Package 'rnoaa'

July 8, 2015

Title 'NOAA' Weather Data from R

Description Client for many 'NOAA' data sources including the 'NCDC' climate 'API' at http://www.ncdc.noaa.gov/cdo-web/webservices/v2, with 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, the 'NOAA' severe weather inventory, 'NOAA' Historical Observing 'Metadata' Repository ('HOMR') data, 'NOAA' storm data via 'IBTrACS', and tornado data via the 'NOAA' storm prediction center.

Version 0.4.2

License MIT + file LICENSE

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

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

LazyData true

VignetteBuilder knitr

Imports methods, stats, utils, httr (>= 1.0.0), lubridate, dplyr, tidyr, ggplot2, scales, rgdal, XML, jsonlite

Suggests testthat, roxygen2, knitr, taxize, ncdf

NeedsCompilation no

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

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2015-07-08 01:16:24

R topics documented:

rnoaa-package 2

buoy . . . 3
caching . . . 4
fipscodes . . . 5
ghcnd . . . 5
homr . . . 7
homr_definitions . . . 9
isd . . . 10
ncdc . . . 11
ncdc_combine . . . 15
ncdc_datacats . . . 16
ncdc_datasets . . . 18
ncdc_datatypes . . . 20
ncdc_locs . . . 21
ncdc_locs_cats . . . 23
ncdc_plot . . . 24
ncdc_stations . . . 25
rnoaa-defunct . . . 28
seaice . . . 29
storm_columns . . . 30
storm_names . . . 30
storm_shp . . . 30
swdi . . . 32
tornadoes . . . 35

Index **36**

`rnoaa-package` *General purpose R interface to NOAA datasets.*

Description

rnoaa is an R interface to NOAA climate data.

Details

Many 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, all of which functions start with `ncdc_`. An access token, or API key, is required to use all the `ncdc_` functions. The key is required by NOAA, not the creators of this R package. Go to the link given above to get an API key.

More NOAA data sources are being added through time. Data sources and their function prefixes are:

- `buoy_*` - NOAA Buoy data from the National Buoy Data Center
- `ghcnd_*` - GHCND daily data from NOAA
- `isd_*` - ISD/ISH data from NOAA
- `homr_*` - Historical Observing Metadata Repository (HOMR) vignette

- ncdc_* - NOAA National Climatic Data Center (NCDC) vignette (examples)
- seaice - Sea ice vignette
- storm_ - Storms (IBTrACS) vignette
- swdi - Severe Weather Data Inventory (SWDI) vignette
- tornadoes - From the NOAA Storm Prediction Center

A note about ncdf

Functions to work with buoy data use netcdf files. You'll need the ncdf package for those functions, and those only. ncdf is in Suggests in this package, meaning you only need ncdf if you are using the buoy functions. You'll get an informative error telling you to install ncdf if you don't have it and you try to use the buoy functions.

Installation of ncdf should be straightforward on Mac and Windows, but on Linux you may have issues. See http://cran.r-project.org/web/packages/ncdf/INSTALL

---

buoy *Get NOAA buoy data from the National Buoy Data Center*

Description

Get NOAA buoy data from the National Buoy Data Center

Usage

```
buoy(dataset, buoyid, year = NULL, datatype = NULL, ...)

buoys(dataset, ...)
```

Arguments

dataset	(character) Dataset name to query. See below for Details. Required
buoyid	(integer) Buoy ID. Required
year	(integer) Year of data collection
datatype	(character) Data type, one of 'c', 'cc', 'p', 'o'
...	Curl options passed on to GET (optional)

Details

Functions:

- buoys Get available buoys given a dataset name
- buoy Get data given some combination of dataset name, buoy ID, year, and datatype

Options for the dataset parameter. One of:

- adcp - Acoustic Doppler Current Profiler data

- adcp2 - MMS Acoustic Doppler Current Profiler data
- cwind - Continuous Winds data
- dart - Deep-ocean Assessment and Reporting of Tsunamis data
- mmbcur - Marsh-McBirney Current Measurements data
- ocean - Oceanographic data
- pwind - Peak Winds data
- stdmet- Standard Meteorological data
- swden - Spectral Wave Density data with Spectral Wave Direction data
- wlevel - Water Level data

References

http://www.ndbc.noaa.gov/ and http://dods.ndbc.noaa.gov/

Examples

```
## Not run:
# Get available buoys
buoys(dataset = 'cwind')

# Get data for a buoy
## if no year or datatype specified, we get the first file
buoy(dataset = 'cwind', buoyid = 46085)

# Including specific year
buoy(dataset = 'cwind', buoyid = 41001, year = 1999)

# Including specific year and datatype
buoy(dataset = 'cwind', buoyid = 41001, year = 2008, datatype = "cc")
buoy(dataset = 'cwind', buoyid = 41001, year = 2008, datatype = "cc")

# Other datasets
buoy(dataset = 'ocean', buoyid = 42856)

# curl debugging
library('httr')
buoy(dataset = 'cwind', buoyid = 46085, config=verbose())

## End(Not run)
```

---

caching *Clear cached files*

Description

Clear cached files

Usage

```
ghcnd_clear_cache(path = "~/.rnoaa/ghcnd", force = FALSE)
```

Arguments

path	Path to location of cached files. Defaults to `disk()$path`
force	(logical) Should we force removal of files if permissions say otherwise?

Details

BEWARE: this will clear all cached files.

---

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.

---

ghcnd *Get GHCND daily data from NOAA FTP server*

Description

Get GHCND daily data from NOAA FTP server

Usage

```
ghcnd(stationid, path = "~/.rnoaa/ghcnd", ...)

ghcnd_search(stationid, date_min = NULL, date_max = NULL, var = "all",
  path = "~/.rnoaa/ghcnd", ...)

ghcnd_splitvars(x)

ghcnd_stations(..., n = 10)

ghcnd_states(...)

ghcnd_countries(...)

ghcnd_version(...)
```

Arguments

stationid	Stationid to get
path	(character) A path to store the files, Default: ~/.rnoaa/isd
...	Curl options passed on to GET
date_min,date_max	(character) Minimum and maximum dates. Use together to get a date range
var	(character) Variable to get, defaults to "all", which gives back all variables in a list. To see what variables are available for a dataset, look at the dataset returned from ghcnd().
x	Input object to print methods. For ghcnd_splitvars(), the output of a call to ghcnd().
n	Number of rows to print

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>, Adam Erickson <adam.erickson@ubc.ca>

Examples

```
## Not run:
# Get metadata
ghcnd_states()
ghcnd_countries()
ghcnd_version()

# Get stations, ghcnd-stations and ghcnd-inventory merged
(stations <- ghcnd_stations())

# Get data
ghcnd(stationid = "AGE00147704")
ghcnd(stations$data$id[40])
```

```
ghcnd(stations$data$id[4000])
ghcnd(stations$data$id[10000])
ghcnd(stations$data$id[80000])
ghcnd(stations$data$id[80300])

library("dplyr")
ghcnd(stations$data$id[80300])$data %>% select(id, element) %>% head

# manipulate data
## using built in fxns
dat <- ghcnd(stationid="AGE00147704")
(alldat <- ghcnd_splitvars(dat))
library("ggplot2")
ggplot(subset(alldat$tmax, tmax >= 0), aes(date, tmax)) + geom_point()

## using dplyr
library("dplyr")
dat <- ghcnd(stationid="AGE00147704")
dat$data %>%
 filter(element == "PRCP", year == 1909)

# Search based on variable and/or date
ghcnd_search("AGE00147704", var = "PRCP")
ghcnd_search("AGE00147704", var = "PRCP", date_min = "1920-01-01")
ghcnd_search("AGE00147704", var = "PRCP", date_max = "1915-01-01")
ghcnd_search("AGE00147704", var = "PRCP", date_min = "1920-01-01", date_max = "1925-01-01")
ghcnd_search("AGE00147704", date_min = "1920-01-01", date_max = "1925-01-01")
ghcnd_search("AGE00147704", var = c("PRCP","TMIN"))
ghcnd_search("AGE00147704", var = c("PRCP","TMIN"), date_min = "1920-01-01")
ghcnd_search("AGE00147704", var="adfdf")

## End(Not run)
```

---

homr *Historical Observing Metadata Repository (HOMR) station metadata*

Description

Historical Observing Metadata Repository (HOMR) station metadata

Usage

```
homr(qid = NULL, qidMod = NULL, station = NULL, state = NULL,
  county = NULL, country = NULL, name = NULL, nameMod = NULL,
  platform = NULL, date = NULL, begindate = NULL, enddate = NULL,
  headersOnly = FALSE, phrData = NULL, combine = FALSE, ...)
```

Arguments

qid	One of COOP, FAA, GHCND, ICAO, NCDCSTNID, NWSLI, TRANS, WBAN, or WMO, or any of those plus [a-z0-9], or just [a-z0-9]. (qid = qualified ID)
qidMod	(character) One of: is, starts, ends, contains. Specifies how the ID portion of the qid parameter should be applied within the search. If a qid is passed but the qidMod parameter is not used, qidMod is assumed to be IS.
station	(character) A station id.
state	(character) A two-letter state abbreviation. Two-letter code for US states, Canadian provinces, and other Island areas.
county	(character) A two letter county code. US county names, best used with a state identifier.
country	(character) A two letter country code. See here for a list of valid country names.
name	(character) One of name=[0-9A-Z]+. Searches on any type of name we have for the station.
nameMod	(character) [is|starts|ends|contains]. Specifies how the name parameter should be applied within the search. If a name is passed but the nameMod parameter is not used, nameMod is assumed to be IS.
platform	(character) (aka network) [ASOS|USCRN|USHCN|NEXRAD|AL USRCRN|USRCRN|COOP]. Limit the search to stations of a certain platform/network type.
date	(character) [YYYY-MM-DD|all] Limits values to only those that occurred on a specific date. Alternatively, date=all will return all values for matched stations. If this field is omitted, the search will return only the most recent values for each field.
begindate,enddate	[YYYY-MM-DD]. Limits values to only those that occurred within a date range.
headersOnly	(logical) Returns only minimal information for each station found (NCDC Station ID, Preferred Name, Station Begin Date, and Station End Date), but is much quicker than a full query. If you are performing a search that returns a large number of stations and intend to choose only one from that list to examine in detail, headersOnly may give you enough information to find the NCDC Station ID for the station that you actually want.
phrData	(logical) The HOMR web service now includes PHR (element-level) data when available, in an elements section. Because of how this data is structured, it can substantially increase the size of any result which includes it. If you don't need this data you can omit it by including phrData=false. If the parameter is not set, it will default to phrData=true.
combine	(logical) Combine station metadata or not.
...	Curl options passed on to GET (optional)

Details

Since the definitions for variables are always the same, we don't include the ability to get description data in this function. Use link[rnoaa]{homr_descriptions} to get descriptions information.

Value

A list, with elements named by the station ids.

References

http://www.ncdc.noaa.gov/homr/api

Examples

```
## Not run:
homr(qid = 'COOP:046742')
homr(headersOnly=TRUE, qid='TRANS:')
homr(qid = ':046742')
homr(qidMod='starts', qid='COOP:0467')
homr(headersOnly=TRUE, state='DE')
homr(headersOnly=TRUE, country='GHANA')
homr(headersOnly=TRUE, state='NC', county='BUNCOMBE')
homr(name='CLAYTON')
res <- homr(state='NC', county='BUNCOMBE', combine=TRUE)
res$id
res$head
res$updates
homr(nameMod='starts', name='CLAY')
homr(headersOnly=TRUE, platform='ASOS')
homr(qid='COOP:046742', date='2011-01-01')
homr(qid='COOP:046742', begindate='2005-01-01', enddate='2011-01-01')
homr(state='DE', headersOnly=TRUE)
homr(station=20002078)
homr(station=20002078, date='all', phrData=FALSE)

# Optionally pass in curl options
homr(headersOnly=TRUE, state='NC', county='BUNCOMBE', config=verbose())

## End(Not run)
```

---

homr_definitions *Historical Observing Metadata Repository (HOMR) station metadata - definitions*

Description

Historical Observing Metadata Repository (HOMR) station metadata - definitions

Usage

```
homr_definitions(...)
```

Arguments

...	Curl options passed on to GET (optional)

Examples

```
## Not run:
head( homr_definitions() )

## End(Not run)
```

`isd` *Get NOAA ISD/ISH data from NOAA FTP server.*

Description

Get NOAA ISD/ISH data from NOAA FTP server.

Usage

```
isd(usaf, wban, year, path = "~/.rnoaa/isd", overwrite = TRUE, ...)

isd_stations(...)
```

Arguments

`usaf,wban`	(character) USAF and WBAN code. Required
`year`	(numeric) One of the years from 1901 to the current year. Required.
`path`	(character) A path to store the files, a directory. Default: `~/.rnoaa/isd`. Required.
`overwrite`	(logical) To overwrite the path to store files in or not, Default: `TRUE`
`...`	Curl options passed on to `GET`

Details

This function first looks for whether the data for your specific query has already been downloaded previously in the directory given by the `path` parameter. If not found, the data is requested form NOAA's FTP server. The first time a dataset is pulled down we must a) download the data, b) process the data, and c) save a .csv file to disk. The next time the same data is requested, we only have to read back in the .csv file, and is quite fast. The processing can take quite a long time since the data is quite messy and takes a bunch of regex to split apart text strings. See examples below for different behavior.

References

ftp://ftp.ncdc.noaa.gov/pub/data/noaa/

Examples

```
## Not run:
# Get station table
stations <- isd_stations()
head(stations)

# Get data
(res <- isd(usaf="011490", wban="99999", year=1986))
(res <- isd(usaf="011690", wban="99999", year=1993))
(res <- isd(usaf="172007", wban="99999", year=2015))
(res <- isd(usaf="702700", wban="00489", year=2015))

# The first time a dataset is requested takes longer
system.time( isd(usaf="782680", wban="99999", year=2011) )
system.time( isd(usaf="782680", wban="99999", year=2011) )

# Optionally pass in curl options
res <- isd(usaf="011490", wban="99999", year=1986, config = verbose())

## End(Not run)
```

---

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, 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

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

`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.

`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")

`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.

`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.

`results` THIS IS A DEPRECATED ARGUMENT. See limit.

`...` Curl options passed on to `GET` (optional)

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)
```

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

## End(Not run)
```

---

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, 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

stationid	Accepts a valid station id. Data returned will contain data for the station(s) specified (optional)
locationid	Accepts a valid location id. Data returned will contain data for the location(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")
...	Further named parameters, such as query, path, etc, passed on to modify_url. Unnamed parameters will be combined with config.

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 10 results
ncdc_datacats(limit=10)

## 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')

## Curl debugging
ncdc_datacats(limit=10, config=verbose())
out <- ncdc_datacats(limit=10, config=progress())

## 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, token = NULL,
  dataset = NULL, page = NULL, year = NULL, month = NULL, ...)
```

Arguments

`datasetid`	(optional) Accepts a single valid dataset id. Data returned will be from the dataset specified, see datasets()
`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)
`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)
`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)
`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.
`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")
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.
...	Curl options passed on to GET (optional)

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,
  token = NULL, dataset = NULL, page = NULL, filter = NULL, ...)
```

Arguments

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

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.
...	Curl options passed on to GET (optional)

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, token = NULL, ...)
```

Arguments

datasetid	A single valid dataset id. Data returned will be from the dataset specified, see datasets() (required)
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)
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")
...	Further named parameters, such as query, path, etc, passed on to modify_url. Unnamed parameters will be combined with config.

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, 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")
`...`	Further named parameters, such as `query`, `path`, etc, passed on to `modify_url`. Unnamed parameters will be combined with `config`.

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(sub("T00:00:00\.000", '', as.character(input$date))) 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, token = NULL, dataset = NULL, station = NULL,
  location = NULL, locationtype = NULL, page = NULL, ...)
```

Arguments

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)
datasetid	(optional) Accepts a single valid dataset id. Data returned will be from the dataset specified, see datasets()
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)
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.
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)
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 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)`.
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")
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.
...	Curl options passed on to GET (optional)

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 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
- erddap_data: See package rerddap
- erddap_clear_cache: See package rerddap
- erddap_datasets: Moved to `ed_datasets` in package rerddap
- erddap_grid: Moved to `griddap` in package rerddap
- erddap_info: Moved to `info` in package rerddap
- erddap_search: Moved to `ed_search` in package rerddap
- erddap_table: Moved to `tabledap` in package rerddap
- ncdc_leg_variables: Removed. See `NCDC Legacy` below
- ncdc_leg_sites: Removed. See `NCDC Legacy` below
- ncdc_leg_site_info: Removed. See `NCDC Legacy` below
- ncdc_leg_data: Removed. See `NCDC Legacy` below

NCDC Legacy

The NCDC legacy API is too unreliable and slow. Use the newer NCDC API via the functions ncdc, ncdc_datacats, ncdc_datasets, ncdc_datatypes, ncdc_locs, ncdc_locs_cats, ncdc_stations, ncdc_plot, and ncdc_combine

`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`

Details

If you want to reproject the shape files, use `readshpfile` to read in shape file, then reproject, and so on.

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()

## End(Not run)
```

storm_columns *NOAA storm column descriptions for data from IBTrACS*

Description

This dataset includes description of the columns of each dataset acquired using `storm_data`

Format

A data frame with 195 rows and 8 variables

storm_names *NOAA storm names from IBTrACS*

Description

This dataset includes a crosswalk from storm serial numbers to their names. Storm serial numbers are used to search for storms in the `storm_data` function.

Format

A data frame with 12,209 rows and 2 variables

storm_shp *Get NOAA wind storm tabular data, metadata, or shp files from IBTrACS*

Description

Get NOAA wind storm tabular data, metadata, or shp files from IBTrACS

Usage

```
storm_shp(basin = NULL, storm = NULL, year = NULL, type = "points",
  path = "~/.rnoaa/storms", overwrite = TRUE)

storm_shp_read(x)

storm_data(basin = NULL, storm = NULL, year = NULL,
  path = "~/.rnoaa/storms", overwrite = TRUE, ...)

storm_meta(what = "storm_columns")
```

Arguments

basin	(character) A basin name, one of EP, NA, NI, SA, SI, SP, or WP.
storm	(character) A storm serial number of the form YYYYJJJHTTNNN. See Details.
year	(numeric) One of the years from 1842 to 2014
type	(character) One of points or lines. This gives shp files with points, or with lines.
path	(character) A path to store the files, Default: ~/.rnoaa/storms
overwrite	(logical) To overwrite the path to store files in or not, Default: TRUE.
x	Output from storm_shp, a path to shp file to read in.
...	Curl options passed on to GET (optional)
what	(character) One of storm_columns or storm_names.

Details

International Best Track Archive for Climate Stewardship (IBTrACS)

Details for storm serial numbers:

- YYYY is the corresponding year of the first recorded observation of the storm
- JJJ is the day of year of the first recorded observation of the storm
- H is the hemisphere of the storm: N=Northern, S=Southern
- TT is the absolute value of the rounded latitude of the first recorded observation of the storm (range 0-90, if basin=SA or SH, then TT in reality is negative)
- NNN is the rounded longitude of the first recorded observation of the storm (range 0-359)

For example: 1970143N19091 is a storm in the North Atlantic which started on May 23, 1970 near 19N 91E

See http://www.ncdc.noaa.gov/ibtracs/index.php?name=numbering for more.

The datasets included in the package storm_names, and storm_columns may help in using these storm functions.

References

http://www.ncdc.noaa.gov/ibtracs/index.php?name=wmo-data

Examples

```
## Not run:
# Metadata
head( storm_meta() )
head( storm_meta("storm_columns") )
head( storm_meta("storm_names") )

# Tabular data
## Get tabular data for basins, storms, or years
storm_data(basin='WP')
storm_data(storm='1970143N19091')
storm_data(year=1940)
```

```
storm_data(year=1941)
storm_data(year=2010)

# shp files
## storm_shp downloads data and gives a path back
## to read in, use storm_shp_read
res <- storm_shp(basin='EP')
storm_shp_read(res)

## Get shp file for a storm
(res2 <- storm_shp(storm='1970143N19091'))

## Plot shp file data, we'll need sp library
library('sp')

### for year 1940, points
(res3 <- storm_shp(year=1940))
res3shp <- storm_shp_read(res3)
plot(res3shp)

### for year 1940, lines
(res3_lines <- storm_shp(year=1940, type="lines"))
res3_linesshp <- storm_shp_read(x=res3_lines)
plot(res3_linesshp)

### for year 2010, points
(res4 <- storm_shp(year=2010))
res4shp <- storm_shp_read(res4)
plot(res4shp)

## 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,
  ...)
```

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)
...	Curl options passed on to GET (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')

# csv output to SpatialPointsDataFrame
res <- swdi(dataset='nx3tvs', startdate='20060505', enddate='20060506', format="csv")
library('sp')
coordinates(res$data) <- ~lon + lat
res$data
class(res$data)

## End(Not run)
```

`tornadoes` *Get NOAA tornado data.*

Description

Get NOAA tornado data.

Usage

```
tornadoes(path = "~/.rnoaa/tornadoes", overwrite = TRUE, ...)
```

Arguments

`path`	A path to store the files, Default: `~/.ots/kelp`
`overwrite`	(logical) To overwrite the path to store files in or not, Default: TRUE.
`...`	Curl options passed on to `GET` (optional)

Value

A Spatial object is returned of class SpatialLinesDataFrame.

References

http://www.spc.noaa.gov/gis/svrgis/

Examples

```
## Not run:
shp <- tornadoes()
library('sp')
plot(shp) # may take 10 sec or so to render

## End(Not run)
```

Index

∗Topic **datasets**
    fipscodes, 5
    storm_columns, 30
    storm_names, 30
∗Topic **package**
    rnoaa-package, 2

buoy, 3
buoys *(*buoy*)*, 3

caching, 4
config, *17*, *22, 23*

date_breaks, *24*
date_format, *24*

erddap_clear_cache, *28*
erddap_data, *28*
erddap_datasets, *28*
erddap_grid, *28*
erddap_info, *28*
erddap_search, *28*
erddap_table, *28*

fipscodes, 5

GET, *3*, *6*, *8–10*, *12*, *19*, *21*, *27*, *31*, *33*, *35*
ghcnd, 5
ghcnd_clear_cache *(*caching*)*, 4
ghcnd_countries *(*ghcnd*)*, 5
ghcnd_search *(*ghcnd*)*, 5
ghcnd_splitvars *(*ghcnd*)*, 5
ghcnd_states *(*ghcnd*)*, 5
ghcnd_stations *(*ghcnd*)*, 5
ghcnd_version *(*ghcnd*)*, 5

homr, 7
homr_definitions, 9

isd, 10
isd_stations *(*isd*)*, 10

modify_url, *17*, *22, 23*

ncdc, 11, *24, 25*, *28*
ncdc_combine, 15, *28*
ncdc_datacats, 16, *28*
ncdc_datasets, 18, *28*
ncdc_datatypes, 20, *28*
ncdc_leg_data, *28*
ncdc_leg_site_info, *28*
ncdc_leg_sites, *28*
ncdc_leg_variables, *28*
ncdc_locs, 21, *28*
ncdc_locs_cats, 23, *28*
ncdc_plot, 24, *28*
ncdc_stations, 25, *28*
noaa, *28*
noaa_combine, *28*
noaa_datacats, *28*
noaa_datasets, *28*
noaa_datatypes, *28*
noaa_locs, *28*
noaa_locs_cats, *28*
noaa_plot, *28*
noaa_seaice, *28*
noaa_stations, *28*

readshpfile, *29*
rnoaa *(*rnoaa-package*)*, 2
rnoaa-defunct, 28
rnoaa-package, 2

seaice, 29
storm_columns, 30, *31*
storm_data, *30*
storm_data *(*storm_shp*)*, 30
storm_meta *(*storm_shp*)*, 30
storm_names, 30, *31*
storm_shp, 30
storm_shp_read *(*storm_shp*)*, 30
swdi, 32

tornadoes, 35