with `plot.bathy` or by providing coordinates of points of interest. This function uses the `locator` function (`graphics` package); after creating a map with `plot.bathy`, the user can click on the map once or several times (if `locator=TRUE`), press the Escape button, and get the depth of those locations in a three-coumn data.frame (longitude, latitude and depth). Alternatively, when `locator=FALSE`, the user can submit a list of longitudes and latitudes, a two-column matrix or data.frame of longitudes and latitudes (as input for `x`), or one vector of longitudes (`x`) and one vector of latitudes (`y`). The non-interactive mode is well suited to get depth information for each point provided by GPS tracking devices. While `get.transect` gets every single depth value available in the bathymetric matrix between two points along a user-defined transect, `get.depth` only provides depth data for the specific points provided as input by the user.

Value

A data.frame with at least, longitude, latitude and depth with one line for each point of interest. If `distance=TRUE`, a fourth column containing the kilometric distance from the first point is added.

Author(s)

Benoit Simon-Bouhet and Eric Pante

See Also

path.profile, get.transect, read.bathy, summary.bathy, subsetBathy, nw.atlantic

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

# create vectors of latitude and longitude
lon <- c(-70, -65, -63, -55)
lat <- c(33, 35, 40, 37)

# a simple example
plot(atl, lwd=.5)
points(lon,lat,pch=19,col=2)

# Use get.depth to get the depth for each point
get.depth(atl, x=lon, y=lat, locator=FALSE)

# alternativeley once the map is plotted, use the iteractive mode:
## Not run:
get.depth(atl, locator=TRUE, pch=19, col=3)

## End(Not run)
# click several times and press Escape
```

---

get.sample *Get sample data by clicking on a map*

Description

Outputs sample information based on points selected by clicking on a map

Usage

```
get.sample(mat, sample, col.lon, col.lat, ...)
```

Arguments

mat	bathymetric data matrix of class bathy, imported using read.bathy (no default)
sample	data.frame containing sampling information (at least longitude and latitude) (no default)

col.lon	column number of data frame sample containing longitude information (no default)
col.lat	column number of data frame sample containing latitude information (no default)
...	further arguments to be passed to rect for drawing a box around the selected area

Details

get.sample allows the user to get sample data by clicking on a map created with plot.bathy. This function uses the locator function (graphics package). After creating a map with plot.bathy, the user can click twice on the map to delimit an area (for example, lower left and upper right corners of a rectangular area of interest), and get a dataframe corresponding to the sample points present within the selected area.

Value

a dataframe of the elements of sample present within the area selected

Warning

clicking once or more than twice on the map will return a warning message: "Please choose two points from the map"

Author(s)

Eric Pante

See Also

read.bathy, summary.bathy, nw.atlantic, metallo

Examples

```
## Not run:
# load metallo sampling data and add a third field containing a specimen ID
data(metallo)
metallo$id <- factor(paste("Metallo",1:38))

# load NW Atlantic data, convert to class bathy, and plot
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)
plot(atl, deep=-8000, shallow=0, step=1000, col="grey")

# once the map is plotted, use get.sample to get sampling info!
get.sample(atl, metallo, 1, 2)
# click twice

## End(Not run)
```

`get.transect` *Compute approximate cross section along a depth transect*

Description

Compute the depth along a linear transect which bounds are specified by the user.

Usage

```
get.transect(mat, x1, y1, x2, y2, locator=FALSE, distance=FALSE, ...)
```

Arguments

`mat`	bathymetric data matrix of class `bathy`, imported using `read.bathy` (no default)
`x1`	start longitude of the transect (no default)
`x2`	stop longitude of the transect (no default)
`y1`	start latitude of the transect (no default)
`y2`	stop latitude of the transect (no default)
`locator`	whether to use locator to choose transect bounds interactively with a map (default is `FALSE`)
`distance`	whether to compute the haversine distance (in km) from the start of the transect, along the transect (default is `FALSE`)
`...`	other arguments to be passed to `locator()` to specify the characteristics of the points and lines to draw on the bathymetric map when `locator=TRUE`.

Details

`get.transect` allows the user to compute an approximate linear depth cross section either by inputing start and stop coordinates, or by clicking on a map created with `plot.bathy`. In its interactive mode, this function uses the `locator` function (`graphics` package); after creating a map with `plot.bathy`, the user can click twice to delimit the bound of the transect of interest (for example, lower left and upper right corners of a rectangular area of interest), press Escape, and get a table with the transect information.

Value

A table with, at least, longitude, latitude and depth along the transect, and if specified (distance=`TRUE`), the distance in kilometers from the start of the transect. The number of elements in the resulting table depends on the resolution of the `bathy` object.

Warning

Clicking once or more than twice on the map will return a warning message: "Please choose only two points from the map". Manually entering coordinates that are outside the geographical range of the input `bathy` matrix will return a warning message.

Note

The distance option of `get.transect` is calculated based on the haversine formula for getting the great circle distance (takes into account the curvature of the Earth). `get.transect` uses an internal function called `diag.bathy` that extracts the approximate diagonal of a matrix, when that matrix has uneven dimentions (different numbers of columns and rows).

Author(s)

Eric Pante and Benoit Simon-Bouhet

See Also

`read.bathy`, `nw.atlantic`, `nw.atlantic.coast`, `get.depth`, `get.sample`

Examples

```
# load datasets
data(nw.atlantic); as.bathy(nw.atlantic) -> atl
data(nw.atlantic.coast)

# Example 1. get.transect(), without use of locator()
get.transect(atl, -65, 43,-59,40) -> test ; plot(test[,3]~test[,2],type="l")
get.transect(atl, -65, 43,-59,40, distance=TRUE) -> test ; plot(test[,4]~test[,3],type="l")

# Example 2. get.transect(), without use of locator(); pretty plot
par(mfrow=c(2,1),mai=c(1.2, 1, 0.1, 0.1))
plot(atl, deep=-6000, shallow=-10, step=1000, lwd=0.5, col="grey50",drawlabels=TRUE)
lines(nw.atlantic.coast)

get.transect(atl, -75, 44,-46,32, loc=FALSE, dis=TRUE) -> test
points(test$lon,test$lat,type="l",col="blue",lwd=2,lty=2)
plotProfile(test)

# Example 3. get.transect(), with use of locator(); pretty plot
## Not run:
par(mfrow=c(2,1),mai=c(1.2, 1, 0.1, 0.1))
plot(atl, deep=-6000, shallow=-10, step=1000, lwd=0.5, col="grey50",drawlabels=TRUE)
lines(nw.atlantic.coast)

get.transect(atl, loc=TRUE, dis=TRUE, col=2, lty=2) -> test
plotProfile(test)

## End(Not run)
```

`getNOAA.bathy` *Import bathymetric data from the NOAA server*

Description

Imports bathymetric data from the NOAA server, given coordinate bounds and resolution.

Usage

```
getNOAA.bathy(lon1,lon2,lat1,lat2, resolution = 4, keep=FALSE, antimeridian=FALSE)
```

Arguments

`lon1`	first longitude of the area for which bathymetric data will be downloaded
`lon2`	second longitude of the area for which bathymetric data will be downloaded
`lat1`	first latitude of the area for which bathymetric data will be downloaded
`lat2`	second latitude of the area for which bathymetric data will be downloaded
`resolution`	resolution of the grid, in minutes (default is 4)
`keep`	whether to write the data downloaded from NOAA into a file (default is FALSE)
`antimeridian`	whether the area should include the antimeridian (longitude 180 or -180). See details.

Details

`getNOAA.bathy` queries the ETOPO1 database hosted on the NOAA website, given the coordinates of the area of interest and desired resolution. Users have the option of directly writing the downloaded data into a file (`keep=TRUE` argument ; see below). If an identical query is performed (i.e. using identical lat-long and resolution), `getNOAA.bathy` will load data from the file previously written to the disk instead of querying the NOAA database. This behavior should be used preferentially (1) to reduce the number of uncessary queries to the NOAA website, and (2) to reduce data load time. If the user wants to make multiple, identical queries to the NOAA website without loading the data written to disk, the data file name must be modified by the user. Alternatively, the data file can be moved outside of the present working directory.

`getNOAA.bathy` allows users to download bathymetric data in the antimeridian region when `antimeridian=TRUE`. The antimeridian is the 180th meridian and is located about in the middle of the Pacific Ocean, east of New Zealand and Fidji, west of Hawaii and Tonga. For a given pair of longitude values, e.g. -150 (150 degrees West) and 150 (degrees East), you have the possibility to get data for 2 distinct regions: the area centered on the antimeridian (60 degrees wide, when `antimeridian=TRUE`) or the area centered on the prime meridian (300 degrees wide, when `antimeridian=FALSE`). It is recommended to use `keep=TRUE` in combination with `antimeridian=TRUE` since gathering data for an area around the antimeridian requires two distinct queries to NOAA servers.

Value

The output of getNOAA.bathy is a matrix of class bathy, which dimensions depends on the resolution of the grid uploaded from the NOAA server. The class bathy has its own methods for summarizing and ploting the data. If keep=TRUE, a csv file containing the downloaded data is produced. This file is named using the following format: 'marmap_coord_COORDINATES_res_RESOLUTION.csv' (COORDINATES separated by semicolons, and the RESOLUTION in degrees).

Author(s)

Eric Pante and Benoit Simon-Bouhet

References

Amante, C. and B. W. Eakins, ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. NOAA Technical Memorandum NESDIS NGDC-24, 19 pp, March 2009. http://www.ngdc.noaa.gov/mgg/global/relief/ETOPO1/docs/ETOPO1.pdf

See Also

read.bathy, readGEBCO.bathy, plot.bathy

Examples

```
## Not run:
# you must have an internet connection. This line queries the NOAA ETOPO1 database
# for data from North Atlantic, for a resolution of 10 minutes.

getNOAA.bathy(lon1=-20,lon2=-90,lat1=50,lat2=20, resolution=10) -> a
plot(a, image=TRUE, deep=-6000, shallow=0, step=1000)

# download speed for a matrix of 10 degrees x 10 degrees x 30 minutes
system.time(getNOAA.bathy(lon1=0,lon2=10,lat1=0,lat2=10, resolution=30))

## End(Not run)
```

---

hawaii *Bathymetric data for Hawaii, USA*

Description

Bathymetric object of class bathy created from NOAA GEODAS data and arbitrary locations around the main Hawaiian islands.

Usage

```
data(hawaii)
data(hawaii.sites)
```

Details

hawaii contains data imported from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html) and transformed into an object of class bathy by read.bathy. hawaii.sites is a 2-columns data.frame containing longitude and latitude of 6 locations spread at sea around Hawaii.

Value

hawaii: a bathymetric object of class bathy with 539 rows and 659 columns. hawaii.sites: data.frame (6 rows, 2 columns)

Author(s)

see http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html

See Also

plot.bathy, summary.bathy

Examples

```
# load hawaii data
data(hawaii)
data(hawaii.sites)

# class "bathy"
class(hawaii)
summary(hawaii)

## Not run:
## use of plot.bathy to produce a bathymetric map
# creation of a color palette
pal <- colorRampPalette(c("black","darkblue","blue","lightblue"))

# Plotting the bathymetry
plot(hawaii,image=TRUE,draw=TRUE,bpal=pal(100),asp=1,col="grey40",lwd=.7)

# Adding coastline
require(mapdata)
map("worldHires",res=0,fill=TRUE,col=rgb(.8,.95,.8,.7),add=TRUE)

# Adding hawaii.sites location on the map
points(hawaii.sites,pch=21,col="yellow",bg=col2alpha("yellow",.9),cex=1.2)

## End(Not run)
```

`is.bathy` *Test whether an object is of class bathy*

Description

Test whether an object is of class bathy

Usage

```
is.bathy(xyz)
```

Arguments

`xyz`	three-column data.frame with longitude (x), latitude (y) and depth (z) (no default)

Value

The function returns `TRUE` or `FALSE`

Author(s)

Eric Pante

See Also

`as.bathy`, `summary.bathy`, `read.bathy`

Examples

```
# load NW Atlantic data
data(nw.atlantic)

# test class "bathy"
is.bathy(nw.atlantic)

# use as.bathy
atl <- as.bathy(nw.atlantic)

# class "bathy"
class(atl)
is.bathy(atl)

# summarize data of class "bathy"
summary(atl)
```

`lc.dist` *Computes least cost distances between two or more locations*

Description

Computes least cost distances between two or more locations

Usage

```
lc.dist(trans,loc,res=c("dist","path"))
```

Arguments

`trans`	transition object as computed by `trans.mat`
`loc`	A two-columns matrix or data.frame containing latitude and longitude for 2 or more locations.
`res`	either `"dist"` or `"path"`. See details.

Details

`lc.dist` computes least cost distances between 2 or more locations. This function relies on the package `gdistance` (van Etten, 2011. http://CRAN.R-project.org/package=gdistance) and on the `trans.mat` function to define a range of depths where the paths are possible.

Value

Results can be presented either as a kilometric distance matrix between all possible pairs of locations (argument `res="dist"`) or as a list of paths (i.e. 2-columns matrices of routes) between pairs of locations (`res="path"`).

Author(s)

Benoit Simon-Bouhet

References

Jacob van Etten (2011). gdistance: distances and routes on geographical grids. R package version 1.1-2. http://CRAN.R-project.org/package=gdistance

See Also

`trans.mat`

Examples

```
# Load and plot bathymetry
data(hawaii)
pal <- colorRampPalette(c("black","darkblue","blue","lightblue"))
plot(hawaii,image=TRUE,bpal=pal(100),asp=1,col="grey40",lwd=.7,
     main="Bathymetric map of Hawaii")

# Load and plot several locations
data(hawaii.sites)
sites <- hawaii.sites[-c(1,4),]
rownames(sites) <- 1:4
points(sites,pch=21,col="yellow",bg=col2alpha("yellow",.9),cex=1.2)
text(sites[,1],sites[,2],lab=rownames(sites),pos=c(3,4,1,2),col="yellow")

## Not run:
# Compute transition object with no depth constraint
trans1 <- trans.mat(hawaii)

# Compute transition object with minimum depth constraint:
# path impossible in waters shallower than -200 meters depth
trans2 <- trans.mat(hawaii,min.depth=-200)

# Computes least cost distances for both transition matrix and plots the results on the map
out1 <- lc.dist(trans1,sites,res="path")
out2 <- lc.dist(trans2,sites,res="path")
lapply(out1,lines,col="yellow",lwd=4,lty=1) # No depth constraint (yellow paths)
lapply(out2,lines,col="red",lwd=1,lty=1) # Min depth set to -200 meters (red paths)

# Computes and display distance matrices for both situations
dist1 <- lc.dist(trans1,sites,res="dist")
dist2 <- lc.dist(trans2,sites,res="dist")
dist1
dist2

# plots the depth profile between location 1 and 3 in the two situations
dev.new()
par(mfrow=c(2,1))
path.profile(out1[[2]],hawaii,pl=TRUE,
                 main="Path between locations 1 & 3\nProfile with no depth constraint")
path.profile(out2[[2]],hawaii,pl=TRUE,
               main="Path between locations 1 & 3\nProfile with min depth set to -200m")

## End(Not run)
```

`linesGC` *Add Great Circle lines on a map*

Description

`linesGC` draws Great Circle lines between a set of start and end points on an existing map.

Usage

```
linesGC(start.points, end.points, n = 10, antimeridian = FALSE, ...)
```

Arguments

start.points	Two-column data.frame or matrix of longitudes and latitudes for start points.
end.points	Two-column data.frame or matrix of longitudes and latitudes for end points. The dimensions of start.points and end.points must be compatible (*i.e.* they must have the same number of rows).
n	Numeric. The number of intermediate points to add along the great circle line between the start end end points.
antimeridian	Logical indicating if the map on which the great circle lines will be plotted covers the antimeridian region. The antimeridian (or antemeridian) is the 180th meridian and is located in the middle of the Pacific Ocean, east of New Zealand and Fidji, west of Hawaii and Tonga.
...	Further arguments to be passed to lines to control the aspect of the lines to draw.

Details

linesGCD takes advantage of the gcIntermediate function from package geosphere to plot lines following a great circle. When working with marmap maps encompassing the antimeridian, longitudes are numbered from 0 to 360 (as opposed to the classical numbering from -180 to +180). It is thus critical to set antimeridian=TRUE to avoid plotting incoherent great circle lines.

Author(s)

Benoit Simon-Bouhet

See Also

dist2isobath, lc.dist

Examples

```
# Load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

# Create vectors of latitude and longitude
lon <- c(-70, -65, -63, -55, -48)
lat <- c(33, 35, 40, 37, 33)

# Compute distances between each point and the -200m isobath
d <- dist2isobath(atl, lon, lat, isobath = -200)
d

# Create a nice palette of bleus for the bathymetry
blues <- c("lightsteelblue4","lightsteelblue3","lightsteelblue2","lightsteelblue1")
```

```
# Visualize the great circle distances
plot(atl, image=TRUE, lwd=0.1, land=TRUE,
 bpal = list(c(0,max(atl),"grey"), c(min(atl),0,blues)))
points(lon,lat, pch=21, col="orange4", bg="orange2", cex=.8)
linesGC(d[2:3],d[4:5])

# Load aleutians data and plot the map
data(aleutians)
plot(aleutians, image=TRUE, lwd=0.1, land=TRUE,
 bpal = list(c(0,max(aleutians),"grey"), c(min(aleutians),0,blues)))

# define start and end points
start <- matrix(c(170,55, 190, 60), ncol=2, byrow=TRUE, dimnames=list(1:2, c("lon","lat")))
end <- matrix(c(200, 56, 201, 57), ncol=2, byrow=TRUE, dimnames=list(1:2, c("lon","lat")))
start
end

# Add points and great circle distances on the map
points(start, pch=21, col="orange4", bg="orange2", cex=.8)
points(end, pch=21, col="orange4", bg="orange2", cex=.8)
linesGC(start, end, antimeridian=TRUE)
```

---

marmap *Import, plot and analyze bathymetric and topographic data*

Description

marmap is a package designed for downloading, plotting and manipulating bathymetric and topographic data in R. It can query the ETOPO1 bathymetry and topography database hosted by the NOAA, use simple latitude-longitude-depth data in ascii format, and take advantage of the advanced plotting tools available in R to build publication-quality bathymetric maps. Functions to query data (bathymetry, sampling information, etc...) are available interactively by clicking on marmap maps. Bathymetric and topographic data can also be used to calculate projected surface areas within specified depth/altitude intervals, and constrain the calculation of realistic shortest path distances.

Details

Package:	marmap
Type:	Package
Version:	0.9.2
Date:	2015-01-28

Import, plot and analyze bathymetric and topographic data

Author(s)

Eric Pante, Benoit Simon-Bouhet and Jean-Olivier Irisson

Maintainer: Eric Pante <pante.eric@gmail.com>

References

Pante E, Simon-Bouhet B (2013) marmap: A Package for Importing, Plotting and Analyzing Bathymetric and Topographic Data in R. PLoS ONE 8(9): e73051. doi:10.1371/journal.pone.0073051

`metallo` *Coral sampling information from the North West Atlantic*

Description

Coral sampling data from Thoma et al 2009 (MEPS)

Usage

```
data(nw.atlantic)
```

Details

Sampling locations (longitude, latitude, depth in meters) for the deep-sea octocoral species Metallogorgia melanotrichos (see Thoma et al 2009 for details, including cruise information)

Value

A 3-column data frame

References

Thoma, J. N., E. Pante, M. R. Brugler, and S. C. France. 2009. Deep-sea octocorals and antipatharians show no evidence of seamount-scale endemism in the NW Atlantic. Marine Ecology Progress Series 397:25-35. http://www.int-res.com/articles/theme/m397p025.pdf

See Also

`nw.atlantic`

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic,metallo)
atl <- as.bathy(nw.atlantic)

## the function plot below plots:
## - the coastline in blue,
## - isobaths between 8000-4000 in light grey,
```

```
## - isobaths between 4000-500 in dark grey (to emphasize seamounts)

# 1st example: function points uses first two columns ; 3rd column contains depth info
plot(atl, deep=c(-8000,-4000,0), shallow=c(-4000,-500,0), step=c(500,500,0),
   lwd=c(0.5,0.5,1.5),lty=c(1,1,1),
   col=c("grey80", "grey20", "blue"),
   drawlabels=c(FALSE,FALSE,FALSE) )
points(metallo, cex=1.5, pch=19,col=rgb(0,0,1,0.5))

# 2nd example: plot points according to coordinates
plot(atl, deep=c(-8000,-4000,0), shallow=c(-4000,-500,0), step=c(500,500,0),
   lwd=c(0.5,0.5,1.5),lty=c(1,1,1),
   col=c("grey80", "grey20", "blue"),
   drawlabels=c(FALSE,FALSE,FALSE) )
subset(metallo, metallo$lon>-55) -> s # isolate points from the Corner Rise seamounts:
points(s, cex=1.5, pch=19,col=rgb(0,0,1,0.5)) # only plot those points

# 3rd example: point colors corresponding to a depth gradient:
par(mai=c(1,1,1,1.5))
plot(atl, deep=c(-6500,0), shallow=c(-50,0), step=c(500,0),
     lwd=c(0.3,1), lty=c(1,1),
     col=c("black","black"),
     drawlabels=c(FALSE,FALSE,FALSE))

max(metallo$depth, na.rm=TRUE) -> mx
colorRamp(c("white","lightyellow","lightgreen","blue","lightblue1","purple")) -> ramp
rgb( ramp(seq(0, 1, length = mx)), max = 255) -> blues

points(metallo, col="black", bg=blues[metallo$depth], pch=21,cex=1.5)
require(shape); colorlegend(zlim=c(-mx,0), col=rev(blues), main="depth (m)",posx=c(0.85,0.88))
```

`nw.atlantic` *Bathymetric data for the North West Atlantic*

Description

Data imported from the NOAA GEODAS server

Usage

```
data(nw.atlantic)
```

Details

Data imported from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html). To prepare data from NOAA, fill the custom grid form, and choose "XYZ (lon,lat,depth)" as the "Output Grid Format", "No Header" as the "Output Grid Header", and either of the space, tab or comma as the column delimiter (either can be used, but "comma" is the default import format of `read.bathy`). Choose "omit empty grid cells" to reduce memory usage.

Value

A three-columns data.frame containing longitude, latitude and depth/elevation data.

Author(s)

see http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html

See Also

plot.bathy, summary.bathy

Examples

```
# load NW Atlantic data
data(nw.atlantic)

# use as.bathy
atl <- as.bathy(nw.atlantic)

# class "bathy"
class(atl)
summary(atl)

# test plot.bathy
plot(atl, deep=-8000, shallow=-1000, step=1000)
```

---

nw.atlantic.coast *Coastline data for the North West Atlantic*

Description

Coastline data for the North West Atlantic, as downloaded using the NOAA Coastline Extractor tool.

Usage

```
data(nw.atlantic.coast)
```

Details

Coastline data for the NW Atlantic was obtained using the NOAA Coastline Extractor tool. To get more coastline data, go to http://www.ngdc.noaa.gov/mgg/shorelines/.

Value

A 2-column data frame

References

see http://www.ngdc.noaa.gov/mgg/shorelines/

See Also

nw.atlantic

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic,nw.atlantic.coast)
atl <- as.bathy(nw.atlantic)

## the function plot below plots only isobaths:
## - isobaths between 8000-4000 in light grey,
## - isobaths between 4000-500 in dark grey (to emphasize seamounts)

plot(atl, deep=c(-8000,-4000), shallow=c(-4000,-500), step=c(500,500),
   lwd=c(0.5,0.5,1.5),lty=c(1,1,1),
   col=c("grey80", "grey20", "blue"),
   drawlabels=c(FALSE,FALSE,FALSE) )

## the coastline can be added from a different source,
## and can therefore have a different resolution:
lines(nw.atlantic.coast)

## add a geographical reference on the coast:
points(-71.064,42.358, pch=19); text(-71.064,42.358,"Boston", adj=c(1.2,0))
```

---

outline.buffer *Get a buffer in a proper format to allow the plotting of its outline*

Description

Get a buffer in a format suitable for plotting its outline. `outline.buffer` replaces any `NA` values in `buffer` by 0 and non `NA` values by -1.

Usage

```
outline.buffer(buffer)
```

Arguments

`buffer`	a buffer object of class `bathy` (i.e. `bathy` matrix containing depth/altitude values within the buffer and `NA`s outside)

Details

This function is essentially used to prepare a buffer or a combination of buffers for plotting its outline on a bathymetric map.

Value

An object of class `bathy` of the same dimension as `buffer` containing only zeros (outside the buffer area) and -1 values (within the buffer).

Author(s)

Benoit Simon-Bouhet

See Also

`create.buffer`, `combine.buffers`, `plot.bathy`

Examples

```
# load and plot a bathymetry
data(florida)
plot(florida, lwd = 0.2)
plot(florida, n = 0, lwd = 0.7, add = TRUE)

# add points around which a buffer will be computed
loc <- data.frame(c(-80,-82), c(26,24))
points(loc, pch = 19, col = "red")

# compute buffer
buf <- create.buffer(florida, loc, radius=1.5)

# plot buffer outline
plot(outline.buffer(buf), add=TRUE, lwd=.7, col=2)
```

---

palette.bathy *Build a bathymetry- and/or topography-constrained color palette*

Description

Build a constrained color palette based on depth / altitude bounds and given colors.

Usage

```
palette.bathy(mat, layers, land=FALSE, default.col="white")
```

Arguments

mat	a matrix of bathymetric data, class bathy not required.
layers	a list of depth bounds and colors (see below)
land	logical. Wether to consider land or not (default is FALSE)
default.col	a color for the area of the matrix not bracketed by the list supplied to layers

Details

palette.bathy allows the production of color palettes for specified bathymetric and/or topographic layers. The layers argument must be a list of vectors. Each vector corresponds to a bathymetry/topography layer (for example, one layer for bathymetry and one layer for topography). The first and second elements of the vector are the minimum and maximum bathymetry/topography, respectively. The other elements of the vector (3, onward) correspond to colors (see example below). palette.bathy is called internally by plot.bathy when the image argument is set to TRUE.

Value

A vector of colors which size depends on the depth / altitude range of the bathy matrix.

Author(s)

Eric Pante and Benoit Simon-Bouhet

See Also

plot.bathy

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

# creating depth-constrained palette for the ocean only
    newcol <- palette.bathy(mat=atl,
layers = list(c(min(atl), 0, "purple", "blue", "lightblue")),
land = FALSE, default.col = "grey" )
plot(atl, land = FALSE, n = 10, lwd = 0.5, image = TRUE,
bpal = newcol, default.col = "grey")

# same:
plot(atl, land = FALSE, n = 10, lwd = 0.5, image = TRUE,
bpal = list(c(min(atl), 0, "purple", "blue", "lightblue")),
default.col = "gray")

# creating depth-constrained palette for 3 ocean "layers"
newcol <- palette.bathy(mat = atl, layers = list(
c(min(atl), -3000, "purple", "blue", "grey"),
c(-3000, -150, "white"),
c(-150, 0, "yellow", "green", "brown")),
```

```
land = FALSE, default.col = "grey")
plot(atl, land = FALSE, n = 10, lwd = 0.7, image = TRUE,
bpal = newcol, default.col = "grey")

# same
plot(atl, land = FALSE, n = 10, lwd = 0.7, image = TRUE,
bpal = list(c(min(atl), -3000, "purple","blue","grey"),
c(-3000, -150, "white"),
c(-150, 0, "yellow", "green", "brown")),
default.col = "grey")

# creating depth-constrained palette for land and ocean
newcol <- palette.bathy(mat= atl, layers = list(
c(min(atl),0,"purple","blue","lightblue"),
c(0, max(atl), "gray90", "gray10")),
land = TRUE)
plot(atl, land = TRUE, n = 10, lwd = 0.5, image = TRUE, bpal = newcol)

# same
plot(atl, land = TRUE, n = 10, lwd = 0.7, image = TRUE,
bpal = list(
c(min(atl), 0, "purple", "blue", "lightblue"),
c(0, max(atl), "gray90", "gray10")))
```

`path.profile` *Geographic coordinates, kilometric distance and depth along a path*

Description

Computes and plots the depth/altitude along a transect or path

Usage

```
path.profile(path,bathy,plot=FALSE, ...)
```

Arguments

`path`	2-columns matrix of longitude and latitude as obtained from `lc.dist` with argument `dist=TRUE`.
`bathy`	bathymetric data matrix of class `bathy`.
`plot`	logical. Should the depth profile be plotted?
`...`	when `plot=TRUE`, other arguments to be passed to `plotProfile`, such as graphical parameters (see `par` and `plotProfile`).

Value

a four-columns matrix containing longitude, latitude, kilometric distance from the start of a route and depth for a set of points along a route. Optionally (i.e. when `plot=TRUE`) a bivariate plot of depth against the kilometric distance from the starting point of a transect or least cost path.

Author(s)

Benoit Simon-Bouhet

See Also

plotProfile

Examples

```
# Loading an object of class bathy and a data.frame of locations
require(mapdata)
data(hawaii)
data(hawaii.sites)

# Preparing a color palette for the bathymetric map
pal <- colorRampPalette(c("black","darkblue","blue","lightblue"))

# Plotting the bathymetric data and the path between locations
# (the path starts on location 1)
plot(hawaii,image=TRUE,bpal=pal(100),col="grey40",lwd=.7,
     main="Bathymetric map of Hawaii")
map("worldHires",res=0,fill=TRUE,col=rgb(.8,.95,.8,.7),add=TRUE)
lines(hawaii.sites,type="o",lty=2,lwd=2,pch=21,
      col="yellow",bg=col2alpha("yellow",.9),cex=1.2)
text(hawaii.sites[,1], hawaii.sites[,2],
     lab=rownames(hawaii.sites),pos=c(3,3,4,4,1,2),col="yellow")

# Computing and plotting the depth profile for this path
profile <- path.profile(hawaii.sites,hawaii,plot=TRUE,
                        main="Depth profile along the path\nconnecting the 6 sites")
summary(profile)
```

---

plot.bathy *Ploting bathymetric data*

Description

Plots contour map from bathymetric data matrix of class bathy

Usage

```
## S3 method for class 'bathy'
plot(x, image=FALSE, bpal=NULL, land=FALSE,
           deepest.isobath, shallowest.isobath, step, n=20,
           lwd=1, lty=1, col="black", default.col="white", drawlabels = FALSE,
           xlab="Longitude", ylab="Latitude", asp=1, ...)
```

Arguments

x	bathymetric data matrix of class bathy, imported using read.bathy
image	whether or not to color depth layers (default is FALSE)
bpal	if image is TRUE, either NULL (default: a simple blue color palette is used), a vector of colors, or a list of depth bounds and colors (see below)
land	whether or not to use topographic data that may be available in the bathy dataset (default is FALSE)
deepest.isobath	deepest isobath(s) to plot
shallowest.isobath	shallowest isobath(s) to plot
step	distance(s) between two isobaths
n	if the user does not specify the range within which isobaths should be plotted, about n isobaths are automatically plotted within the depth range of the bathy matrix (default is 20).
lwd	isobath line(s) width (default is 1)
lty	isobath line type(s) (default is 1)
col	isobath line color(s) (default is black)
default.col	if image is TRUE, a color for the area of the matrix not bracketed by the list supplied to bpal (see below; default is white)
drawlabels	whether or not to plot isobath depth as a label (default is FALSE); may contain several elements
xlab	label for the x axis of the plot
ylab	label for the y axis of the plot
asp	numeric, giving the aspect ratio y/x of the plot. See plot.window
...	Other arguments to be passed either to countour (default) or to image when argument image=TRUE.

Details

plot.bathy uses the base contour and image functions. If a vector of isobath characteristics is provided, different types of isobaths can be added to the same plot using a single call of plot.bathy (see examples)

If image=TRUE, the user has three choices for colors: (1) bpal can be set to NULL, in which case a default blue color palette is generated; (2) colors can be user-defined as in example 4, in which case the palette can be generated with function colorRampPalette (colors are then supplied as a vector to plot.bathy) ; (3) colors can be constrained to bathymetry- and/or topography. In this last case, a list of vectors is supplied to plot.bathy (example 7): each vector corresponds to a bathymetry/topography layer (for example, one layer for bathymetry and one layer for topography). The first and second elements of the vector are the minimum and maximum bathymetry/topography, respectively. The other elements of the vector (3, onward) correspond to colors (see example 7).

Value

a bathymetric map with isobaths

Note

`plot.bathy` uses a matrix of class `bathy`, and can therefore be substituted for `plot`.

Author(s)

Eric Pante and Benoit Simon-Bouhet

References

Eric Pante, Benoit Simon-Bouhet (2013) marmap: A Package for Importing, Plotting and Analyzing Bathymetric and Topographic Data in R. PLoS ONE 8(9): e73051. doi:10.1371/journal.pone.0073051. http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0073051

See Also

`read.bathy`, `summary.bathy`, `nw.atlantic`, `metallo`

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

## Example 1: a simple marine chart
plot(atl) # without specifying any isobath parameters
plot(atl, n=5, drawlabels=TRUE) # with about 5 isobaths
plot(atl, deep=-8000, shallow=0, step=1000) # with isobath parameters

## Example 2: taking advantage of multiple types of isobaths
plot(atl, deep=c(-8000,-2000,0), shallow=c(-2000,-100,0), step=c(1000,100,0),
 lwd=c(0.5,0.5,1),lty=c(1,1,1),col=c("grey80","red", "blue"),
 drawlabels=c(FALSE,FALSE,FALSE) )

## Example 3: plotting a colored map with the default color palette
plot(atl, image=TRUE, deep=c(-8000,0), shallow=c(-1000,0), step=c(1000,0),
     lwd=c(0.5,1), lty=c(1,1), col=c("grey","black"), drawlabels=c(FALSE,FALSE))

## Example 4: make a pretty custom color ramp
colorRampPalette(c("purple","lightblue","cadetblue2","cadetblue1","white")) -> blues

plot(atl, image=TRUE, bpal=blues(100), deep=c(-6500,0), shallow=c(-50,0), step=c(500,0),
     lwd=c(0.3,1), lty=c(1,1), col=c("black","black"), drawlabels=c(FALSE,FALSE))

scaleBathy(atl, deg=3, x="bottomleft", inset=5)

## Example 5: add points corresponding to sampling locations
##            point colors correspond to the sampling depth
```

```
par(mai=c(1,1,1,1.5))
plot(atl, deep=c(-4500,0), shallow=c(-50,0), step=c(500,0),
     lwd=c(0.3,1), lty=c(1,1), col=c("black","black"), drawlabels=c(FALSE,FALSE))

# add a title to the plot
title(main="Distribution of coral samples\non the New England and Corner Rise seamounts")
# add a scale
scaleBathy(atl, deg=3, x="bottomleft", inset=5)

# add a geographical reference on the coast:
points(-71.064,42.358, pch=19)
text(-71.064,42.358,"Boston", adj=c(1.2,0))

# prepare colors for the sampling locations:
data(metallo) ## see dataset metallo
max(metallo$depth, na.rm=TRUE) -> mx
colorRampPalette(c("white","lightyellow","lightgreen","blue","lightblue1","purple")) -> ramp
blues <- ramp(max(metallo$depth))

# plot sampling locations:
points(metallo, col="black", bg=blues[metallo$depth], pch=21,cex=1.5)
library(shape)
colorlegend(zlim=c(-mx,0), col=rev(blues), main="depth (m)",posx=c(0.85,0.88))

## Example 6: use packages maps and mapdata in combination with marmap
# use maps and mapdata to plot the coast
library(maps)
library(mapdata)
map('worldHires',xlim=c(-75,-46),ylim=c(32,44), fill=TRUE, col="grey")
box();axis(1);axis(2)

# add bathymetric data from 'bathy' data
    plot(atl, add=TRUE, lwd=.3, deep=-4500, shallow=-10, step=500,
drawlabel=FALSE, col="grey50")

## Example 7: provide a list of depths and colors to argument bpal to finely tune palette
# check out ?palette.bathy to see details on how the palette is handled

# creating depth-constrained palette for the ocean only
plot(atl, land = FALSE, n = 10, lwd = 0.5, image = TRUE,
     bpal = list(c(min(atl), 0, "purple", "blue", "lightblue")),
     default.col = "gray")

# creating depth-constrained palette for 3 ocean "layers"
plot(atl, land = FALSE, n = 10, lwd = 0.7, image = TRUE,
     bpal = list(c(min(atl), -3000, "purple","blue","grey"),
                 c(-3000, -150, "white"),
                 c(-150, 0, "yellow", "green", "brown")),
     default.col = "grey")

# creating depth-constrained palette for land and ocean
plot(atl, land = TRUE, n = 10, lwd = 0.7, image = TRUE,
     bpal = list(c(min(atl), 0, "purple", "blue", "lightblue"),
```

```
                    c(0, max(atl), "gray90", "gray10")))
```

---

plotArea *Plotting projected surface areas*

Description

Highlights the projected surface area for specific depth layers on an existing bathymetric/hypsometric map

Usage

```
plotArea(area, col)
```

Arguments

area	a list of 4 elements as produced by get.area.
col	color of the projected surface area on the map.

Author(s)

Benoit Simon-Bouhet

See Also

get.area, plot.bathy, areaPolygon

Examples

```
# load and plot a bathymetry
data(florida)
plot(florida, lwd = 0.2)
plot(florida, n = 1, lwd = 0.7, add = TRUE)

# Create a point and a buffer around this point
loc <- data.frame(-80, 26)
buf <- create.buffer(florida, loc, radius=1.8)

# Get the surface within the buffer for several depth slices
surf1 <- get.area(buf, level.inf=-200, level.sup=-1)
surf2 <- get.area(buf, level.inf=-800, level.sup=-200)
surf3 <- get.area(buf, level.inf=-3000, level.sup=-800)

s1 <- round(surf1$Square.Km)
s2 <- round(surf2$Square.Km)
s3 <- round(surf3$Square.Km)

# Add buffer elements on the plot
col.surf1 <- rgb(0.7, 0.7, 0.3, 0.3)
```

```
col.surf2 <- rgb(0, 0.7, 0.3, 0.3)
col.surf3 <- rgb(0.7, 0, 0, 0.3)

plotArea(surf1, col = col.surf1)
plotArea(surf2, col = col.surf2)
plotArea(surf3, col = col.surf3)
plot(outline.buffer(buf), add = TRUE, lwd = 0.7)
points(loc, pch = 19, col = "red")

## Add legend
legend("topleft", fill = c(col.surf1, col.surf2, col.surf3),
       legend = c(paste("]-200 ; -1] -",s1,"km2"),
      paste("]-800 ; -200] -",s2,"km2"),
  paste("]-3000 ; -800] -",s3,"km2")))
```

`plotProfile` *Ploting bathymetric data along a transect or path*

Description

Plots the depth/altitude along a transect or path

Usage

```
plotProfile(profile,shadow=TRUE,xlim,ylim,col.sea,col.bottom,xlab,ylab, ...)
```

Arguments

`profile`	4-columns matrix obtained from `get.transect` with argument `dist=TRUE`, or from `path.profile`.
`shadow`	logical. Should the depth profile cast a shadow over the plot background?
`xlim, ylim`	numeric vectors of length 2, giving the x and y coordinates ranges. If unspecified, `xlim` values are based on the length of the transect or path and `ylim` values are based on the depth range of the bathymetric matrix `bathy`.
`col.sea`	color for the sea area of the plot. Defaults to `rgb(130/255,180/255,212/255)`
`col.bottom`	color for the bottom area of the plot. Defaults to `rgb(198/255,184/255,151/255)`
`xlab, ylab`	titles for the x and y axes. If unspecified, `xlab="Distance from start of transect (km)"` and `ylab="Depth (m)"`
`...`	arguments to be passed to methods, such as graphical parameters (see `par`)

Value

a bivariate plot of depth against the kilometric distance from the starting point of a transect or least cost path.

Note

path.profile with argument plot set to TRUE plots depth profiles with default values for all arguments of plotProfile.

Author(s)

Benoit Simon-Bouhet

See Also

path.profile, plot.bathy

Examples

```
# Example 1:
data(celt)
layout(matrix(1:2,nc=1),height=c(2,1))
par(mar=c(4,4,1,1))
plot(celt,n=40,draw=TRUE)
points(c(-6.34,-5.52),c(52.14,50.29),type="o",col=2)

tr <- get.transect(celt, x1 = -6.34, y1 = 52.14, x2 = -5.52, y2 = 50.29, distance = TRUE)
plotProfile(tr)

# Example 2:
layout(matrix(1:2,nc=1),height=c(2,1))
par(mar=c(4,4,1,1))
plot(celt,n=40,draw=TRUE)
points(c(-5,-6.34),c(49.8,52.14),type="o",col=2)

tr2 <- get.transect(celt, x1 = -5, y1 = 49.8, x2 = -6.34, y2 = 52.14, distance = TRUE)
plotProfile(tr2)

# Example 3: click several times on the map and press ESC
## Not run:
layout(matrix(1:2,nc=1),height=c(2,1))
par(mar=c(4,4,1,1))
data(florida)
plot(florida,image=TRUE,dra=TRUE,land=TRUE,n=40)

out <- path.profile(as.data.frame(locator(type="o",col=2,pch=19,cex=.8)),florida)
plotProfile(out)

## End(Not run)
```

`read.bathy` *Read bathymetric data in XYZ format*

Description

Reads a three-column table containing longitude (x), latitude (y) and depth (z) data.

Usage

```
read.bathy(xyz, header = FALSE, sep = ",", ...)
```

Arguments

`xyz`	three-column table with longitude (x), latitude (y) and depth (z) (no default)
`header`	whether this table has a row of column names (default = FALSE)
`sep`	character separating columns, (default=",")
`...`	further arguments to be passed to `read.table()`

Details

Allows direct import of data from the NOAA GEODAS Grid Translator webpage (http://www.ngdc.noaa.gov/mgg/gdas/gd_designagrid.html). To prepare data from NOAA, fill the custom grid form, and choose "XYZ (lon,lat,depth)" as the "Output Grid Format", "No Header" as the "Output Grid Header", and either of the space, tab of comma as the column delimiter (either can be used, but "comma" is the default import format of `read.bathy`). Choose "omit empty grid cells" to reduce memory usage.

Value

The output of `read.bathy` is a matrix of class `bathy`, which dimensions depends on the resolution of the grid uploaded from the NOAA GEODAS server (Grid Cell Size). The class `bathy` has its own methods for summarizing and ploting the data.

Author(s)

Eric Pante

See Also

`summary.bathy`, `plot.bathy`, `readGEBCO.bathy`

Examples

```
# load NW Atlantic data
data(nw.atlantic)

# write example file to disk
write.table(nw.atlantic, "NW_Atlantic.csv", sep=",", quote=FALSE, row.names=FALSE)

# use read.bathy
read.bathy("NW_Atlantic.csv", header=TRUE) -> atl

# remove temporary file
system("rm NW_Atlantic.csv") # remove file, for unix-like systems

# class "bathy"
class(atl)

# summarize data of class "bathy"
summary(atl)
```

---

readGEBCO.bathy *Read bathymetric data from a GEBCO file*

Description

Imports 30-sec and 1-min bathymetric data from a .nc file downloaded on the GEBCO website.

Usage

```
readGEBCO.bathy(file, resolution = 1)
```

Arguments

file	name of the .nc file
resolution	resolution of the grid, in units of the selected database (default is 1; see details)

Details

readGEBCO.bathy reads a 30 arcseconds or 1 arcminute bathymetry file downloaded from the GEBCO (General Bathymetric Chart of the Oceans) website (British Oceanographic Data Center). The website allows the download of bathymetric data in the netCDF format. readGEBCO.bathy uses the ncdf package to load the data into R, and parses it into an object of class bathy.

Data can be downloaded from the 30 arcseconds database (GEBCO_08) or the 1 arcminute database (GEBCO_1min, the default). A third database type, GEBCO_08 SID, is available from the website. This database includes a source identifier specifying which grid cells have depth information based on soundings ; it does not include bathymetry or topography data. readGEBCO.bathy can read this type of database and only the SID information will be included in the object of class bathy.

Therefore, to display a map with both the bathymetry and the SID information, you will have to download both datasets from GEBCO, and import and plot both independently.

The argument `resolution` specifies the resolution of the object of class `bathy`. Because the resolution of GEBCO data is rather fine, we offer the possibility of downsizing the dataset with `resolution`. `resolution` is in units of the selected database: in "GEBCO_1min", `resolution` is in minutes; in "GEBCO_08", `resolution` is in 30 arcseconds (that is, `resolution = 3` corresponds to 3x30sec, or 1.5 arcminute).

Value

The output of `readGEBCO.bathy` is a matrix of class `bathy`, which dimensions depends on the resolution specified (one-minute, the original GEBCO resolution, is the default). The class `bathy` has its own methods for summarizing and ploting the data.

Author(s)

Eric Pante and Benoit Simon-Bouhet

References

British Oceanographic Data Center: General Bathymetric Chart of the Oceans gridded bathymetric data sets (accessed Oct 5, 2013) http://www.bodc.ac.uk/data/online_delivery/gebco/

General Bathymetric Chart of the Oceans website (accessed Oct 5, 2013) http://www.gebco.net

David Pierce (2011). ncdf: Interface to Unidata netCDF data files. R package version 1.6.6. http://CRAN.R-project.org/package=ncdf

See Also

`getNOAA.bathy`, `read.bathy`, `plot.bathy`

Examples

```
## Not run:
# This example will not run, and we do not provide the dummy "gebco_file.nc" file,
# because a copyright license must be signed on the GEBCO website before the data can be
# downloaded and used. We just provide this line as an example for synthax.
  readGEBCO.bathy(file="gebco_file.nc", resolution=1) -> nw.atl

# Second not-run example, with GEBCO_08 and SID:
  readGEBCO.bathy("gebco_08_7_38_10_43_corsica.nc") -> med
  summary(med) # the bathymetry data

  readGEBCO.bathy("gebco_SID_7_38_10_43_corsica.nc")-> sid
  summary(sid) # the SID data

  colorRampPalette(c("lightblue","cadetblue1","white")) -> blues # custom col palette
  plot(med, n=1, im=T, bpal=blues(100)) # bathymetry

  as.numeric(rownames(sid)) -> x.sid
  as.numeric(colnames(sid)) -> y.sid
```

```
  contour(x.sid, y.sid, sid, drawlabels=FALSE, lwd=.1, add=TRUE) # SID

## End(Not run)
```

---

scaleBathy *Adds a scale to a map*

Description

Uses geographic information from object of class bathy to calculate and plot a scale in kilometer.

Usage

```
scaleBathy(mat, deg=1, x="bottomleft", y=NULL, inset=10, angle=90, ...)
```

Arguments

mat	bathymetric data matrix of class bathy, imported using read.bathy
deg	the number of degrees of longitudes to convert into kilometers (default is 1)
x, y	the coordinates used to plot the scale on the map (see Details)
inset	when x is a keyword (e.g. "bottomleft"), inset is a percentage of the plotting space controlling the relative position of the plotted scale (see Examples)
angle	angle from the shaft of the arrow to the edge of the arrow head
...	further arguments to be passed to text

Details

scaleBathy is a simple utility to add a scale to the lower left corner of a bathy plot. The distance in kilometers between two points separated by 1 degree longitude is calculated based on the minimum latitude of the bathy object used to plot the map. Option deg allows the user to plot the distance separating more than one degree (default is one).

The plotting coordinates x and y either correspond to two points on the map (i.e. longitude and latitude of the point where the scale should be plotted), or correspond to a keyword (set with x, y being set to NULL) from the list "bottomright", "bottomleft", "topright", "topleft". When a keyword is used, the option inset controls how far the scale will be from the edges of the plot.

Value

a scale added to the active graphical device

Note

The calculation formula is from function map.scale of package maps. 6372.798 km is used as the Earth radius.

Author(s)

Eric Pante

See Also

plot.bathy

Examples

```
# load NW Atlantic data and convert to class bathy
data(nw.atlantic)
atl <- as.bathy(nw.atlantic)

# a simple example
plot(atl, deep=-8000, shallow=-1000, step=1000, lwd=0.5, col="grey")
scaleBathy(atl, deg=4)

# using keywords to place the scale with inset=10%
par(mfrow=c(2,2))
plot(atl, deep=-8000, shallow=-1000, step=1000, lwd=0.5, col="grey")
scaleBathy(atl, deg=4, x="bottomleft", y=NULL)
plot(atl, deep=-8000, shallow=-1000, step=1000, lwd=0.5, col="grey")
scaleBathy(atl, deg=4, x="bottomright", y=NULL)

# using keywords to place the scale with inset=20%
plot(atl, deep=-8000, shallow=-1000, step=1000, lwd=0.5, col="grey")
scaleBathy(atl, deg=4, x="topleft", y=NULL, inset=20)
plot(atl, deep=-8000, shallow=-1000, step=1000, lwd=0.5, col="grey")
scaleBathy(atl, deg=4, x="topright", y=NULL, inset=20)
```

---

space.pies *Automatic placement of piecharts on maps*

Description

Attemps to automatically place piecharts on maps, avoiding overlap. Work in progress...

Usage

```
space.pies(x, y, pie.slices, pie.colors=NULL, pie.radius=1, pie.space=5,
              link=TRUE, seg.lwd=1, seg.col=1, seg.lty=1, coord=NULL)
```

Arguments

x	the longitude of the anchor point for the piechart
y	the latitude of the anchor point for the piechart

pie.slices	a table with the counts to draw pies (col: pie categories, or slices; rows: sites on the map)
pie.colors	a table with the colors to draw pies (col: pie categories, or slices; rows: sites on the map)
pie.radius	size of the piechart
pie.space	factor of spacing between the anchor and the pie (the larger, the farther the pie from the anchor)
link	logical; whether to add a segment to link pie and anchor
seg.lwd	the line width of the link
seg.col	the line color of the link
seg.lty	the line type of the link
coord	when coord = NULL (default), placement is automatic. Otherwise, a 2-col table of lon/lat for pies.

Details

space.pies tries to position piecharts on a map while avoiding overlap between them. The function heavily relies on two other functions. floating.pie from package plotrix is used to draw individual piecharts. floating.pie treats one pie at a time; space.pies can handle one or multiple pies by looping floating.pie. pointLabels from package maptools was modified to find the best placement for the pies, given their size and distance from their anchor point. pointLabels was originally meant to automatically place text labels, not objects; the modified version contained in space.pies uses the coordinates chosen by pointLabels for text. The algorithm used is simulating annealing (SANN). You can get a different result each time you run space.pies, because pointLabel finds one good solution out of many. If you are not satisfied by the solution, you can try running the function again.

The argument coord allows to choose between the automatic placement outlined above, and a user-defined list of longitudes and latitudes (in a two-column table format) for plotting the piecharts.

Anchor point: spatial location of the data corresponding to the piechart (e.g. a sampling point).

Value

Piechart(s) added to a plot.

Author(s)

Eric Pante, using functions floating.pie from package plotrix and pointLabel from maptools.

References

Bivand, R. and Lewin-Koh, N. (2013) maptools: Tools for reading and handling spatial objects. R package version 0.8-25. http://CRAN.R-project.org/package=maptools

Lemon, J. (2006) Plotrix: a package in the red light district of R. R-News, 6(4): 8-12.

SANN code implemented in pointLabel based on: Jon Christensen, Joe Marks, and Stuart Shieber. Placing text labels on maps and diagrams. In Paul Heckbert, editor, Graphics Gems IV, pages 497-504. Academic Press, Boston, MA, 1994.

See Also

plot.bathy, floating.pie, pointLabel

Examples

```
# fake frequencies to feed to space.pies()
sample(seq(10,90,5), 11)-> freq.a
100-freq.a -> freq.b
rep("lightblue",11) -> col.a
rep("white",11) -> col.b

# some coordinates on the NW Atlantic coast, and on seamounts
x = c(-74.28487,-73.92323,-73.80753,-72.51728,-71.12418,
  -69.81176,-69.90715,-70.43201,-70.17135,-69.43912,-65.49608)
y = c(39.36714,39.98515,40.40316,40.79654,41.49872,41.62076,
  41.99805,42.68061,43.40714,43.81499,43.36471)
pts.coast = data.frame(x,y, freq.a, freq.b, col.a, col.b)

x = c(-66.01404,-65.47260,-63.75456,-63.26082,-62.12838,
      -60.46885,-59.96952,-56.90925,-52.20397,-51.32288,-50.72461)
y = c(39.70769,39.39064,38.83020,38.56479,38.01881,38.95405,
      37.55675,34.62617,36.15592,36.38992,35.91779)
pts.smt = data.frame(x,y, freq.a, freq.b, col.a, col.b)

# prepare the plot
data(nw.atlantic) ; atl <- as.bathy(nw.atlantic)
plot(atl, deep=-8000, shallow=0, step=1000,col="grey")
points(pts.coast,pch=19,col="blue", cex=0.5)
points(pts.smt,pch=19,col="blue", cex=0.5)

# automatic placement of piecharts with space.pies
space.pies(pts.coast[,1], pts.coast[,2],
           pie.slices=pts.coast[,3:4], pie.colors=pts.coast[,5:6], pie.radius=0.5)
space.pies(pts.smt[,1], pts.smt[,2],
           pie.slices=pts.smt[,3:4], pie.colors=pts.coast[,5:6], pie.radius=0.5)
```

---

subsetBathy *Creates bathy objects from larger bathy objects*

Description

Generates rectangular or non rectangular bathy objects by extracting bathymetric data from larger bathy objects.

Usage

```
subsetBathy(mat, x, y=NULL, locator=TRUE, ...)
```

Arguments

mat	Bathymetric data matrix of class bathy, as imported with read.bathy.
x	Either a list of two elements (numeric vectors of longitude and latitude), a 2-column matrix or data.frame of longitudes and latitudes, or a numeric vector of longitudes.
y	Either NULL (default) or a numerical vector of latitudes. Ignored if x is not a numeric vector.
locator	Logical. Whether to choose data points interactively with a map or not. If TRUE (default), a bathymetric map must have been plotted and both x and y are both ignored.
...	Further arguments to be passed to locator when the interactive mode is used (locator=TRUE).

Details

subsetBathy allows the user to generate new bathy objects by extracting data from larger bathy objects. The extraction of bathymetric data can be done interactively by clicking on a bathymetric map, or by providing longitudes and latitudes for the boundaries for the new bathy object. If two data points are provided, a rectangular area is selected. If more than two points are provided, a polygon is defined by linking the points and the bathymetic data is extracted within the polygon only. subsetBathy relies on the point.in.polygon function from package sp to identify which points of the initial bathy matrix lie witin the boundaries of the user-defined polygon.

Value

A matrix of class bathy.

Author(s)

Benoit Simon-Bouhet

References

Pebesma, EJ, RS Bivand, (2005). Classes and methods for spatial data in R. R News 5 (2), http://cran.r-project.org/doc/Rnews/.

Bivand RS, Pebesma EJ, Gomez-Rubio V (2013). Applied spatial data analysis with R, Second edition. Springer, NY. http://www.asdar-book.org/

See Also

plot.bathy, get.depth, summary.bathy, aleutians

Examples

```
# load aleutians dataset
data(aleutians)

# create vectors of latitude and longitude to define the boundary of a polygon
```

```
lon <- c(188.56, 189.71, 191, 193.18, 196.18, 196.32, 196.32, 194.34, 188.83)
lat <- c(54.33, 55.88, 56.06, 55.85, 55.23, 54.19, 52.01, 50.52, 51.71)


# plot the initial bathy and overlay the polygon
plot(aleutians, image=TRUE, land=TRUE, lwd=.2)
polygon(lon,lat)

# Use of subsetBathy to extract the new bathy object
zoomed <- subsetBathy(aleutians, x=lon, y=lat, locator=FALSE)

# plot the new bathy object
dev.new() ; plot(zoomed, land=TRUE, image=TRUE, lwd=.2)

# alternativeley once the map is plotted, use the interactive mode:
## Not run:
plot(aleutians, image=TRUE, land=TRUE, lwd=.2)
zoomed2 <- subsetBathy(aleutians, pch=19, col=3)
dev.new() ; plot(zoomed2, land=TRUE, image=TRUE, lwd=.2)

## End(Not run)
# click several times and press Escape
```

---

subsetSQL *Creating and querying local SQL database for bathymetric data*

Description

subsetSQL queries the local SQL database created with setSQL to extract smaller data subsets.

Usage

```
setSQL(bathy, header = TRUE, sep = ",", db.name = "bathy_db")
subsetSQL(min_lon, max_lon, min_lat, max_lat, db.name = "bathy_db")
```

Arguments

bathy	A text file containing a comma-separated, three-column table with longitude, latitude and depth data (no default)
header	does the xyz file contains a row of column names (default = TRUE)
sep	character separating columns in the xyz file, (default=",")
min_lon	minimum longitude of the data to be extracted from the local SQL database
max_lon	maximum longitude of the data to be extracted from the local SQL database
min_lat	minimum latitude of the data to be extracted from the local SQL database
max_lat	maximum latitude of the data to be extracted from the local SQL database
db.name	The name of (or path to) the SQL database to be created on disk by setSQL or from which subsetSQL will extract data ("bathy_db" by default)

Details

Functions setSQL and subsetSQL were built to work together. setSQL builds an SQL database and saves it on disk. subsetSQL queries that local database and the fields min_lon, max_lon, etc, are used to extract a subset of the database. The functions were built as two entities so that multiple queries can be done multiple times, without re-building the database each time. These functions were designed to access the very large (>5Go) ETOPO1 file that can be downloaded from the NOAA website (http://www.ngdc.noaa.gov/mgg/global/global.html)

Value

setSQL returns TRUE if the database was successfully created. subsetSQL returns a matrix of class bathy that can directly be used with plot.bathy.

Note

If unspecified, db.name is set to "bathy_db" by default. Thus, theere must be no database file called bathy_db in the working directory prior to running setSQL unless a different name is used for the new database. Make sure that your "bathy" input is a xyz text file (for function setSQL) with 3 columns containing longitude, latitude and depth data, in that order. setSQL and subsetSQL were modified on Nov. 2, 2014 to comply with RSQLite 1.0.0.

Author(s)

Eric Pante

References

Amante, C. and B. W. Eakins, ETOPO1 1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis. NOAA Technical Memorandum NESDIS NGDC-24, 19 pp, March 2009. http://www.ngdc.noaa.gov/mgg/global/relief/ETOPO1/docs/ETOPO1.pdf

Examples

```
# load NW Atlantic data
data(nw.atlantic)

# write data to disk as a comma-separated text file
write.table(nw.atlantic, "NW_Atlantic.csv", sep=",", quote=FALSE, row.names=FALSE)

## Not run:
# prepare SQL database
setSQL(bathy="NW_Atlantic.csv")

# uses data from the newly-created SQL database:
subsetSQL(min_lon=-70,max_lon=-50,
            min_lat=35, max_lat=41) -> test

# visualize the results (of class bathy)
summary(test)
```

```
# remove temporary database and CSV files
system("rm bathy_db") # remove file, for unix-like systems
system("rm NW_Atlantic.csv") # remove file, for unix-like systems

## End(Not run)
```

---

summary.bathy *Summary of bathymetric data of class* bathy

Description

Summary of bathymetric data of class bathy. Provides geographic bounds and resolution (in minutes) of the dataset, statistics on depth data, and a preview of the bathymetric matrix.

Usage

```
## S3 method for class 'bathy'
summary(object, ...)
```

Arguments

object	object of class bathy
...	additional arguments affecting the summary produced (see base function summary).

Value

Information on the geographic bounds of the dataset (minimum and maximum latitude and longitude), resolution of the matrix in minutes, statistics on the depth data (e.g. min, max, median...), and a preview of the data.

Author(s)

Eric Pante and Benoit Simon-Bouhet

See Also

read.bathy, plot.bathy

Examples

```
# load NW Atlantic data
data(nw.atlantic)

# use as.bathy
atl <- as.bathy(nw.atlantic)

# class bathy
class(atl)
```

```
# summarize data of class bathy
summary(atl)
```

`trans.mat` *Transition matrix*

Description

Creates a transition object to be used by `lc.dist` to compute least cost distances between locations.

Usage

```
trans.mat(bathy,min.depth=0,max.depth=NULL)
```

Arguments

`bathy`	A matrix of class `bathy`.
`min.depth, max.depth`	Numeric. The range of depth between which the path will be possible. The default (`min.depth=0` and max.depth=NULL) indicates that the transition between cells of the grid is possible between 0 meters depth and the maximum depth of `bathy`. See details

Details

`trans.mat` creates a transition object usable by `lc.dist` to computes least cost distances between a set of locations. `trans.mat` rely on the function `raster` from package `raster` (Hijmans & van Etten, 2012. http://CRAN.R-project.org/package=raster) and on `transition` from package `gdistance` (van Etten, 2011. http://CRAN.R-project.org/package=gdistance).

The transition object contains the probability of transition from one cell of a bathymetric grid to adjacent cells and depends on user defined parameters. `trans.mat` is especially usefull when least cost distances need to be calculated between several locations at sea. The default values for `min.depth` and `max.depth` ensure that the path computed by `lc.dist` will be the shortest path possible at sea avoiding land masses. The path can be constrained to a given depth range by setting manually `min.depth` and `max.depth`. For instance, it is possible to limit the possible paths to the continental shelf by setting `max.depth=-200`. Inaccuracies of the bathymetric data can occasionally result in paths crossing land masses. Setting `min.depth` to low negative values (e.g. -10 meters) can limit this problem.

`trans.mat` takes also advantage of the function `geoCorrection` from package `gdistance` (van Etten, 2012. http://CRAN.R-project.org/package=gdistance) to take into account map distortions over large areas.

Value

A transition object.

Warning

Please be aware that the use of `trans.mat` can be time consumming for large bathymetric datasets. The function takes about one minute to compute a transition matrix for the `hawaii` bathymetric data (bathymetric data of class `bathy` with 599 rows and 419 columns, see `hawaii`) on a MacBook Pro with a 2.66 GHz Intel Core i7 processor and 4 Go of RAM.

Author(s)

Benoit Simon-Bouhet

References

Jacob van Etten (2011). gdistance: distances and routes on geographical grids. R package version 1.1-2. http://CRAN.R-project.org/package=gdistance Robert J. Hijmans & Jacob van Etten (2012). raster: Geographic analysis and modeling with raster data. R package version 1.9-92. http://CRAN.R-project.org/package=raster

See Also

`lc.dist`, `hawaii`

Examples

```
# Load and plot bathymetry
data(hawaii)
summary(hawaii)
plot(hawaii)

## Not run:
# Compute transition object with no depth constraint
trans1 <- trans.mat(hawaii)

# Compute transition object with minimum depth constraint:
# path impossible in waters shallower than -200 meters depth
trans2 <- trans.mat(hawaii,min.depth=-200)

# Visualizing results
par(mfrow=c(1,2))
plot(raster(trans1), main="No depth constraint")
plot(raster(trans2), main="Constraint in shallow waters")

## End(Not run)
```

Index