Package 'oceanmap'

September 12, 2023

Type Package

Title A Plotting Toolbox for 2D Oceanographic Data

Version 0.1.4

Date 2023-09-11

Author Robert K. Bauer

Maintainer Robert K. Bauer <rkbauer@hawaii.edu>

Depends R (>= 3.5.0), maps, mapdata, raster, extrafont, reshape2,

Imports abind, fields, plotrix, methods, utils, grDevices, maptools, sp, ncdf4, stats, lubridate, ggplot2, ggedit, sf(>= 0.9-7), plotly

Description Plotting toolbox for 2D oceanographic data (satellite data, sea surface temperature, chlorophyll, ocean fronts & bathymetry). Recognized classes and formats include netcdf, Raster, '.nc' and '.gz' files.

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-11 23:20:25 UTC

SystemRequirements ImageMagick

R topics documented:

`.get.worldmap` *World Map*

Description

Creates a world map database that allows longitude ranges between -180 and 360 degrees, and thus from the Pacific to the Atlantic and vise versa. It is based on the worldHires database (which itself is based on CIA World Data Bank II data and contains approximately 2 million points representing the world coastlines and national boundaries), from which polygon irritations of the Antarctic were also corrected.

Usage

```
.get.worldmap(resolution)
```

Arguments

resolution	number that specifies the resolution with which to draw the map. Resolution 0 is the full resolution of the database [default]. Otherwise, just before polylines are plotted they are thinned: roughly speaking, successive points on the polyline that are within resolution device pixels of one another are collapsed to a single point (see the Reference for further details). Thinning is not performed if plot = FALSE or when polygons are drawn (fill = TRUE or database is a list of polygons).

Value

A list of class "map" with longitude (x) and latitude (y) positions of coastlines and state boundaries (different coastline or booundary elements are seperated by NA), single polygon names are provided by a names vector.

Author(s)

Robert K. Bauer

See Also

worldHires
https://www.evl.uic.edu/pape/data/WDB/

Examples

```
worldmap <- oceanmap:::.get.worldmap(worldmap)
str(worldmap)

## wordlmap usage in plotmap, with different center-options
# par(mfrow=c(3,1))
# plotmap(lon=c(80, -120), lat=c(-50, 10), main= "map from East to West")
# plotmap(lon=c(-120, 80), lat=c(-50, 10), main= "map from West to East")
# plotmap('tp')
```

---

add.region *adding a region to the region_definitions file*

Description

adding a region to the region_definitions-file, taking or restoring a backup of region definitions. The basic idea is to provide a region-keyword that is used to access the region-information in later related function-calls (see: v and plotmap, regions). Information consists of a region-keyword, -longname, its spatial extent (longitudes and latitudes), grid resolution, as well as default colorbar position and figure size.

The required information can be provided by an interactive **session** (widget) that leads step by step through the region definition (is set `default`), in parts by an extent-object with the missing information then completed by the **session** or by a one-row data frame that holds the entire information (see: region_definitions).

ATTENTION! When reinstalling or updating the oceanmap package, previous region definitions are getting lost! It is therefore highly recommanded to take and restore own backups (see: `backup` and `restore`).

Usage

```
add.region(add, add.px, cbx, cby, figdim, lib.folder,
       widget=T, backup=F, backup.folder='.', backup.name, restore=F,backup.regions)
```

Arguments

`add`	extent-,raster-object or dataset containing all required region definition entries (`label`, `name`, `latn`, `lats`, `lonw`, `lone`, `ncol`, `nrow`, `px`, `cbx1`, `cbx2`, `cby1`, `cby2`, `figxdim`, `figydim` and `grid.res`). Ignored when `add.px` is supplied. The values `latn`, `lats`, `lonw`, `lone` define the regions extent, `cbx1`, `cbx2`, `cby1` and `cby2` define the position of the colorbar, `gradient` the orientation of the colorbar (x for horizontal, y for vertical), `oticks` the margin where to put the colorbar ticks relative to the colorbar rectangle (`'l'` left, `'r'` right and `'b'` for bottom; `figxdim` and `figydim` set the default window size of `'.gz'`-file figures and `grid.res` the default grid resolution.
`add.px`	dataframe or list containing region data needed to read gz-compressed `'.gz'`-files. Required entries include '`label`' to identify the region, '`ncol`' and '`nrow`', to define the number of columns and rows of the 'gz'-file, respectively. These values are automatically set if missing when writing gz-compressed `'.gz'`-files (see: writebin).
`cbx`	the horizontal limits (x1, x2) of the colorbar. If missing, the user will be asked for manual colorbar placement.
`cby`	the vertical limits (y1, y2) of the colorbar. If missing, the user will be asked for manual colorbar placement.
`figdim`	numeric vector indicating the `width` and `height` of the plot device in inches. If missing and `force.figdim.widget` is set `FALSE`, figdim is assigned a default `width` and `height` of 7in, otherwise the user will be asked to resize the plot device to set plot dimensions.
`lib.folder`	Character string indicating R-library path in which the oceanmap-package is installed.
`widget`	whether an interactive session (**widget**) shall assist the data entry procedure (default is `TRUE`).
`backup`	whether the current region_definitions-file should be backuped in the folder '`backup.folder` in the file `backup.name` (default is `FALSE`). **ATTENTION!** When reinstalling or updating the oceanmap package, previous region_definitions are getting lost!
`backup.folder`	Character string indicating the folder where to store the region_definitions-file backup (default is the current working directory).

backup.name	Character string indicating the filename of the region_definitions-file backup (If restore the default is the original oceanmap-region_definitions file; if backup the default is set to 'region_definitions.bkp.%Y%m%d.rda').
restore	whether to restore a backup of the region_definitions-file (default is FALSE).
backup.regions	Vector of region indicators defining which regions should be saved in backup file.

Author(s)

Robert K. Bauer

See Also

delete.region, region_definitions, regions, plotmap, v

Examples

```
## Example 1: Add region by supplying a one-row data.frame
##            that holds the entire required information
# data(region_definitions) # load region_definitions
# lion <- region_definitions[region_definitions$label == 'lion',] # selecting Gulf of Lions region
# lion
# junk <- lion
# junk$label <- 'junk' # rename region label
# add.region(junk) # add junk region
# data(region_definitions) # reload region_definitions
# region_definitions[,1:9]

## Example 2: Delete region
#delete.region("junk") # delete junk region
#data(region_definitions) # reload region_definitions
#region_definitions[,1:9]

## Example 3: Add region by supplying an extent- or raster-object and running the widget
library(raster)

ext <- extent(0,10,50,60)
plotmap(ext)
#add.region(ext) # extent-object

r <- raster(ext)
#add.region(r) # raster-object

## Example 4: Add region by supplying raster-object, colorbar positions and running the widget
#add.region(r,cbx=c(5,9.5),cby=c(51.7,52.4))

## Example 5: Add region by running the widget
#add.region()

## Example 6: Add region by running the widget
```

```
#add.region(add.px=list(label="lion",nrow=10,ncol=10))
#data(region_definitions)
#region_definitions[region_definitions$label =="lion",]

## Example 7: Creating a backup
#add.region(backup=T)
#add.region(backup=T, backup.folder=".",backup.regions=c("lion","medw4"))

## Example 8: Restoring the backup of the original region_definitions file
#add.region(restore=T)
```

---

area_extrac *Extracts a pre-defined region from '.gz'-file and saves subset as a new '.gz'-file*

Description

Extracts a pre-defined region from '.gz'-file and saves subset as a new '.gz'-file (gzip compressed format). Basically it represents a combined call of regions, crop, raster2matrix and writebin.

Usage

```
area_extrac(obj,area)
```

Arguments

obj	Character string indicating search criteria for '.gz'-files.
area	Character string identifying the region that should be extracted. area must be a subregion of the original region defined by the '.gz'-file. See region_definitions for area definitions and use add.region to add new regions.

Author(s)

Robert K. Bauer

See Also

readbin, writebin, crop, raster2matrix, param_unconvert

Examples

```
## Example 1: extract, write '.gz'-files, following default plot-procedure
library(raster)

# load sample-'.gz'-file
path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/*.gz'))[1] # load sample-'.gz'-files
print(gz.files)
# area_extrac(gz.files[1],area='lion')
```

```
# gz <- Sys.glob(path,'/medw4*.gz') # load new-'.gz'-file
# v(gz) # visualize new-'.gz'-file
# system(paste('rm', gz))
# v(gz.files[1],v_area='lion')
```

`bindate2Title` *returns formatted date string for v-plot titles*

Description

returns formatted date string for v-plot titles by provided date information (e.g. filename of `'.gz'`-files, name of raster-layers. `bindate2Title` is returned by default by v-calls. `bindate2main` and `bindate2ylab` are plotted when v is called with `sidelabels=T`.

Usage

```
bindate2Title(timestep, date1, date2=date1)

bindate2main(timestep, date1, date2=date1)

bindate2ylab(timestep, date1, date2=date1)
```

Arguments

`timestep`	character string, indicating the range of the time unit in numbers and the time unit (e.g. "1d" for daily data; "7d" or "1w" for weekly data; "1m" for monthly data)
`date1, date2`	character string, indicating the first and last date of the timeframe covered (recognized format is %Y%m%d%H or %Y%m%d). E.g. 20030301 and 20030331 for monthly data (`timestep` = 1m) of March 2003.

Author(s)

Robert K. Bauer

See Also

name_split, v

Examples

```
## Example 1: output of different bindate2???-functions
path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files
u <- name_split(gz.files)
```

```
print(gz.files[1]) # print filename
print(u[1,]) # print splitted filename
bindate2main(u$timestep[1],u$date1[1],u$date2[1]) # main
bindate2Title(u$timestep[1],u$date1[1],u$date2[1]) # Title
bindate2ylab(u$timestep[1],u$date1[1],u$date2[1]) # ylab


## Example 2: Visualize output for multiple '.gz'-files
u$option <- '... .'

dev.new(width=9.7,height=7.8,xpos=-1)
empty.plot()
box()
for (i in 1:nrow(u)){
  mtext(name_join(u[i,]),side=1,line=i-10)
  main <- bindate2main(u$timestep[i],u$date1[i],u$date2[i]) # main
  Title <- bindate2Title(u$timestep[i],u$date1[i],u$date2[i]) # Title
  ylab <- bindate2ylab(u$timestep[i],u$date1[i],u$date2[i]) # ylab
  mtext(c(Title,ylab,main),side=1:3,line=c(i,nrow(u)+1-i,nrow(u)+1-i))
  mtext(paste("file",i),side=c(1,1:3),line=c(i-10,i,nrow(u)+1-i,nrow(u)+1-i),adj=0)
}
mtext(c("filename",
        "bindate2Title (default)",
         "bindate2ylab (sidelabels=T)",
         "bindate2main  (sidelabels=T)"),
      side=c(1,1:3),line=c(-11,rep(i+2,3)),font=2)
```

`check_gzfiles` *Returns summary on '.gz'-file types*

Description

Returns summary table on '.gz'-file types available in a specified folder. Provided information include `region` (region covered, as described by the region_definitions), `sat` (satellite source), `param` (parameter), `res` (spatial resolution), `ts` (temporal resolution), `filetype` (file filetype)

Usage

```
check_gzfiles(sstring="*",folder,filetype=".gz")
```

Arguments

`sstring`	Character string indicating the search criteria for sat files (default is *, including all '.gz'-files).
`folder`	Character string indicating the folder in which searched files are located (default is current working directory)
`filetype`	Character string indicating thefile type of sat files (default is .gz)

Value

An aggregated data frame, returning '.gz'-file type-information (see description) on available files in a specified folder.

Author(s)

Robert K. Bauer

See Also

name_split, check_ts

Examples

```
## Example 1: plot '.gz'-files, following default plot-procedure
path <- system.file("test_files", package="oceanmap")
check_gzfiles(folder=path) # return file summary-table per filetype

## check for missing dates
check_ts('medw4*',folder=path)
check_ts('medw4*',folder=path,output=TRUE)
```

---

check_ts *checks if daily '.gz'-file time series is complete*

Description

checks if daily '.gz'-file time series in the present working directory is complete.

Usage

```
check_ts(sstring="*.gz",folder,output=F)
```

Arguments

sstring	Character string indicating search criteria for gz-files (default is '*.gz').
folder	Character string indicating the folder in which searched files are located (default is current working directory)
output	weather the missing dates should be returned as vector (default is F).

Value

optional vector of missing dates (see output argument).

Author(s)

Robert K. Bauer

See Also

name_split, check_gzfiles

Examples

```
## Example 1: plot '.gz'-files, following default plot-procedure
path <- system.file("test_files", package="oceanmap")
check_gzfiles(folder=path) # return file summary-table per filetype

## check for missing dates
check_ts('medw4*',folder=path)
check_ts('medw4*',folder=path,output=TRUE)
```

`clim_plot` *plots '.gz'-file climatologies*

Description

Creates climatology plots of `'.gz'`-files. **ATTENTION!** This function requires an ImageMagick installation, but runs also under Windows operating systems.

Usage

```
clim_plot(obj, folder,plotfolder=".", plotname, question=T, sst.frontcolor='red',
      chla.frontcolor='blue', sidelabels = F, Ylab = F, axeslabels = T, v_area, ...)
```

Arguments

`obj`	Character string indicating search criteria for climatology `'.gz'`-files.
`folder`	directory where data files are located (optional).
`plotfolder`	directory where image should be saved.
`plotname`	the name of the output file. If not provided, value will be derived from `'.gz'`-filenames.
`question`	whether the user shall be informed about the number of figures to plot before running the procedure (default is `TRUE`).
`chla.frontcolor`	color map to be plotted for chlorophyll fronts (default is blue; obtained from cmap-dataset)
`sst.frontcolor`	color map to be plotted for sea surface temperature fronts (default is red; obtained from cmap-dataset)
`sidelabels`	whether an additional y-axis label and title should be added to the plot device (default is `FALSE`). If `TRUE`, y-axis label is defined by `Ylab`, the additional title is derived from the `date`-information and gives the month information.
`Ylab`	an additional title for the y axis (default is date information), only used when `sidelabels` is set `TRUE`. Default value is year-information.

axeslabels	whether axeslabels should be shown (default is TRUE, set as 'longitude' and 'latitude')
v_area	character string identifying the region that should be plotted, or in case of obj == 'bathy', also a Raster* or Extent object. If missing, region is derived from the '.gz'-filename. See region_definitions for area definitions and use add.region to add new regions.
...	Additional arguments to be passed to v and plotmap (e.g. main, sidelabels, Ylab, scale_arrow, minv, maxv, adaptive.vals, cb.xlab, suffix, v_area, v_image, v_contour, v_arrows, fill, col, border, grid, grid.res, bwd, axeslabels, ticklabels, cex.lab, cex.ticks)

Author(s)

Robert K. Bauer

See Also

v, readbin, name_split, regions, plotmap

Examples

```
## Example 1: plot seasonal '.gz'-files, following default plot-procedure
path <- system.file("test_files", package="oceanmap")
gz.file <- Sys.glob(paste0(path,'/*.gz'))[1] # load sample-'.gz'-files
check_gzfiles(folder=path) # return file summary-table
gz.files <- Sys.glob(paste0(path,'/*1s_*.gz')) # load seasonal '.gz'-files
# v(gz.files) # as single plots

## as combined climatology plot, saved in plotfolder
# clim_plot(gz.files,plotname='chla.summary.png')
```

---

close_fig *function to close current graphic device*

Description

function to close current graphic device, complement to figure-function that generates graphic devices in flexible fileformats.

Usage

```
close_fig(do.close=F, do.save=do.close)
```

Arguments

do.close, do.save	whether file should be saved or not (default is TRUE). if FALSE, new graphic device will be opened inside R.

Author(s)

Robert K. Bauer

See Also

figure

Examples

```
# do.save <- TRUE
# figure("Gulf_of_Lions", do.save=do.save, width=5, height=5, type="pdf")
# plotmap("lion")
# close_fig(do.save)

do.save <- TRUE
plotmap("lion")
close_fig(do.save)

do.save <- FALSE
figure("Gulf_of_Lions", do.save=do.save, width=5, height=5, type="pdf")
plotmap("lion")
close_fig(do.save)
```

---

cmap *color maps*

Description

list holding different color maps that can be used in image plots (see: v, get.bathy, image, image.plots, clim_plot)

available color maps are: `ano`, `bathy`, `blue`, `chla`, `haxby`, `jet` (obtained from matlab), `rainbow`, `red`, `orange`, `green`, `sst` and `haxbyrev`.

Usage

```
data(cmap)
data(cmap_topo)
```

Format

list

Author(s)

Robert K. Bauer

Examples

```
data('cmap') # load color maps data
names(cmap) # list available color maps

path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files
# figure(width=15,height=15)
# par(mfrow=c(4,5))
# for(n in names(cmap)) v(gz.files[2], v_area='lion', subplot=TRUE,
#                         pal=n, adaptive.vals=TRUE, main=n)

## simple example of the \link{image}-function
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100))
image(x, y, volcano, col = cmap$jet) # jet color map
image(x, y, volcano, col = cmap$haxby) # haxby color map
image(x, y, volcano, col = cmap$chla) # chlorophyll color map
image(x, y, volcano, col = cmap$sst) # sst color map

data(cmap_topo)
image(x, y, volcano, col = cmap_topo$col) # topography color map

## another example: plot bathymetry and topography of the western Mediterranean Sea
#get.bathy("medw4",visualize=T,terrain=T,res=3)
#get.bathy("medw4",visualize=T,terrain=F,res=3,levels=c(200,2000)) # show contours
```

`delete.region` *deletes a region from the region_definitions-definition file*

Description

deletes a specified region from the region_definitions-definition file

Usage

```
delete.region(region,lib.folder,restore=F)
```

Arguments

`region`	Character string identifying the region that should be deleted. See region_definitions for area definitions and use add.region to add new regions.
`lib.folder`	Character string indicating R-library path in which the oceanmap-package is installed.
`restore`	whether the original region_definitions-file should be restored.

Author(s)

Robert K. Bauer

See Also

add.region, region_definitions, regions, writebin

Examples

```
## Example 1: Add region by supplying a one-row data.frame
##            that holds the entire required information
data(region_definitions)
lion <- region_definitions[region_definitions$label == 'lion',] # selecting Gulf of Lions region
lion
junk <- lion
junk$label <- 'junk' # rename region label
#add.region(junk) # add junk region
data(region_definitions) # reload region_definitions
region_definitions[,1:9]

## Example 2: Delete region
#delete.region("junk") # delete junk region
data(region_definitions) # reload region_definitions
region_definitions[,1:9]
```

---

empty.plot *Creates an empty scatter plot*

Description

Creates an empty scatter plot that is equal to the function call:

`plot(1,lwd=0,axes=F,xlab="",ylab="",...)`

Usage

```
empty.plot(..., xlab = "", ylab = "", new=T, add=!new, n=1, axes = F)
```

Arguments

`...`	other arguments of the generic x-y plotting fucntion plot.
`xlab, ylab`	label for the x- and y-axis of the plot (default is empty).
`new, add`	whether to show add plot to a current plot device or to start a new figure (default is: new=TRUE and add=FALSE).
`n`	number of figures to be plotted (default is 1)
`axes`	whether to show plot axes (default is FALSE).

Author(s)

Robert K. Bauer

Examples

```
empty.plot()
title("empty plot")
box()
axis(1)
axis(2)
```

---

figure *generate (and save) graphic devices with flexible fileformat selection*

Description

figure generates graphic devices with flexible fileformat selection. **Function call with (figure(do.save=T) needs to be finished by close_fig(do.save=T), to close open file connection.**

Usage

```
figure(filename, folder, type, save=F, do.save=save,
     width=10, height=10, xpos=-1, do.overwrite=T, delete.old=do.overwrite, ...)
```

Arguments

filename	name of the figure to be generated (without file extension)
folder	plot folder (by default current working directory)
type	character string indicating the graphics format of the figure file. can be: • "jpg" • "jpeg" • "png" • "tiff" • "eps" • "pdf"
width, height	width and height of figure to be generated. default units are inches.
save, do.save	whether file should be saved or not (default is TRUE). if FALSE, new graphic device will be opened inside R.
xpos	horizontal screen position of graphic device (ignored if do.save == TRUE)
do.overwrite, delete.old	overwrite existing figure with same filename and extension (default is FALSE)
...	additional arguments to be passed to the graphic device

Author(s)

Robert K. Bauer

See Also

close_fig

Examples

```
## Example 1: plotmap() and figure()
do.save <- FALSE
figure("Gulf_of_Lions_extended", do.save=do.save, width=5, height=5, type="pdf")
plotmap("lion")
close_fig(do.save)

## now resize figure manually and get new figure dimensions:
width <- dev.size()[1]
height <- dev.size()[2]

# do.save <- TRUE
# figure("Gulf_of_Lions_extended", do.save=do.save, width=width, height=height, type="pdf")
# plotmap("lion")
# close_fig(do.save)
```

---

get.avg.bathy *returns the average value of circles with specified coordinates and a defined radius*

Description

returns the average value of circles with specified coordinates and a defined radius

Usage

```
get.avg.bathy(x, radius, unit="km", raster, bathy, v_area="medw4")
get.avg(x, radius, unit="km", raster)
```

Arguments

x	a vector or matrix providing the coordinates of the circle
radius	the radius (if unit != "km", the radius is assumed to be of the same scale as the plotting window)

unit	unit of the radius (by default "km"). if unit != "km", the radius is assumed to be of the same scale as the plotting window.
raster	raster object, from which the average should be calculated.
bathy	raster object with bathymetry data, from which the average should be calculated.
v_area	character string identifying the region for which the bathymetry data should downloaded from the NOAA server.

Author(s)

Robert K. Bauer

See Also

SpatialCircle

Examples

```
## Example 1: load & plot bathymetry of the Baltic Sea, defined by longitudes and latidtues
lon <- c(9, 31)
lat <- c(53.5, 66)
# bathy <- get.bathy(lon=lon, lat=lat, main="Baltic Sea", cbpos='r')
plotmap(lon=lon, lat=lat)
spc <- SpatialCircle(x= 20,y = 57.5,r=1)
plot(spc,add=TRUE)
# get.avg.bathy(c(20,57.5), radius = 1, bathy = bathy)
# get.avg(c(20,57.5), radius = 1, unit="km",raster = bathy)
```

---

get.bathy *Returns bathymetric data from the NOAA ETOPO1 database as RasterLayer, given coordinate bounds and resolution.*

Description

Returns and optionally stores bathymetric data from the ETOPO1 database hosted on the NOAA server as a RasterLayer, based on the defined resolution and provided coordinate bounds or region definition. Stored bathymetry files can be reloaded through the same function call.

Usage

```
get.bathy(v_area, lon, lat, resolution=4, keep=F ,
          savename.bathy, folder.bathy, visualize=T, terrain=F,...)
```

Arguments

v_area	character string identifying the region that should be plotted, or in case of x == 'bathy', also a Raster* or Extent object. If missing, region is derived from the '.gz'-filename. See region_definitions for area definitions and use add.region to add new regions.
lon,lat	longitude and latitude describing the extend of the region of interest.
resolution	resolution of the bathymetric grid, in minutes (default is 4).
keep	whether to write the data downloaded from NOAA into a file (default is FALSE).
savename.bathy	savename for the bathymetric data file, if not specified set to type 'bathy_lon-lat_res.resolution.dat' or 'bathy_v_area_res.resolution.dat'.
folder.bathy	directory where bathymetric data should be saved (default is current working directory).
visualize	whether the bathymetric data should be plotted instantly.
terrain	whether the to keep terrain data (default is FALSE). If set FALSE and visualize is TRUE, grid command in plotmap is disabled!
...	additional arguments to be passed to v, used if visualize is set TRUE.

Author(s)

Robert K. Bauer

See Also

v, add.region, region_definitions, regions, writebin, get.bathy

Examples

```
## Example 1: load & plot bathymetry of the Baltic Sea, defined by longitudes and latidtues
lon <- c(9, 31)
lat <- c(53.5, 66)
# get.bathy(lon=lon, lat=lat, main="Baltic Sea", cbpos='r')

## Example 2: plot bathymetry using a v_area-keyword
#get.bathy("lion",res=4, keep=T) # can take some time, requires server connection!
#get.bathy("lion",res=1, keep=T,visualize=FALSE)

## Example 3: plot landmask of the Baltic Sea defined by an extent- or raster-object
library('raster')
ext <- extent(lon,lat)
#get.bathy(ext,visualize=T,main="Baltic Sea",res=4,levels=200) # extent-object

## Example 4: plot bathymetry and topography of the western Mediterranean Sea
### a) download, assign and save bathymetry
# bathy <- get.bathy("medw4",visualize=F,terrain=T,res=3,keep=T)
# # load('bathy_medw4_res.3.dat',verbose = T); bathy <- h
# par(mfrow=c(2,1))
# v(bathy,param="bathy",subplot = T)
```

```
# get.bathy("medw4",visualize=T,terrain=F,res=3,levels=c(200,2000),
# subplot = T,grid=F) # show contours

### b) only contour lines:
# par(mfrow=c(1,2))
# h <- get.bathy("lion",visualize=T,terrain=F,res=3,levels=c(200,2000),
#                v_image=F, subplot=T,grid=F)

### use v-function for same plot but on subregion:
# v(h,v_area = "survey", param="bathy",subplot = T, v_contour = T,
#   v_image = F, levels=c(200,2000))
```

`ggplotmaply` *Converts a ggplot2 object from ggplotmap() to plotly*

Description

This function converts a ggplot2 object created by oceanmap::ggplotmap() to a plotly object.

Usage

```
ggplotmaply(ggobj, fixedrange=F, grid=F,expand=3)
```

Arguments

`ggobj`	Character string identifying regions predefined by the region_definitions-dataset, Raster* or Extent object (corresponds to `v_area` of the v-function). If missing, region is derived from geographical coordinates, denoted by lat and lon. See add.region to define new region definitions and delete.region to delete unproper region definitions.
`fixedrange`	Vector returning longitude coordinates of the area to be plotted.
`grid`	whether a grid should be plotted (default is `TRUE`)
`expand`	By default, the underlying ggplotly-function does not stick to the plotting region of the ggobj, but extends it. This can result in missing countries or islands. The `expand`-argument extends the plotly-plotting window in each direction in order to cover the corresponding landmasks.)

Details

`ggoplotmaply` uses the `ggplotly` functions to convert the ggplot object into the plotly format.

Author(s)

Robert K. Bauer

See Also

ggplotmap, ggplotly

Examples

```
library(ggplot2)

#### Example 1: plot landmask of the Western Mediterranean Sea
## a) by using longitude and latitude coordinates:
# lon <- c(-6, 16.5)
# lat <- c(34, 44.5)
# ggobj <- ggplotmap(xlim=lon, ylim=lat)
# ggobj
# ggplotmaply(ggobj,expand = 10) ## we need to expand the plotting region


## b) plot landmask of the Western Mediterranean Sea by using an extent-object:
# library('raster')
# ext <- extent(lon, lat)
# plotmap(ext, main="Western Mediterranean Sea") # extent-object
# ggobj <- ggplotmap(ext)
# ggobj
# ggplotmaply(ggobj)


## c) plot landmask of the Western Mediterranean Sea by using a raster-object:
# r <- raster(ext)
# ggobj <- ggplotmap(r)
# ggobj
# ggplotmaply(ggobj)
# ggplotmaply(ggobj)

## d) plot landmask of the entire Mediterranean Sea by using keyword:
 ggobj <- ggplotmap("med4") +
          geom_point(data=data.frame(x=3.7008, y=43.4079),aes(x,y),size=5,colour="blue")
# ggobj
# ggplotmaply(ggobj,expand = 10)


## e) add landmask to raster image plot (similar to v()-call)
# library(dplyr)
# library(ggplot2)
# data(cmap)
# setwd(system.file("test_files", package="oceanmap"))
# nc <- nc2raster(ncfiles[1])
# rs2df <- nc[[1]] %>% ## take first layer
#          rasterToPoints() %>% ## convert raster to xyz matrix
#          as.data.frame() ## convert to data frame
# names(rs2df) <- c("Lon","Lat","Conc") ## reset names (important for ggplotmaply hover text)
# ggobj <- ggplot() + geom_raster(data = rs2df, aes(x=Lon,y=Lat,fill=Conc))
# ggobj_with_land_mask <- ggplotmap(add_to = ggobj) +
#                         scale_fill_gradientn(colours=cmap$jet) # change colorbar
```

```
# ggobj_with_land_mask
# ggplotmaply(ggobj_with_land_mask)
```

`internal.datasets` *internal datasets*

Description

internal (`lazyload`) datasets `medm9_proj` and `regions.dim.bathy`, accessed by `v.plot` and readbin respectively.

Author(s)

Robert K. Bauer

`matrix2raster` *Converts a matrix to a RasterLayer or arrays to a RasterStack-object*

Description

`matrix2raster` Converts a matrix to a RasterLayer or arrays to a RasterStack-object.

Usage

```
matrix2raster(z,x,y,layer,proj="+proj=longlat")
```

Arguments

`z`	matrix or array to be converted.
`x`	optional x-coordinates giving the horizontal range of the raster layer, its size does not need to coincide with ncol(z)!
`y`	optional y-coordinates giving the verical range of the raster layer, its size does not need to coincide with nrow(z)!
`layer`	layer to be selected (only valid if z is an array).
`proj`	optional argument, setting the coordinate reference system (CRS) of a Raster* object (default is `+proj=longlat`).

Author(s)

Robert K. Bauer

Examples

```
## Example 1: convert a matrix
m <- matrix(3,2,2)
matrix2raster(m)

## Example 2: convert an array
a <- array(3,dim=c(2,2,2))
matrix2raster(a)
matrix2raster(a,layer=1)

## Example 3: convert '.nc'-file to raster-object manually
owd <- getwd()
path <- system.file("test_files", package="oceanmap")
ncfile <- Sys.glob(paste0(path,'/herring*.nc')) # load sample-'.nc'-files

library('ncdf4')
library('raster')
nc <- nc_open(ncfile) # open netcdf file
z <- ncvar_get(nc,'Conc')[,,1]
lon <- as.vector(ncvar_get(nc,'lon')) # fillvalues are automatically replaced by NA
lat <- as.vector(ncvar_get(nc,'lat')) # fillvalues are automatically replaced by NA
matrix2raster(z,x=lon,y=lat)

## Example 4: convert '.nc'-file to raster-object using nc2raster
nc2raster(ncfile,varname='Conc',layer=1:4)
```

`name_join` *create '' .gz'-filenames from a list or dataframe*

Description

creates filenames based on a list or dataframe with the (header)-names:

`area` `source` `parameter` `resolution` `timestep` `date1` `date2` `option`

by aligning the defined filetype:

e.g. `area_source_parameter_resolution_timestep_date1_date2.option.filetype`

Usage

```
name_join(parts,filetype='gz')
```

Arguments

`parts`	a list or dataframe with the parts:

- `area` , the region keyword
- `source` , the data source

- `param` , the parameter saved in the `'.gz'`-file. Can only be one value!
- `resolution` , the spatial resolution
- `timestep` , the temporal resolution
- `date1` & `date2` , the temporal resolution (the time interval covered).
- `option` a character string holding supplmentary information of `'.gz'`-file treatment

`filetype` character string inidicating the filtype to be checked. ('.gz' by default)

Author(s)

Herve Demarq, translated from IDL by Robert K. Bauer

See Also

See check_gzfiles to return summary of available `'.gz'`-files and name_split to split `'.gz'`-filenames

Examples

```
## Example: read and plot '.gz'-file
path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files
check_gzfiles(folder=path) # return file summary-table

# return summary of availble '.gz'-files
# suffix-column corresponds to option column of the name_join-call
# addition n-column returns the number of available files per filetype
check_gzfiles(gz.files)

## Example: split and rejoin '.gz'-filenames
name_split(gz.files) # return summary-table per file
name_join(name_split(gz.files))
```

---

name_split *Returns a summary data frame of '.gz' encoded oceanography files by splitting their name*

Description

Returns a summary data.frame of `'.gz'` encoded oceanography files by splitting their name

Usage

```
name_split(gz.files)
get.gz.info(gz.files)
```

Arguments

`gz.files` Optional character vector or search criteria for .gz-encoded oceanography files.

Value

Returns a summary data.frame of `'.gz'` encoded oceanography files by splitting their name

`area source parameter resolution timestep date1 date2 option`

`area`	region keyword
`source`	data source
`param`	the parameter saved in the `'.gz'`-file. Can only be one value!
`resolution`	the spatial resolution
`timestep`	the temporal resolution
`date1 & date2`	the time interval covered in date format
`option`	a character string holding supplmentary information of `'.gz'`-file treatment

Author(s)

Robert K. Bauer

See Also

See check_gzfiles to return summary of available `'.gz'`-files and name_join to create `'.gz'`-filenames from splitted names (name_split)-calls

Examples

```
## Example: read and plot '.gz'-file
path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files
check_gzfiles(folder=path) # return file summary-table

# return summary of availble '.gz'-files
# suffix-column corresponds to option column of the name_split-call
# addition n-column returns the number of available files per filetype
check_gzfiles(gz.files)

## Example: split and rejoin '.gz'-filenames
gz.files
name_split(gz.files) # return summary-table per file
name_split() # return summary-table of all gz-file in current folder
name_join(name_split(gz.files))
```

nc2raster *Convert Raster layer to a matrix or array*

Description

nc2raster converts a netcdf-file ('.nc'-file) or ncdf4-object to a Raster* object, setting the time variable as layer name.

Usage

```
nc2raster(nc, varname, t=layer, layer, verbose=FALSE)
```

Arguments

nc	character string indicating the filepath to a netcdf-file ('.nc'-file), or a ncdf4-object.
varname	character string indicating the name of the netcdf-variable to be selected.
layer, t	layer/time stemp to select in multi-layer files.
verbose	should information about the netcdf file, including the variables and dimensions it contains, be printed during loading? (default is FALSE)

Value

RasterLayer or RasterStack

Author(s)

Robert K. Bauer

Examples

```
path <- system.file("test_files", package="oceanmap")
nfiles <- Sys.glob(paste0(path,'/*.nc'))[1] # load sample-'.nc'-files

nc2raster(nfiles[1],"Conc",layer=1) # RasterLayer
nc2raster(nfiles[1],"Conc",layer=1:4) # RasterStack

library('ncdf4')
nc <- nc_open(nfiles[1])
nc2raster(nc,"Conc",layer=1:4) # RasterStack

###### load & plot sample netcdf-file ('.nc'-file)

### option a) load netcdf-file with ncdf4-package and plot it
library('ncdf4')
ncdf <- nc_open(nfiles[1])
print(ncdf)
v(obj = ncdf, cbpos="r")
```

```
### option b) load and plot netcdf-file as RasterStack object
nc <- nc2raster(nfiles[1])
v(nc,cbpos="r") # plot RasterStack object
v(nfiles[1], cbpos="r",replace.na=TRUE) # plot directly netcdf-file

### option c) plot netcdf-file directly
v(nfiles[1], cbpos="r") # plot RasterStack object

###### plot multiple layers:
par(mfrow=c(2,2))
v(nfiles[1], t=1:4, cbpos="r", replace.na=TRUE, subplot = TRUE)
```

`nc2time` *reads and converts the time variable of a netcdf-file ('.nc'-file) or* `ncdf4`*-object as* `as.Date`*-object*

Description

reads and converts the time variable of a netcdf-file ('.nc'-file) or `ncdf4`-object as `as.Date`-object.

Usage

```
nc2time(nc,varname)
```

Arguments

`nc`	character string indicating the filepath to a netcdf-file ('.nc'-file), or a `ncdf4`-object.
`varname`	character string indicating the name of the time vaiable of the netcdf-file.

Author(s)

Robert K. Bauer

Examples

```
path <- system.file("test_files", package="oceanmap")
nfile <- Sys.glob(paste0(path,'/herring*.nc')) # load sample-'.nc'-files
head(nc2time(nfile))

library('ncdf4')
nc <- nc_open(nfile)
head(nc2time(nc))
```

oceanmap *oceanmap - plot tools for 2D oceanographic data*

Description

oceanmap is a plotting toolbox for oceanographic data. Visualizing data is a crucial step in analyzing and exploring data. During the last two decades the statistical programming language R has become a major tool for data analyses and visualization across different fields of science. However, creating figures ready for scientific publication can be a tricky and time consuming task.

The oceanmap package provides some helpful functions to facilitate and optimize the visualization of geographic and oceanographic data, such as satellite and bathymetric data sets. Its plotting functions are written in a way that they do not require a large amount of their numerous arguments to be specified but still return nice plots. Its major functions are:

Major functions:

- plotmap: plots landmask as basis or overlay
- v: plots oceanographic data (fronts, SST, chla, bathymetry, etc.) from raster-objects, ncdf4- or gz-files
- set.colorbar: adds a colorbar to current figure, allowing several placement methods
- get.bathy: download bahymetric data at user defined resolution from the NOAA ETOPO1 database
- add.region: generate region definitions to facilitate land mask and colorbar plotting using plotmap and v
- figure & close_fig: generate and save graphic devices in flexible file formats (jpeg, png, eps, pdf and eps)

Getting Started
Check out some examples of the principle functions, listed above.

Author(s)

Robert K. Bauer

parameter_definitions *parameter definitions dataframe*

Description

a dataframe containing definitions of parameters to plot or to save by v, readbin and writebin.

Usage

```
data(parameter_definitions)
```

Format

data.frame

Value

a dataframe with the following header, containing definitions of parameters to plot or to save by v, readbin and writebin:

`param a b c log name1 unit pal1 minv maxv min max invalid_data_dc coast_dc land_dc no_data_dc`

`param`	character string indicating the keyword of a parameter.
`a,b,c`	value for parameter parameter data conversion from/to byte data. (See param_convert and param_unconvert)
`log`	whether a logarithmic formula should be applied for data conversion (0 for `FALSE` and 1 for `TRUE`; See param_convert and param_unconvert).
`name`	character string indicating the long name of a parameter.
`unit`	character string or bgroup statement indicating the parameter unit.
`pal1`	default color map used by `v` calls on parameter related data.
`minv, maxv`	default minimum and maximum z-value used by `v` calls on parameter related data.
`min, max`	minimum and maximum byte-values to be considered when calculating absolute values.
`invalid_data_dc, coast_dc, land_dc & no_data_dc`	byte values used to mask invalid data, coast lines, land masses and missing data.

Author(s)

Robert K. Bauer

See Also

v

Examples

```
## Example
data(parameter_definitions)
head(parameter_definitions)

# selecting sea surface temperature parameter definition
parameter_definitions[parameter_definitions$param == "sst2",]
```

param_convert *converts byte data to absolte values or vise versa (*param_unconvert*)*

Description

converts byte data as stored in '.gz'-files to absolte values (param_convert) or vise versa (param_unconvert) using the parameter_definitions-dataset. param_convert is used by readbin, param_unconvert is used by writebin.

Usage

```
param_convert(x,param)

param_unconvert(x,param)
```

Arguments

x	vector, matrix or raster-object holding byte-data that that should be converted to absolute values (param_convert) or vise versa (param_unconvert).
param	Character string indicating parameter of the dataset to be treated. See parameter_definitions for available parameters.

Author(s)

Robert K. Bauer

See Also

param_unconvert, readbin

Examples

```
library('fields')
path <- system.file("test_files", package="oceanmap")
gz.file <- Sys.glob(paste0(path,'/*.gz'))[1] # load sample-'.gz'-files
param <- name_split(gz.file)$parameter
print(param)

## converted data, according to param information
m <- readbin(gz.file, Raster=FALSE)
image.plot(m)

## byte data ("unconverted") according to param information, as stored in ".gz"files
bin <- param_unconvert(m,param)
image.plot(bin)

## reconverting byte data, according to param information
conv <- param_convert(bin,param)
image.plot(conv)
```

plotmap *plots landmask of a defined region*

Description

plots the landmask of a region defined by a region-key word, georgraphical coordinates (longitude and latitude), a raster- or extent-object. See add.region to add and save new region definitions. Attention! Unlike add.region, plotmap does not include colorbar placement (see: set.colorbar)

Usage

```
plotmap(region=v_area, lon, lat, add=F, asp,
        grid=T, grid.res, resolution=0,
       main, axes=T, axeslabels=axes, ticklabels=T, cex.lab=0.8, cex.ticks=0.8,
        fill.land=T, col.land="grey", col.bg=NA, border='black', bwd=2, las=1,
        v_area, xlim, ylim
        )

ggplotmap(region=v_area, lon=xlim, lat=ylim, add_to, asp,
          grid=T, grid.res, resolution=0,
          main, axes=T, axeslabels=axes, ticklabels=T,
          fill.land=T, col.land="grey", col.bg=NA, border='black',
          col.scale = "black", bwd=1.5, v_area, xlim, ylim)
```

Arguments

region, v_area	Character string identifying regions predefined by the region_definitions-dataset, Raster* or Extent object (corresponds to v_area of the v-function). If missing, region is derived from geographical coordinates, denoted by lat and lon. See add.region to define new region definitions and delete.region to delete unproper region definitions.
lon, xlim	Vector returning longitude coordinates of the area to be plotted.
lat, ylim	Vector returning latitude coordinates of the area to be plotted.
add, add_to	whether the a the landmask should be added to an existent figure (default is FALSE) or an existing ggplot object, in case of ggplotmap.
asp	numeric, giving the aspect y/x-ratio of the y- and x-axes. See plot.window for more details.
main	title to be plotted
axes, axeslabels	whether axes and axes-labels (longitude and latitude) should be plotted (default is TRUE). axes-labels can be a single value or a vector of size two, representing values for x and y axis, respectively.
ticklabels	whether tick-labels should be added to the axes (default is TRUE). Can be a single value or a vector.

cex.lab	font size of axis labels
cex.ticks	font size of tick labels
grid	whether a grid should be plotted (default is TRUE)
grid.res	resolution of the grid, in degrees (default is is derived from the region extent)
resolution	number that specifies the resolution with which to draw the map. Resolution 0 is the full resolution of the database [default]. Otherwise, just before polylines are plotted they are thinned: roughly speaking, successive points on the polyline that are within resolution device pixels of one another are collapsed to a single point (see the Reference for further details). Thinning is not performed if plot = FALSE or when polygons are drawn (fill = TRUE or database is a list of polygons).
bwd	width is of the axes bars (default is 1)
fill.land	whether the a the landmask should be filled by a color (default is TRUE)
col.land	fill color of the landmask to be plotted (default is grey)
col.bg	background color (ocean) to be plotted (default is NA)
border	country border color of the landmask to be plotted (default is black)
col.scale	color of the map scale to be plotted around the map (default is black)
las	numeric in 0,1,2,3; the style of axis labels. 0: always parallel to the axis, 1: always horizontal [default], 2: always perpendicular to the axis, 3: always vertical

Details

plotmap uses the maps and maptools functions to plot the landmask.

Author(s)

Robert K. Bauer

See Also

v, regions

Examples

```
#### Example 1: plot landmask of the Mediterranean Sea
## a) by using longitude and latitude coordinates:
lon <- c(-6, 37)
lat <- c(30, 46)
figure(width=9.75,height=5.28)
plotmap(lon=lon, lat=lat, main="Mediterranean Sea")
plotmap(xlim=lon, ylim=lat, main="Mediterranean Sea")
ggobj <- ggplotmap(xlim=lon, ylim=lat)
ggobj
```

```
# ggplotmaply(ggobj)


## b) plot landmask of the Mediterranean Sea by using an extent-object:
# library('raster')
# ext <- extent(lon, lat)
# plotmap(ext, main="Mediterranean Sea") # extent-object
# ggplotmap(ext)

## c) plot landmask of the Mediterranean Sea by using a raster-object:
# r <- raster(ext)
# plotmap(r, main="Mediterranean Sea") # raster-object
# ggplotmap(r)

## d) plot landmask of the Mediterranean Sea by using a region label:
# plotmap('med4', main="Mediterranean Sea") # region-label
# regions() ## check preinstalled region label

## e) add landmask to an existing plot:
# plot(3.7008, 43.4079, xlim=lon, ylim=lat)
# plotmap(add=T)
# points(3.7008, 43.4079, pch=19)
# ggplotmap(xlim=lon, ylim=lat)

# library(ggplot2)
# ggobj <- ggplotmap("lion") +
#         geom_point(data=data.frame(x=3.7008, y=43.4079),aes(x,y),size=5,colour="blue")
# ggobj
## ggplotmaply(ggobj)
## f <- ggplotmaply(ggobj)
## pos <- as.data.frame(list(x=c(5.83, 4.91, 5.67, 5.91, 6.31, 6.37,
##                               5.66, 5.54, 5.51, 5.67, 5.89, 5.97),
##                           y=c(42.89, 42.27, 42.42, 42.33, 42.1, 41.92,
##                               41.74, 41.45, 41.32, 41.21, 41.04, 40.96)
##                          ))
## f %>% add_trace(data = pos,x = ~x, y= ~y,type='scatter',mode='marker',name="new pos")



#### Example 2: subplots and some additional arguments of plotmap()
# par(mfrow=c(2, 1))
# plotmap('medw4', main="Western Mediterranean Sea",col.bg="darkblue")
# plotmap('medw4', main="Western Mediterranean Sea", bwd=3, border='grey', grid=FALSE)



#### Example 3: plotmap() and figure()
# do.save <- FALSE ## open a plotting window
# figure("Gulf_of_Lions_extended", do.save=do.save, width=5, height=5, type="pdf")
# plotmap("lion",col.bg='darkblue',grid=FALSE)
# close_fig(do.save)

## now resize figure manually and get new figure dimensions:
```

```
# width <- dev.size()[1]
# height <- dev.size()[2]

# do.save <- TRUE ## do NOT open a plotting window, but save figure internally
# figure("Gulf_of_Lions_extended", do.save=do.save, width=width, height=height, type="pdf")
# plotmap("lion",col.bg='darkblue',grid=FALSE)
# close_fig(do.save)


#### Example 4: between hemispheres
# par(mfrow=c(2,1))
# plotmap(lon=c(-180, 180), lat=c(-80, 80), main="map from West to East")
# plotmap(lon=c(0, 360), lat=c(-80, 80), main="map from West to East")
# plotmap(lon=c(-360, 00), lat=c(-80, 80), main="map from West to East") # same as before

#### Example 5: plot bathymetry and topography of the western Mediterranean Sea
#get.bathy("medw4",visualize=T,terrain=T,res=3)
#get.bathy("medw4",visualize=T,terrain=F,res=3,levels=c(200,2000)) # show contours
#get.bathy("lion",visualize=T,terrain=F,res=3,levels=c(200,2000),v_image=F) # show only contours

#### Example 6: testing some additional arguments
# lon <- c(-180,200); lat <- c(-80,90);
# ext <- extent(lon, lat)
# plotmap(ext, border=NA, bwd=NA, grid=FALSE, col.land = "#9ac0cd",axes=FALSE)
```

`raster2matrix` *Convert Raster layer to a matrix or array*

Description

`raster2matrix` converts a raster layer to a matrix or array. Used by readbin and writebin.

Usage

```
raster2matrix(RasterLayer)

raster2array(RasterLayer)
```

Arguments

`RasterLayer`	raster layer to be converted.

Author(s)

Robert K. Bauer

Examples

```
library('raster')
path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files
check_gzfiles(folder=path) # return file summary-table

raster.file <- readbin(gz.files[1]) # loading gz-file as raster-layer
image(raster.file)

## Example 1: converting single raster layer to matrix
image(as.matrix(raster.file)) # unflipped conversion
m <- raster2matrix(raster.file) # converting raster-layer to matrix
image(m)

## Example 2: converting double raster layer to an array
stack.file <- stack(raster.file,raster.file)
image(as.array(stack.file)[,,1]) # unflipped conversion
a <- raster2array(stack.file) # converting raster-layer to array (works also with raster2matrix)
image(a[,,1])
```

`readbin` *Returns '.gz'-file as matrix or raster-object*

Description

Returns `'.gz'`-file as matrix or raster-object.

Usage

```
readbin(filename, area, Image = F, byte = F, Raster = T)
```

Arguments

`filename`	Character string indicating search criteria for the `'.gz'`-file of interest. Only `'.gz'`-files with valid filenames can be read, consisting of: `area`, `source`, `parameter`, `resolution`, `timestep`, `date1`, `date2` and `option`-criteria, separated by an underscore with only `option` being aligned by a point and ending with `'.gz'`, e.g.: `area_source_parameter_resolution_timestep_date1_date2.option.gz`. See region_definitions for valid area- and parameter_definitions for valid `parameter`-values, respecively.
`Image`	whether the a the `'.gz'`-file should be plotted immediately using image.plot-function of the fields-package (default is `FALSE`)
`byte`	whether the a the data of the `'.gz'`-file should be returned unconverted as a byte-values (default is `FALSE`)
`Raster`	whether the a the data of the `'.gz'`-file should be returned in a raster-object (default is `TRUE`)

area — Character string identifying the region that should be extracted. If missing, region is derived from the '.gz'-filename. See region_definitions for area definitions and use add.region to add new regions.

Author(s)

Robert K. Bauer

See Also

writebin, regions, crop, raster2matrix, param_convert

Examples

```
### Example: read and plot '.gz'-file
path <- system.file("test_files", package="oceanmap")
check_gzfiles(folder=path) # return file summary-table
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files

### all manual:
obj <- readbin(gz.files[2],area='lion')
obj
ticks <- seq(20,30,5)
data('cmap')
image(obj,zlim=range(ticks),col=cmap$jet)
plotmap('lion',add=TRUE) # add landmask
#set.colorbar(ticks=ticks,cb.title='cb.title',cb.xlab='cb.xlab')

### using v:

## ticks set by adaptive.vals
v(obj,varname="sst2",cb.title='cb.title',cb.xlab='cb.xlab')

## ticks set by parameter definition
v(obj,varname="sst2",cb.title='cb.title',cb.xlab='cb.xlab',adaptive.vals=FALSE)

### extracting subregion:
obj <- readbin(gz.files[2])
area.extent <- extent(c(5,10,35,40))
subarea <- crop(obj,area.extent)
# v(subarea)

## getting average value:
mean(subarea[,],na.rm=TRUE)
```

---

regions — *Returns two-row summary table of a specified region.*

Description

Reorganizes summary information of a specified region from the region_definitionsset into a two-row dataframe. Region definitions can be added, backed up or restored by add.region or deleted by calling delete.region.

ATTENTION! When reinstalling or updating the oceanmap package, previous region definitions are getting lost! It is therefore highly recommanded to take and restore own backups (see: `backup` and `restore`).

Usage

```
regions(label)
```

Arguments

`label`	Character string indicating the name of the region of interest. If missing, list of available regions in the region_definitions-dataset will be returned by a error message.

Value

a two-row dataframe with the following header, containing the summary information of the region specified:

`xlim` `ylim` `dim` `name` `cbx` `cby` `align` `gradient` `figdim` `grid.res`

`xlim` & `ylim`	the spatial extent of the region
`dim`	the number of grid points for both x & y-dimension
`name`	the long name of the region
`cbx` & `cby`	x & y-coordinates for colorbar
`align`	a vector defining the color-`gradient` of the colorbar (`x` for horizontal, and `y` for vertical), as well as the `margin` where the colorbar ticks should be plotted, relative to the colorbar rectangle (`'l'` left, `'r'` right and `'b'` for bottom)
`figdim`	the region-specific default plot device size
`grid.res`	the default grid resolution in degrees

Author(s)

Robert K. Bauer

See Also

`v`, `plotmap`

Examples

```
## Example: return summary table for the Gulf of Lions
data('region_definitions')
region_definitions[region_definitions$label=='lion',] # select raw region data summary
regions('lion') # return formatted summary table
```

region_definitions *region definitions dataframe*

Description

dataset providing spatial extent and color bar placement information by a region-keyword in later related function-calls (see: v, plotmap and regions). Information consists of a region-keyword, -longname, its spatial extent (longitudes and latitudes), grid resolution, as well as default colorbar position and figure size. Region definitions can be added, backed up or restored by add.region or deleted by calling delete.region.

ATTENTION! When reinstalling or updating the oceanmap package, previous region definitions are getting lost! It is therefore highly recommanded to take and restore own backups (see: `backup` and `restore`).

Usage

```
data(region_definitions)
```

Format

data.frame

Value

dataframe with the following header, containing the summary information of the region specified:

`label` `name` `latn` `lats` `lonw` `lone` `ncol` `nrow` `px` `cbx1` `cbx2` `cby1` `cby2` `gradient` `oticks` `figxdim` `figydim` `grid.res`

`label`	region-keywords
`name`	the long name of the region
`latn` & `lats`	northern and southern most latitude of the region
`lonw` & `lone`	western and eastern most longitude of the region
`ncol`, `nrow` & `px`	default matrix size per region described by the number of columns, rows and pixels. **ATTENTION!**! Regions of the same spatial extent but different default (matrix-) resolution may cause errors when reading or writing '.gz'-files and must therefore be distinguished by different keywords.
`cbx1` & `cbx2`	x-coordinates for colorbar
`cby1` & `cby2`	y-coordinates for colorbar
`gradient`	the color-`gradient` of the colorbar (`x` for horizontal, and `y` for vertical)
`oticks`	the margin where the colorbar ticks should be plotted, relative to the colorbar rectangle (`'l'` left, `'r'` right and `'b'` for bottom)
`figxdim` & `figydim`	the region-specific default plot device size (width and height in inches)
`grid.res`	the default grid resolution in degrees

Author(s)

Robert K. Bauer

See Also

See add.region to add new, backup or restore region definitions, and plotmap for basic landmask plots

Examples

```
data(region_definitions)
head(region_definitions)
region_definitions$label
# ?region_definitions

# figure(width=15,height=15)
# par(mfrow=c(5,6))
# for(n in region_definitions$label) plotmap(region = n,main=n)


# Mediterranean Sea with a spatial resolution of 4km (e.g. MODIS-Aqua)
region_definitions[region_definitions$label == 'med4',]

# Mediterranean Sea with a spatial resolution of 9km (e.g. dekkar)
region_definitions[region_definitions$label == 'med9',]

# plotting same landmasks by different region-keywords
plotmap('med4')
plotmap('med9')


## Example for selecting wrong area definition when saving files
path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/med4*.gz')) # load sample-med4'.gz'-files

fname <- name_split(gz.files[1])
param <- fname$parameter
gz <- readbin(gz.files[1])
dim(gz)
v(gz.files[1])

## reset region name
# fname$area <- 'med9'
# fname <- name_join(fname)
# writebin(gz,fname,param=param)
# v(fname,folder=".")
# system(paste('rm', fname))
```

`set.colorbar` *Adds colorbar to an extisting plot device*

Description

Adds colorbar to an extisting plot device. If position vectors are not provided, the user will be asked to define the colorbar placement by the mouse cursor.

Usage

```
set.colorbar(cbx, cby, cbpos, cbline=0, pal='jet', zlim, ticks=1:10, labels=ticks,
             gradient, oticks, cb.title="", cb.xlab="", font=1, cex=1,
           cex.cb.title=0.9, cex.cb.xlab=0.8, cex.cb.ticks=0.7, cb.ticks.srt=90,
             cb.ticks.length, cb.ticks.ypos, cb.ticks.lwd=1,
             integer=F, cb.xlab.line=0, total.reg, cbxp, cbyp,...)

set.colorbarp(cbxp, cbyp, total.reg=T, year_bar=F, pal="jet",...)
```

Arguments

`cbx, cby`	(`set.colorparp`-arguments) the horizontal and vertical limits of the colorbar. If missing, the user will be asked for manual colorbar placement.
`cbxp, cbyp`	(`set.colorparp`-arguments) the horizontal and vertical limits of the colorbar in percent. If missing, the user will be asked for manual colorbar placement.
`cbpos`	letter ("b", "l", "t", "r") indicating the position of the colorbar (bottom, left, top, right). Overwrites `cbx` and `cby` values.
`cbline`	distance to default location of the colorbar, starting at 0.
`total.reg`	(`set.colorparp`-argument) if colorbar placement is relative to current subplot or entire figure region.
`year_bar`	whether to plot a colorbar with monthly ticks (by default `FALSE`)
`pal`	color map to be plotted (default is `'jet'` for direct calls). See cmap for available color maps and parameter_definitions for predefined colormaps of different parameters (for internal function calls, e.g. v))
`zlim`	(optional) value limits of the color bar. Overwrites ticks if ticks are povided.
`ticks`	the points at which tick-marks are to be drawn (default is 1:10). Non-finite (infinite, NaN or NA) values are omitted. Gets verwritten by zlim if povided.
`labels`	character or expression vector of labels to be placed at the tickpoints. (default equals `ticks`-values.)
`gradient`	whether to have a horizontal (x) or vertical (y) color gradient.
`oticks`	the margin where to put the colorbar ticks relative to the colorbar rectangle (`'l'` left, `'r'` right and `'b'` for bottom;
`cb.title`	character string indicating the title of the colorbar (default is set to date information/empty string if date information is missing.)

cb.xlab	character string indicating the x-axis label of the colorbar.
font	Integer specifying font to use for text. 1=plain [default], 2=bold, 3=italic, 4=bold italic, 5=symbol
cb.xlab.line	line of x-axis colorbar label
cex, cex.cb.title, cex.cb.xlab, cex.cb.ticks	*cex*: general font size, used as reference for colorbar labels and title *cex.cb.xlab*: font size of the x-axis label of the colorbar *cex.cb.title*: font size of the title of the colorbar
cb.ticks.srt, cb.ticks.length, cb.ticks.ypos, cb.ticks.lwd	rotation, length, relative y-position and line width of colorbar ticks
integer	(default is FALSE).
...	additional arguments to be passed to text or set.colorpar

Details

set.colobar adds a colorbar to the current plot device. If colorbar positions are missing (cbx, cby), the user will be asked for manual placement. ticks and tick-labels should correspond to zlim-values of the plot. pal defines the colormap and should equal col of the selected plot.

Value

a list of colorbar definition vectors: oticks, gradient, cbx and cby. See function argmuments for more details.

Author(s)

Robert K. Bauer

Examples

```
## Example 1: plot colorbars manually
par(mar=c(8,8,8,8))
plot(0.5,0.5,xlim=c(0,1),ylim=c(0,1))
set.colorbar(cbx=c(0, 1), cby=c(-.3, -.4)) # bottom
set.colorbar(cby=c(0, 1), cbx=c(-.4, -.3)) # left
set.colorbar(cbx=c(0, 1), cby=c(1.2, 1.3)) # top
set.colorbar(cby=c(0, 1), cbx=c(1.2, 1.3)) # right


## Example 2: use cbpos
par(mar=c(8,8,8,8))
plot(0.5,0.5,xlim=c(0,1),ylim=c(0,1))
set.colorbar(cbpos='b') # bottom
set.colorbar(cbpos='l') # left
set.colorbar(cbpos='t') # top
set.colorbar(cbpos='r') # right


## Example 3: interactive placement
# par(mar=c(8,8,8,8))
```

```
# plot(0.5,0.5,xlim=c(0,1),ylim=c(0,1))
# cb <- set.colorbar() # interactive
# plot(0.5,0.5,xlim=c(0,1),ylim=c(0,1))
# set.colorbar(cbx=cb$cbx, cby=cb$cby) # reuse stored colorbar positions
```

SpatialCircle *Creates a circle of radius r around a specified point.*

Description

Returns a SpatialLines object, that defines a circle of radius r around a specified point.

Usage

```
SpatialCircle(x,y,r,n=100,proj4str)
```

Arguments

x, y	x and y coordinates of the circle
r	radius (of the same scale as plotting window)
n	precision indicator of the circle.
proj4str	projection string.

Author(s)

Robert K. Bauer

See Also

get.avg.bathy

Examples

```
## Example 1: load & plot bathymetry of the Baltic Sea, defined by longitudes and latidtues
lon <- c(9, 31)
lat <- c(53.5, 66)
# bathy <- get.bathy(lon=lon, lat=lat, main="Baltic Sea", cbpos='r')
plotmap(lon=lon, lat=lat)
spc <- SpatialCircle(x= 20,y = 57.5,r=1)
plot(spc,add=TRUE)
# get.avg.bathy(c(20,57.5), radius = 1, bathy = bathy)
# get.avg(c(20,57.5), radius = 1, unit="km",raster = bathy)
```

v *Plotting spatial data*

Description

Plots spatial data (e.g. 2D oceanographic data). Valid input data are objects of class 'Raster' ('RasterLayer', 'RasterStack' or 'RasterBrick'), 'ncdf4' (already loaded netcdf files) or a character strings indicating 'bathy'metric data, 'gz'- or '.nc-files' (netcdf). See also name_split for further information on '.gz'-file nomenclature.

Usage

```
## S4 method for signature 'bathy'
v(obj, v_area, lon, lat, resolution=4, keep=F,
 savename.bathy, folder.bathy=".", adaptive.vals=T, cb.title, show.colorbar=T,...)

## S4 method for signature 'nc'
v(obj, varname, t=1, layer=t, adaptive.vals=T, dates,
  cb.xlab=varname, show.colorbar=T ,...)

## S4 method for signature 'ncdf4'
v(obj, varname, t=1, layer=t, adaptive.vals=T, dates,
  cb.xlab=varname, show.colorbar=T, ...)

## S4 method for signature 'RasterLayer'
v(obj, varname, t=1, layer=t, ...)

## S4 method for signature 'RasterBrick'
v(obj, varname, t=1, layer=t,  ...)

## S4 method for signature 'RasterStack'
v(obj, varname, t=1, layer=t,  ...)

## S4 method for signature 'gz'
v(obj, v_area, adaptive.vals=F, show.colorbar=T,...)
```

Arguments

obj	object of class 'Raster' ('RasterLayer', 'RasterStack' or 'RasterBrick'), 'ncdf4' or a character string indicating , 'bathy'metric data, '.gz'- or '.nc'-files to plot.
v_area	character string identifying the region that should be plotted, or in case of obj == 'bathy', also a Raster* or Extent object. If missing, region is derived from the '.gz'-filename. See region_definitions for area definitions and use add.region to add new regions.

`adaptive.vals` sets minimum and maximum z-value according to the `'.gz'`-files value range. (**ATTENTION!** `minv` and `maxv` are disregarded if set!). (default is `TRUE` for non-`'.gz'`-files. If `FALSE` or not set, default value from the parameter_definitions-dataset will be applied according to the `param`-value.

`t, layer` layer/time stemp to select in multi-layer files/objects (e.g. ncdf4, RasterStack).

`dates` vector of type 'character' indicating dates per layer, used to define the title of the colorbar. Argument is omitted for `'.gz'`-files but date-information is derived from the filename. For `'.nc'`-files or 'ncdf4'-objects, date information is derived from the time-vector. For raster-objects the layer name is applied.

`varname` character string indicating the name of the variable to plot. For `'.nc'`-files or 'ncdf4'-objects, this name must correspond to a variable name defined in the file/object. Sets also colorbar-title for non-`'.gz'`-files if `cb.xlab` is missing.

`cb.title` character string indicating the title of the colorbar (default is set to date information/empty string if date information is missing.)

`cb.xlab` character string indicating the x-axis label of the colorbar and `cb.xlab.line` its placement line (default is 0). If not defined, it will be set to `varname` for raster, `ncdf4`-objects and `'.nc'`-files or for `'.gz'`-files to a predefined title in the parameter_definitions-dataset according to the `param`-value.

`lon` Vector returning longitude coordinates of the area to be plotted, only valable for `obj == 'bathy'`.

`lat` Vector returning latitude coordinates of the area to be plotted, only valable for `obj == 'bathy'`.

`resolution` resolution of the bathymetric grid, in minutes (default is 4), only valable for `obj == 'bathy'`.

`keep` whether to write the data downloaded from NOAA into a file (default is FALSE), only valable for `obj == 'bathy'`.

`savename.bathy` savename for the bathymetric data file, if not specified set to type 'bathy_lonlat_res.resolution.dat' or 'bathy_v_area_res.resolution.dat', only valable for `obj == 'bathy'`.

`folder.bathy` directory where bathymetric data should be saved (default is current working directory), only valable for `obj == 'bathy'`.

`show.colorbar` weather a colorbar should be plotted for image plots(default is `T`).

`...` additional arguments to be passed:

`region` see `v_area`.

`minv, maxv` minimum and maximum z-value to be plotted. If not set, default value from the parameter_definitions-dataset will be applied. Argument is overwritten by `adaptive.vals` and `zlim`.

`replace.na` whether missing values should be replaced by minimum values (default is `FALSE`.)

`param` character string indicating the parameter name for the dataset treatment. See `parameter_definitions` for available parameters. For `'.gz'`-files, `param` is derived from the filename. For non-`'.gz'`-files this value is non-obligatory, but can replace the `varname`-argument and vise versa. See examples.

`main` an overall title for the plot: see title.

`cbpos` letter ("b", "l", "t", "r") indicating the position of the colorbar (bottom, left, top, right). Overwrites `cbx` and `cby` values.

`cbx` the horizontal limits (x1, x2) of the colorbar. If missing and the value can not be reconstructed by the region information (e.g. `v_area`, `'.gz'`-file), the user will be asked for manual colorbar placement.

`cby` the vertical limits (y1, y2) of the colorbar. If missing and the value can not be reconstructed by the region information (e.g. `v_area`, `'.gz'`-file), the user will be asked for manual colorbar placement.

`nticks` number of tick marks for the colorbar (default is 5).

`pal` color map to be plotted (default is the 'jet'-colormap, or in case of `'.gz'`-files derived from the parameter_definitions-dataset. See cmap for available color maps and parameter_definitions for predefined colormaps for different parameters.)

`sidelabels` whether an additional y-axis label and title should be added to the plot device (default is `FALSE`). If `TRUE`, y-axis label is defined by `Ylab`, the additional title is derived from the `date`-information and gives the month information.

`Ylab` an additional title for the y axis (default is date information), only used when `sidelabels` is set `TRUE`. Default value is year-information.

`axeslabels` whether axeslabels should be shown (default is `TRUE`, set as 'longitude' and 'latitude')

`subplot` whether `'.gz'`-file will be plotted as a sub plot to an existing plot device (default is `FALSE`; see: par)

`width, height` the width and height of the plotting window, in inches. For `'.gz'`-files, default values are derived from the `region`-name as indicated by the filename. See region_definitions for predescribed definitions and use add.region to add new region definitions.

`figdim` numeric vector indicating the `width` and `height` of the plot device in inches. For `'.gz'`-files, default values are derived from the `region`-name as indicated by the filename. Value is overwritten if both, `width` and `height` are provided. See region_definitions for predescribed definitions and use add.region to add new region definitions.

`xpos` integer: initial position of the top left corner of the figure window on the pc-screen, given in pixels. Negative values are from the opposite corner. (default is -1). Disregarded under Mac OS and if `Save` is set `TRUE`.

`Save` whether the a plot device should be saved automatically as an image file of type `fileformat` in a folder specified by `plotfolder` (default is `FALSE`)

`plotfolder` directory where images should be saved (default is current working directory).

`plotname` the name of the output file(s). If not set, value will be derived from the provided file information (For `'.gz'`-files, default `plotname` is equal to the `'.gz'`-filename, replacing the `'.gz'`-fileformat-suffix with the defined image-`fileformat`.

`fileformat` fielformat of image file to be saved (only `png` and `eps` are accepted; default is `png`).

suffix suffix to be added to the image filename, before the filetype specification (e.g. '...suffix.png').

v_image whether an image-plot should be plotted (default is TRUE)

v_contour whether contour lines should be plotted (default is FALSE). If levels are specified, v_contour is set TRUE.

levels numeric vector of levels at which to draw contour lines.

contour.labels a vector giving the labels for the contour lines. By default levels are used as labels.

v_arrows whether current or wind vectors should be plotted (default is TRUE; Argument is disregarded for non-.gz-files and omitted if non current or wind data-files are provided)

scale_arrow scale factor needed for current and wind vector plots (default is 1; Argument is disregarded for non-.gz-files and omitted if no current or wind data-files are provided, indicated by the param-argument (valid param-definitions are: 'uz' and 'vz', for current data, 'wu' and 'wz' for wind data))

terrain whether the to keep terrain data (default is FALSE). If set FALSE and visualize is TRUE, grid command in plotmap is disabled!

verbose whether the plot information shall be printed in the R-console (by default TRUE)

... Additional arguments to be passed to plotmap (bwd, fill, col, border, grid, grid.res, axeslabels, ticklabels, cex.lab, cex.ticks).

Details

v uses the maps and maptools functions to plot the landmask. See clim_plot for aligned plots of satallite-data climatologies.

Author(s)

Robert K. Bauer

References

Bauer, R. K., Stepputtis, D., Grawe, U., Zimmermann, C., and Hammer, C. 2013. Wind-induced variability in coastal larval retention areas: a case study on Western Baltic spring-spawning herring. Fisheries Oceanography, 22: 388-399.

See Also

clim_plot, readbin, name_split, regions, plotmap, v

Examples

```
################## simple example section:

## Example 1: load & plot a sample Raster-object
path <- system.file("test_files", package="oceanmap")
load(paste0(path,"/medw4_modis_sst2_4km_1d_20020705_20020705.r2010.0.qual0.Rdata"),verbose=TRUE)
```

```
dat <- raster::crop(dat,extent(c(0,10,40,44))) ## crop data, xlim/ylim not yet implemented in v()
print(dat)
v(dat, main="Raster-object", cbpos='r')


## Example 2: load & plot sample netcdf-file ('.nc'-file)
nfiles <- Sys.glob(paste0(path,'/*.nc')) # load list of sample-'.nc'-files
head(nfiles)

### option a) load netcdf-file with ncdf4-package and plot it
library('ncdf4')
ncdf <- nc_open(nfiles[1])
# print(ncdf)
# v(obj = ncdf, cbpos="r")

### option b) load and plot netcdf-file as RasterStack object
# nc <- nc2raster(nfiles[1])
# v(nc,cbpos="r") # plot RasterStack object

### option c) plot netcdf-file directly
# v(nfiles[1], cbpos="r")
# v(nfiles[1], cbpos="r", replace.na=TRUE)

###### plot multiple layers:
# par(mfrow=c(2,2))
# v(nfiles[1], t=1:4, cbpos="r", replace.na=TRUE, subplot = TRUE)


# ## Example 2: load & plot bathymetry data from the NOAA-ETOPO1 database
# par(mfrow=c(2,1))
# bathy <- get.bathy("medw4", terrain=T, res=3, keep=T, visualize=T, subplot = TRUE, grid=F)
# # load('bathy_medw4_res.3.dat',verbose = T); bathy <- h
# v(bathy, param="bathy", subplot = TRUE, terrain=F, levels=c(200,2000)) # show contours
#
# ## b) only contour lines:
# par(mfrow=c(1,2))
# h <- get.bathy("lion",terrain=F,res=3, visualize=T,
#                v_image = FALSE, levels=c(200,2000))
#
# ## use v-function for same plot but on subregion:
# v(h,v_area = "survey", param="bathy",
#   v_image = FALSE, levels=c(200,2000))


## Example 3: plot sample-'.gz'-file
gz.files <- Sys.glob(paste0(path,'/*.gz'))
# v(gz.files[2]) ## plot content of gz-file


## Example 4: load sample-'.gz'-file manually as Raster-object and plot it
obj <- readbin(gz.files[2],area='lion')
# par(mfrow=c(1,2))
# v(obj,param="sst",subplot = TRUE)
```

```
# v(obj,param="Temp",subplot = TRUE) ## note unset "pal" (colormap) for unkown "param"-values!


## Example 5: available color maps
data('cmap') # load color maps data
names(cmap) # list available color maps

gz.files <- Sys.glob(paste0(path,'/*.gz'))
# figure(width=15,height=15)
# par(mfrow=c(4,5))
# for(n in names(cmap)) v(gz.files[2], v_area='lion', subplot=TRUE,
#                         pal=n, adaptive.vals=TRUE, main=n)

## define new color maps from blue to red to white:
n <- colorRampPalette(c('blue','red','white'))(100)
# v(gz.files[2], v_area='lion', subplot=TRUE,
#  pal=n, adaptive.vals=TRUE, main="own colormap")


## Example 6: available parameters
data(parameter_definitions)
names(parameter_definitions)
# ?parameter_definitions

# figure(width=12, height=6.2)
# par(mfrow=c(2,3))
# v('*sst2*707*',v_area="medw4",main="sst", folder=path, subplot=TRUE)
# v('*chla*531*',v_area="medw4",main="chla", folder=path, subplot=TRUE)
# v('*chlagrad*',v_area="medw4",main="chlagrad",folder=path, subplot=TRUE)
# v('*p100*',v_area="medw4",main="p100 (oceanic fronts)",folder=path, subplot=TRUE)
# v('*sla*',v_area="medw4",main="sla",folder=path, subplot=TRUE)
# h <- get.bathy("medw4",visualize=TRUE,terrain=F,res=4, subplot=TRUE,main="bathy")

################### advanced example section:

## Example I: plot bathymetry using a v_area-keyword
## requires server connection!
# par(mfrow=c(2,1))
# v("bathy","lion",res=4, keep=TRUE,border='grey',
#   main='Gulf of Lions bathymetry',cb.title="resolution 4 min")

# v("bathy","lion",res=1, keep=TRUE,border='grey',
#   cb.title="resolution 1 min")  # can take some time depending on server connection!


## Example II: plot bathymetry of the Baltic Sea defined by longitude and latidtue coordinates
## requires server connection!
lon <- c(9, 31)
lat <- c(53.5, 66)
#v("bathy",lon=lon,lat=lat,main="Baltic Sea")


## Example III: plot landmask of the Baltic Sea defined by an extent- or raster-object
```

```
## requires server connection!
library('raster')
ext <- extent(lon,lat)
# v("bathy",ext,main="Baltic Sea",res=4,levels=200) # extent-object


## Example IV: plot '.gz'-files, following default plot-procedure

check_gzfiles(folder=path) # return file summary-table
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files
# v(gz.files[1:4])
# v(gz.files[4],bwd=2)

## Example V: plot climatologies from '.gz'-files
##            (ATTENTION: not working for non-'gz'-files, requiring ImageMagick)
# clim_plot('*1s*.gz' ,folder = path,bwd=0.7,adaptive.vals=TRUE,plotname="seasonal_climatology.png")

## Example VI: plot subregion of gz-files as subplots
# graphics.off()
# par(mfrow=c(2,1))
# v(gz.files[1:2],v_area='lion') # run ?region_definitions to see predefined regions


## Example VII: plot subregion of raster file

# all manual:
obj <- readbin(gz.files[2],area='lion')
dev.new()
ticks <- seq(20,30,5)
par(mar=c(5,4,5,8))
image(obj,zlim=range(ticks),col=cmap$jet)
plotmap('lion',add=TRUE) # add landmask
# set.colorbar(ticks=ticks,cb.title='cb.title',cb.xlab='cb.xlab')

## using v, reconstructing region information
# obj <- readbin(gz.files[2],area='lion')
# v(obj,varname="sst2",cb.title='cb.title',cb.xlab='cb.xlab')

# using v for another subregion
ncorse <- crop(obj,extent(6,9,40,42))
# v(ncorse,grid.res=1)
# v(ncorse,zlim=c(20,30),cbx=c(8.3,8.9),cby=c(40.7,40.8)) # skipping colorbar widget


## Example VIII: Add region by supplying raster-object, colorbar positions and running the widget
#add.region(ncorse,cbx=c(8.3,8.9),cby=c(40.7,40.8))


## Example IX: plot netcdf-files ('.nc'-files)
nfiles <- Sys.glob(paste0(path,'/*.nc')) # load sample-'.nc'-files
head(nfiles)

## plot herring larval dispersal from Bauer et al. (2013)
```

```
# par(mfrow=c(2,2))
# v(nfiles[1], subplot=TRUE, t=1:4,minv=0, maxv=1000, adaptive.vals=FALSE, replace.na=TRUE)
# par(new=TRUE,mfrow=c(1,1))
# empty.plot(main='herring larval dispersal in the Greifswald lagoon, Germany')
# mtext('see Bauer et al. (2013) as reference')

# plot bathymetric data (obtained from the Leibniz Institute for Baltic Sea Research Warnemuende)
# v(nfiles[2],varname='bathymetry') # following default plot-procedure
# v(nfiles[2],varname='bathymetry',pal='haxbyrev',Log=TRUE, cb.xlab='depth [log m]',levels=50)
```

`v-class` *v-classes*

Description

internal dummy classes used by v.

`writebin` *Saves geographic data as byte file ('.gz')*

Description

Saves geographic data as byte file, in gzip compressed format (`'.gz'`). **ATTENTION!**! Only 2D (one layer) can be stored!

Usage

```
writebin(satdata,filename,folder,param)
```

Arguments

`satdata`	one layer-raster-object or matrix holding spatial data.
`param`	character string indicating the parameter name for the dataset treatment. See `parameter_definitions` for available parameters.
`filename`	character string naming the `'.gz'`-file to be created.
`folder`	character string indicating the target directory.

Author(s)

Robert K. Bauer

See Also

readbin, regions, crop, raster2matrix, param_unconvert

Examples

```
## Example for selecting wrong area definition when saving files
path <- system.file("test_files", package="oceanmap")
gz.files <- Sys.glob(paste0(path,'/*.gz')) # load sample-'.gz'-files
v(gz.files[1])

fname <- name_split(gz.files[1])
param <- fname$parameter
gz <- readbin(gz.files[1])
dim(gz)
v(gz.files[1])

### reset region name
fname$area <- 'med9'
fname <- name_join(fname)
# writebin(gz,fname,folder=path,param=param)
# v(fname)
# system(paste('rm', fname))

### multi layer raster file
gz2 <- stack(gz,gz)
# writebin(gz2,rep(gz.files[1],2),folder=path,param) # error message since multi layer
# writebin(gz,gz.files[1],folder=path,param) # single layer raster file
# v(gz.files[1])
```

Index