Package ‘satmap’

May 6, 2014

Type Package

Title Advanced plot tools for spatial data

Version 0.0.1

Date 2014-03-16

Author Robert K. Bauer

Maintainer Robert K. Bauer <robert.bauer@ifremer.fr>

Depends R (>= 2.10), utils, raster, grDevices, ncdf4, maps, maptools, mapdata

Imports abind, fields, marmap, plotrix, rgdal, methods

Description Advanced plot tools for spatial (satellite, bathymetric and topographic) data. Recognized classes and formats include ncdf4, Raster, .nc- and .gz-files.

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-30 16:03:02

SystemRequirements ImageMagick

R topics documented:

`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 a **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 **widget** or by a one-row data frame that holds the entire information (see: region_definitions).

ATTENTION! When reinstalling or updating the satmap 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)
```

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 satmap-package is installed.
`widget`	whether a **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 satmap 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 satmap-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`).

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)

## 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
setwd(system.file("test_files", package="satmap"))
gz.files <- Sys.glob(*.gz)
print(gz.files)
area_extrac(gz.files[1],area=lion)

gz <- Sys.glob(lion*.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
setwd(system.file("test_files", package="satmap"))
gz.files <- Sys.glob(*.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 <- ... .
f <- nrow(u)*2

x11(width=9.7,height=7.8,xpos=-1)
par(mar=c(f,f,f,0)) # mar= c(bottom, left, top, right)
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 search criteria for sat files (default is `*`, including all `.gz`-files).
`folder`	Character string indicating folder in which searched filesare located (default is current working directory)
`filetype`	Character string indicating file 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

Examples

```
## Example 1: plot .gz-files, following default plot-procedure
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
check_gzfiles() # return file summary-table per filetype

gz.files <- Sys.glob(*.gz) # load sample-.gz-files
name_split(gz.files) # return summary-table per file
```

---

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, 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.
`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
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
check_gzfiles() # return file summary-table
gz.files <- Sys.glob(*1s_*.gz) # load seasonal .gz-files
v(gz.files) # as single plots

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

---

cmap *color maps*

Description

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

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

Usage

```
data(cmap)
```

Format

list

Author(s)

Robert K. Bauer

Examples

```
data(cmap)
names(cmap)

## 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
```

---

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 satmap-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(..., axes = F, xlab = "", ylab = "")
```

Arguments

`axes`	whether to show plot axes (default is `FALSE`).
`xlab, ylab`	label for the x- and y-axis of the plot (default is empty).
`...`	Other arguments to be passed either to `plot`.

Author(s)

Robert K. Bauer

Examples

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

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

Description

Returns bathymetric data from the NOAA server as RasterLayer, given coordinate bounds and resolution.

Usage

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

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`	Vector returning longitude coordinates of the area to be plotted, only valable for `x == bathy`.
`lat`	Vector returning latitude coordinates of the area to be plotted, only valable for `x == bathy`.
`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.
`...`	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: 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=F)


## Example 2: plot bathymetry of the Baltic Sea defined by longitude and latidtue coordinates
lon <- c(9, 31)
lat <- c(53.5, 66)
#get.bathy(lon=lon,lat=lat,visualize=T,main="Baltic Sea")


## 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, steps=200) # extent-object
```

`inst.pkg` *Loading Packages and automatically installs packages if missing*

Description

Loads and automatically installs packages if missing

Usage

```
inst.pkg(package)
```

Arguments

`package`	the name of a package to be loaded, given as a name or character string.

Details

`inst.pkg` is based on install.packages and library.

Author(s)

Robert K. Bauer

Examples

```
inst.pkg(satmap) # loading existing package
inst.pkg(marmap) # install and load new package
```

`internal.datasets` *internal datasets*

Description

internal (`lazyload`) datasets `medm9_proj` and `regions.dim.bathy`, accessed by `v.plot` and read-bin 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()
setwd(system.file("test_files", package="satmap"))
ncfile <- Sys.glob(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)

## Example 5: convert .nc-file to raster-object using raster-function
stack(ncfile,varname=Conc,bands=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
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
check_gzfiles() # return file summary-table
gz.files <- Sys.glob(*.gz) # load sample-.gz-files

# 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 table of an '* `.gz` *'-filen based on its name*

Description

Returns a summary table of an `.gz`-filen based on its name

Usage

```
name_split(filename)
```

Arguments

`filename`	Character string indicating search criteria for sat files.

Value

Returns a summary table of a `.gz`-file (by splitting its name) with the header:

`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 temporal resolution the time interval covered
`option`	a character string holding supplmentary information of `.gz`-file treatment

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_join to create `.gz`-filenames from splitted names (name_split)-calls

Examples

```
## Example: read and plot .gz-file
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
check_gzfiles() # return file summary-table per filetype
gz.files <- Sys.glob(*.gz) # load sample-.gz-files

# 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_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, lonname="lon", latname="lat",
    layer, date=T)
```

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.
`lonname`	character string indicating the name of the longitude-variable of `ncdf4`-objects and `.nc`-files to plot (default is `lon`)
`latname`	character string indicating the name of the latitude-variable of `ncdf4`-objects and `.nc`-files to plot (default is `lat`)
`layer, t`	layer/time stemp to select in multi-layer files.
`date`	whether the layer names should be set to the date of the ncdf-file layer (default is `TRUE`, format is 'X%Y%m%d').

Value

RasterLayer or RasterStack

Author(s)

Robert K. Bauer

Examples

```
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
nfiles <- Sys.glob(*.nc) # 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
```

```
## ordinary raster call
raster(nfiles[1])
brick(nfiles[1],band=c(1,4))

setwd(owd)
```

---

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

```
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
nfile <- Sys.glob(herring*.nc) # load sample-.nc-files
head(nc2time(nfile))

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

setwd(owd)
```

`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)
setwd(system.file("test_files", package="satmap"))
gz.file <- Sys.glob(*.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, axeslabels = T,
        ticklabels = T, grid = T, cex.lab = 0.8, cex.ticks = 0.8,
        grid.res, main, col = "grey", border = "black",
        fill = T, bwd = 1, v_area)
```

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`	Vector returning longitude coordinates of the area to be plotted.
`lat`	Vector returning latitude coordinates of the area to be plotted.
`add`	whether the a the landmask should be added to an existent figure (default is `FALSE`)
`main`	title to be plotted
`axeslabels`	whether axis-labels (longitude and latitude) should be added to the axes (default is `TRUE`). Can be a single value or a vector of size two.
`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)
`bwd`	width is of the axes bars (default is 1)
`fill`	whether the a the landmask should be filled by a color (default is `TRUE`)
`col`	fill color of the landmask to be plotted (default is grey)
`border`	country border color of the landmask to be plotted (default is black)

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
par(mfrow=c(2,1))
plotmap(med4,main="Mediterranean Sea")
plotmap(med4,main="Mediterranean Sea",bwd=2,border=grey,grid=FALSE)


## Example 2: plot landmask of the Baltic Sea defined by longitude and latidtue coordinates
lon <- c(9, 31)
lat <- c(53.5, 66)
plotmap(lon=lon,lat=lat,main="Baltic Sea",grid.res=5,bwd=2)


## Example 3: plot landmask of the Baltic Sea defined by an extent- or raster-object
library(raster)
ext <- extent(lon,lat)
plotmap(ext,main="Baltic Sea") # extent-object

r <- raster(ext)
plotmap(r,main="Baltic Sea") # raster-object
```

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)
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
check_gzfiles() # return file summary-table
gz.files <- Sys.glob(*.gz) # load sample-.gz-files

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
owd <- getwd()
setwd(system.file("test_files", package="satmap"))
check_gzfiles() # return file summary-table
gz.files <- Sys.glob(*.gz) # load sample-.gz-files
```

```
### all manual:
obj <- readbin(gz.files[2],area=lion)
obj
x11()
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)
```

---

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.

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 to the dataset by add.region or deleted by calling delete.region.

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

```
## Example
data(region_definitions)
head(region_definitions)

# 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
setwd(system.file("test_files", package="satmap"))
gz.files <- Sys.glob(med4*.gz) # load sample-med4.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,param=param)
v(fname)
system(paste(rm, fname))
```

---

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

Description

Adds colorbar to an existing 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, pal="jet", ticks=1:10, labels=ticks,
             gradient, oticks, cb.title="", cb.xlab="")
```

Arguments

`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.
`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))
`ticks`	the points at which tick-marks are to be drawn (default is 1:10). Non-finite (infinite, NaN or NA) values are omitted.
`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.

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: matrix
vals <- rnorm(1000,500,100)
vals <- vals[order(vals)]
hist(vals)
z <- matrix(vals,100,100)
zlim <- c(0,1000)
par(mar=rep(5,4)) # add space for colorbar
data(cmap)
image(z,col=cmap$jet,zlim=zlim)
ticks <- seq(zlim[1],zlim[2],100)

set.colorbar(cbx=c(0,1),cby=c(-.16, -.248), ticks=ticks) # bottom
set.colorbar(cby=c(0,1),cbx=c(-.16, -.088), ticks=ticks) # left
set.colorbar(cbx=c(0,1),cby=c(1.088, 1.16), ticks=ticks) # top
set.colorbar(cby=c(0,1),cbx=c(1.088, 1.16), ticks=ticks) # right

# manual placement calling the widget
cb <- set.colorbar(ticks=ticks)

# reuse definition
image(z,col=cmap$jet,zlim=zlim)
set.colorbar(cbx=cb$cbx, cby=cb$cby, ticks=ticks)
```

---

v *Ploting geographic data*

Description

Plots geographic data. Valid input data are objects of class 'Raster' ('RasterLayer', 'RasterStack' or 'RasterBrick'), 'ncdf4' or a character strings indicating `bathymetric` data, 'gz'- or '.nc-files. 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, ...)

## S4 method for signature nc
v(obj, varname, t=1, adaptive.vals=T, dates,
  cb.xlab=varname, lonname="lon", latname=lat,...)

## S4 method for signature ncdf4
v(obj, varname, t=1, adaptive.vals=T, dates,
  cb.xlab=varname, lonname="lon", latname=lat,...)

## S4 method for signature RasterLayer
v(obj, varname, t=1, ...)

## S4 method for signature RasterBrick
v(obj, varname, t=1, ...)

## S4 method for signature RasterStack
v(obj, varname, t=1, ...)

## S4 method for signature gz
v(obj, v_area, adaptive.vals=F, ...)
```

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/time stemp to select in multi-layer files.
`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. 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.
`lonname`	character string indicating the name of the longitude-variable of `ncdf4`-objects and `.nc`-files to plot (default is `lon`)
`latname`	character string indicating the name of the latitude-variable of `ncdf4`-objects and `.nc`-files to plot (default is `lat`)
`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_lon-lat_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`.
`...`	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.

`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.

`pal` color map to be plotted (defualt 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.)

`steps` number: step size between minimum and maximium colorbar ticks.

`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). Omitted 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))

... 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

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

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


## Example 2: plot bathymetry of the Baltic Sea defined by longitude and latidtue coordinates
lon <- c(9, 31)
lat <- c(53.5, 66)
#v("bathy",lon=lon,lat=lat,main="Baltic Sea")


## Example 3: plot landmask of the Baltic Sea defined by an extent- or raster-object
library(raster)
ext <- extent(lon,lat)
#v("bathy",ext,main="Baltic Sea",res=4,levels=200, steps=200) # extent-object


## Example 4: plot .gz-files, following default plot-procedure
owd <- getwd()
```

```
setwd(system.file("test_files", package="satmap"))
check_gzfiles() # return file summary-table
gz.files <- Sys.glob(*.gz) # load sample-.gz-files
v(gz.files[1:3])
v(gz.files[3],bwd=2)

## Example 5: plot climatologies from .gz-files
##            (ATTENTION: not working for non-gz-files, requiring ImageMagick)
clim_plot(*1s*.gz,bwd=0.7,adaptive.vals=TRUE,plotname="seasonal_climatology.png")

## Example 6: plot subregion of gz-files as subplots
graphics.off()
par(mfrow=c(2,1))
v(gz.files[1:2],v_area=lion,subplot=TRUE) # run ?region_definitions to see predefined regions


## Example 7: plot subregion of gz-file with different color maps
data(cmap)
par(mfrow=c(3,4))
for(i in 1:11) v(gz.files[2],v_area=lion,subplot=TRUE,pal=names(cmap)[i],
                 adaptive.vals=TRUE,main=names(cmap)[i])


## Example 8: plot subregion of raster file

# all manual:
obj <- readbin(gz.files[2],area=lion)
x11()
ticks <- seq(20,30,5)
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
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 9: 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 10: plot netcdf-files (.nc-files)
nfiles <- Sys.glob(*.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,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.

Author(s)

Robert K. Bauer

See Also

readbin, regions, crop, raster2matrix, param_unconvert

Examples

```
## Example for selecting wrong area definition when saving files
setwd(system.file("test_files", package="satmap"))
gz.files <- Sys.glob(med4*.gz) # load sample-med4.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,param=param)
v(fname)
system(paste(rm, fname))

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

Index