Package 'rerddap'

August 29, 2016

Title General Purpose Client for 'ERDDAP' Servers

Description General purpose R client for 'ERDDAP' servers. Includes functions to search for 'datasets', get summary information on 'datasets', and fetch 'datasets', in either 'csv' or 'netCDF' format. 'ERDDAP' information: http://upwell.pfeg.noaa.gov/erddap/information.html.

Version 0.3.4

License MIT + file LICENSE

URL https://github.com/ropensci/rerddap

BugReports http://www.github.com/ropensci/rerddap/issues

LazyData true

VignetteBuilder knitr

Imports methods, utils, stats, httr (>= 1.0.0), dplyr, data.table, jsonlite (>= 0.9.17), xml2, digest, ncdf4

Suggests roxygen2, knitr, testthat

Enhances taxize

RoxygenNote 5.0.1

NeedsCompilation no

Author Scott Chamberlain [aut, cre]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2016-01-14 22:50:52

R topics documented:

`rerddap-package` *General purpose R client for ERDDAP servers*

Description

General purpose R client for ERDDAP servers

ERDDAP info

NOAA's ERDDAP service holds many datasets of interest. It's built on top of OPenDAP `http://www.opendap.org/`. You can search for datasets via `ed_search`, list datasets via `ed_datasets`, get information on a single dataset via `info`, then get data you want for either tabledap type via `tabledap`, or for griddap type via `griddap`.

tabledap/griddap

tabledap and griddap have different interfaces to query for data, so `tabledap` and `griddap` are separated out as separate functions even though some of the internals are the same. In particular, with tabledap you can query on/subset all variables, whereas with gridddap, you can only query on/subset the dimension varibles (e.g., latitude, longitude, altitude).

NOTE

With griddap data via `griddap` you can get a lot of data quickly. Try small searches of a dataset to start to get a sense for the data, then you can increase the amount of data you get. See `griddap` for more details.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

`cache_delete` *Delete cached files*

Description

Delete cached files

Usage

```
cache_delete(x, cache_path = "~/.rerddap", force = FALSE)

cache_delete_all(cache_path = "~/.rerddap", force = FALSE)
```

Arguments

`x`	File names
`cache_path`	path to cached files
`force`	(logical) Should files be force deleted? Default: `FALSE`

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/index.html

See Also

`cache_list`, `cache_details`

Examples

```
## Not run:
# delete files by name in cache
# cache_delete('9911750294a039b8b517c8bf288978ea.csv')
# cache_delete(c('9911750294a039b8b517c8bf288978ea.csv',
#                'b26825b6737da13d6a52c28c8dfe690f.csv'))

# You can delete from the output of griddap or tabledap fxns
## tabledap
(table_res <- tabledap('erdCalCOFIfshsiz'))
cache_delete(table_res)

## griddap
```

```
(grid_res <- griddap('noaa_esrl_027d_0fb5_5d38',
 time = c('2012-01-01','2012-06-12'),
 latitude = c(21, 18),
 longitude = c(-80, -75)
))
cache_delete(grid_res)

## End(Not run)
```

---

cache_details *Get details of cached files*

Description

Get details of cached files

Usage

```
cache_details(x, cache_path = "~/.rerddap")
```

Arguments

x	File names
cache_path	path to cached files

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/index.html

See Also

cache_list, cache_delete

Examples

```
## Not run:
# List details for all cached files
cache_details()

# List details for specific files
(x <- cache_list())
cache_details(x$nc[1])
cache_details(x$csv[1])

# For a list or character vector of files
```

```
ff <- cache_list()[[1]]
cache_details(ff[1:3])
cache_details(as.list(ff[1:3]))

# List details from output of griddap or tabledap
## tabledap
(table_res <- tabledap('erdCalCOFIfshsiz'))
cache_details(table_res)

## griddap
(grid_res <- griddap('noaa_esrl_027d_0fb5_5d38',
 time = c('2012-01-01','2012-06-12'),
 latitude = c(21, 18),
 longitude = c(-80, -75)
))
cache_details(grid_res)

## End(Not run)
```

---

cache_list *List cached files*

Description

List cached files

Usage

```
cache_list(cache_path = "~/.rerddap")
```

Arguments

cache_path	path to cached files

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/index.html

See Also

cache_delete, cache_details

Examples

```
## Not run:
# list files in cache
(x <- cache_list())

# List info for files
cache_details(x$nc[1])
cache_details(x$csv[1])
cache_details()

# delete files by name in cache
# cache_delete(x$nc[1])
# cache_delete(x$nc[2:3])

## End(Not run)
```

`convert_time` *Convert a UDUNITS compatible time to ISO time*

Description

Convert a UDUNITS compatible time to ISO time

Usage

```
convert_time(n = NULL, isoTime = NULL,
  units = "seconds since 1970-01-01T00:00:00Z",
  url = "http://coastwatch.pfeg.noaa.gov", method = "local", ...)
```

Arguments

`n`	numeric; A unix time number.
`isoTime`	character; A string time representation.
`units`	character; Units to return. Default: "seconds since 1970-01-01T00:00:00Z"
`url`	Base URL of the ERDDAP server
`method`	(character) One of local or web. Local simply uses `as.POSIXct`, while web method uses the ERDDAP time conversion service `/erddap/convert/time.txt`
`...`	Curl args passed on to `GET`

Details

When `method = "web"` time zone is GMT/UTC

Examples

```
 ## Not run:
# local conversions
convert_time(n = 473472000)
convert_time(isoTime = "1985-01-02T00:00:00Z")

# using an erddap web service
convert_time(n = 473472000, method = "web")
convert_time(isoTime = "1985-01-02T00:00:00Z", method = "web")

## End(Not run)
```

`convert_units` *Convert a CF Standard Name to/from a GCMD Science Keyword*

Description

Convert a CF Standard Name to/from a GCMD Science Keyword

Usage

```
convert_units(udunits = NULL, ucum = NULL,
  url = "http://coastwatch.pfeg.noaa.gov", ...)
```

Arguments

`udunits`	character; A UDUNITS character string http://www.unidata.ucar.edu/software/udunits/
`ucum`	character; A UCUM character string http://unitsofmeasure.org/ucum.html
`url`	Base URL of the ERDDAP server
`...`	Curl args passed on to GET

Examples

```
 ## Not run:
convert_units(udunits = "degree_C meter-1")
convert_units(ucum = "Cel.m-1")

## End(Not run)
```

disk *Options for saving ERDDAP datasets.*

Description

Options for saving ERDDAP datasets.

Usage

```
disk(path = "~/.rerddap", overwrite = TRUE)

memory()
```

Arguments

path	Path to store files in. A directory, not a file. Default: ~/.rerddap
overwrite	(logical) Overwrite an existing file of the same name? Default: TRUE

ed_search *Search for ERDDAP tabledep or griddap datasets*

Description

Search for ERDDAP tabledep or griddap datasets

Usage

```
ed_search(query, page = NULL, page_size = NULL, which = "griddap",
  url = eurl(), ...)

ed_datasets(which = "tabledap", url = eurl())
```

Arguments

query	(character) Search terms
page	(integer) Page number
page_size	(integer) Results per page
which	(character) One of tabledep or griddap.
url	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/
...	Further args passed on to GET (must be a named parameter)

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/index.html

Examples

```
## Not run:
(out <- ed_search(query='temperature'))
out$alldata[[1]]
(out <- ed_search(query='size'))
out$info

# List datasets
head( ed_datasets('table') )
head( ed_datasets('grid') )

# use a different ERDDAP server
## Marine Institute (Ireland)
ed_search("temperature", url = "http://erddap.marine.ie/erddap/")
## Marine Domain Awareness (MDA) (Italy)
ed_search("temperature", url = "https://bluehub.jrc.ec.europa.eu/erddap/")

## End(Not run)
```

---

ed_search_adv *Advanced search for ERDDAP tabledep or griddap datasets*

Description

Advanced search for ERDDAP tabledep or griddap datasets

Usage

```
ed_search_adv(query = NULL, page = 1, page_size = 1000, protocol = NULL,
  cdm_data_type = NULL, institution = NULL, ioos_category = NULL,
  keywords = NULL, long_name = NULL, standard_name = NULL,
  variableName = NULL, maxLat = NULL, minLon = NULL, maxLon = NULL,
  minLat = NULL, minTime = NULL, maxTime = NULL, url = eurl(), ...)
```

Arguments

query	(character) Search terms
page	(integer) Page number. Default: 1
page_size	(integer) Results per page: Default: 1000
protocol	(character) One of any (default), tabledep or griddap

cdm_data_type	(character) One of grid, other, point, profile, timeseries, timeseriesprofile, trajectory, trajectoryprofile
institution	(character) An institution. See the dataset institutions.
ioos_category	(character) An ioos category See the dataset ioos_categories.
keywords	(character) A keywords. See the dataset keywords.
long_name	(character) A long name. See the dataset longnames.
standard_name	(character) A standar dname. See the dataset standardnames.
variableName	(character) A variable name. See the dataset variablenames.
minLon, maxLon	(numeric) Minimum and maximum longitude. Some datasets have longitude values within -180 to 180, others use 0 to 360. If you specify min and max Longitude within -180 to 180 (or 0 to 360), ERDDAP will only find datasets that match the values you specify. Consider doing one search: longitude -180 to 360, or two searches: longitude -180 to 180, and 0 to 360.
minLat, maxLat	(numeric) Minimum and maximum latitude, between -90 and 90
minTime, maxTime	(numeric/character) Minimum and maximum time. Time string with the format "yyyy-MM-ddTHH:mm:ssZ, (e.g., 2009-01-21T23:00:00Z). If you specify something, you must include at least yyyy-MM-dd; you can omit Z, :ss, :mm, :HH, and T. Always use UTC (GMT/Zulu) time. Or specify the number of seconds since 1970-01-01T00:00:00Z.
url	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/
...	Further args passed on to GET (must be a named parameter)

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/index.html

Examples

```
## Not run:
ed_search_adv(query = 'temperature')
ed_search_adv(query = 'temperature', protocol = "griddap")
ed_search_adv(query = 'temperature', protocol = "tabledap")
ed_search_adv(maxLat = 63, minLon = -107, maxLon = -87, minLat = 50, protocol = "griddap")
ed_search_adv(maxLat = 63, minLon = -107, maxLon = -87, minLat = 50, protocol = "tabledap")
ed_search_adv(minTime = "2010-01-01T00:00:00Z", maxTime="2010-02-01T00:00:00Z")
(out <- ed_search_adv(maxLat = 63, minLon = -107, maxLon = -87, minLat = 50,
            minTime = "2010-01-01T00:00:00Z", maxTime="2010-02-01T00:00:00Z"))
out$alldata[[1]]
ed_search_adv(variableName = 'upwelling')
ed_search_adv(query = 'upwelling', protocol = "tabledap")
```

```
# use a different URL
ed_search_adv(query = 'temperature', url = servers()$url[6])

## End(Not run)
```

`eurl` *Default ERDDAP server URL*

Description

Default ERDDAP server URL

Usage

```
eurl()
```

`fipscounty` *Convert a FIPS County Code to/from a County Name*

Description

Convert a FIPS County Code to/from a County Name

Usage

```
fipscounty(county = NULL, code = NULL, url = eurl(), ...)
```

Arguments

`county`	character; A county name.
`code`	numeric; A FIPS code.
`url`	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/
`...`	Curl args passed on to GET

Examples

```
 ## Not run:
fipscounty(code = "06053")
fipscounty(county = "CA, Monterey")
fipscounty(county = "OR, Multnomah")

## End(Not run)
```

griddap *Get ERDDAP gridded data*

Description

Get ERDDAP gridded data

Usage

```
griddap(x, ..., fields = "all", stride = 1, fmt = "nc", url = eurl(),
  store = disk(), read = TRUE, callopts = list())
```

Arguments

x	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.
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)
fmt	(character) One of csv or nc (for netcdf). Default: nc
url	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/
store	One of disk (default) or memory. You can pass options to disk
read	(logical) Read data into memory or not. Does not apply when store parameter is set to memory (which reads data into memory). For large csv, or especially netcdf files, you may want to set this to FALSE, which simply returns a summary of the dataset - and you can read in data piecemeal later. Default: TRUE
callopts	Pass on curl options to GET

Details

Details:

Value

An object of class griddap_csv if csv chosen or griddap_nc if nc file format chosen. These two classes are a thin wrapper around a data.frame, so the data you get back is a data.frame with metadata attached as attributes, along with a summary of the netcdf file (if fmt="nc"). If read=FALSE, you get back an empty data.frame.

Dimensions and Variables

ERDDAP grid dap data has this concept of dimenions vs. variables. Dimensions are things like time, latitude, longitude, altitude, and depth. Whereas variables are the measured variables, e.g., temperature, salinity, air.

You can't separately adjust values for dimensions for different variables. So, here's how it's gonna work:

Pass in lower and upper limits you want for each dimension as a vector (e.g., `c(1,2)`), or leave to defaults (i.e., don't pass anything to a dimension). Then pick which variables you want returned via the `fields` parameter. If you don't pass in options to the `fields` parameter, you get all variables back.

To get the dimensions and variables, along with other metadata for a dataset, run `info`, and each will be shown, with their min and max values, and some other metadata.

Where does the data go?

You can choose where data is stored. Be careful though. You can easily get a single file of hundreds of MB's (upper limit: 2 GB) in size with a single request. To the `store` parameter, pass `memory` if you want to store the data in memory (saved as a data.frame), or pass `disk` if you want to store on disk in a file. Note that `memory` and `disk` are not character strings, but function calls. `memory` does not accept any inputs, while `disk` does. Possibly will add other options, like "sql" for storing in a SQL database.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/rest.html

Examples

```
## Not run:
# single variable dataset
## You can pass in the outpu of a call to info
(out <- info('noaa_esrl_027d_0fb5_5d38'))
(res <- griddap(out,
 time = c('2012-01-01','2012-06-12'),
 latitude = c(21, 18),
 longitude = c(-80, -75)
))
## Or, pass in a dataset id
(res <- griddap('noaa_esrl_027d_0fb5_5d38',
 time = c('2012-01-01','2012-06-12'),
 latitude = c(21, 18),
 longitude = c(-80, -75)
))

# multi-variable dataset
```

```
(out <- info('erdQMekm14day'))
(res <- griddap(out,
 time = c('2015-12-28','2016-01-01'),
 latitude = c(24, 23),
 longitude = c(88, 90)
))
(res <- griddap(out, time = c('2015-12-28','2016-01-01'), latitude = c(24, 23),
   longitude = c(88, 90), fields = 'mod_current'))
(res <- griddap(out, time = c('2015-12-28','2016-01-01'), latitude = c(24, 23),
   longitude = c(88, 90), fields = 'mod_current', stride = c(1,2,1,2)))
(res <- griddap(out, time = c('2015-12-28','2016-01-01'), latitude = c(24, 23),
   longitude = c(88, 90), fields = c('mod_current','u_current')))

(out <- info('noaa_esrl_4965_b6d4_7198'))
(res <- griddap(out,
 time = c('1990-10-01', '1991-02-01'),
 latitude = c(20, 21),
 longitude = c(2, 5)
))

# Write to memory (within R), or to disk
(out <- info('erdQSwindmday'))
## disk, by default (to prevent bogging down system w/ large datasets)
## you can also pass in path and overwrite options to disk()
(res <- griddap(out,
 time = c('2006-07-11','2006-07-20'),
 longitude = c(166, 170),
 store = disk()
))
## the 2nd call is much faster as it's mostly just the time of reading in the table from disk
system.time( griddap(out,
 time = c('2006-07-11','2006-07-15'),
 longitude = c(10, 15),
 store = disk()
) )
system.time( griddap(out,
 time = c('2006-07-11','2006-07-15'),
 longitude = c(10, 15),
 store = disk()
) )

## memory
(res <- griddap("hawaii_463b_5b04_35b7",
 time = c('2015-01-01','2015-01-03'),
 latitude = c(14, 15),
 longitude = c(75, 76),
 store = memory()
))

## Use ncdf4 package to parse data
info("hawaii_463b_5b04_35b7")
(res <- griddap("hawaii_463b_5b04_35b7",
 time = c('2015-01-01','2015-01-03'),
```

```
 latitude = c(14, 15),
 longitude = c(75, 76)
))

# Get data in csv format
## by default, we get netcdf format data
(res <- griddap('hawaii_463b_5b04_35b7',
 time = c('2015-01-01','2015-01-03'),
 latitude = c(14, 15),
 longitude = c(75, 76),
 fmt = "csv"
))

# Use a different ERDDAP server url
## NOAA IOOS PacIOOS
url = "http://oos.soest.hawaii.edu/erddap/"
out <- info("NOAA_DHW", url = url)
(res <- griddap(out,
 time = c('2005-11-01','2006-01-01'),
 latitude = c(21, 20),
 longitude = c(10, 11)
))
## pass directly into griddap()
griddap("NOAA_DHW", url = url,
 time = c('2005-11-01','2006-01-01'),
 latitude = c(21, 20),
 longitude = c(10, 11)
)

# You don't have to pass in all of the dimensions
## They do have to be named!
griddap(out, time = c('2005-11-01','2005-11-03'))

# Using 'last'
## with time
griddap('noaa_esrl_fce0_4aad_340a',
 time = c('last-5','last'),
 latitude = c(21, 18),
 longitude = c(3, 5)
)
## with latitude
griddap('noaa_esrl_fce0_4aad_340a',
  time = c('2008-01-01','2009-01-01'),
  latitude = c('last', 'last'),
  longitude = c(3, 5)
)
## with longitude
griddap('noaa_esrl_fce0_4aad_340a',
 time = c('2008-01-01','2009-01-01'),
 latitude = c(21, 18),
 longitude = c('last', 'last')
)
```

```
## End(Not run)
```

`info` *Get information on an ERDDAP dataset.*

Description

Get information on an ERDDAP dataset.

Usage

```
info(datasetid, url = eurl(), ...)

as.info(x, url)
```

Arguments

`datasetid`	Dataset id
`url`	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/
`...`	Further args passed on to GET (must be a named parameter)
`x`	A datasetid or the output of `info`

Value

Prints a summary of the data on return, but you can index to various information.

The data is a list of length two with:

- variables - Data.frame of variables and their types
- alldata - List of data variables and their full attributes

Where `alldata` element has many data.frame's, one for each variable, with metadata for that variable. E.g., for griddap dataset `noaa_pfeg_696e_ec99_6fa6`, `alldata` has:

- NC_GLOBAL
- time
- latitude
- longitude
- sss

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/index.html

Examples

```
## Not run:
# grid dap datasets
info('noaa_esrl_027d_0fb5_5d38')

(out <- ed_search(query='temperature'))
info(out$info$dataset_id[5])
info(out$info$dataset_id[15])
info(out$info$dataset_id[25])
info(out$info$dataset_id[150])
info(out$info$dataset_id[400])
info(out$info$dataset_id[678])

out <- info(datasetid='noaa_esrl_027d_0fb5_5d38')
## See brief overview of the variables and range of possible values, if given
out$variables
## all information on longitude
out$alldata$longitude
## all information on air
out$alldata$air

# table dap datasets
(out <- ed_search(query='temperature', which = "table"))
info(out$info$dataset_id[1])
info(out$info$dataset_id[2])
info(out$info$dataset_id[3])
info(out$info$dataset_id[4])

info('erdCalCOFIfshsiz')
out <- info('erdCinpKfmBT')
## See brief overview of the variables and range of possible values, if given
out$variables
## all information on longitude
out$alldata$longitude
## all information on Haliotis_corrugata_Mean_Density
out$alldata$Haliotis_corrugata_Mean_Density

# use a different ERDDAP server
## Marine Institute (Ireland)
info("IMI_CONN_2D", url = "http://erddap.marine.ie/erddap/")
## Marine Domain Awareness (MDA) (Italy)
info("erdMH1chlamday", url = "https://bluehub.jrc.ec.europa.eu/erddap/")

## End(Not run)
```

`institutions` *institutions*

Description

institutions

Format

A character vector

`ioos_categories` *ioos_categories*

Description

ioos_categories

Format

A character vector

`keywords` *keywords*

Description

keywords

Format

A character vector

`key_words` *Convert a CF Standard Name to/from a GCMD Science Keyword*

Description

Convert a CF Standard Name to/from a GCMD Science Keyword

Usage

```
key_words(cf = NULL, gcmd = NULL, url = eurl(), ...)
```

Arguments

`cf`	character; A cf standard name http://cfconventions.org/Data/cf-standard-names/27/build/cf-standard-name-table.html
`gcmd`	character; A GCMD science keyword http://gcmd.gsfc.nasa.gov/learn/keyword_list.html
`url`	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/
`...`	Curl args passed on to GET

Examples

```
 ## Not run:
key_words(cf = "air_pressure")
cat(key_words(cf = "air_pressure"))
key_words(gcmd = "Atmosphere > Atmospheric Pressure > Sea Level Pressure")
cat(key_words(gcmd = "Atmosphere > Atmospheric Pressure > Sea Level Pressure"))

# a different ERDDAP server
key_words(cf = "air_pressure", url = servers()$url[5])

## End(Not run)
```

`longnames` *longnames*

Description

longnames

Format

A character vector

`servers` *ERDDAP server URLS and other info*

Description

ERDDAP server URLS and other info

Usage

```
servers()
```

Examples

```
servers()
```

`standardnames` *standardnames*

Description

standardnames

Format

A character vector

`tabledap` *Get ERDDAP tabledap data.*

Description

Get ERDDAP tabledap data.

Usage

```
tabledap(x, ..., fields = NULL, distinct = FALSE, orderby = NULL,
  orderbymax = NULL, orderbymin = NULL, orderbyminmax = NULL,
  units = NULL, url = eurl(), store = disk(), callopts = list())
```

Arguments

`x`	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`.
`...`	Any number of key-value pairs in quotes as query constraints. See Details & examples
`fields`	Columns to return, as a character vector
`distinct`	If TRUE ERDDAP will sort all of the rows in the results table (starting with the first requested variable, then using the second requested variable if the first variable has a tie, ...), then remove all non-unique rows of data. In many situations, ERDDAP can return distinct values quickly and efficiently. But in some cases, ERDDAP must look through all rows of the source dataset.
`orderby`	If used, ERDDAP will sort all of the rows in the results table (starting with the first variable, then using the second variable if the first variable has a tie, ...). Normally, the rows of data in the response table are in the order they arrived from the data source. orderBy allows you to request that the results table be sorted in a specific way. For example, use `orderby=c("stationID,time")` to get the results sorted by stationID, then time. The orderby variables MUST be included in the list of requested variables in the fields parameter.
`orderbymax`	Give a vector of one or more fields, that must be included in the fields parameter as well. Gives back data given constraints. ERDDAP will sort all of the rows in the results table (starting with the first variable, then using the second variable if the first variable has a tie, ...) and then just keeps the rows where the value of the last sort variable is highest (for each combination of other values).
`orderbymin`	Same as `orderbymax` parameter, except returns minimum value.
`orderbyminmax`	Same as `orderbymax` parameter, except returns two rows for every combination of the n-1 variables: one row with the minimum value, and one row with the maximum value.
`units`	One of 'udunits' (units will be described via the UDUNITS standard (e.g.,degrees_C)) or 'ucum' (units will be described via the UCUM standard (e.g., Cel)).
`url`	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/

store	One of disk (default) or memory. You can pass options to disk
callopts	Further args passed on to httr::GET (must be a named parameter)

Details

For key-value pair query constraints, the valid operators are =, != (not equals), =~ (a regular expression test), <, <=, >, and >= . For regular expressions you need to add a regular expression. For others, nothing more is needed. Construct the entry like 'time>=2001-07-07' with the parameter on the left, value on the right, and the operator in the middle, all within a set of quotes. Since ERDDAP accepts values other than =, we can't simply do time = '2001-07-07' as we normally would.

Server-side functionality: Some tasks are done server side. You don't have to worry about what that means. They are provided via parameters in this function. See distinct, orderby, orderbymax, orderbymin, orderbyminmax, and units.

Data is cached based on all parameters you use to get a dataset, including base url, query parameters. If you make the same exact call in the same or a different R session, as long you don't clear the cache, the function only reads data from disk, and does not have to request the data from the web again.

Value

An object of class tabledap. This class is a thin wrapper around a data.frame, so the data you get back is a data.frame with metadata attached as attributes.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://upwell.pfeg.noaa.gov/erddap/index.html

Examples

```
## Not run:
# Just passing the datasetid without fields gives all columns back
tabledap('erdCalCOFIfshsiz')

# Pass time constraints
tabledap('erdCalCOFIfshsiz', 'time>=2001-07-07', 'time<=2001-07-08')

# Pass in fields (i.e., columns to retrieve) & time constraints
tabledap('erdCalCOFIfshsiz',fields=c('longitude','latitude','fish_size','itis_tsn'),
   'time>=2001-07-07','time<=2001-07-10')
tabledap('erdCinpKfmBT', fields=c('latitude','longitude',
   'Aplysia_californica_Mean_Density','Muricea_californica_Mean_Density'),
   'time>=2007-06-24','time<=2007-07-01')

# Get info on a datasetid, then get data given information learned
info('erdCalCOFIlrvsiz')$variables
```

```
tabledap('erdCalCOFIlrvsiz', fields=c('latitude','longitude','larvae_size',
   'itis_tsn'), 'time>=2011-10-25', 'time<=2011-10-31')

# An example workflow
## Search for data
(out <- ed_search(query='fish', which = 'table'))
## Using a datasetid, search for information on a datasetid
id <- "hawaii_43a8_6d6d_9052"
info(id)$variables
## Get data from the dataset
tabledap(id, fields = c('scientificName', 'tsn', 'sex'))

# Time constraint
## Limit by time with date only
(info <- info('erdCalCOFIfshsiz'))
tabledap(info, fields = c('latitude','longitude','scientific_name'),
   'time>=2001-07-14')

# Use distinct parameter
tabledap('erdCalCOFIfshsiz',fields=c('longitude','latitude','fish_size','itis_tsn'),
   'time>=2001-07-07','time<=2001-07-10', distinct=TRUE)

# Use units parameter
## In this example, values are the same, but sometimes they can be different given the units
## value passed
tabledap('erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', units='udunits')
tabledap('erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', units='ucum')

# Use orderby parameter
tabledap('erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', orderby='temperature')
# Use orderbymax parameter
tabledap('erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', orderbymax='temperature')
# Use orderbymin parameter
tabledap('erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', orderbymin='temperature')
# Use orderbyminmax parameter
tabledap('erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', orderbyminmax='temperature')
# Use orderbymin parameter with multiple values
tabledap('erdCinpKfmT', fields=c('longitude','latitude','time','depth','temperature'),
   'time>=2007-06-10', 'time<=2007-09-21', orderbymax=c('depth','temperature'))

# Spatial delimitation
tabledap('erdCalCOFIfshsiz', fields = c('latitude','longitude','scientific_name'),
 'latitude>=34.8', 'latitude<=35', 'longitude>=-125', 'longitude<=-124')

# Integrate with taxize
out <- tabledap('erdCalCOFIfshsiz',
   fields = c('latitude','longitude','scientific_name','itis_tsn'))
```

```
tsns <- unique(out$itis_tsn[1:100])
library("taxize")
classif <- classification(tsns, db = "itis")
head(rbind(classif)); tail(rbind(classif))

# Write to memory (within R), or to disk
(out <- info('erdCalCOFIfshsiz'))
## disk, by default (to prevent bogging down system w/ large datasets)
## the 2nd call is much faster as it's mostly just the time of reading in the table from disk
system.time( tabledap('erdCalCOFIfshsiz', store = disk()) )
system.time( tabledap('erdCalCOFIfshsiz', store = disk()) )
## memory
tabledap(x='erdCalCOFIfshsiz', store = memory())

# use a different ERDDAP server
## NOAA IOOS NERACOOS
url <- "http://www.neracoos.org/erddap/"
tabledap("E01_optics_hist", url = url)

## End(Not run)
```

`variablenames` *variablenames*

Description

variablenames

Format

A character vector

`version` *Get ERDDAP version*

Description

Get ERDDAP version

Usage

```
version(url = eurl(), ...)
```

Arguments

`url`	A URL for an ERDDAP server. Default: http://upwell.pfeg.noaa.gov/erddap/
`...`	Curl args passed on to GET

Examples

```
## Not run:
version()
ss <- servers()
version(ss$url[1])
version(ss$url[2])
version(ss$url[3])

## End(Not run)
```

Index