Package 'rnoaa'

July 21, 2014

Version 0.2.0

Date 2014-07-21

License MIT + file LICENSE

Title NOAA climate data from R.

Description This is an R wrapper to many NOAA APIs including the NCDC climate API at http://www.ncdc.noaa.gov/cdo-web/webservices/v2. There are functions for each of the API endpoints: data, data categories, data sets, data types, locations, location categories, and stations. In addition, we have an interface for NOAA sea ice data that is not part of the climate data API, the NOAA severe weather inventory, and ERDDAP data. NOAA buoy data is only on the buoy branch in the Github repository for this package (see url below).

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

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

LazyLoad yes

LazyData yes

VignetteBuilder knitr

Imports httr, lubridate, plyr, ggplot2, scales, sp, rgdal, rgeos,maptools, stringr, XML, data.table, assertthat, jsonlite

Suggests testthat, roxygen2, knitr, taxize

Author Hart Edmund [aut, cre],Scott Chamberlain [aut],Karthik Ram [aut]

Maintainer Hart Edmund <Edmund.m.hart@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-21 23:25:40

R topics documented:

`rnoaa-package` *General purpose R interface to noaa.*

Description

rnoaa is an R interface to NOAA climate data.

Details

Specifically, most functions in this package interact with the National Climatic Data Center application programming interface (API) at `http://www.ncdc.noaa.gov/cdo-web/webservices/v2`.

An access token, or API key, is required to use this R package. The key is required by NOAA, not the creators of this R package. Go to the link given above to get an API key.

Author(s)

Scott Chamberlain `<myrmecocystus@gmail.com>`

`erddap_data` *Get ERDDAP data.*

Description

Get ERDDAP data.

Usage

```
erddap_data(datasetid, ..., fields = NULL, distinct = FALSE,
  orderby = NULL, orderbymax = NULL, orderbymin = NULL,
  orderbyminmax = NULL, units = NULL, callopts = list())
```

Arguments

`datasetid`	Dataset id
`...`	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)).
`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`.

Examples

```
## Not run:
# Just passing the datasetid without fields gives all columns back
out <- erddap_data(datasetid='erdCalCOFIfshsiz')
nrow(out)

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

# Pass in fields (i.e., columns to retrieve) & time constraints
erddap_data(datasetid='erdCalCOFIfshsiz',fields=c('longitude','latitude','fish_size','itis_tsn'),
   'time>=2001-07-07','time<=2001-07-10')
erddap_data(datasetid='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
erddap_info('erdCalCOFIlrvsiz')$variables
erddap_data(datasetid='erdCalCOFIlrvsiz', fields=c('latitude','longitude','larvae_size',
   'itis_tsn'), 'time>=2011-10-25', 'time<=2011-10-31')

# An example workflow
## Search for data
(out <- erddap_search(query='fish size'))
## Using a datasetid, search for information on a datasetid
id <- out$info$dataset_id[1]
erddap_info(datasetid=id)$variables
## Get data from the dataset
head(erddap_data(datasetid = id, fields = c('latitude','longitude','scientific_name')))

# Time constraint
## Limit by time with date only
erddap_data(datasetid = id, fields = c('latitude','longitude','scientific_name'),
   'time>=2001-07-14')
## Limit by time with hours and date
erddap_data(datasetid='ndbcSosWTemp', fields=c('latitude','longitude','sea_water_temperature'),
   'time>=2014-05-14T15:15:00Z')

# Use distinct parameter
```

```
erddap_data(datasetid='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
erddap_data(datasetid='erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', units='udunits')
erddap_data(datasetid='erdCinpKfmT', fields=c('longitude','latitude','time','temperature'),
   'time>=2007-09-19', 'time<=2007-09-21', units='ucum')

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

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

# Integrate with taxize
out <- erddap_data(datasetid = '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))

## End(Not run)
```

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

Description

Gives back a brief data.frame for quick inspection and a list of more complete metadata on the dataset.

Usage

```
erddap_info(datasetid, callopts = list())
```

Arguments

datasetid	Dataset id
callopts	Further args passed on to httr::GET (must be a named parameter)

Value

A list of length two

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

Examples

```
## Not run:
erddap_info(datasetid='erdCalCOFIfshsiz')
out <- erddap_info(datasetid='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

## End(Not run)
```

---

erddap_search *Search for ERDDAP datasets.*

Description

Search for ERDDAP datasets.

Usage

```
erddap_search(query, page = NULL, page_size = NULL, callopts = list())

## S3 method for class 'erddap_search'
print(x, ...)
```

Arguments

query	Search terms
page	Page number
page_size	Results per page
callopts	Further args passed on to httr::GET (must be a named parameter)
x	Input to print method for class erddap_search
...	Further args to print, ignored.

Examples

```
## Not run:
(out <- erddap_search(query='fish size'))
out$alldata
(out <- erddap_search(query='size'))
out$info

## End(Not run)
```

---

fipscodes *FIPS codes for US states.*

Description

A dataset containing the FIPS codes for 51 US states and territories. The variables are as follows:

Format

A data frame with 3142 rows and 5 variables

Details

- state. US state name.
- county. County name.
- fips_state. Numeric value, from 1 to 51.
- fips_county. Numeric value, from 1 to 840.
- fips. Numeric value, from 1001 to 56045.

is.ncdc_data *Check object class*

Description

Check if an object is of class ncdc_data, ncdc_datasets, ncdc_datatypes, ncdc_datacats, ncdc_locs, ncdc_locs_cats, or ncdc_stations

Usage

```
is.ncdc_data(x)

is.ncdc_datasets(x)

is.ncdc_datatypes(x)

is.ncdc_datacats(x)

is.ncdc_locs(x)

is.ncdc_locs_cats(x)

is.ncdc_stations(x)
```

Arguments

x	input

long2utm *Function to get UTM zone from a single longitude and latitude pair originally from David LeBauer I think*

Description

Function to get UTM zone from a single longitude and latitude pair originally from David LeBauer I think

Usage

```
long2utm(lon, lat)
```

Arguments

lon	Longitude, in decimal degree style
lat	Latitude, in decimal degree style

ncdc *Search for and get NOAA NCDC data.*

Description

Search for and get NOAA NCDC data.

Usage

```
ncdc(datasetid = NULL, datatypeid = NULL, stationid = NULL,
  locationid = NULL, startdate = NULL, enddate = NULL, sortfield = NULL,
  sortorder = NULL, limit = 25, offset = NULL, callopts = list(),
  token = NULL, dataset = NULL, datatype = NULL, station = NULL,
  location = NULL, locationtype = NULL, page = NULL, year = NULL,
  month = NULL, day = NULL, includemetadata = TRUE, results = NULL)
```

Arguments

datatypeid	Accepts a valid data type id or a chain of data type ids in a comma-separated vector. Data returned will contain all of the data type(s) specified (optional)
locationid	Accepts a valid location id or a chain of location ids in a comma-separated vector. Data returned will contain data for the location(s) specified (optional)
stationid	Accepts a valid station id or a chain of of station ids in a comma-separated vector. Data returned will contain data for the station(s) specified (optional)
sortfield	The field to sort results by. Supports id, name, mindate, maxdate, and datacoverage fields (optional)
sortorder	Which order to sort by, asc or desc. Defaults to asc (optional)
limit	Defaults to 25, limits the number of results in the response. Maximum is 1000 (optional)
offset	Defaults to 0, used to offset the resultlist (optional)
token	This must be a valid token token supplied to you by NCDC's Climate Data Online access token generator. (required) Get an API key (=token) at http://www.ncdc.noaa.gov/cdo-web/token. You can pass your token in as an argument or store it in your .Rprofile file with an entry like • options("noaakey" = "your-noaa-token")
callopts	Further arguments passed on to the API GET call. (optional)
datasetid	(required) Accepts a single valid dataset id. Data returned will be from the dataset specified, see datasets()
startdate	(required) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data after the specified date. The date range must be less than 1 year.
enddate	(required) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data before the specified date. The date range must be less than 1 year.

dataset	THIS IS A DEPRECATED ARGUMENT. See datasetid.
datatype	THIS IS A DEPRECATED ARGUMENT. See datatypeid.
station	THIS IS A DEPRECATED ARGUMENT. See stationid.
location	THIS IS A DEPRECATED ARGUMENT. See locationid.
locationtype	THIS IS A DEPRECATED ARGUMENT. There is no equivalent argument in v2 of the NOAA API.
page	THIS IS A DEPRECATED ARGUMENT. There is no equivalent argument in v2 of the NOAA API.
year	THIS IS A DEPRECATED ARGUMENT. Use combination of startdate and enddate arguments.
month	THIS IS A DEPRECATED ARGUMENT. Use combination of startdate and enddate arguments.
day	THIS IS A DEPRECATED ARGUMENT. Use combination of startdate and enddate arguments.
results	THIS IS A DEPRECATED ARGUMENT. See limit.
includemetadata	Used to improve response time by preventing the calculation of result metadata. Default: TRUE. This does not affect the return object, in that the named part of the output list called "meta' is still returned, but is NULL. In practice, I haven't seen response time's improve, but perhaps they will for you.

Details

Note that NOAA NCDC API calls can take a long time depending on the call. The NOAA API doesn't perform well with very long timespans, and will time out and make you angry - beware.

Keep in mind that three parameters, datasetid, startdate, and enddate are required.

Note that the default limit (no. records returned) is 25. Look at the metadata in `$meta` to see how many records were found. If more were found than 25, you could set the parameter `limit` to something higher than 25.

The attributes, or "flags", for each row of the output for data may have a flag with it. Each `datasetid` has it's own set of flags. The following are flag columns, and what they stand for. `fl_` is the beginning of each flag column name, then one or more characters to describe the flag, keeping it short to maintain a compact data frame. Some of these fields are the same across datasetids. See the vignette `vignette("rnoaa_attributes", "rnoaa")` for description of possible values for each flag.

- fl_c completeness
- fl_d day
- fl_m measurement
- fl_q quality
- fl_s source
- fl_t time
- fl_cmiss consecutive missing
- fl_miss missing
- fl_u units

Value

An S3 list of length two, a slot of metadata (meta), and a slot for data (data). The meta slot is a list of metadata elements, and the data slot is a data.frame, possibly of length zero if no data is found.

Examples

```
## Not run:
# GHCN-Daily (or GHCND) data, for a specific station
ncdc(datasetid='GHCND', stationid='GHCND:USW00014895', startdate = '2013-10-01',
   enddate = '2013-12-01')

# GHCND data, for a location by FIPS code
ncdc(datasetid='GHCND', locationid = 'FIPS:02', startdate = '2010-05-01',
   enddate = '2010-05-10')

# GHCND data from October 1 2013 to December 1 2013
ncdc(datasetid='GHCND', startdate = '2013-10-01', enddate = '2013-10-05')

# GHCN-Monthly (or GHCNDMS) data from October 1 2013 to December 1 2013
ncdc(datasetid='GHCNDMS', startdate = '2013-10-01', enddate = '2013-12-01')

# Normals Daily (or NORMAL_DLY) GHCND:USW00014895 dly-tmax-normal data
ncdc(datasetid='NORMAL_DLY', stationid='GHCND:USW00014895', startdate = '2010-05-01',
   enddate = '2010-05-10')

# Dataset, and location in Australia
ncdc(datasetid='GHCND', locationid='FIPS:AS', startdate = '2010-05-01', enddate = '2010-05-31')

# Dataset, location and datatype for PRECIP_HLY data
ncdc(datasetid='PRECIP_HLY', locationid='ZIP:28801', datatypeid='HPCP',
   startdate = '2010-05-01', enddate = '2010-05-10')

# Dataset, location, station and datatype
ncdc(datasetid='PRECIP_HLY', locationid='ZIP:28801', stationid='COOP:310301', datatypeid='HPCP',
   startdate = '2010-05-01', enddate = '2010-05-10')

# Dataset, location, and datatype for GHCND
ncdc(datasetid='GHCND', locationid='FIPS:BR', datatypeid='PRCP', startdate = '2010-05-01',
   enddate = '2010-05-10')

# Normals Daily GHCND dly-tmax-normal data
ncdc(datasetid='NORMAL_DLY', datatypeid='dly-tmax-normal', startdate = '2010-05-01',
   enddate = '2010-05-10')

# Normals Daily GHCND:USW00014895 dly-tmax-normal
ncdc(datasetid='NORMAL_DLY', stationid='GHCND:USW00014895', datatypeid='dly-tmax-normal',
   startdate = '2010-05-01', enddate = '2010-05-10')

# Hourly Precipitation data for ZIP code 28801
ncdc(datasetid='PRECIP_HLY', locationid='ZIP:28801', datatypeid='HPCP',
   startdate = '2010-05-01', enddate = '2010-05-10')
```

```
# 15 min Precipitation data for ZIP code 28801
ncdc(datasetid='PRECIP_15', datatypeid='QPCP', startdate = '2010-05-01', enddate = '2010-05-02')

# Search the NORMAL_HLY dataset
ncdc(datasetid='NORMAL_HLY', stationid = 'GHCND:USW00003812', startdate = '2010-05-01',
   enddate = '2010-05-10')

# Search the ANNUAL dataset
ncdc(datasetid='ANNUAL', locationid='ZIP:28801', startdate = '2010-05-01',
   enddate = '2010-05-10')

# Search the NORMAL_ANN dataset
ncdc(datasetid='NORMAL_ANN', datatypeid='ANN-DUTR-NORMAL', startdate = '2010-01-01',
   enddate = '2010-01-01')

# Include metadata or not
ncdc(datasetid='GHCND', stationid='GHCND:USW00014895', startdate = '2013-10-01',
   enddate = '2013-12-01')
ncdc(datasetid='GHCND', stationid='GHCND:USW00014895', startdate = '2013-10-01',
   enddate = '2013-12-01', includemetadata=FALSE)

## End(Not run)


# NEXRAD2 data
## doesn't work yet
ncdc(datasetid='NEXRAD2', startdate = '2013-10-01', enddate = '2013-12-01')
```

`ncdc_combine` *Coerce multiple outputs to a single data.frame object.*

Description

Coerce multiple outputs to a single data.frame object.

Usage

```
ncdc_combine(...)
```

Arguments

`...`	Objects from another ncdc_* function.

Value

A data.frame

Examples

```
## Not run:
# data
out1 <- ncdc(datasetid='GHCND', locationid = 'FIPS:02', startdate = '2010-05-01',
enddate = '2010-05-31', limit=10)
out2 <- ncdc(datasetid='GHCND', locationid = 'FIPS:02', startdate = '2010-07-01',
enddate = '2010-07-31', limit=10)
ncdc_combine(out1, out2)

# data sets
out1 <- ncdc_datasets(datatypeid='TOBS')
out2 <- ncdc_datasets(datatypeid='PRCP')
ncdc_combine(out1, out2)

# data types
out1 <- ncdc_datatypes(datatypeid="ACMH")
out2 <- ncdc_datatypes(datatypeid='PRCP')
ncdc_combine(out1, out2)

# data categories
out1 <- ncdc_datacats(datacategoryid="ANNAGR")
out2 <- ncdc_datacats(datacategoryid='PRCP')
ncdc_combine(out1, out2)

# data locations
out1 <- ncdc_locs(locationcategoryid='ST', limit=52)
out2 <- ncdc_locs(locationcategoryid='CITY', sortfield='name', sortorder='desc')
ncdc_combine(out1, out2)

# data locations
out1 <- ncdc_locs_cats(startdate='1970-01-01')
out2 <- ncdc_locs_cats(locationcategoryid='CLIM_REG')
ncdc_combine(out1, out2)

# stations
out1 <- ncdc_stations(datasetid='GHCND', locationid='FIPS:12017',
stationid='GHCND:USC00084289')
out2 <- ncdc_stations(stationid='COOP:010008')
out3 <- ncdc_stations(datasetid='PRECIP_HLY', startdate='19900101',
enddate='19901231')
out4 <- ncdc_stations(datasetid='GHCND', locationid='FIPS:12017')
ncdc_combine(out1, out2, out3, out4)

# try to combine two different classes
out1 <- ncdc_locs_cats(startdate='1970-01-01')
out2 <- ncdc_stations(stationid='COOP:010008')
out3 <- ncdc_locs_cats(locationcategoryid='CLIM_REG')
ncdc_combine(out1, out2, out3)

## End(Not run)
```

`ncdc_datacats` *Get possible data categories for a particular datasetid, locationid, stationid, etc.*

Description

Data Categories represent groupings of data types.

Usage

```
ncdc_datacats(datasetid = NULL, datacategoryid = NULL, stationid = NULL,
  locationid = NULL, startdate = NULL, enddate = NULL, sortfield = NULL,
  sortorder = NULL, limit = 25, offset = NULL, callopts = list(),
  token = NULL)
```

Arguments

`datasetid`	Accepts a single valid dataset id. Data returned will be from the dataset specified, see datasets() (required)
`datacategoryid`	A valid data category id. Data types returned will be associated with the data category(ies) specified
`locationid`	Accepts a valid location id. Data returned will contain data for the location(s) specified (optional)
`stationid`	Accepts a valid station id. Data returned will contain data for the station(s) specified (optional)
`startdate`	Accepts valid ISO formated date (yyyy-mm-dd). Data returned will have data after the specified date. Paramater can be use independently of enddate (optional)
`enddate`	Accepts valid ISO formated date (yyyy-mm-dd). Data returned will have data before the specified date. Paramater can be use independently of startdate (optional)
`sortfield`	The field to sort results by. Supports id, name, mindate, maxdate, and datacoverage fields (optional)
`sortorder`	Which order to sort by, asc or desc. Defaults to asc (optional)
`limit`	Defaults to 25, limits the number of results in the response. Maximum is 1000 (optional)
`offset`	Defaults to 0, used to offset the resultlist (optional)
`token`	This must be a valid token token supplied to you by NCDC's Climate Data Online access token generator. (required) Get an API key (=token) at http://www.ncdc.noaa.gov/cdo-web/token. You can pass your token in as an argument or store it in your .Rprofile file with an entry like • options("noaakey" = "your-noaa-token")
`callopts`	Further arguments passed on to the API GET call. (optional)

Details

Note that calls with both startdate and enddate don't seem to work, though specifying one or the other mostly works.

Value

A `data.frame` for all datasets, or a list of length two, each with a data.frame.

References

Vignette at http://ropensci.org/tutorials/rnoaa_tutorial.html

Examples

```
## Not run:
## Limit to 41 results
ncdc_datacats(limit=41)

## Single data category
ncdc_datacats(datacategoryid="ANNAGR")

## Fetch data categories for a given set of locations
ncdc_datacats(locationid='CITY:US390029')
ncdc_datacats(locationid=c('CITY:US390029', 'FIPS:37'))

## Data categories for a given date
ncdc_datacats(startdate = '2013-10-01')

## End(Not run)
```

---

ncdc_datasets *Search NOAA datasets*

Description

From the NOAA API docs: All of our data are in datasets. To retrieve any data from us, you must know what dataset it is in.

Usage

```
ncdc_datasets(datasetid = NULL, datatypeid = NULL, stationid = NULL,
  locationid = NULL, startdate = NULL, enddate = NULL, sortfield = NULL,
  sortorder = NULL, limit = 25, offset = NULL, callopts = list(),
  token = NULL, dataset = NULL, page = NULL, year = NULL,
  month = NULL)
```

Arguments

datatypeid	Accepts a valid data type id or a chain of data type ids in a comma-separated vector. Data returned will contain all of the data type(s) specified (optional)
locationid	Accepts a valid location id or a chain of location ids in a comma-separated vector. Data returned will contain data for the location(s) specified (optional)
stationid	Accepts a valid station id or a chain of of station ids in a comma-separated vector. Data returned will contain data for the station(s) specified (optional)
sortfield	The field to sort results by. Supports id, name, mindate, maxdate, and datacoverage fields (optional)
sortorder	Which order to sort by, asc or desc. Defaults to asc (optional)
limit	Defaults to 25, limits the number of results in the response. Maximum is 1000 (optional)
offset	Defaults to 0, used to offset the resultlist (optional)
token	This must be a valid token token supplied to you by NCDC's Climate Data Online access token generator. (required) Get an API key (=token) at http://www.ncdc.noaa.gov/cdo-web/token. You can pass your token in as an argument or store it in your .Rprofile file with an entry like • options("noaakey" = "your-noaa-token")
callopts	Further arguments passed on to the API GET call. (optional)
datasetid	(optional) Accepts a single valid dataset id. Data returned will be from the dataset specified, see datasets()
startdate	(optional) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data after the specified date. The date range must be less than 1 year.
enddate	(optional) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data before the specified date. The date range must be less than 1 year.
dataset	THIS IS A DEPRECATED ARGUMENT. See datasetid.
page	THIS IS A DEPRECATED ARGUMENT. There is no equivalent argument in v2 of the NOAA API.
year	THIS IS A DEPRECATED ARGUMENT. Use combination of startdate and enddate arguments.
month	THIS IS A DEPRECATED ARGUMENT. Use combination of startdate and enddate arguments.

Value

A data.frame for all datasets, or a list of length two, each with a data.frame.

Examples

```
## Not run:
# Get a table of all datasets
ncdc_datasets()
```

```
# Get details from a particular dataset
ncdc_datasets(datasetid='ANNUAL')

# Get datasets with Temperature at the time of observation (TOBS) data type
ncdc_datasets(datatypeid='TOBS')

# Get datasets with data for a series of the same parameter arg, in this case
# stationid's
ncdc_datasets(stationid=c('COOP:310090','COOP:310184','COOP:310212'))

# Multiple datatypeid's
ncdc_datasets(datatypeid=c('ACMC','ACMH','ACSC'))
ncdc_datasets(datasetid='ANNUAL', datatypeid=c('ACMC','ACMH','ACSC'))

## End(Not run)
```

`ncdc_datatypes` *Get possible data types for a particular dataset*

Description

From the NOAA API docs: Describes the type of data, acts as a label. If it's 64 degrees out right now, then the data type is Air Temperature and the data is 64.

Usage

```
ncdc_datatypes(datasetid = NULL, datatypeid = NULL, datacategoryid = NULL,
  stationid = NULL, locationid = NULL, startdate = NULL, enddate = NULL,
  sortfield = NULL, sortorder = NULL, limit = 25, offset = NULL,
  callopts = list(), token = NULL, dataset = NULL, page = NULL,
  filter = NULL)
```

Arguments

`datatypeid`	Accepts a valid data type id or a chain of data type ids in a comma-separated vector. Data returned will contain all of the data type(s) specified (optional)
`locationid`	Accepts a valid location id or a chain of location ids in a comma-separated vector. Data returned will contain data for the location(s) specified (optional)
`stationid`	Accepts a valid station id or a chain of of station ids in a comma-separated vector. Data returned will contain data for the station(s) specified (optional)
`sortfield`	The field to sort results by. Supports id, name, mindate, maxdate, and datacoverage fields (optional)
`sortorder`	Which order to sort by, asc or desc. Defaults to asc (optional)
`limit`	Defaults to 25, limits the number of results in the response. Maximum is 1000 (optional)
`offset`	Defaults to 0, used to offset the resultlist (optional)

token — This must be a valid token token supplied to you by NCDC's Climate Data Online access token generator. (required) Get an API key (=token) at http://www.ncdc.noaa.gov/cdo-web/token. You can pass your token in as an argument or store it in your .Rprofile file with an entry like

- options("noaakey" = "your-noaa-token")

callopts — Further arguments passed on to the API GET call. (optional)

datasetid — (optional) Accepts a single valid dataset id. Data returned will be from the dataset specified, see datasets()

startdate — (optional) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data after the specified date. The date range must be less than 1 year.

enddate — (optional) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data before the specified date. The date range must be less than 1 year.

dataset — THIS IS A DEPRECATED ARGUMENT. See datasetid.

page — THIS IS A DEPRECATED ARGUMENT. There is no equivalent argument in v2 of the NOAA API.

filter — THIS IS A DEPRECATED ARGUMENT. There is no equivalent argument in v2 of the NOAA API.

datacategoryid — Optional. Accepts a valid data category id or a chain of data category ids seperated by ampersands (although it is rare to have a data type with more than one data category). Data types returned will be associated with the data category(ies) specified

Value

A data.frame for all datasets, or a list of length two, each with a data.frame.

Examples

```
## Not run:
# Fetch available data types
ncdc_datatypes()

# Fetch more information about the ACMH data type id
ncdc_datatypes(datatypeid="ACMH")

# Fetch data types with the air temperature data category
ncdc_datatypes(datacategoryid="TEMP", limit=56)

# Fetch data types that support a given set of stations
ncdc_datatypes(stationid=c('COOP:310090','COOP:310184','COOP:310212'))

## End(Not run)
```

ncdc_locs *Get metadata about NOAA NCDC locations.*

Description

From the NOAA NCDC API docs: Locations can be a specific latitude/longitude point such as a station, or a label representing a bounding area such as a city.

Usage

```
ncdc_locs(datasetid = NULL, locationid = NULL, locationcategoryid = NULL,
  startdate = NULL, enddate = NULL, sortfield = NULL, sortorder = NULL,
  limit = 25, offset = NULL, callopts = list(), token = NULL)
```

Arguments

datasetid	A single valid dataset id. Data returned will be from the dataset specified, see datasets() (required)
locationcategoryid	A valid location id or a chain of location category ids in a comma-separated vector. Locations returned will be in the location category(ies) specified
startdate	A valid ISO formated date (yyyy-mm-dd). Data returned will have data after the specified date. Paramater can be use independently of enddate (optional)
enddate	Accepts valid ISO formated date (yyyy-mm-dd). Data returned will have data before the specified date. Paramater can be use independently of startdate (optional)
sortfield	The field to sort results by. Supports id, name, mindate, maxdate, and datacoverage fields (optional)
sortorder	Which order to sort by, asc or desc. Defaults to asc (optional)
limit	Defaults to 25, limits the number of results in the response. Maximum is 1000 (optional)
offset	Defaults to 0, used to offset the resultlist (optional)
token	This must be a valid token token supplied to you by NCDC's Climate Data Online access token generator. (required) Get an API key (=token) at http://www.ncdc.noaa.gov/cdo-web/token. You can pass your token in as an argument or store it in your .Rprofile file with an entry like • options("noaakey" = "your-noaa-token")
callopts	Further arguments passed on to the API GET call. (optional)
locationid	A valid location id or a chain of location ids seperated by ampersands. Data returned will contain data for the location(s) specified (optional)

Value

A list containing metadata and the data, or a single data.frame.

Examples

```
## Not run:
# All locations, first 25 results
ncdc_locs()

# Fetch more information about location id FIPS:37
ncdc_locs(locationid='FIPS:37')

# Fetch available locations for the GHCND (Daily Summaries) dataset
ncdc_locs(datasetid='GHCND')

# Fetch all U.S. States
ncdc_locs(locationcategoryid='ST', limit=52)

# Fetch list of city locations in descending order
ncdc_locs(locationcategoryid='CITY', sortfield='name', sortorder='desc')

## End(Not run)
```

---

ncdc_locs_cats *Get metadata about NOAA location categories.*

Description

Location categories are groupings of similar locations.

Usage

```
ncdc_locs_cats(datasetid = NULL, locationcategoryid = NULL,
  startdate = NULL, enddate = NULL, sortfield = NULL, sortorder = NULL,
  limit = 25, offset = NULL, callopts = list(), token = NULL)
```

Arguments

datasetid	A single valid dataset id. Data returned will be from the dataset specified, see datasets() (required)
locationcategoryid	A valid location id or a chain of location category ids in a comma-separated vector. Locations returned will be in the location category(ies) specified
startdate	A valid ISO formated date (yyyy-mm-dd). Data returned will have data after the specified date. Paramater can be use independently of enddate (optional)
enddate	Accepts valid ISO formated date (yyyy-mm-dd). Data returned will have data before the specified date. Paramater can be use independently of startdate (optional)
sortfield	The field to sort results by. Supports id, name, mindate, maxdate, and datacoverage fields (optional)
sortorder	Which order to sort by, asc or desc. Defaults to asc (optional)

limit	Defaults to 25, limits the number of results in the response. Maximum is 1000 (optional)
offset	Defaults to 0, used to offset the resultlist (optional)
token	This must be a valid token token supplied to you by NCDC's Climate Data Online access token generator. (required) Get an API key (=token) at http://www.ncdc.noaa.gov/cdo-web/token. You can pass your token in as an argument or store it in your .Rprofile file with an entry like • options("noaakey" = "your-noaa-token")
callopts	Further arguments passed on to the API GET call. (optional)

Details

Locations can be a specific latitude/longitude point such as a station, or a label representing a bounding area such as a city.

Value

A list containing metadata and the data, or a single data.frame.

Examples

```
## Not run:
# All location categories, first 25 results
ncdc_locs_cats()

# Find locations with category id of CLIM_REG
ncdc_locs_cats(locationcategoryid='CLIM_REG')

# Displays available location categories within GHCN-Daily dataset
ncdc_locs_cats(datasetid='GHCND')

# Displays available location categories from start date 1970-01-01
ncdc_locs_cats(startdate='1970-01-01')

## End(Not run)
```

---

ncdc_plot *Plot NOAA climate data.*

Description

This function accepts directly output from the ncdc function, not other functions.

Usage

```
ncdc_plot(..., breaks = "7 days", dateformat = "%d/%m/%y")

## S3 method for class 'ncdc_data'
ncdc_plot(..., breaks = "7 days",
  dateformat = "%d/%m/%y")
```

Arguments

...	Input noaa object or objects.
breaks	Regularly spaced date breaks for x-axis. See date_breaks
dateformat	Date format using standard POSIX specification for labels on x-axis. See date_format

Details

This is a simple wrapper function around some ggplot2 code. There is indeed a lot you can modify in your plots, so this function just does some basic stuff. Here's the code within this function, where input is the output from a ncdc call - go crazy:

input <- input$data input$date <- ymd(str_replace(as.character(input$date), "T00:00:00\.000", ")) ggplot(input, aes(date, value)) + theme_bw(base_size=18) + geom_line(size=2) + scale_x_datetime(breaks = date_breaks("7 days"), labels = date_format(' labs(y=as.character(input[1,'dataType']), x="Date")

Value

Plot of climate data.

Examples

```
## Not run:
# Search for data first, then plot
out <- ncdc(datasetid='GHCND', stationid='GHCND:USW00014895', datatypeid='PRCP',
startdate = '2010-05-01', enddate = '2010-10-31', limit=500)
ncdc_plot(out)
ncdc_plot(out, breaks="14 days")
ncdc_plot(out, breaks="1 month", dateformat="%d/%m")
ncdc_plot(out, breaks="1 month", dateformat="%d/%m")

out2 <- ncdc(datasetid='GHCND', stationid='GHCND:USW00014895', datatypeid='PRCP',
startdate = '2010-05-01', enddate = '2010-05-03', limit=100)
ncdc_plot(out2, breaks="6 hours", dateformat="%H")

# Combine many calls to ncdc function
out1 <- ncdc(datasetid='GHCND', stationid='GHCND:USW00014895', datatypeid='PRCP',
startdate = '2010-03-01', enddate = '2010-05-31', limit=500)
out2 <- ncdc(datasetid='GHCND', stationid='GHCND:USW00014895', datatypeid='PRCP',
startdate = '2010-09-01', enddate = '2010-10-31', limit=500)
df <- ncdc_combine(out1, out2)
ncdc_plot(df)
## or pass in each element separately
ncdc_plot(out1, out2, breaks="45 days")
```

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

---

ncdc_stations *Get metadata about NOAA NCDC stations.*

Description

From the NOAA NCDC API docs: Stations are where the data comes from (for most datasets) and can be considered the smallest granual of location data. If you know what station you want, you can quickly get all manner of data from it

Usage

```
ncdc_stations(stationid = NULL, datasetid = NULL, datatypeid = NULL,
  locationid = NULL, startdate = NULL, enddate = NULL, sortfield = NULL,
  sortorder = NULL, limit = 25, offset = NULL, datacategoryid = NULL,
  extent = NULL, radius = 10, callopts = list(), token = NULL,
  dataset = NULL, station = NULL, location = NULL, locationtype = NULL,
  page = NULL)
```

Arguments

datatypeid	Accepts a valid data type id or a chain of data type ids in a comma-separated vector. Data returned will contain all of the data type(s) specified (optional)
locationid	Accepts a valid location id or a chain of location ids in a comma-separated vector. Data returned will contain data for the location(s) specified (optional)
stationid	Accepts a valid station id or a chain of of station ids in a comma-separated vector. Data returned will contain data for the station(s) specified (optional)
sortfield	The field to sort results by. Supports id, name, mindate, maxdate, and datacoverage fields (optional)
sortorder	Which order to sort by, asc or desc. Defaults to asc (optional)
limit	Defaults to 25, limits the number of results in the response. Maximum is 1000 (optional)
offset	Defaults to 0, used to offset the resultlist (optional)
token	This must be a valid token token supplied to you by NCDC's Climate Data Online access token generator. (required) Get an API key (=token) at http://www.ncdc.noaa.gov/cdo-web/token. You can pass your token in as an argument or store it in your .Rprofile file with an entry like • options("noaakey" = "your-noaa-token")
callopts	Further arguments passed on to the API GET call. (optional)
datasetid	(optional) Accepts a single valid dataset id. Data returned will be from the dataset specified, see datasets()

startdate	(optional) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data after the specified date. The date range must be less than 1 year.
enddate	(optional) Accepts valid ISO formated date (yyyy-mm-dd) or date time (YYYY-MM-DDThh:mm:ss). Data returned will have data before the specified date. The date range must be less than 1 year.
datacategoryid	(character, optional) Accepts a valid data category id or an array of data category ids. Stations returned will be associated with the data category(ies) specified
extent	(numeric, optional) The geographical extent for which you want to search. Give either a vecctor with two values: a latitude and a longitude. For example, `c(lat,long)`. Or give four values that defines a bounding box, lat and long for the southwest corner, then lat and long for the northeast corner. For example: `c(minlat,minlong, maxlat, maxlong)`.
radius	(numeric) If a single latitude/longitude pair is given to the extent parameter, the radius to create around the point. Ignored if a vector of appropriate structure is passed to the extent parameter.
dataset	THIS IS A DEPRECATED ARGUMENT. See datasetid.
station	THIS IS A DEPRECATED ARGUMENT. See stationid.
location	THIS IS A DEPRECATED ARGUMENT. See locationid.
locationtype	THIS IS A DEPRECATED ARGUMENT. There is no equivalent argument in v2 of the NOAA API.
page	THIS IS A DEPRECATED ARGUMENT. There is no equivalent argument in v2 of the NOAA API.

Value

A list of metadata.

Examples

```
## Not run:
# Get metadata on all stations
ncdc_stations()
ncdc_stations(limit=5)

# Get metadata on a single station
ncdc_stations(stationid='COOP:010008')

# Displays all stations within GHCN-Daily (100 Stations per page limit)
ncdc_stations(datasetid='GHCND')

# Station
ncdc_stations(datasetid='NORMAL_DLY', stationid='GHCND:USW00014895')

# Displays all stations within GHCN-Daily (Displaying page 10 of the results)
ncdc_stations(datasetid='GHCND')

# Specify datasetid and locationid
```

```
ncdc_stations(datasetid='GHCND', locationid='FIPS:12017')

# Specify datasetid, locationid, and station
ncdc_stations(datasetid='GHCND', locationid='FIPS:12017', stationid='GHCND:USC00084289')

# Specify datasetid, locationidtype, locationid, and station
ncdc_stations(datasetid='GHCND', locationid='FIPS:12017', stationid='GHCND:USC00084289')

# Displays list of stations within the specified county
ncdc_stations(datasetid='GHCND', locationid='FIPS:12017')

# Displays list of Hourly Precipitation locationids between 01/01/1990 and 12/31/1990
ncdc_stations(datasetid='PRECIP_HLY', startdate='19900101', enddate='19901231')

# Search for stations by spatial extent
## Search using a single point, given by a lat long pair
ncdc_stations(extent=c(33.95,-118.40))
## Search using a bounding box, w/ lat/long of the SW corner, then of NE corner
ncdc_stations(extent=c(47.5204,-122.2047,47.6139,-122.1065))

## End(Not run)
```

`rnoaa-defunct` *Defunct functions in rnoaa*

Description

- noaa: Function name changed, prefixed with ncdc now
- noaa_datacats: Function name changed, prefixed with ncdc now
- noaa_datasets: Function name changed, prefixed with ncdc now
- noaa_datatypes: Function name changed, prefixed with ncdc now
- noaa_locs: Function name changed, prefixed with ncdc now
- noaa_locs_cats: Function name changed, prefixed with ncdc now
- noaa_stations: Function name changed, prefixed with ncdc now
- noaa_plot: Function name changed, prefixed with ncdc now
- noaa_combine: Function name changed, prefixed with ncdc now
- noaa_seaice: Function name changed to seaice

`seaice` *Get sea ice data.*

Description

Get sea ice data.

Usage

```
seaice(url, ...)
```

Arguments

`url`	A url for a NOAA sea ice ftp file
`...`	Further arguments passed on to readshpfile function, see `readshpfile`

Value

A data.frame

Examples

```
## Not run:
# Look at data.frame's for a series of years for Feb, South pole
urls <- sapply(seq(1979,1990,1), function(x) seaiceeurls(yr=x, mo='Feb', pole='S'))
out <- lapply(urls, seaice)
lapply(out, head)

# Map a single year/month/pole combo
urls <- seaiceeurls(mo='Apr', pole='N', yr=1990)
out <- seaice(urls)
library('ggplot2')
ggplot(out, aes(long, lat, group=group)) +
   geom_polygon(fill="steelblue") +
   theme_ice()

# Map all years for April only for North pole
library('plyr')
library('doMC')
urls <- seaiceeurls(mo='Apr', pole='N')
registerDoMC(cores=4)
out <- llply(urls, seaice, .parallel=TRUE)
names(out) <- seq(1979,2013,1)
df <- ldply(out)
ggplot(df, aes(long, lat, group=group)) +
  geom_polygon(fill="steelblue") +
  theme_ice() +
  facet_wrap(~ .id)

## End(Not run)
```

swdi *Get NOAA data for the severe weather data inventory (swdi).*

Description

Get NOAA data for the severe weather data inventory (swdi).

Usage

```
swdi(dataset = NULL, format = "xml", startdate = NULL, enddate = NULL,
  limit = 25, offset = NULL, radius = NULL, center = NULL,
  bbox = NULL, tile = NULL, stat = NULL, id = NULL, filepath = NULL,
  callopts = list())
```

Arguments

dataset	Dataset to query. See below for details.
format	File format to download. One of xml, csv, shp, or kmz.
startdate	Start date. See details.
enddate	End date. See details.
limit	Number of results to return. Defaults to 25. Any number from 1 to 10000000.
offset	Any number from 1 to 10000000. Default is NULL, no offset, start from 1.
radius	Search radius in miles (current limit is 15 miles)
center	Center coordinate in lon,lat decimal degree format, e.g.: c(-95.45,36.88)
bbox	Bounding box in format of minLon,minLat,maxLon,maxLat, e.g.: c(-91,30,-90,31)
tile	Coordinate in lon,lat decimal degree format, e.g.: c(-95.45,36.88) The lat/lon values are rounded to the nearest tenth of degree. For the above example, the matching tile would contain values from -95.4500 to -95.5499 and 36.8500 to 36.9499
stat	One of count or tilesum:$longitude,$latitude. Setting stat='count' returns number of results only (no actual data). stat='tilesum:$longitude,$latitude' returns daily feature counts for a tenth of a degree grid centered at the nearest tenth of a degree to the supplied values.
id	An identifier, e.g., 533623. Not sure how you find these ids?
filepath	If kmz or shp chosen the file name and optionally path to write to. Ignored format=xml or format=csv (optional)
callopts	Further arguments passed on to the API GET call. (optional)

Details

Options for the dataset parameter. One of (and their data formats):

- nx3tvs NEXRAD Level-3 Tornado Vortex Signatures (point)
- nx3meso NEXRAD Level-3 Mesocyclone Signatures (point)
- nx3hail NEXRAD Level-3 Hail Signatures (point)
- nx3structure NEXRAD Level-3 Storm Cell Structure Information (point)
- plsr Preliminary Local Storm Reports (point)
- warn Severe Thunderstorm, Tornado, Flash Flood and Special Marine warnings (polygon)
- nldn Lightning strikes from Vaisala (.gov and .mil ONLY) (point)

For startdate and enddate, the date range syntax is 'startDate:endDate' or special option of 'periodOfRecord'. Note that startDate is inclusive and endDate is exclusive. All dates and times are in GMT. The current limit of the date range size is one year.

All latitude and longitude values for input parameters and output data are in the WGS84 datum.

Value

If xml or csv chosen, a list of length three, a slot of metadata (meta), a slot for data (data), and a slot for shape file data with a single column 'shape'. The meta slot is a list of metadata elements, and the data slot is a data.frame, possibly of length zero if no data is found.

If kmz or shp chosen, the file is downloaded to your machine and a message is printed.

Examples

```
## Not run:
# Search for nx3tvs data from 5 May 2006 to 6 May 2006
swdi(dataset='nx3tvs', startdate='20060505', enddate='20060506')

# Get all 'nx3tvs' within 15 miles of latitude = 32.7 and longitude = -102.0
swdi(dataset='nx3tvs', startdate='20060506', enddate='20060507',
radius=15, center=c(-102.0,32.7))

# use an id
swdi(dataset='warn', startdate='20060506', enddate='20060507', id=533623)

# Get all 'plsr' within the bounding box (-91,30,-90,31)
swdi(dataset='plsr', startdate='20060505', enddate='20060510',
bbox=c(-91,30,-90,31))

# Get all 'nx3tvs' within the tile -102.1/32.6 (-102.15,32.55,-102.25,32.65)
swdi(dataset='nx3tvs', startdate='20060506', enddate='20060507',
tile=c(-102.12,32.62))

# Counts
## Note: stat='count' will only return metadata, nothing in the data or shape slots
## Note: stat='tilesum:...' returns counts in the data slot for each date for that tile,
##   and shape data
## Get number of 'nx3tvs' within 15 miles of latitude = 32.7 and longitude = -102.0
```

```
swdi(dataset='nx3tvs', startdate='20060505', enddate='20060516', radius=15,
center=c(-102.0,32.7), stat='count')

## Get daily count nx3tvs features on .1 degree grid centered at latitude = 32.7
## and longitude = -102.0
swdi(dataset='nx3tvs', startdate='20060505', enddate='20090516',
stat='tilesum:-102.0,32.7')

# CSV format
swdi(dataset='nx3tvs', startdate='20060505', enddate='20060506', format='csv')

# SHP format
swdi(dataset='nx3tvs', startdate='20060505', enddate='20060506', format='shp',
   filepath='myfile')

# KMZ format
swdi(dataset='nx3tvs', startdate='20060505', enddate='20060506', format='kmz',
   radius=15, filepath='myfile.kmz')

## End(Not run)
```

Index