Package: rerddapUtils (via r-universe)

October 29, 2026

Title Miscellaneous Utilities for 'rerddap'

Version 1.0.3

Date 2026-10-28

Description The 'rerddapUtils' package is an 'R' package that is a set of four main functions designed to work with and extend the 'rerddap' package. These functions includes one for restricting by season, one for splitting large requests, and two for working with projected datasets. There are also two utility functions that provide estimates of the size of a proposed 'rerddap::griddap()' request.

License CC0

Encoding UTF-8

Depends R(>= 4.4.0)

Imports cli, DBI, duckdb, lubridate, ncdf4, rerddap, sf, stats, stringr,

Suggests arrow, dplyr, duckplyr, httr, knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

LazyData true

NeedsCompilation no

Author Roy Mendelssohn [aut, cre]

Maintainer Roy Mendelssohn <roy.mendelssohn@noaa.gov>

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

Date/Publication 2026-10-29 21:15:22 UTC

RemoteUrl https://github.com/cran/rerddapUtils

RemoteRef HEAD

RemoteSha e825c47c4356ecbafb547cec2abc0f72920e5947

Contents

estimate_griddap_size 2
estimate_griddap_split_size 4
griddap_season 5
griddap_split 6
iceInfo 8
latlon_to_xy 9
proj_extract 10
wind_info 10
xy_to_latlon 11

Index **13**

`estimate_griddap_size` *Estimate the size of a rerddap::griddap() request*

Description

Uses coordinate metadata from an ERDDAP info() object to estimate how many grid cells will be returned and the total uncompressed byte count for each requested data variable. No network request is made.

Usage

```
estimate_griddap_size(
  info,
  ...,
  fields = "all",
  stride = 1L,
  spacing = list(),
  verbose = TRUE
)
```

Arguments

`info`	An object returned by rerddap::info().
`...`	Named dimension constraints, one per coordinate variable to constrain. Names must exactly match the coordinate variable names in the dataset (as returned by dimvars(info)). Each value is c(min, max); for time use ISO 8601 strings.
`fields`	Character vector of data variable names, or "all" (default).
`stride`	Integer scalar or named list of per-dimension stride values, using the same convention as griddap(). Default 1.
`spacing`	Optional named list to override auto-detected spacing for one or more dimensions. For time, supply seconds as time_sec (e.g. spacing = list(time_sec = 86400)). For other dimensions use the coordinate variable name (e.g. spacing = list(latitude = 0.01, xi_rho = 1)).
`verbose`	Logical; print a formatted summary (default TRUE).

Details

Coordinate dimension names are taken directly from the info object using the same logic as rerddap:::dimvars() — the set-difference between all keys in info$alldata and the data variable names plus "NC_GLOBAL". This means any coordinate system works: geographic lat/lon, projected x/y, sigma-layer depth, ROMS xi_rho/eta_rho, etc.

Dimension constraints are passed via ... using the exact coordinate variable names reported by the dataset, the same way griddap() accepts them. Each constraint is a numeric (or character for time) vector of length 2: c(min, max).

Spacing is resolved in this order for each dimension:

1. User-supplied override in the spacing argument.
2. For time: (t_max - t_min) / (nValues - 1) derived from the dimension's nValues row and actual_range attribute. This correctly handles running composites where time steps are daily even though the composite window is e.g. 8 days.
3. NC_GLOBAL attributes: geospatial_lat_resolution, geospatial_lon_resolution, time_coverage_resolution (ISO 8601).
4. Coordinate variable attributes: point_spacing, resolution, spacing.
5. For time: averageSpacing string from the nValues row as a last resort.
6. If the constraint min == max the dimension contributes 1 point.
7. Otherwise: NA — a warning is issued and 1 point is assumed.

Value

Invisibly, a named list containing per-dimension point counts, spacing values, per-variable byte estimates, and total bytes.

Examples

```
## Not run:
  library(rerddap)

  myURL <- "https://coastwatch.pfeg.noaa.gov/erddap/"
  response <- try(httr::HEAD(myURL, httr::timeout(10)), silent = TRUE)
  if (inherits(response, "try-error")) {
     stop("The ERDDAP\u2122 server is not responding")
  }
  info <- rerddap::info("erdMH1chla8day", url = myURL)
  estimate_griddap_size(info,
    latitude  = c(30, 50),
    longitude = c(-140, -110),
    time      = c("2020-01-01", "2020-12-31"))

## End(Not run)
```

`estimate_griddap_split_size` *Estimate the size of a single split in a rerddap::griddap() request*

Description

Takes the result of estimate_griddap_size() together with a named list of split counts per dimension and reports the estimated uncompressed size of one split. The total number of splits is the product of all split counts (e.g. splits = list(time = 5, latitude = 2, longitude = 2) means 20 total splits). Per-split point counts use ceiling(n_pts / split_count) so uneven divisions are handled correctly and estimates are conservative.

Usage

```
estimate_griddap_split_size(size_est, splits, verbose = TRUE)
```

Arguments

`size_est`	The list returned invisibly by estimate_griddap_size().
`splits`	Named list of split counts per dimension. Dimensions not listed are assumed to have a split count of 1 (no split).
`verbose`	Logical; print a summary (default TRUE).

Value

Invisibly, a named list with the total split count, per-split cell count, per-variable byte estimates, total per-split bytes, and a formatted size string.

Examples

```
## Not run:
  library(rerddap)

  myURL = 'https://coastwatch.pfeg.noaa.gov/erddap/'
  response <- try(httr::HEAD(myURL, httr::timeout(10)), silent = TRUE)
  if (inherits(response, "try-error")) {
     stop("The ERDDAP\u2122 server is not responding")
  }
  wind_info <- info("erdQMekm14day", url = myURL)
  sz <- estimate_griddap_size(wind_info,
    latitude  = c(20, 40),
    longitude = c(-140, -110),
    time      = c("2015-01-01", "2016-01-01"),
    fields    = "mod_current")

  estimate_griddap_split_size(sz,
    splits = list(time = 5, altitude = 1, latitude = 2, longitude = 2))

## End(Not run)
```

griddap_season *Title Get ERDDAP gridded data restricted to a given season of the year*

Description

griddap_season uses the R program 'rerddap::griddap()' to extract environmental data from an 'ERDDAP' server in an (time, z, y ,x) bounding box where time is restricted to a given season of the year (see below). Arguments are the same in 'rerddap::griddap()' except for the added 'season' parameter. 'read' and 'fmt' options are ignored.

Usage

```
griddap_season(
  datasetx,
  ...,
  fields = "all",
  stride = 1,
  season = NULL,
  fmt = "nc",
  url = rerddap::eurl(),
  store = rerddap::disk(),
  read = TRUE,
  callopts = list()
)
```

Arguments

datasetx	Anything coercable to an object of class info. So the output of a call to info, or a datasetid, which will internally be passed through info
...	Dimension arguments. See examples. Can be any 1 or more of the dimensions for the particular dataset - and the dimensions vary by dataset. For each dimension, pass in a vector of length two, with min and max value desired. at least 1 required.
fields	(character) Fields to return, in a character vector.
stride	(integer) How many values to get. 1 = get every value, 2 = get every other value, etc. Default: 1 (i.e., get every value)
season	(character) a character array with the times of the trajectory in the form c("MM-DD", "MM-DD")
fmt	(character) ignored
url	A URL for an ERDDAP server. Default: https://upwell.pfeg.noaa.gov/erddap/ - See rerddap::eurl() for more information
store	ignored
read	ignored
callopts	Curl options passed on to verb-GET

Value

An object of class griddap_csv if csv chosen or griddap_nc if nc file format chosen.

Examples

```
myURL <- "https://coastwatch.pfeg.noaa.gov/erddap/"
response <- try(httr::HEAD(myURL, httr::timeout(5)), silent = TRUE)
if (inherits(response, "try-error")) {
   message("The ERDDAP\u2122 server is not responding")
} else
{
data(wind_info)
season <- c('03-01', '03-04')
season_extract <- try(griddap_season(wind_info,
                                  time = c('2015-01-01','2016-01-01'),
                                  latitude = c(20, 21),
                                  longitude = c(220, 221),
                                  fields = 'mod_current',
                                  season = season
                                  ), silent = TRUE)
if (inherits(season_extract, "try-error")) {
   message("Unable to retrieve data from the ERDDAP\u2122 server")
   }
 }
```

---

griddap_split *Title Get ERDDAP gridded data restricted to a given season of the year*

Description

griddap_season uses the R program 'rerddap::griddap()' to extract environmental data from an 'ERDDAP' server in an (time, z, y ,x) bounding box where time is restricted to a given season of the year (see below). Arguments are the same as in 'rerddap::griddap()' except for the added 'season' parameter. 'read' and 'fmt' options are ignored.

Usage

```
griddap_split(
  datasetx,
  ...,
  fields = "all",
  stride = 1,
  request_split = NULL,
  fmt = "nc",
  url = rerddap::eurl(),
  store = rerddap::disk(),
  read = TRUE,
```

```
  callopts = list(),
  aggregate_file = NULL
)
```

Arguments

datasetx	Anything coercable to an object of class info. So the output of a call to info, or a datasetid, which will internally be passed through info
...	Dimension arguments. See examples. Can be any 1 or more of the dimensions for the particular dataset - and the dimensions vary by dataset. For each dimension, pass in a vector of length two, with min and max value desired. at least 1 required.
fields	(character) Fields to return, in a character vector.
stride	(integer) How many values to get. 1 = get every value, 2 = get every other value, etc. Default: 1 (i.e., get every value)
request_split	A numeric vector indicating the number of splits for each dimension, used to segment the request into manageable chunks. This is particularly useful for large datasets.
fmt	(character) One of: • nc: save output to a netcdf file • memory: save output in a dataframe in memory • duckdb: save data in a duckdb database • parquet: save data as a partitioned parquet dataset in a directory
url	A URL for an ERDDAP server. Default: https://upwell.pfeg.noaa.gov/erddap/ - See rerddap::eurl() for more information
store	ignored
read	ignored
callopts	Curl options passed on to verb-GET
aggregate_file	A string specifying: • if is.null(aggregate_file) then a temporary file is created that does not overwrite user space • if "fmt = nc" the path to the NetCDF file to store the results • if "fmt = duckdb" the path to the duckdb file to store the results • if "fmt = memory" the value is ignored

Value

Varies by format requested:

- if "fmt = nc" the path to the NetCDF file where the results are stored
- if "fmt = duckdb" the path to the duckdb file where the results are stored
- if "fmt = memory" the usual rerddap::griddap dataframe

Examples

```
myURL <- "https://coastwatch.pfeg.noaa.gov/erddap/"
response <- try(httr::HEAD(myURL, httr::timeout(20)), silent = TRUE)
if (inherits(response, "try-error")) {
  message("The ERDDAP\u2122 server is not responding")
} else
{
data(wind_info)
request_split <- list(time = 2, altitude = 1, latitude = 1, longitude = 1)
res <- try(griddap_split(wind_info,
                     time = c('2015-12-31','2016-01-01'),
                     latitude = c(20, 21),
                     longitude = c(220, 221),
                     fields = 'mod_current',
                     request_split = request_split
                     ), silent = TRUE)
if (inherits(res, "try-error")) {
   message("Unable to retrieve data from the ERDDAP\u2122 server")
   }
}
```

`iceInfo` *Projected ice dataset info*

Description

`rerddap::info()` call to Projected ice dataset

Usage

```
iceInfo
```

Format

A list of class `info` with elements:

variables a data frame with a list of the variables in the extraction

alldata metadata summary

base_url a character string givng the base URL of the extract

Source

Data originally accessed from the NOAA CoastWatch ERDDAP server: https://coastwatch.noaa.gov/erddap/

`latlon_to_xy` *Convert Latitude-Longitude to Projected Coordinates*

Description

This function converts geographic coordinates (latitude and longitude) into projected coordinates based on a specified Coordinate Reference System (CRS). The function can automatically detect the CRS from provided metadata (`dataInfo`) or use a specified CRS code. It is designed to work with datasets obtained from `rerddap::griddap()` but can be used with any geographic data that requires coordinate transformation.

Usage

```
latlon_to_xy(
  dataInfo,
  longitude,
  latitude,
  xName = "rows",
  yName = "cols",
  crs = NULL
)
```

Arguments

`dataInfo`	Metadata object containing CRS information, typically obtained from a `rerddap::info()` call on a dataset. It should include global attributes that contain CRS information.
`longitude`	Numeric vector of longitudes to be converted.
`latitude`	Numeric vector of latitudes to be converted.
`xName`	Name of the longitude coordinate in the output projection. Defaults to 'longitude'.
`yName`	Name of the latitude coordinate in the output projection. Defaults to 'latitude'.
`crs`	Optional. A character string specifying the CRS to use for the projection. This can be an EPSG code (e.g., 'EPSG:4326' for WGS84) or a PROJ string. If NULL, the function attempts to detect the CRS from `dataInfo`.

Value

A matrix with columns corresponding to the projected x and y coordinates (in the order specified by `xName` and `yName`). Each row in the matrix corresponds to a pair of input latitude and longitude values.

Examples

```
# myURL <- 'https://coastwatch.noaa.gov/erddap/'
# iceInfo <- rerddap::info('noaacwVIIRSn20icethickNP06Daily', url = myURL)
data(iceInfo)
latitude <- c( 80., 85.)
longitude <- c(-170., -165)
coords <- latlon_to_xy(iceInfo,  longitude, latitude)
```

---

proj_extract *Projected ice dataset response*

Description

`rerddap::griddap()` call to Projected ice dataset

Usage

```
proj_extract
```

Format

A list of class `griddap_nc` with elements:

data a data frame of the extracted ice values

summary metadata summary

Source

Data originally accessed from the NOAA CoastWatch ERDDAP server: https://coastwatch.noaa.gov/erddap/

---

wind_info *Projected 14 day wind info*

Description

`rerddap::info()` call to 14 day wind dataset

Usage

```
wind_info
```

Format

A list of class `info` with elements:

variables a data frame with a list of the variables in the extraction

alldata metadata summary

base_url a character string givng the base URL of the extract

Source

Data originally accessed from the NOAA CoastWatch ERDDAP server: https://coastwatch.noaa.gov/erddap/

`xy_to_latlon` *Convert Projected 'rerddap::griddap()' Coordinates to Latitude-Longitude*

Description

This function converts the projected coordinates from a 'rerddap::griddap()' response into geographical coordinates (latitude and longitude). It supports responses from `griddap`, `rxtracto`, and `rxtracto3D` by detecting the response type and applying the appropriate Coordinate Reference System (CRS) transformation.

Usage

```
xy_to_latlon(resp, yName = "cols", xName = "rows", crs = NULL)
```

Arguments

`resp`	A response object from a call to `rerddap::griddap()`. It is expected to be of type `griddap_nc`, `rxtracto3D`, or `rxtractoTrack`, which contains the projected coordinate data along with metadata including the dataset's CRS.
`yName`	The name of the variable in `resp` that represents the Y coordinate (typically latitude or northing). Defaults to 'cols'.
`xName`	The name of the variable in `resp` that represents the X coordinate (typically longitude or easting). Defaults to 'rows'.
`crs`	An optional CRS code to be used for the transformation. If a CRS is found in the `resp` metadata this is ignored. CRS code should be a valid EPSG code (e.g., `4326` for WGS84) or a PROJ string.

Value

A matrix with two columns (`longitude`, `latitude`) containing the geographic coordinates corresponding to the projected coordinates in the input `resp`. Each row in the matrix corresponds to a point in `resp`.

Examples

```
rows <- c(300112.5, 250027.5)
cols <- c(-400282.5, -350197.5)
# myURL <- 'https://coastwatch.noaa.gov/erddap/'
# iceInfo <- rerddap::info('noaacwVIIRSn20icethickNP06Daily', url devtoo= myURL)
# proj_extract <- rerddap::griddap(iceInfo,
#                            time = c('2026-05-18T12:28:57Z', '2026-05-18T12:28:57Z'),
#                             rows = rows,
#                              cols = cols,
#                              altitude = c(0., 0.),
#                              fields = 'IceThickness',
#                             url = myURL
#
data(proj_extract)
crs = "EPSG:3413"
test <- xy_to_latlon(proj_extract, crs = crs)
```

Index

∗ **datasets**
    iceInfo, 8
    proj_extract, 10
    wind_info, 10

estimate_griddap_size, 2
estimate_griddap_split_size, 4

griddap_season, 5
griddap_split, 6

iceInfo, 8
info, *5*, *7*

latlon_to_xy, 9

proj_extract, 10

rerddap::eurl(), *5*, *7*

wind_info, 10

xy_to_latlon, 11