Package: SOmap (via r-universe)

September 2, 2024

Type Package

Title Southern Ocean maps

Version 0.6.2.9005

Description Create publication-quality Southern Ocean maps in a simple manner with multiple management layer options. Future versions will have the option of adding extra rounded legends for other layers.

URL https://github.com/AustralianAntarcticDivision/SOmap

BugReports https://github.com/AustralianAntarcticDivision/SOmap/issues

License GPL (>= 2)

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

Depends R (>= 3.6.0), raster, sp

Imports assertthat, dplyr, ggplot2, glue, graticule, methods, proj4, reproj (>= 0.4.0), rlang, sf (>= 0.7-3), spbabel, spex (>= 0.6.0), stars, terra, vapour (>= 0.9.5), tabularaster, tibble, uuid

Suggests adehabitatLT, covr, ggnewscale, knitr, maps, rmarkdown, testthat (>= 2.1.0), palr, vdiffr

VignetteBuilder knitr

Repository https://scar.r-universe.dev

RemoteUrl https://github.com/AustralianAntarcticDivision/SOmap

RemoteRef HEAD

RemoteSha 8d0acf7c7181123e79f8c2d380e053aa90489c2d

Contents

`Bathy` *Bathymetric data for maps*

Description

Bathymetric data reprocessed from the GEBCO_2014 Grid data set.

Usage

```
data(Bathy)
```

Format

An object of class `"RasterLayer"`

Source

GEBCO

References

The GEBCO_2014 Grid, version 20150318

`ice` *Sea ice*

Description

Example sea ice concentration data from the Southern Ocean (2018-10-15). (See "data-raw/ice.R").

Examples

```
## Not run:
  ll <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"
  xy <- coordinates(spTransform(as(SOmap_data$fronts_orsi, "SpatialPoints"), ll))
  ## just because you can doesn't mean you should ...
 SOmap_auto(xy[,1], xy[,2], bathy = ice, input_points = FALSE, levels = c(15, 30, 60, 90))

## End(Not run)
```

`is.discrete` *used in SOleg to test color palettes*

Description

used in SOleg to test color palettes

Usage

```
is.discrete(x)
```

Arguments

`x`	Object to test for is it discrete

Value

returns true false

latmask *Latitude mask for polar rasters.*

Description

Latitude mask for polar projections; written by M.D. Sumner and part of the spex package.

Usage

```
latmask(x, latitude = 0, southern = TRUE)
```

Arguments

x	A raster layer.
latitude	maximum latitude (effectively a minimum latitude if southern = FALSE)
southern	flag for whether south-polar context is used, default is TRUE

Examples

```
## Not run:
## assumes that you have already defined a raster object called 'ice'
plot(latmask(ice, -60))

## End(Not run)
```

reproj *Reproject SOmap*

Description

Reproject a SOmap object by specifying a 'target' projection string (PROJ4)

Usage

```
## S3 method for class 'SOmap'
reproj(x, target, ..., source = NULL)

## S3 method for class 'SOmap_auto'
reproj(x, target, ..., source = NULL)

## S3 method for class 'SOmap_management'
reproj(x, target, ..., source = NULL)

## S3 method for class 'SOmap_legend'
reproj(x, target, ..., source = NULL)
```

```
## S3 method for class 'BasicRaster'
reproj(x, target, ..., source = NULL)

## S3 method for class 'Spatial'
reproj(x, target, ..., source = NULL)

## S3 method for class 'sf'
reproj(x, target, ..., source = NULL)

## S3 method for class 'sfc'
reproj(x, target, ..., source = NULL)
```

Arguments

`x`	coordinates
`target`	target specification (PROJ.4 string or epsg code)
`...`	arguments passed to the underlying projection engine, see `reproj::reproj()`
`source`	source specification (PROJ.4 string or epsg code)

Details

See `reproj::reproj()` for details.

Warning

So many ...

See Also

`reproj::reproj()`

Examples

```
## Not run:
  set.seed(27)
  amap <- SOmap_auto()
  reproj(amap, "+proj=moll")
  reproj(amap, "+proj=laea +lat_0=-55 +lon_0=154 +datum=WGS84")

  bmap <- SOmap(trim = -35)

  ## works great!
  reproj(bmap, "+proj=stere +lat_0=-90 +lon_0=147 +lat_ts=-71 +datum=WGS84")

  ## these aren't exactly ideal
  reproj(bmap, "+proj=ortho +lat_0=-70")
  reproj(bmap, "+proj=laea +lat_0=-55 +lon_0=154 +datum=WGS84")

## End(Not run)
```

SOauto_crop *Reproject and crop Spatial and sf objects to SOmap objects*

Description

Reproject and crop Spatial and sf objects to SOmap objects

Usage

```
SOauto_crop(layer, x, sp = TRUE)
```

Arguments

`layer`	: an `sf` or `Spatial` (SpatialPolygonsDataFrame, SpatialLinesDataFrame, SpatialPointsDataFrame etc) object to reproject and crop
`x`	: a SOmap or SOauto_map object
`sp`	logical: if `TRUE`, return the cropped object in `Spatial` form, otherwise `sf`

Value

If successful, a reprojected and cropped version of `layer`. If the reprojection or cropping operations fail, the returned object will be of class `try-error`. If the cropping operations return an empty object (i.e. no parts of `layer` lie within the bounds of `x`) then the returned object will either be `NULL` (if `sp = TRUE`) or an `sf` object with no features if `sp = FALSE`.

Examples

```
## Not run:
  a <- SOmap_auto(c(0, 50), c(-70, -50))
  x <- SOauto_crop(SOmap_data$fronts_orsi, a)
  plot(a)
  plot(x, add = TRUE)

  a <- SOmap(trim = -60)
  x <- SOauto_crop(SOmap_data$EEZ, a)
  plot(a)
  plot(x, add = TRUE)

## End(Not run)
```

SObin — *Bin longitude latitude points by count in a SOmap context*

Description

Creates a raster density layer from a set of points.

Usage

```
SObin(
  x,
  y = NULL,
  baselayer = NULL,
  ...,
  col = hcl.colors(26, "Viridis"),
  dim = c(512, 512),
  add = TRUE,
  target = NULL,
  source = NULL,
  data.frame = FALSE
)
```

Arguments

`x`	longitudes
`y`	latitudes
`baselayer`	optional spatial layer to get extent from
`...`	passed to plot if `add = TRUE`
`col`	colours to use if `add = TRUE`
`dim`	dimensions of raster to bin to
`add`	if `TRUE`, the raster is added to the current plot. An error is thrown if there is no existing plot
`target`	target projection passed to SOproj
`source`	source projection of data projection passed to SOproj
`data.frame`	if true return a data frame instead of a raster.

Value

A raster. If `add = TRUE`, it is returned invisibly.

Examples

```
## Not run:
  SOmap_auto()
 pts <- cbind(lon = runif(1e6, min = -180, max = 180), lat = runif(1e6, min = -90, max = 90))
  bin <- SObin(pts[, 1], pts[, 2], add = TRUE)

## End(Not run)
```

SOcode *Extract plotting code from a SOmap*

Description

This is thoroughly experimental!

Usage

```
SOcode(x, data_object_name = "SOmap_data")
```

Arguments

`x`	: a map object as returned by SOmap, SOmanagement, SOleg, or SOgg
`data_object_name`	string: the name to use for the object that will hold the map data. See Examples, below

Value

A list with two elements: `code` contains R code that will draw the map, and `SOmap_data` (or whatever was passed as the `data_object_name` argument) contains any data required by that code

See Also

SOmap

Examples

```
## Not run:
  p <- SOmap()
  mapcode <- SOcode(p, data_object_name = "SOmap_data")

  ## write this code to a file
  my_R_file <- tempfile(fileext = ".R")
  writeLines(mapcode$code, con = my_R_file)

  ## you can edit the code in that file if desired

  ## save the data
  my_data_file <- tempfile(fileext = ".rds")
  saveRDS(mapcode$SOmap_data, my_data_file)

  ## later on, we can re-load the data and execute the code
  SOmap_data <- readRDS(my_data_file)
  source(my_R_file)

  ## or just to show that this works, evaluate the returned code directly against its data
  with(mapcode, for (codeline in code) eval(parse(text = codeline)))

## End(Not run)
```

SOcrs *SOmap coordinate system*

Description

Set or return the coordinate system currently in use.

Usage

```
SOcrs(crs = NULL)
```

Arguments

`crs`	provide PROJ string to set the value

Details

If argument `crs` is NULL, the function returns the current value (which may be `NULL`).

Examples

```
## Not run:
SOmap()
SOcrs()

## End(Not run)
```

SOgg *Generate a ggplot2 representation of an SOmap object*

Description

Note: this function is still experimental! Use at your own risk.

Usage

```
SOgg(...)
```

Arguments

`...`	: one or more objects as returned by `SOmap`, `SOmap2`, `SOmanagement`, or `SOmap_auto`

Value

An object of class "SOmap_gg", "SOmanagement_gg", or "SOmap_auto_gg". Printing or plotting this object will cause it to generate a ggplot2 object, which will be returned to the user. If this object is printed or plotted (e.g. to the console) then it will be displayed in the current graphics device as is usual for ggplot2 objects.

Examples

```
## Not run:
  ## generate a SOmap object
  p <- SOmap2(trim = -45, iwc = TRUE, iwc_labels = TRUE, graticules = TRUE, fronts = TRUE,
              mpa = TRUE, mpa_labels = TRUE)

  ## convert this to a ggplot2-based representation
  pg <- SOgg(p)

  ## display it
  pg

  ## we can see that this object has a bunch of ggplot code embedded inside of it
  str(pg)

  ## and that code can be modified if desired
  ## e.g. change the bathymetry colours
  pg$scale_fill[[1]]$plotargs$colours <- topo.colors(21)
  ## plot it
  pg

  ## If we want to change the legend breaks we can add breaks to the plotting arguments.
  pg$scale_fill[[1]]$plotargs$breaks <- c(0,500,1000,4000)


  ## when the print or plot method is called on pg, it creates an actual ggplot2
  ##  object, which we can capture and modify
  pg_gg <- plot(pg)
  class(pg_gg)

  ## modifying this is done in the same way any other ggplot object is modified
  ## e.g. add a new scale_fill_gradientn to override the existing one
  pg_gg + ggplot2::scale_fill_gradientn(colours = heat.colors(21))

## End(Not run)
```

`SOgg_cex` *Convert cex to ggplot size*

Description

Text size in base graphics is generally specified via `cex` values, which are multipliers applied to the device pointsize. `SOgg_cex` is a convenience function that converts a cex value into a `size` value as used by ggplot2 geometries.

Usage

```
SOgg_cex(cex)
```

Arguments

`cex`	numeric: character expansion, see `par`

Value

The corresponding 'size' value to use in ggplot calls

`SOleg` *Rounded legends for SOmap*

Description

Rounded legends for SOmap

Usage

```
SOleg(
  x = NULL,
  position = "topright",
  col = NULL,
  ticks = NULL,
  tlabs = NULL,
  breaks = NULL,
  trim = -45,
  type = "discrete",
  label = "",
  ladj = 0.5,
  lsrt = 0,
  lcex = 0.75,
  tadj = 0.5,
  tcex = 1,
  rnd = NULL,
  border_width = 2
)
```

Arguments

`x`	numeric: object to obtain min and max values from for `type = "continuous"`.
`position`	string: where you want the legend ("topleft", "topright", "bottomleft", or "bottomright").
`col`	character: colours to use.
`ticks`	numeric: number of ticks to include on the legend. Only used with `type = "continuous"`.
`tlabs`	character: tick labels. Required for `type = "discrete"`, optional for `type = "continuous"` if `x` is given.
`breaks`	numeric: vector of tick positions for `type = "continuous"` when `x` is given.
`trim`	numeric: `trim` value that was used to create the SOmap object (see `SOmap`).
`type`	string: type of legend ("discrete" or "continuous").
`label`	string: legend label.
`ladj`	numeric: distance to adjust the tick labels from the ticks.
`lsrt`	numeric: angle of the tick labels.
`lcex`	numeric: size of the tick labels.
`tadj`	numeric: distance to adjust the title from the ticks.
`tcex`	numeric: size of the title text.
`rnd`	numeric: optional rounding factor for continuous legends using the `link{round}` function.
`border_width`	numeric: thickness (in degrees of latitude) of the border.

Value

An object of class "SOmap_legend". Printing or plotting this object will cause it to be added to the SOmap in the current graphics device.

Examples

```
## Not run:
  SOmap()

  ## Discrete Legend
  SOleg(position = "topleft", col = hcl.colors(5, "Viridis"),
        tlabs = c("a", "b", "c", "d", "e"), trim = -45, label = "Species")

  ## Continuous Legend
  SOleg(x = runif(100), position = "topright", col = hcl.colors(80, "Viridis"),
        breaks = c(0.1, 0.2, 0.5, 0.9), trim = -45, label = "Species",
        rnd = 1, type = "continuous")

## End(Not run)
```

SOmanagement *Southern Ocean management map layers*

Description

Function for adding management layers to SOmap

Usage

```
SOmanagement(
  ccamlr = FALSE,
  ccamlr_labels = FALSE,
  ssru = FALSE,
  ssru_labels = FALSE,
  ssmu = FALSE,
  ssmu_labels = FALSE,
  rb = FALSE,
  rb_labels = FALSE,
  sprfmorb = FALSE,
  trim = -45,
  eez = FALSE,
  eez_labels = FALSE,
  mpa = FALSE,
  mpa_labels = FALSE,
  iwc = FALSE,
  iwc_labels = FALSE,
  domains = FALSE,
  domains_labels = FALSE,
  rb_col = "green",
  sprfmo_col = "grey50",
  ccamlr_col = "red",
  ssru_col = "grey50",
  ssmu_col = "grey70",
  eez_col = "maroon",
  mpa_col = "yellow",
  iwc_col = "blue",
  domains_col = "magenta",
  basemap
)
```

Arguments

`ccamlr`	logical: if `TRUE`, insert the CCAMLR area boundaries.
`ccamlr_labels`	logical: if `TRUE`, add labels for the CCAMLR areas.
`ssru`	logical: if `TRUE`, insert the CCAMLR small scale research unit boundaries.
`ssru_labels`	logical: if `TRUE`, add labels for the CCAMLR small scale research units.
`ssmu`	logical: if `TRUE`, insert the CCAMLR small scale management unit boundaries.
`ssmu_labels`	logical: if `TRUE`, add labels for the CCAMLR small scale management units.
`rb`	logical: if `TRUE`, insert the CCAMLR research block boundaries.
`rb_labels`	logical: if `TRUE`, add labels for the CCAMLR research blocks.
`sprfmorb`	logical: if `TRUE`, insert the SPRFMO toothfish research block boundaries.
`trim`	numeric: latitude to trim the map to. Set this to -10 for effectively no trim.
`eez`	logical: if `TRUE`, insert Exclusive Economic Zones.
`eez_labels`	logical: if `TRUE`, add labels for the Exclusive Economic Zones.
`mpa`	logical: if `TRUE`, insert CCAMLR Marine Protected Areas.
`mpa_labels`	logical: if `TRUE`, add labels for the CCAMLR Marine Protected Areas.
`iwc`	logical: if `TRUE`, insert International Whaling Commission boundaries.
`iwc_labels`	logical: if `TRUE`, add labels for the International Whaling Commission areas.
`domains`	logical: if `TRUE`, insert CCAMLR Marine Protected Areas planning domains.
`domains_labels`	logical: if `TRUE`, add labels for the CCAMLR Marine Protected Area planning domains.
`rb_col`	character: colour for CCAMLR research blocks.
`sprfmo_col`	character: colour for SPRFMO toothfish research blocks
`ccamlr_col`	character: colour for CCAMLR boundaries
`ssru_col`	character: colour for CCAMLR small scale research units.
`ssmu_col`	character: colour for CCAMLR small scale management units.
`eez_col`	character: colour for Exclusive Economic Zone boundaries.
`mpa_col`	character: colour for CCAMLR Marine Protected Areas.
`iwc_col`	character: colour for IWC boundaries.
`domains_col`	character: colour for the CCAMLR planning domains boundaries.
`basemap`	SOmap or SOmap_auto: optional map object to extract extent, projection, and other information from.

Value

An object of class "SOmap_management" containing the requested management layers. Printing or plotting this object will display those layers on the current map (note that an `SOmap` object needs to have been plotted first)

Examples

```
## Not run:
  tfile <- tempfile("SOmap", fileext = ".png")
  png(tfile, width=22, height=20, units='cm', res=600)
  SOmap(trim = -45)
  SOmanagement(ccamlr = TRUE, ccamlr_labels = TRUE, trim=-45)
  dev.off()
  unlink(tfile)

  ## map with non-default latitudinal extent
  SOmap(trim = -55)
  ## either provide the same extent via 'trim'
  SOmanagement(ccamlr = TRUE, ccamlr_labels = TRUE, trim = -55)

  ## or equivalently, pass the basemap to SOmanagement
  x <- SOmap(trim = -55)
  plot(x)
  SOmanagement(ccamlr = TRUE, ccamlr_labels = TRUE, basemap = x)

## End(Not run)
```

`SOmap` *Southern Ocean Map*

Description

Function for creating round Southern Ocean maps.

Usage

```
SOmap(
  bathy_legend = TRUE,
  border = TRUE,
  trim = -45,
  graticules = FALSE,
  straight = FALSE,
  land = TRUE,
  land_col = "black",
  ice = TRUE,
  ice_col = "black",
  fronts = FALSE,
  fronts_col = c("hotpink", "orchid", "plum"),
  border_col = c("white", "black"),
  border_width = 2,
  graticules_col = "grey70"
)
```

Arguments

`bathy_legend`	logical: if `TRUE`, insert the bathymetry legend. If `bathy_legend = NULL` or `bathy_legend = "space"`, then space will be left for the legend but no legend will actually be plotted. Use this if you plan to add a legend later.
`border`	logical: if `TRUE`, insert longitude border.
`trim`	numeric: latitude to trim the map to. Set this to -10 for effectively no trim.
`graticules`	logical: if `TRUE`, insert graticule grid.
`straight`	logical: if `TRUE`, leave a blank space on the side for a straight legend.
`land`	logical: if `TRUE`, plot coastline.
`land_col`	character: colour to use for coastline.
`ice`	logical: if `TRUE`, plot ice features (ice shelves, glacier tongues, and similar).
`ice_col`	character: colour to use for ice features.
`fronts`	logical or string: if `TRUE` or "Orsi", plot Orsi et al., (1995) ocean fronts: Subantarctic Front, Polar Front, Southern Antarctic Circumpolar Current Front. If "Park" plot the Park & Durand (2019) fronts; Northern boundary, Subantarctic Front, Polar Front, Southern Antarctic Circumpolar Current Front and Southern Boundary.
`fronts_col`	character: colours for fronts.
`border_col`	character: colours for longitude border.
`border_width`	numeric: thickness (in degrees of latitude) of the border.
`graticules_col`	string: colour for graticule grid.

Value

An object of class "SOmap", which represents a polar-stereographic map of the southern hemisphere. Printing or plotting this object will cause it to be displayed in the current graphics device.

Examples

```
## Not run:
  tfile <- tempfile("SOmap", fileext = ".png")
  png(tfile, width = 22, height = 20, units = "cm", res = 600)
  SOmap(trim = -45, graticules = TRUE)
  dev.off()
  unlink(tfile)
  SOmap(trim = -45, graticules = TRUE)

## End(Not run)
```

`SOmap-defunct` *Defunct function*

Description

Removed from SOmap

Usage

```
default_somap(...)

SOauto_map(...)
```

Arguments

`...`	all arguments passed to new function

`SOmap2` *Southern Ocean Map 2*

Description

Function for creating round Southern Ocean maps with inbuild base layers.

Usage

```
SOmap2(
  bathy_legend = TRUE,
  land = TRUE,
  ice = TRUE,
  ccamlr = FALSE,
  ccamlr_labels = FALSE,
  ssru = FALSE,
  ssru_labels = FALSE,
  ssmu = FALSE,
  ssmu_labels = FALSE,
  rb = FALSE,
  rb_labels = FALSE,
  sprfmorb = FALSE,
  border = TRUE,
  trim = -45,
  graticules = FALSE,
  eez = FALSE,
  eez_labels = FALSE,
  mpa = FALSE,
  mpa_labels = FALSE,
  domains = FALSE,
  domains_labels = FALSE,
  iwc = FALSE,
  iwc_labels = FALSE,
  straight = FALSE,
  fronts = FALSE,
  fronts_col = c("hotpink", "orchid", "plum"),
  land_col = "black",
  ice_col = "black",
  rb_col = 3,
  sprfmo_col = "grey50",
  ccamlr_col = 2,
  ssru_col = "grey50",
  ssmu_col = "grey70",
  eez_col = "maroon",
  mpa_col = "yellow",
  border_col = c("white", "black"),
  graticules_col = "grey70",
  iwc_col = "blue",
  domains_col = "magenta"
)
```

Arguments

`bathy_legend`	logical: if `TRUE`, insert the bathymetry legend.
`land`	logical: if `TRUE`, plot the coastline.
`ice`	logical: if `TRUE`, plot ice features (ice shelves, glacier tongues, and similar).
`ccamlr`	logical: if `TRUE`, insert the CCAMLR area boundaries.
`ccamlr_labels`	logical: if `TRUE`, add labels for the CCAMLR areas.
`ssru`	logical: if `TRUE`, insert the CCAMLR small scale research unit boundaries.
`ssru_labels`	logical: if `TRUE`, add labels for the CCAMLR small scale research units.
`ssmu`	logical: if `TRUE`, insert the CCAMLR small scale management unit boundaries.
`ssmu_labels`	logical: if `TRUE`, add labels for the CCAMLR small scale management units.
`rb`	logical: if `TRUE`, insert the CCAMLR research block boundaries.
`rb_labels`	logical: if `TRUE`, add labels for the CCAMLR research blocks.
`sprfmorb`	logical: if `TRUE`, insert the SPRFMO toothfish research block boundaries.
`border`	logical: if `TRUE`, insert longitude border.
`trim`	numeric: latitude to trim the map to. Set this to -10 for effectively no trim.
`graticules`	logical: if `TRUE`, insert a graticule grid.
`eez`	logical: if `TRUE`, insert Exclusive Economic Zones.
`eez_labels`	logical: if `TRUE`, add labels for the Exclusive Economic Zones.
`mpa`	logical: if `TRUE`, insert CCAMLR Marine Protected Areas.

`mpa_labels`	logical: if `TRUE`, add labels for the CCAMLR Marine Protected Areas.
`domains`	logical: if `TRUE`, insert CCAMLR Marine Protected Areas planning domains.
`domains_labels`	logical: if `TRUE`, add labels for the CCAMLR Marine Protected Area planning domains.
`iwc`	logical: if `TRUE`, insert International Whaling Commission boundaries.
`iwc_labels`	logical: if `TRUE`, add labels for the International Whaling Commission areas.
`straight`	logical: if `TRUE`, leave a blank space on the side for a straight legend.
`fronts`	logical or string: if `TRUE` or "Orsi", plot Orsi et al., (1995) ocean fronts: Subantarctic Front, Polar Front, Southern Antarctic Circumpolar Current Front. If "Park" plot the Park & Durand (2019) fronts; Northern boundary, Subantarctic Front, Polar Front, Southern Antarctic Circumpolar Current Front and Southern Boundary.
`fronts_col`	character: colours to use for fronts.
`land_col`	character: colour to use for coastline.
`ice_col`	character: colour to use for ice features.
`rb_col`	character: colour for CCAMLR research blocks.
`sprfmo_col`	character: colour for SPRFMO toothfish research blocks
`ccamlr_col`	character: colour for CCAMLR boundaries
`ssru_col`	character: colour for CCAMLR small scale research units.
`ssmu_col`	character: colour for CCAMLR small scale management units.
`eez_col`	character: colour for Exclusive Economic Zone boundaries.
`mpa_col`	character: colour for CCAMLR Marine Protected Areas.
`border_col`	character: colours for longitude border.
`graticules_col`	character: colour for graticule grid.
`iwc_col`	character: colour for IWC boundaries.
`domains_col`	character: colour for the CCAMLR planning domains boundaries.

Value

An object of class "SOmap", which represents a polar-stereographic map of the southern hemisphere, with the chosen management layers added. Printing or plotting this object will cause it to be displayed in the current graphics device.

Examples

```
## Not run:
  SOmap2(ccamlr = TRUE, mpa = TRUE, trim = -45)

## End(Not run)
```

SOmap_auto *Custom Southern Ocean map*

Description

Given some minimal input information, SOmap_auto will attempt to guess an appropriate extent and projection to use. For demonstration purposes, run the function without any inputs at all and it will use random location data.

Usage

```
SOmap_auto(
  x,
  y,
  centre_lon = NULL,
  centre_lat = NULL,
  target = "stere",
  dimXY = c(512, 512),
  bathy = TRUE,
  land = TRUE,
  land_col = "black",
  ice = TRUE,
  ice_col = "black",
  input_points = TRUE,
  input_lines = TRUE,
  graticule = TRUE,
  expand = 0.05,
  contours = FALSE,
  levels = c(-500, -1000, -2000),
  ppch = 19,
  pcol = 2,
  pcex = 1,
  bathyleg = FALSE,
  llty = 1,
  llwd = 1,
  lcol = 1,
  gratlon = NULL,
  gratlat = NULL,
  gratpos = "all",
  ...
)
```

Arguments

x	optional input data longitudes. x can also be a Raster or Spatial object, in which case the extent of x will be used for the map, but note that the contents of x will not be plotted automatically (use SOplot to do so)

`y`	optional input data latitudes
`centre_lon`	optional centre longitude (of the map projection, also used to for plot range if `expand = TRUE`)
`centre_lat`	as per `centre_lon`
`target`	optional projection family (default is `stereographic`), or full PROJ string (see Details)
`dimXY`	dimensions of background bathmetry (if used), a default is provided
`bathy`	logical: if `TRUE`, plot bathymetry. Alternatively, provide the bathymetry data to use as a `Raster` object
`land`	logical: if `TRUE`, plot coastline. Alternatively, provide the coastline data to use as a `Spatial` object
`land_col`	character: colour to use for plotting the coastline
`ice`	logical: if `TRUE`, plot ice features (ice shelves, glacier tongues, and similar)
`ice_col`	character: colour to use for ice features
`input_points`	add points to plot (of x, y)
`input_lines`	add lines to plot (of x, y)
`graticule`	flag to add a basic graticule
`expand`	fraction to expand plot range (default is 0.05, set to zero for no buffer, may be negative)
`contours`	logical: add contours?
`levels`	numeric: contour levels to use if `contours` is `TRUE`
`ppch`	set point character (default=19)
`pcol`	set point color (default=19)
`pcex`	set point cex (default=1)
`bathyleg`	optional bathymetry legend (default=FALSE)
`llty`	set line type
`llwd`	set line width
`lcol`	set line color
`gratlon`	longitude values for graticule meridians
`gratlat`	latitude values for graticule parallels
`gratpos`	positions (sides) of graticule labels
`...`	reserved, checked for defunct and deprecated usage

Details

To input your data, use input locations as `x` (longitude) and `y` (latitude) values. There must be at least two locations. The `x` input object can also be provided as a `Raster` or `Spatial` object, in which case the extent of `x` will be used for the map, but note that the contents of `x` will not be plotted automatically (use `SOplot` to do so).

Try `target` families such as 'lcc', 'laea', 'gnom', 'merc', 'aea' if feeling adventurous.

Value

An object of class SOmap_auto, containing the data and other details required to generate the map. Printing or plotting the object will cause it to be plotted.

Examples

```
## Not run:
  SOmap_auto(c(0, 50), c(-70, -50))
  SOmap_auto(runif(10, 130, 200), runif(10, -80, -10))
  SOplot(c(147, 180), c(-42, -60), pch = 19, cex = 2,col = "firebrick")
  SOmap_auto(runif(10, 130, 200), runif(10, -85, -60))

  ## save the result to explore later!
  protomap <- SOmap_auto(runif(10, 60, 160), runif(10, -73, -50))

  SOmap_auto(runif(50, 40, 180), runif(50, -73, -10), family = "laea", centre_lat = -15,
              input_lines = FALSE)

## End(Not run)
```

`SOmap_data` *Contextual data for Southern Ocean maps*

Description

Various spatial datasets that are commonly used on Southern Ocean maps.

Usage

```
data(SOmap_data)
```

Format

A list containing the following elements:

- CCAMLR_MPA
  - Description: current marine protected areas
  - Source: CCAMLR
  - URL: https://data.ccamlr.org/dataset/marine-protected-areas
  - License: not specified
- CCAMLR_research_blocks
  - Description: A defined spatial area in which research fishing on toothfish is conducted under a research plan agreed by the Commission
  - Source: CCAMLR
  - URL: https://data.ccamlr.org/dataset/research-blocks
  - License: Not specified

- CCAMLR_SSMU
  - Description: Small-scale management units (SSMUs) are designed to be used as a basis for subdividing the precautionary catch limit for krill in Subareas 48.1, 48.2, 48.3 and 48.4, and in developing management procedures for the krill fishery that can adequately account for localised effects on krill predators (SC-CAMLR-XXI, paragraphs 3.16 to 3.18). The boundaries of the SSMUs are based on predator foraging ranges (refer SC-CAMLR-XXI, Annex 4 and Trathan et al, 2008).
  - Source: CCAMLR
  - URL: https://data.ccamlr.org/dataset/small-scale-management-units
  - License: Not specified
- CCAMLR_SSRU
  - Description: Small-scale research units (SSRUs) are designed to be used as a basis for subdividing the precautionary catch limit for toothfish in exploratory fisheries, and in condcuting research fishing and developing stock assessments. The boundaries of the SSRUs are defined in Conservation Measure 41-01 (2013). CCAMLR Secretariat (2013).
  - Source: CCAMLR
  - URL: https://data.ccamlr.org/dataset/small-scale-research-units
  - License: Not specified
- CCAMLR_statistical_areas
  - Description: Statistical areas, subareas and divisions are used globally for the purpose of reporting fishery statistics. CCAMLR's Convention Area in the Southern Ocean is divided, for statistical purposes, into Area 48 (Atlantic Antarctic) between 70W and 30E, Area 58 (Indian Ocean Antarctic) between 30 and 150E, and Area 88 (Pacific Antarctic) between 150E and 70W. These areas, which are further subdivided into subareas and divisions, are managed by CCAMLR.
  - Source: CCAMLR
  - URL: https://data.ccamlr.org/dataset/statistical-areas-subareas-and-divisions
  - License: Public domain
- CCAMLR_VME_polygons
  - Description: Defined areas of registered vulnerable marine ecosystems as defined under CM 22-09.
  - Source: CCAMLR
  - URL: https://gis.ccamlr.org/geoserver/ows?service=wfs&version=1.0.0&request=GetCapabilities
  - License: Not specified
- CCAMLR_VME_fsr
  - Description: Vulnerable marine ecosystem fine-scale rectangles identified under CM 22-07.
  - Source: CCAMLR
  - URL: https://gis.ccamlr.org/geoserver/ows?service=wfs&version=1.0.0&request=GetCapabilities
  - License: Not specified
- CCAMLR_VME_risk_areas
  - Description: Vulnerable marine ecosystem risk areas declared under CM 22-07.
  - Source: CCAMLR

  - URL: https://gis.ccamlr.org/geoserver/ows?service=wfs&version=1.0.0&request=GetCapabilities
  - License: Not specified
- CCAMLR_planning_domains
  - Description: Nine planning domains were defined during the 2011 CCAMLR workshop on marine protected areas (SC-CAMLR-XXX, Annex 6). These planning domains provide comprehensive coverage of bioregions in the Southern Ocean and may be used as reporting and auditing units for work related to the development of MPAs and as a means to organise future activities related to this effort.
  - Source: CCAMLR
  - URL: https://gis.ccamlr.org/geoserver/ows?service=wfs&version=1.0.0&request=GetCapabilities
  - License: Not specified
- continent
  - Description: Coastline, details TBA
  - Source: TBA
  - URL: TBA
  - License: TBA
- EEZ
  - Description: An exclusive economic zone (EEZ) is a sea zone prescribed by the United Nations Convention on the Law of the Sea over which a state has special rights regarding the exploration and use of marine resources.
  - Source: Flanders Marine Institute (2020). Union of the ESRI Country shapefile and the Exclusive Economic Zones (version 3). https://doi.org/10.14284/403
  - URL: https://www.marineregions.org/
  - License: CC-BY
- fronts_orsi
  - Description: Southern Ocean fronts as defined by Orsi et al. 1995
  - Source: orsifronts
  - URL: https://github.com/AustralianAntarcticDivision/orsifronts
  - License: see orsifronts
- seaice_feb and seaice_oct
  - Description: median October and February sea ice extent
  - Source: Fetterer, F., K. Knowles, W. Meier, M. Savoie, and A. K. Windnagel. 2017, updated daily. Sea Ice Index, Version 3. Boulder, Colorado USA. NSIDC: National Snow and Ice Data Center
  - URL: https://doi.org/10.7265/N5K072F8
  - License: Please cite
- mirounga_leonina
  - Description: Example elephant seal Argos tracking data
  - Source: Data were sourced from the Integrated Marine Observing System (IMOS) - IMOS is supported by the Australian Government through the National Collaborative Research Infrastructure Strategy and the Super Science Initiative
  - URL: https://github.com/ianjonsen/bsam

    - License: Please cite
- ADD_coastline_med
    - Description: Medium-resolution coastline data from the SCAR Antarctic Digital Database. This coastline only covers continental Antarctica: see the GSHHS_i_L1 data for the remainder of the southern hemisphere
    - Source: SCAR
    - URL: https://add.data.bas.ac.uk/repository/entry/show?entryid=f477219b-9121-44d6-afa6-d8552762dc45
    - License: CC-BY. Citation: SCAR Antarctic Digital Database (2018)
- GSHHS_i_L1
    - Description: Coastline data (excluding Antarctica) from the Global Self-consistent, Hierarchical, High-resolution Geography Database. Only southern hemisphere, level 1 (boundary between land and ocean, except Antarctica), intermediate resolution data are included here
    - Source: Wessel P, Smith WHF (1996) A Global Self-consistent, Hierarchical, High-resolution Shoreline Database. J. Geophys. Res. 101:8741-8743
    - URL: http://www.soest.hawaii.edu/wessel/gshhg/
    - License: LGPL
- fronts_park
    - Description: Altimetry-derived Antarctic Circumpolar Current fronts
    - Source: Park Young-Hyang, Durand Isabelle (2019). Altimetry-drived Antarctic Circumpolar Current fronts. SEANOE.
    - URL: https://doi.org/10.17882/59800
    - License: CC-BY-4.0, please cite in full as at DOI

`SOmap_text` *Helper function for labels This is basically a thin wrapper around* `text`*, that passes* `x[[labelcol]]` *to* `text` *as the* `labels` *parameter*

Description

Helper function for labels This is basically a thin wrapper around `text`, that passes `x[[labelcol]]` to `text` as the `labels` parameter

Usage

```
SOmap_text(x, labelcol, ...)
```

Arguments

`x`	data.frame, Spatial data.frame, or sfc: data to pass to `text`
`labelcol`	string: name of the column in `x` to use for text labels
`...`	other plot arguements

Value

as for `text`

See Also

`text`

`SOmerge` *Merge multiple SOmap or related objects*

Description

The inputs must contain exactly one object of class `SOmap`.

Usage

```
SOmerge(..., reproject = TRUE)
```

Arguments

`...`	: one or more objects of class `SOmap`, `SOmap_management`, or `SOmap_legend`, or a list of such objects
`reproject`	logical: if `TRUE` (the default), and any of the input objects are in a different projection to the input `SOmap` object, an attempt will be made to reproject them. If you run into problems with `SOmerge`, try setting this to `FALSE`

Details

Note that objects of class `SOmap_auto` are not yet supported.

Value

A single object of class `SOmap`.

See Also

`SOmap`

Examples

```
## Not run:
  mymap <- SOmap(bathy_legend = "space")
 mylegend <- SOleg(x = runif(100), position = "topright", col = hcl.colors(80, "Viridis"),
                  breaks = c(0.1, 0.2, 0.5, 0.9), trim = -45, label = "Thing",
                  rnd = 1, type = "continuous")
  mymgmt <- SOmanagement(eez = TRUE, basemap = mymap)
  merged <- SOmerge(mymap, mymgmt, mylegend)
  plot(merged)
  ## note that you need to take some care in constructing the component objects
  ##  to ensure their visual consistency

  ## e.g. this will work, but the EEZ layers will extend beyond the map bounds
  mymap <- SOmap(trim = -55)
  mymgmt <- SOmanagement(eez = TRUE, trim = -45) ## note different trim
  plot(SOmerge(mymap, mymgmt))

  ## better to do
  mymap <- SOmap(trim = -55)
  mymgmt <- SOmanagement(eez = TRUE, basemap = mymap)
  plot(SOmerge(mymap, mymgmt))

  ## SOmerge will reproject objects on the fly if needed

 sw_atlantic <- SOmap_auto(c(-70, -20), c(-65, -45), input_points = FALSE, input_lines = FALSE)
  mymap_auto$projection
  ## the EEZs within this region
  sw_atlantic_mgmt <- SOmanagement(eez = TRUE, basemap = sw_atlantic)

  mymap <- SOmap()
  mymap$projection

  ## sw_atlantic_mgmt lies within the bounds of mymap, so we might want to combine them
  ##  even though their projections are different
  merged <- SOmerge(mymap, sw_atlantic_mgmt)
  plot(merged)

## End(Not run)
```

SOplot *Add items to an existing SOmap*

Description

Reproject and add an object to an existing SOmap or SOmap_auto.

Usage

```
SOplot(x, y = NULL, target = NULL, ..., source = NULL, add = TRUE)
```

Arguments

x	: longitude vector, or an object with coordinates
y	: latitude vector, or missing if x is an object
target	: target projection. If not provided, it will default to the projection of the current map, and if that is not set it will use the default SOmap polar stereographic projection

`...`	: other parameters passed to the `plot` function
`source`	: if `x` is not an object with a projection already set, specify its projection here (default = longlat)
`add`	logical: if `TRUE`, add this object to an existing plot

Examples

```
## Not run:
  x <-c (-70, -60,-50, -90)
  y <-c (-50, -75, -45, -60)
  map <- SOmap_auto(x, y, input_lines = FALSE)

  ## plot the map, with the x, y points already added
  map
  ## re-plot the points in a different colour and marker
  SOplot(x = x, y = y, pch = 0, cex = 2, col = 6)

## End(Not run)
```

`SOproj` *Southern projection*

Description

Function for reprojecting data.

Usage

```
SOproj(x, y = NULL, target = NULL, data, ..., source = NULL)
```

Arguments

`x`	longitude vector, or object with coordinates
`y`	latitude vector
`target`	target projection (default = stereo)
`data`	optional data to be included
`...`	arguments passed to `reproj::reproj()`
`source`	starting projection (default = longlat)

Value

Reprojects the given data object to polar projection. Works with Points, spatial, raster, SOmap, sf and sfc objects.

Examples

```
## Not run:
 lat <- c(-70, -60,-50, -90)
 lon <- c(-50, -75, -45, -60)
 pnts <- SOproj(x = lon, y = lat)
 SOmap2(CCAMLR = TRUE)
 plot(pnts, pch = 19, col = 3, add = TRUE)

## End(Not run)
```

`SO_plotter` *Construct a SO_plotter object*

Description

`SOmap` and similar objects contain all of the data and code required to draw a map. This information is embedded in `SO_plotter` objects within the `SOmap` object.

Usage

```
SO_plotter(plotfun, plotargs = NULL, name = NULL)
```

Arguments

`plotfun`	function or string: either the name of a function to use, or the function itself
`plotargs`	list: arguments to pass to the function
`name`	string: optional name for this element

Value

An object of class SO_plotter

See Also

`SOmap`

Examples

```
## Not run:
  p <- SOmap()
  ## replace the `box` element with different plotting code
  p$box <- SO_plotter(plotfun = "graphics::box", plotargs = list(col = "red"))

  ## you can also specify multiple plotting instructions for a single graphical element
  ##  of a map
  p$box <- c(SO_plotter(plotfun = "graphics::box", plotargs = list(col = "red")),
           SO_plotter(plotfun = "graphics::box", plotargs = list(lwd = 2)))

## End(Not run)
```

Index