Package: angstroms (via r-universe)

November 5, 2026

Title Tools for 'ROMS' the Regional Ocean Modeling System

Version 0.0.2.9007

Description Helper functions for working with Regional Ocean Modeling System 'ROMS' output. See <https://www.myroms.org/> for more information about 'ROMS'.

Depends R (>= 3.3.0), raster

License GPL-3

LazyData true

Encoding UTF-8

Imports dplyr, FNN, ncdf4, rbgm, reproj (>= 0.2.0), sp, magrittr, silicate, rlang

RoxygenNote 7.3.2

Suggests bgmfiles, covr, testthat (>= 2.1.0)

Roxygen list(markdown = TRUE)

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

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

Remotes AustralianAntarcticDivision/rbgm

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libnetcdf-dev libproj-dev libsqlite3-dev

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

Date/Publication 2026-10-21 08:19:26 UTC

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

RemoteRef HEAD

RemoteSha ae7327ca12e7dc7580b6504e387187bc66c2939d

Contents

angstroms 2
antarctica 2
coords_points 3

create_transport 3
croproms 5
ice_fake 5
plot_cgrid 6
rawdata 6
romsboundary 7
romscoords 8
romsdata 9
romshcoords 10
romsmap 11
romsrotate 12

Index **14**

`angstroms` *Tools for ROMS model output.*

Description

Facilities for easy access to Regional Ocean Modeling System (ROMS) output.

Details

`coords_points`	produce sp::SpatialPoints from ROMS coordinate arrays
`croproms`	create a raster::extent by cropping a ROMS data layer
`rawdata`	read raw NetCDF variables by name
`romsdata`	read data layers from 4D variables by arbitrary slice
`romsboundary`	generate the spatial boundary of the ROMS data set in x-y co
`romscoords`	extract the x-y (long-lat) coordinates
`romshcoords`	extract the multi-layer 'h'eight grid with S-coordinate stretch
`romsmap`	re-map a spatial layer (polygons, lines, points) into ROMS gr

---

`antarctica` *Antarctica simple coastline.*

Description

Taken from "rnaturalearth::countries110"

coords_points *Create SpatialPoints.*

Description

Convenience wrapper around SpatialPoints for a two layer brick with longitude and latitude coordinate arrays.

Usage

```
coords_points(x, ...)
```

Arguments

x	two layer RasterBrick with longitude and latitude values
...	ignored

Value

SpatialPoints

Examples

```
## library(raadtools)
##coords_points(romscoords(cpolarfiles()$fullname[1]))

pts <- coords_points(ice_coords)
```

create_transport *Create "transport" file for Atlantis.*

Description

Transport is the hydrodynamic forcing for the Atlantis model from oceanographic models. These functions aim to minimize the amount of manual handling of details, creating an NetCDF file that can be modified directly. All the Box Geometry Model details are determined from the bgm file itself. The output file is the precursor file for input to HydroConstruct (as per section 4.1 of the Atlantis User Guide).

Usage

```
create_transport(
  filename,
  model_title = "Transport file [placeholder]",
  bgmfilepath = "",
  bgmlevels = NULL,
  time_steps = NULL,
  transp_params = "",
  overwrite = FALSE
)
```

Arguments

filename	the NetCDF file to create
model_title	name of the model
bgmfilepath	path to BGM file
bgmlevels	the Atlantis depth levels
time_steps	actual time steps, need to be regularly space
transp_params	optional details to put in the NetCDF notes
overwrite	set to TRUE to clobber an existing file

Details

Please use the original BGM file in its native map projection, there is no need to copy this to a longitude-latitude version of the same file.

TODO: check that bgmlevels should only be an index, check that time_steps must be regular ?(or just monotonic incresing?)

Value

the filename of the output (use ncdf4 to inspect, modify it)

Examples

```
tfile <- file.path(tempdir(), "my_hydrofile.nc")
(ncfile <- create_transport(tfile, bgmfilepath = bgmfiles::bgmfiles("antarctica_99")))
## use ncdf4::nc_open to inspect the file
```

croproms *Crop a ROMS layer*

Description

Crop a ROMS data layer from `romsdata` with a raster extent.

Usage

```
croproms(x, ext, ...)
```

Arguments

`x`	ROMS xy- coordinates, see `romscoords`
`ext`	`raster::extent` in the coordinate system of `x`
`...`	ignored

Details

The spatial crop is performed in the coordinate space of roms data.

Examples

```
## notice that extent is in long-lat, but ice_local is in the grid
## space of ice_coords
ice_local <- croproms(ice_coords, extent(100, 120, -75, -60))
plot(ice_coords[[2]], col = grey(seq(0, 1, length  = 20)))
plot(crop(ice_fake, ice_local), add = TRUE)
```

ice_fake *Fake model data.*

Description

`ice_coords` and `ice_fake` are generated from a projected map of southern Ocean sea ice data.

Details

The coords layer is the longitude and latitude values for the centres of the polar cells. This is veyr loosely analogous to the coordinate arrays used by ROMS data, included here for working examples, illustration and code tests.

The proper metadata for these layers is "-3950000, 3950000, -3950000, 4350000 (xmin, xmax, ymin, ymax)"

"+proj=stere +lat_0=-90 +lat_ts=-70 +lon_0=0 +k=1 +x_0=0 +y_0=0 +a=6378273 +b=6356889.449 +units=m +no_defs"

plot_cgrid *https://www.myroms.org/wiki/easygrid The rho-grid represents the grid centers (red dots) The u-grid represents the grid East-West sides (blue triangles) The v-grid represents the grid North-South sides (green triangles) The psi-grid represents the grid corners (purple crosses) A useable Thredds example is here https://ecowatch.ncddc.noaa.gov/thredds/dodsC/northwest_fisheries/roms/nep_c.nc*

Description

https://www.myroms.org/wiki/easygrid The rho-grid represents the grid centers (red dots) The u-grid represents the grid East-West sides (blue triangles) The v-grid represents the grid North-South sides (green triangles) The psi-grid represents the grid corners (purple crosses) A useable Thredds example is here https://ecowatch.ncddc.noaa.gov/thredds/dodsC/northwest_fisheries/roms/nep_c.nc

Usage

```
plot_cgrid(
  x,
  ex = extent(0, 15, 0, 20),
  include = c("u", "v", "rho", "psi"),
  cell = TRUE,
  ...
)
```

Arguments

x	ROMS file
ex	extent in index space
include	grid elements to plot (all by default, uv, rho, psi)
cell	draw the cells (defaults to TRUE)
...	arguments passed to plot

rawdata *Read the variable as is*

Description

Read the variable as is

Usage

```
rawdata(x, varname, ..., native = FALSE)

## S3 method for class 'character'
rawdata(x, varname, ..., native = FALSE)

## S3 method for class 'NetCDF'
rawdata(x, varname, ..., native = FALSE)
```

Arguments

x	netcdf file path
varname	variable name
...	dots (ignored)
native	maintain values in internal storge mode (default is FALSE)

---

romsboundary *Boundary polygon from raster of coordinates.*

Description

Create a boundary polygon by tracking around coordinates stored in a RasterStack.

Usage

```
romsboundary(cds)

databoundary(x, mask = NULL, ...)
```

Arguments

cds	two-layer Raster
x	a thing
mask	logical
...	dots

Details

The first layer in the stack is treated as the X coordinate, second as Y.

Examples

```
ice_grid_boundary <- romsboundary(ice_coords)
plot(antarctica)
## does not make sense in this space
plot(ice_grid_boundary, add = TRUE, border = "grey")

## ok in this one
#library(rgdal)
#   proj4string(ice_grid_boundary) <- CRS("+init=epsg:4326")
# pweird <- "+proj=laea +lon_0=147 +lat_0=-42 +ellps=WGS84"
#  laea_world <- spTransform(antarctica, pweird)
#  plot(extent(laea_world) + 8e6, type = "n", asp = 1)
#  plot(laea_world, add = TRUE)
#  plot(spTransform(ice_grid_boundary, pweird), add  = TRUE, border = "darkgrey")
```

`romscoords` *Extract coordinate arrays from ROMS.*

Description

Returns a RasterStack of the given variable names.

Usage

```
romscoords(
  x,
  spatial = c("lon_u", "lat_u"),
  ncdf = TRUE,
  transpose = TRUE,
  ...,
  varname = "",
  flip_y = TRUE
)
```

Arguments

`x`	ROMS file name
`spatial`	names of coordinate variables (e.g. lon_u, lat_u)
`ncdf`	default to NetCDF no matter what file name
`transpose`	the extents (ROMS is FALSE, Access is TRUE)
`...`	unused
`varname`	in desperate cases, specify the variable that these coordinate variables belong to
`flip_y`	Y coordinates are assumed to be in top-down order, set to FALSE to assume down-up

Details

The two layers from the model output are used to define the real-world space. This is used to create a boundary `romsboundary`, to map real-world objects into grid space `romscoords` and to generate graticules for mapping into the grid space with `graphics::contour`.

Value

RasterStack with two layers of the 2D-variables

Examples

```
## Not run:
  coord <- romscoord("roms.nc")

## End(Not run)
## with in-built fake data
plot(ice_fake, asp = 0.5)
contour(ice_coords[[1]], add = TRUE, levels = seq(-165, 165, by = 15))
contour(ice_coords[[2]], add = TRUE)
```

`romsdata` *ROMS single slice 2D layer Extract a data layer from ROMS by name and 4-D slice.*

Description

`romsdata` always works in the first two dimensions (x-y), the more specialist functions will work in the space indicated by their name `roms_xy`, `roms_xt` and so on.

Usage

```
roms_xy(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)

roms_xz(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)

roms_xt(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)

roms_yz(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)

roms_yt(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)

roms_zt(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)

romsdata(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)

romsdata2d(x, varname = "", slice = c(1L, 1L), transpose = TRUE, ...)
```

```
romsdata3d(
  x,
  varname = "",
  slice = 1L,
  transpose = TRUE,
  verbose = TRUE,
  ...,
  lvar = 4L
)
```

Arguments

x	ROMS file name
varname	name of ROMS variable
slice	index in w and t (depth and time), defaults to first encountered
transpose	the extents (ROMS is FALSE, Access is TRUE)
...	unused
verbose	be chatty
lvar	passed to `raster::brick` to specify 3rd or 4th dimension

Value

RasterLayer

Examples

```
#x <- raadtools:::cpolarfiles()$fullname[1]
#plot(roms_xy(x, "u"))
#plot(roms_xz(x, "u", slice = c(392L,1L)), asp = NA)
#plot(roms_xt(x, "u", slice = c(392L,1L)), asp = NA)

#plot(roms_yz(x, "u"))
#plot(roms_yt(x, "u", slice = c(1L,1L)), asp = NA)
#plot(roms_zt(x, "u", slice = c(1L, 392L)), asp = NA)
```

---

romshcoords *Coordinates at depth*

Description

Extract the multi-layer 'h'eight grid with S-coordinate stretching applied

Usage

```
romshcoords(x, grid_type = "rho", slice, ..., depth = "h", simple = FALSE)
```

Arguments

x	ROMS file name
grid_type	string: "rho","psi","u","v","w"
slice	integer: if non-missing, use this time slice to index into zeta (free-surface). Otherwise assume zeta is zero (and hence depth is time-independent)
...	dots
depth	string: the name of the appropriate variable to use for depth
simple	logical: if TRUE, use the old "simple" depth method, which may not be correct

Details

Compute ROMS grid depth from vertical stretched variables Given a bathymetry (h), free-surface (zeta) and terrain-following parameters, this function computes the 3D depths for the requested C-grid location. If the free-surface is not provided, a zero value is assumed resulting in unperturb depths. This function can be used when generating initial conditions or climatology data for an application. Check the following link for details: https://www.myroms.org/wiki/index.php/Vertical_S-coordinate See https://github.com/dcherian/tools/blob/master/ROMS/arango/utility/set_depth.m Original Matlab code by Hernan Arango.

Value

RasterStack with a layer for every depth

---

romsmap *Remap an object to the ROMS grid.*

Description

Find the nearest-neighbour coordinates of x in the coordinate arrays of coords.

Usage

```
romsmap(x, ...)

## S3 method for class 'SpatialPolygonsDataFrame'
romsmap(x, coords, crop = FALSE, lonlat = TRUE, ...)

## Default S3 method:
romsmap(x, coords, crop = FALSE, lonlat = TRUE, ...)

## S3 method for class 'SpatialLinesDataFrame'
romsmap(x, coords, crop = FALSE, lonlat = TRUE, ...)

## S3 method for class 'SpatialPointsDataFrame'
romsmap(x, coords, crop = FALSE, lonlat = TRUE, ...)
```

Arguments

x	object to transform to the grid space, e.g. a Spatial object
...	unused
coords	romscoords RasterStack
crop	logical, if TRUE crop x to the extent of the boundary of the values in coords
lonlat	logical, if TRUE check for need to back-transform to longitude/latitude and do it

Details

The input coords is a assumed to be a 2-layer RasterStack or RasterBrick and using FNN::get.knnx() the nearest matching position of the coordinates of x is found in the grid space of coords. The motivating use-case is the curvilinear longitude and latitude arrays of ROMS model output.

No account is made for the details of a ROMS cell, though this may be included in future. We tested only with the "lon_u" and "lat_u" arrays.

Value

input object with coordinates transformed to space of the coords

Note

Do not use this for extraction purposes without checking the output, this is best used for exploration and visualization. Re-mapping ROMS data is better done by looking up the coords_points within spatial objects, and transferring via the grid index.

Examples

```
library(raster)
ant_ice_coords <- romsmap(antarctica, ice_coords)
raster::image(ice_fake, col = c("transparent", hcl.colors(24)), asp = 1)
sp::plot(ant_ice_coords, add = TRUE)
```

---

romsrotate *Rotate 'ROMS' vectors*

Description

romsrotate performs the rotation on a 2-layer brick romsangle reads the (likely) angle variable

Usage

```
romsrotate(uv, angle, ...)

romsangle(x, varname = "angle", ...)
```

Arguments

uv	raster brick of two layers with 'u' ("east-west") and 'v' ("north-south") vector components
angle	raster layer of the 'angle' variable from 'ROMS'
...	ignored currently
x	filename
varname	variable name from file 'x'

References

ROMS website

Index

angstroms, 2
antarctica, 2

coords_points, *2*, 3
create_transport, 3
croproms, *2*, 5

databoundary *(*romsboundary*)*, 7

FNN::get.knnx(), *12*

ice_coords *(*ice_fake*)*, 5
ice_fake, 5

plot_cgrid, 6

raster::extent, *2*
rawdata, *2*, 6
roms_xt *(*romsdata*)*, 9
roms_xy *(*romsdata*)*, 9
roms_xz *(*romsdata*)*, 9
roms_yt *(*romsdata*)*, 9
roms_yz *(*romsdata*)*, 9
roms_zt *(*romsdata*)*, 9
romsangle *(*romsrotate*)*, 12
romsboundary, *2*, 7
romscoords, *2*, 8
romsdata, *2*, 9
romsdata2d *(*romsdata*)*, 9
romsdata3d *(*romsdata*)*, 9
romshcoords, *2*, 10
romsmap, *2*, 11
romsrotate, 12

sp::SpatialPoints, *2*
Spatial, *12*