Package 'traits'

August 29, 2016

Title Species Trait Data from Around the Web

Description Species trait data from many different sources, including sequence data from 'NCBI', plant trait data from 'BETYdb', invasive species data from the Global Invasive Species Database and 'EOL', 'Traitbank' data from 'EOL', Coral traits data from http://coraltraits.org, 'nativity' status ('Flora Europaea' or 'ITIS'), and 'Birdlife' International.

Version 0.2.0

License MIT + file LICENSE

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

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

LazyData true

VignetteBuilder knitr

Depends R (>= 2.10)

Imports methods, stats, utils, jsonlite (>= 0.9.19), httr (>= 1.1.0), dplyr (>= 0.4.3), data.table (>= 1.9.6), readr (>= 0.2.2), taxize (>= 0.7.4), xml2 (>= 0.1.2), rvest (>= 0.3.1)

Suggests roxygen2 (>= 5.0.1), knitr, testthat, plyr, covr

RoxygenNote 5.0.1

NeedsCompilation no

Author Scott Chamberlain [aut, cre], Zachary Foster [aut], Ignasi Bartomeus [aut], David LeBauer [aut], David Harris [aut]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2016-03-18 15:58:47

R topics documented:

`traits-package` *traits - Species trait data from around the web*

Description

Currrently included in `traits` with the associated function prefix:

- BETYdb http://www.betydb.org - `betydb_`
- National Center for Biotechnology Information - NCBI http://www.ncbi.nlm.nih.gov/ - `ncbi_`
- Global Invasive Species Database - GISD http://www.issg.org/database/welcome/ - `g_`
- Encyclopedia of Life Invasive Species - `eol_`
- Encyclopedia of Life Traitbank - `traitbank_`
- Coral Traits Database http://coraltraits.org/ - `coral_`
- Flora Europaea http://rbg-web2.rbge.org.uk/FE/fe.html - `fe_`
- Birdlife International http://rbg-web2.rbge.org.uk/FE/fe.html - `birdlife_`
- LEDA Traitbase http://www.leda-traitbase.org/LEDAportal/index.jsp - `leda_`
- More to come ...

Author(s)

Ignasi Bartomeus <nacho.bartomeus@gmail.com>

Zachary Foster <zacharyfoster1989@gmail.com>

David LeBauer <dlebauer@gmail.com>

David Harris <davharris@ucdavis.edu>

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
library("traits")

## Search the Coral database
## Get the species list and their ids
coral_species()

## Get data by taxon
coral_taxa(80)

## End(Not run)
```

---

betydb *Search for traits from BETYdb*

Description

Search for traits from BETYdb

Usage

```
betydb_search(query = "Maple SLA", fmt = "json", key = NULL,
  user = NULL, pwd = NULL, ...)

betydb_trait(id, genus = NULL, species = NULL, fmt = "json", key = NULL,
  user = NULL, pwd = NULL, ...)

betydb_specie(id, genus = NULL, species = NULL, fmt = "json",
  key = NULL, user = NULL, pwd = NULL, ...)

betydb_citation(id, genus = NULL, species = NULL, fmt = "json",
  key = NULL, user = NULL, pwd = NULL, ...)

betydb_site(id, fmt = "json", key = NULL, user = NULL, pwd = NULL, ...)
```

Arguments

query	Query terms
fmt	(character) Format to return data in, one of json, xml, csv. Only json currently supported.
key	(character) An API key. Use this or user/pwd combo. Save in your .Rprofile file as betydb_key. Optional
user, pwd	(character) A user name and password. Use a user/pwd combo or an API key. Save in your .Rprofile file as betydb_user and betydb_pwd. Optional
...	Curl options passed on to GET. Optional
id	(integer) One or more ids for a species, site, variable, etc.
genus	(character) A genus name. Optional
species	(character) A specific epithet. Optional

Details

Details:

Authentication

Defers to use API key first since it's simpler, but if you don't have an API key, you can supply a username and password.

Functions

Singular functions like betydb_trait accept an id and additional parameters, and return a list of variable outputs depending on the inputs.

However, plural functions like betydb_traits accept query parameters, but not ids, and always return a single data.frame.

References

API documentation https://www.authorea.com/users/5574/articles/7062

Examples

```
## Not run:
# General Search
out <- betydb_search(query = "Switchgrass Yield")
library("dplyr")
out %>%
 group_by(id) %>%
 summarise(mean_result = mean(as.numeric(mean), na.rm = TRUE)) %>%
 arrange(desc(mean_result))
# Get by ID
## Traits
betydb_trait(id = 10)
## Species
betydb_specie(id = 1)
```

```
## Citations
betydb_citation(id = 1)
## Site information
betydb_site(id = 795)

## End(Not run)
```

`birdlife_habitat` *Get bird habitat information from BirdLife/IUCN*

Description

Get bird habitat information from BirdLife/IUCN

Usage

```
birdlife_habitat(id)
```

Arguments

`id`	A single IUCN species ID

Value

a `data.frame` with level 1 and level 2 habitat classes, as well as importance ratings and occurrence type (e.g. breeding or non-breeding). The habitat classification scheme is described at http://bit.ly/1e6gKBr

Author(s)

David J. Harris <harry491@gmail.com>

See Also

`birdlife_threats`

Examples

```
## Not run:
birdlife_habitat(22721692)

## End(Not run)
```

birdlife_threats *Get bird threat information from BirdLife/IUCN*

Description

Get bird threat information from BirdLife/IUCN

Usage

```
birdlife_threats(id)
```

Arguments

id	A single IUCN species ID

Value

a `data.frame` with the species ID and two levels of threat descriptions, plus stresses, timing, scope, severity, and impact associated with each stressor.

Author(s)

David J. Harris <harry491@gmail.com>

See Also

`birdlife_habitat`

Examples

```
## Not run:
birdlife_threats(22721692)

## End(Not run)
```

coral *Search for coral data on coraltraits.org*

Description

Search for coral data on coraltraits.org

Usage

```
coral_taxa(taxon, taxonomy = FALSE, contextual = TRUE, global = FALSE,
  ...)

coral_traits(trait, taxonomy = FALSE, contextual = TRUE, global = FALSE,
  ...)

coral_locations(location, taxonomy = FALSE, contextual = TRUE,
  global = FALSE, ...)

coral_methodologies(methodology, taxonomy = FALSE, contextual = TRUE,
  global = FALSE, ...)

coral_resources(resource, taxonomy = FALSE, contextual = TRUE,
  global = FALSE, ...)

coral_species(...)
```

Arguments

taxon	A taxon id
taxonomy	logical; Include contextual data. Default: FALSE
contextual	logical; Include contextual data. Default: TRUE
global	logical; Include contextual data. Default: FALSE
...	Curl options passed on to GET
trait	A trait id
location	A location id
methodology	A methodology id
resource	A resource id

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://coraltraits.org/

Examples

```
## Not run:
# Get the species and their Ids
head( coral_species() )

# Get data by taxon
coral_taxa(8)
```

```
# Get data by trait
coral_traits(3)

# Get data by methodology
coral_methodologies(2)

# Get data by location
coral_locations(132)

# Get data by resource
coral_resources(9)

# curl options
library("httr")
coral_taxa(8, config=verbose())

## End(Not run)
```

`eol_invasive_` *Search for presence of taxonomic names in EOL invasive species databases.*

Description

See Details for important information.

Usage

```
eol_invasive_(name = NULL, dataset = "all", searchby = grep,
  page = NULL, per_page = NULL, key = NULL, verbose = TRUE,
  count = FALSE, ...)
```

Arguments

`name`	A taxonomic name, or a vector of names.
`dataset`	One of all, gisd100, gisd, isc, daisie, i3n, or mineps. See the Details for what each dataset ID.
`searchby`	One of 'grep' (exact match) or 'agrep' (fuzzy match)
`page`	A maximum of 30 results are returned per page. This parameter allows you to fetch more pages of results if there are more than 30 matches (Default 1)
`per_page`	Results to get per page
`key`	Your EOL API key; loads from .Rprofile.
`verbose`	(logical) If TRUE the actual taxon queried is printed on the console.
`count`	(logical) If TRUE, give back a count of number of taxa listed as invasive, if FALSE (default), the normal output is given.
`...`	Further args passed on to `GET`

Details

IMPORTANT: When you get a returned NaN for a taxon, that means it's not on the invasive list in question. If the taxon is found, a taxon identifier is returned.

Beware that some datasets are quite large, and may take 30 sec to a minute to pull down all data before we can search for your species. Note there is no parameter in this API method for searching by taxon name.

This function is vectorized, so you can pass a single name or a vector of names.

It's possible to return JSON or XML with the EOL API. However, this function only returns JSON for now.

Options for the dataset parameter are

- all - All datasets
- gisd100 - 100 of the World's Worst Invasive Alien Species (Global Invasive Species Database) http://eol.org/collections/54500
- gisd - Global Invasive Species Database 2013 http://eol.org/collections/54983
- isc - Centre for Agriculture and Biosciences International Invasive Species Compendium (ISC) http://eol.org/collections/55180
- daisie - Delivering Alien Invasive Species Inventories for Europe (DAISIE) Species List http://eol.org/collections/55179
- i3n - IABIN Invasives Information Network (I3N) Species http://eol.org/collections/55176
- mineps - Marine Invaders of the NE Pacific Species http://eol.org/collections/55331

Datasets are not updated that often. Here's last updated dates for some of the datasets as of 2014-08-25

- gisd100 updated 6 mos ago
- gisd updated 1 yr ago
- isc updated 1 yr ago
- daisie updated 1 yr ago
- i3n updated 1 yr ago
- mineps updated 1 yr ago

Value

A list of data.frame's/strings with results, with each element named by the input elements to the name parameter.

References

See info for each data source at http://eol.org/collections/55367/taxa

Examples

```
## Not run:
eol_invasive_(name='Brassica oleracea', dataset='gisd')
eol_invasive_(name=c('Lymantria dispar','Cygnus olor','Hydrilla verticillata','Pinus concolor'),
   dataset='gisd')
eol_invasive_(name='Sargassum', dataset='gisd')
eol_invasive_(name='Ciona intestinalis', dataset='mineps')
eol_invasive_(name=c('Lymantria dispar','Cygnus olor','Hydrilla verticillata','Pinus concolor'),
   dataset='i3n')
eol_invasive_(name=c('Branta canadensis','Gallus gallus','Myiopsitta monachus'),
   dataset='daisie')
eol_invasive_(name=c('Branta canadensis','Gallus gallus','Myiopsitta monachus'), dataset='isc')

# Count
eol_invasive_(name=c('Lymantria dispar','Cygnus olor','Hydrilla verticillata','Pinus concolor'),
   dataset='gisd', count = TRUE)

## End(Not run)
```

`fe_native` *Check species status (native, exotic, ...) for one species from Flora Europaea webpage*

Description

This function check the status (native or exotic) of a species in each of the eu countries.

For that end, it checks Flora Europaea (http://rbg-web2.rbge.org.uk/FE/fe.html) and scrapes the data from there.

Note that the webpage contains more information.

As expected, the function is as good as the database is. I think for native species is robust but new exotic species are not added as to my knowledge the database is not updated anymore. The database is not able to recognize species synonyms.

See `http://rbg-web2.rbge.org.uk/FE/data/countries` for explanation of the database codes.

Usage

```
fe_native(sp, ...)
```

Arguments

`sp`	character; a vector of length one with a single scientific species names in the form of `c("Genus species")`.
`...`	Curl options passed on to `GET`

Value

A list of vectors containing the countries where the species is native, exotic, ...

Author(s)

Ignasi Bartomeus <nacho.bartomeus@gmail.com>

Examples

```
## Not run:
sp <- c("Lavandula stoechas", "Carpobrotus edulis", "Rhododendron ponticum",
       "Alkanna lutea", "Anchusa arvensis")
fe_native(sp[1])
sapply(sp, fe_native, simplify = FALSE)

## End(Not run)
```

---

g_invasive *Check invasive species status for a set of species from GISD database*

Description

This function check which species (both plants and animals) are considered "invaders" somewhere in the world.

For that end, it checks GISD (http://www.issg.org/database/welcome/) and returns a value, either "Not in GISD" or the brief description presented in GISD.

Note that the webpage contains more information. Also note that the function won't tell you if it's exotic in your area, a lot of exotic species are not considered invaders (yet).

As expected, the function is as good as the database is, which I find quite reliable and well maintained. The database is also able to recognize a lot (but not all) of the species synonyms.

Note that `eol_invasive` with source of gisd or gisd100 may end up with different results as this function goes directly to the GISD website, whereas eol_invasive only updates their GISD data occassionally. See notes in `eol_invasive`.

Usage

```
g_invasive(x, simplify = FALSE, verbose = TRUE)
```

Arguments

`x`	character; a vector of scientific species names in the form of c("Genus species").
`simplify`	logical; returns a data.frame with the species name and the values "Invasive", "Not in GISD". I recomend to check first the not simplified version (default), which contains raw information about the level of invasiveness.
`verbose`	logical; If TRUE (default), informative messages printed.

Value

A data.frame with species names and invasiveness.

Author(s)

Ignasi Bartomeus <nacho.bartomeus@gmail.com>

Examples

```
## Not run:
sp <- c("Carpobrotus edulis", "Rosmarinus officinalis")
## first species is invasive, second one is not.
g_invasive(sp)
g_invasive(sp, simplify = TRUE)

## End(Not run)
```

`is_native` *Check if a species is native somewhere*

Description

This function check the status (native or exotic) of a species in a given place

For that end, calls `itis_native` and `fe_native`. See help documentation of those functions for details.

So many more things can be done, like checking species first with **taxize**, adding more native lists to check...

Usage

```
is_native(sp, where, region = c("america", "europe"), ...)
```

Arguments

`sp`	character; a vector of length one with a single scientific species names in the form of `c("Genus species")`.
`where`	character; a vector of length one with a single place. For America has to match one of those: "Continental US", "Alaska", "Canada", "Caribbean Territories", "Central Pacific Territories", "Hawaii", "Mexico". For Europe has to match one of those: "Albania", "Austria", "Azores", "Belgium", "Islas_Baleares", "Britain", "Bulgaria", "Corse", "Kriti", "Czechoslovakia", "Denmark", "Faroer", "Finland", "France", "Germany", "Greece", "Ireland", "Switzerland", "Netherlands", "Spain", "Hungary", "Iceland", "Italy", "Jugoslavia", "Portugal", "Norway", "Poland", "Romania", "USSR", "Sardegna", "Svalbard", "Sicilia", "Sweden", "Turkey", "USSR_Northern_Division", "USSR_Baltic_Division", "USSR_Central_Division", "USSR_South_western", "USSR_Krym", "USSRSouth_eastern_Division"
`region`	character; a vector of length one with a single region. Only "europe" and "america" implemented "europe" checks Flora Europaea and only contain plants. "america" checks ITIS and contain both plant and animals.
`...`	Curl options passed on to `GET`

Value

A data.frame, with species name and result of origin check

Author(s)

Ignasi Bartomeus <nacho.bartomeus@gmail.com>

Examples

```
## Not run:
sp <- c("Lavandula stoechas", "Carpobrotus edulis", "Rhododendron ponticum",
       "Alkanna lutea", "Anchusa arvensis")
is_native(sp[1], where = "Islas_Baleares", region = "europe")
lapply(sp, is_native, where = "Continental US", region = "america")
lapply(sp, is_native, where = "Islas_Baleares", region = "europe")

# combine output for many taxa
res <- lapply(sp, is_native, where = "Continental US", region = "america")
library("dplyr")
rbind_all(res)

## End(Not run)
```

---

leda *Access LEDA trait data*

Description

Access LEDA trait data

Usage

```
leda(trait = "age_first_flowering", ...)
```

Arguments

`trait`	(character) Trait to get. See Details.
`...`	Curl options passed on to `GET`

Details

For parameter `trait`, one of age_first_flowering, branching, buds_seasonality, buds_vertical_dist, canopy_height, dispersal_type, leaf_distribution, ldmc_geo, leaf_mass, leaf_size, morphology_disperal, growth_form, life_span, releasing_height, seed_longevity, seed_mass, seed_number, seed_shape, shoot_growth_form, snp, ssd, tv, or clonal_growth_organs

The following are not supported as they are too much of a pain to parse: buoyancy, seed_bank, sla_geo

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
# Age of first flowering
leda(trait = "age_first_flowering")

# Seed number
leda("seed_number")

# Releasing height
leda(trait = "releasing_height")

# Clonal growth organs
leda(trait = "clonal_growth_organs")

all <- c("age_first_flowering", "branching", "buds_seasonality",
  "buds_vertical_dist", "canopy_height",
  "dispersal_type", "leaf_distribution", "ldmc_geo", "leaf_mass",
  "leaf_size", "morphology_disperal", "growth_form", "life_span",
  "releasing_height", "seed_longevity", "seed_mass",
  "seed_number", "seed_shape", "shoot_growth_form",
  "snp", "ssd", "tv", "clonal_growth_organs")
out <- list()
for (i in seq_along(all)) {
  cat(all[i], sep="\n")
  out[[i]] <- leda(all[i])
}
sapply(out, NROW)

## End(Not run)
```

---

ncbi_byid *Retrieve gene sequences from NCBI by accession number.*

Description

Retrieve gene sequences from NCBI by accession number.

Usage

```
ncbi_byid(ids, format = NULL, verbose = TRUE)
```

Arguments

ids	(character) GenBank ids to search for. One or more. Required.
format	(character) Return type, e.g., "fasta". NOW IGNORED.
verbose	(logical) If TRUE (default), informative messages printed.

Details

If bad ids are included with good ones, the bad ones are silently dropped. If all ids are bad you'll get a stop with error message.

Value

Data.frame of the form:

- taxon - taxonomic name (may include some junk, but hard to parse off)
- gene_desc - gene description
- gi_no - GI number
- acc_no - accession number
- length - sequence length
- sequence - sequence character string

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

See Also

ncbi_search, ncbi_getbyname

Examples

```
## Not run:
# A single gene
ncbi_byid(ids="360040093")

# Many genes (with different accession numbers)
ncbi_byid(ids=c("360040093","347448433"))

## End(Not run)
```

---

ncbi_byname *Retrieve gene sequences from NCBI by taxon name and gene names.*

Description

Retrieve gene sequences from NCBI by taxon name and gene names.

Usage

```
ncbi_byname(taxa, gene = "COI", seqrange = "1:3000", getrelated = FALSE,
  verbose = TRUE)
```

Arguments

taxa	(character) Scientific name to search for.
gene	(character) Gene or genes (in a vector) to search for. See examples.
seqrange	(character) Sequence range, as e.g., "1:1000". This is the range of sequence lengths to search for. So "1:1000" means search for sequences from 1 to 1000 characters in length.
getrelated	(logical) If TRUE, gets the longest sequences of a species in the same genus as the one searched for. If FALSE, returns nothing if no match found.
verbose	(logical) If TRUE (default), informative messages printed.

Details

Removes predicted sequences so you don't have to remove them. Predicted sequences are those with accession numbers that have "XM_" or "XR_" prefixes. This function retrieves one sequences for each species, picking the longest available for the given gene.

Value

Data.frame of results.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

See Also

ncbi_search, ncbi_getbyid

Examples

```
## Not run:
# A single species
ncbi_byname(taxa="Acipenser brevirostrum")

# Many species
species <- c("Colletes similis","Halictus ligatus","Perdita trisignata")
ncbi_byname(taxa=species, gene = c("coi", "co1"), seqrange = "1:2000")

## End(Not run)
```

ncbi_searcher *Search for gene sequences available for taxa from NCBI.*

Description

Search for gene sequences available for taxa from NCBI.

Usage

```
ncbi_searcher(taxa = NULL, id = NULL, seqrange = "1:3000",
  getrelated = FALSE, fuzzy = FALSE, limit = 500, entrez_query = NULL,
  hypothetical = FALSE, verbose = TRUE)
```

Arguments

taxa	(character) Scientific name to search for.
id	(character) Taxonomic id to search for. Not compatible with argument taxa.
seqrange	(character) Sequence range, as e.g., "1:1000". This is the range of sequence lengths to search for. So "1:1000" means search for sequences from 1 to 1000 characters in length.
getrelated	(logical) If TRUE, gets the longest sequences of a species in the same genus as the one searched for. If FALSE, returns nothing if no match found.
fuzzy	(logical) Whether to do fuzzy taxonomic ID search or exact search. If TRUE, we use xXarbitraryXx[porgn:__txid<ID>], but if FALSE, we use txid<ID>. Default: FALSE
limit	(numeric) Number of sequences to search for and return. Max of 10,000. If you search for 6000 records, and only 5000 are found, you will of course only get 5000 back.
entrez_query	(character; length 1) An Entrez-format query to filter results with. This is useful to search for sequences with specific characteristics. The format is the same as the one used to seach genbank. (http://www.ncbi.nlm.nih.gov/books/NBK3837/#EntrezHelp.Entrez_Searching_Options)
hypothetical	(logical; length 1) If FALSE, an attempt will be made to not return hypothetical or predicted sequences judging from accession number prefixs (XM and XR). This can result in less than the limit being returned even if there are more sequences available, since this filtering is done after searching NCBI.
verbose	(logical) If TRUE (default), informative messages printed.

Value

data.frame of results if a single input is given. A list of data.frames if multiple inputs are given.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>, Zachary Foster <zacharyfoster1989@gmail.com>

See Also

ncbi_getbyid, ncbi_getbyname

Examples

```
## Not run:
# A single species
out <- ncbi_searcher(taxa="Umbra limi", seqrange = "1:2000")
# Get the same species information using a taxonomy id
out <- ncbi_searcher(id = "75935", seqrange = "1:2000")
# If the taxon name is unique, using the taxon name and id are equivalent
all(ncbi_searcher(id = "75935") ==  ncbi_searcher(taxa="Umbra limi"))
# If the taxon name is not unique, use taxon id
#  "266948" is the uid for the butterfly genus, but there is also a genus of orchids with the
#  same name
nrow(ncbi_searcher(id = "266948")) ==  nrow(ncbi_searcher(taxa="Satyrium"))
# get list of genes available, removing non-unique
unique(out$gene_desc)
# does the string 'RAG1' exist in any of the gene names
out[grep("RAG1", out$gene_desc, ignore.case=TRUE),]

# A single species without records in NCBI
out <- ncbi_searcher(taxa="Sequoia wellingtonia", seqrange="1:2000", getrelated=TRUE)

# Many species, can run in parallel or not using plyr
species <- c("Salvelinus alpinus","Ictalurus nebulosus","Carassius auratus")
out2 <- ncbi_searcher(taxa=species, seqrange = "1:2000")
lapply(out2, head)
library("plyr")
out2df <- ldply(out2) # make data.frame of all
unique(out2df$gene_desc) # get list of genes available, removing non-unique
out2df[grep("12S", out2df$gene_desc, ignore.case=TRUE), ]

# Using the getrelated and entrez_query options
ncbi_searcher(taxa = "Olpidiopsidales", limit = 5, getrelated = TRUE,
            entrez_query = "18S[title] AND 28S[title]")

# get refseqs
one <- ncbi_searcher(taxa = "Salmonella enterica", entrez_query="srcdb_refseq[PROP]")
two <- ncbi_searcher(taxa = "Salmonella enterica")

## End(Not run)
```

---

plantatt *PLANTATT plant traits dataset*

Description

PLANTATT plant traits dataset

taxa_search *Search for traits by taxa names*

Description

Search for traits by taxa names

Usage

```
taxa_search(x, db, ...)
```

Arguments

x	(character) Taxonomic name(s) to search for
db	(character) One of betydb, traitbank, ncbi, coral.
...	Curl options passed on to GET

Value

A data.frame

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
taxa_search("Poa annua", db = "traitbank")
taxa_search("Poa annua", db = "ncbi")

## End(Not run)
```

traitbank *Search for traits from EOL's Traitbank.*

Description

Search for traits from EOL's Traitbank.

Usage

```
traitbank(pageid, cache_ttl = NULL, ...)
```

Arguments

pageid	A page id. I know, not ideal. Would be better if this was a trait id or trait name. This is the page ID for a taxon, not a trait. Apparently, traits don't have pages. Note: this parameter used to be `trait`, but badly mis-represented what the input actually represents.
cache_ttl	Cache code
...	Curl options passed on to GET

Details

See http://eol.org/data_glossary for human readable definitions for the attribute terms that EOL uses. Go to http://eol.org/data_search for the web interface to Traitbank.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://eol.org/info/516

Examples

```
## Not run:
# Get data for Balaenoptera musculus (http://eol.org/pages/328574/)
res <- traitbank(328574)
res$context
names( res$graph )
head( res$graph )

# Get data for Closterocerus formosus (http://eol.org/pages/846827/)
traitbank(846827)

## End(Not run)
```

---

traits-deprecated *Deprecated functions in traits*

Description

These functions still work but will be removed (defunct) in the next version.

Details

- eol_invasive_: This function is moving to a new package to be on CRAN soon. It will be eol in originr package
- fe_native: This function is moving to a new package to be on CRAN soon. It will be flora_europaea in originr package
- g_invasive: This function is moving to a new package to be on CRAN soon. It will be gisd in originr package
- is_native: This function is moving to a new package to be on CRAN soon. It will be is_native in originr package

Index