Package 'taxize'

February 24, 2019

Title Taxonomic Information from Around the Web

Description Interacts with a suite of web 'APIs' for taxonomic tasks, such as getting database specific taxonomic identifiers, verifying species names, getting taxonomic hierarchies, fetching downstream and upstream taxonomic names, getting taxonomic synonyms, converting scientific to common names and vice versa, and more.

Version 0.9.6

License MIT + file LICENSE

URL https://github.com/ropensci/taxize (devel), https://ropensci.github.io/taxize-book/ (user manual)

BugReports https://github.com/ropensci/taxize/issues

LazyLoad yes

LazyData yes

VignetteBuilder knitr

Encoding UTF-8

Language en-US

Depends R(>= 3.2.1)

Imports graphics, methods, stats, utils, crul (>= 0.7.0), xml2 (>= 1.2.0), jsonlite, reshape2, stringr, plyr, foreach, ape, zoo, bold (>= 0.8.6), data.table, rredlist (>= 0.5.0), rotl (>= 3.0.0), ritis (>= 0.7.6), tibble (>= 1.2), worrms (>= 0.3.2), natserv (>= 0.3.0), wikitaxa (>= 0.3.0)

Suggests testthat, knitr, vegan, vcr

RoxygenNote 6.1.1

X-schema.org-applicationCategory Taxonomy

X-schema.org-keywords taxonomy, biology, nomenclature, JSON, API, web, api-client, identifiers, species, names

X-schema.org-isPartOf https://ropensci.org

NeedsCompilation no

Author Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>),
Eduard Szoecs [aut],
Zachary Foster [aut],
Zebulun Arendsee [aut],
Carl Boettiger [ctb],
Karthik Ram [ctb],
Ignasi Bartomeus [ctb],
John Baumgartner [ctb],
James O'Donnell [ctb],
Jari Oksanen [ctb],
Bastian Greshake Tzovaras [ctb],
Philippe Marchand [ctb],
Vinh Tran [ctb],
Maëlle Salmon [ctb],
Gaopeng Li [ctb],
rOpenSci [fnd] (https://ropensci.org/)

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2019-02-24 18:00:05 UTC

R topics documented:

`taxize-package` *Taxonomic Information from Around the Web*

Description

This package interacts with a suite of web 'APIs' for taxonomic tasks, such as verifying species names, getting taxonomic hierarchies, and verifying name spelling.

About

Allows users to search over many websites for species names (scientific and common) and download up- and downstream taxonomic hierarchical information - and many other things.

The functions in the package that hit a specific API have a prefix and suffix separated by an underscore. They follow the format of `service_whatitdoes`. For example, `gnr_resolve` uses the Global Names Resolver API to resolve species names.

General functions in the package that don't hit a specific API don't have two words separated by an underscore, e.g., `classification`

You need API keys for some data sources. See `taxize-authentication` for more information.

Currently supported APIs

API	prefix	SOAP?
Encyclopedia of Life (EOL)	eol	FALSE
Taxonomic Name Resolution Service	tnrs	FALSE
Integrated Taxonomic Information Service (ITIS)	itis	FALSE
Global Names Resolver (from EOL/GBIF)	gnr	FALSE
Global Names Index (from EOL/GBIF)	gni	FALSE
IUCN Red List	iucn	FALSE
Tropicos (from Missouri Botanical Garden)	tp	FALSE
Theplantlist.org	tpl	FALSE
Catalogue of Life	col	FALSE
National Center for Biotechnology Information	ncbi	FALSE
CANADENSYS Vascan name search API	vascan	FALSE
International Plant Names Index (IPNI)	ipni	FALSE
World Register of Marine Species (WoRMS)	worms	TRUE
Barcode of Life Data Systems (BOLD)	bold	FALSE
Pan-European Species directories Infrastructure (PESI)	pesi	TRUE
Mycobank	myco	TRUE
National Biodiversity Network (UK)	nbn	FALSE
Index Fungorum	fg	FALSE
EU BON	eubon	FALSE
Index of Names (ION)	ion	FALSE
Open Tree of Life (TOL)	tol	FALSE

World Register of Marine Species (WoRMS)	worms	FALSE
NatureServe	natserv	FALSE

If the source above has a TRUE in the SOAP? column, it is not available in this package. They are available from a different package called **taxizesoap**. See the GitHub repo for how to install https://github.com/ropensci/taxizesoap

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Eduard Szoecs <eduardszoecs@gmail.com>

Zachary Foster <zacharyfoster1989@gmail.com>

Carl Boettiger <cboettig@gmail.com>

Karthik Ram <karthik@ropensci.org>

Ignasi Bartomeus <nacho.bartomeus@gmail.com>

John Baumgartner <johnbb@student.unimelb.edu.au>

James O'Donnell <jodonnellbio@gmail.com>

---

apg *Get APG names*

Description

Generic names and their replacements from the Angiosperm Phylogeny Group III system of flowering plant classification.

Usage

```
apgOrders(...)

apgFamilies(...)
```

Arguments

...	Curl args passed on to verb-GET

References

http://www.mobot.org/MOBOT/research/APweb/

Examples

```
## Not run:
head(apgOrders())
head(apgFfamilies())

## End(Not run)
```

`apg_families` *MOBOT family names*

Description

Family names and their replacements from the Angiosperm Phylogeny Website system of flowering plant classification.

Format

A data frame with 1597 rows and 4 variables:

original original data record from APG website

this Order name

that Replacement order name

order Order name

Details

This dataset is from Version 13, incorporated on 2015-04-29.

Source

http://www.mobot.org/MOBOT/research/APweb/

`apg_lookup` *Lookup in the APGIII taxonomy and replace family names*

Description

Lookup in the APGIII taxonomy and replace family names

Usage

```
apg_lookup(taxa, rank = "family")
```

Arguments

taxa	(character) Taxonomic name to lookup a synonym for in APGIII taxonomy.
rank	(character) Taxonomic rank to lookup a synonym for. One of family or order.

Details

Internally in this function, we use the datasets apg_families and apg_orders - see their descriptions for the data in them. The functions apgOrders apgFamilies are for scraping current content from the http://www.mobot.org/MOBOT/research/APweb/ website.

BEWARE: The datasets used in this function are (I think) from Version 12 of the data on http://www.mobot.org/MOBOT/research/APweb/ - I'll update data asap.

Value

A APGIII family or order name, or the original name if no match.

Examples

```
# New name found
apg_lookup(taxa = "Hyacinthaceae", rank = "family")
apg_lookup(taxa = "Poaceae", rank = "family")

# Name not found
apg_lookup(taxa = "Asteraceae", rank = "family")
```

---

apg_orders *MOBOT order names*

Description

Order names and their replacements from the Angiosperm Phylogeny Website system of flowering plant classification.

Format

A data frame with 494 rows and 3 variables:

original original data record from APG website

this Order name

that Replacement order name

Details

This dataset is from Version 13, incorporated on 2015-04-29.

Source

http://www.mobot.org/MOBOT/research/APweb/

`bold_search` *Search Barcode of Life for taxonomic IDs*

Description

Search Barcode of Life for taxonomic IDs

Usage

```
bold_search(name = NULL, id = NULL, fuzzy = FALSE,
  dataTypes = "basic", includeTree = FALSE, response = FALSE, ...)
```

Arguments

`name`	(character) One or more scientific names.
`id`	(integer) One or more BOLD taxonomic identifiers.
`fuzzy`	(logical) Whether to use fuzzy search or not (default: FALSE). Only used if `name` passed.
`dataTypes`	(character) Specifies the datatypes that will be returned. See Details for options. This variable is ignored if `name` parameter is passed, but is used if the `id` parameter is passed.
`includeTree`	(logical) If TRUE (default: FALSE), returns a list containing information for parent taxa as well as the specified taxon. Only used if `id` passed.
`response`	(logical) Note that response is the object that returns from the Curl call, useful for debugging, and getting detailed info on the API call.
`...`	Further args passed on to `verb-GET`, main purpose being curl debugging

Details

You must provide one of name or id to this function. The other parameters are optional. Note that when passing in `name`, `fuzzy` can be used as well, while if `id` is passed, then `fuzzy` is ignored, and `dataTypes` `includeTree` can be used.

Options for `dataTypes` parameter:

- all returns all data
- basic returns basic taxon information
- images returns specimen image. Includes copyright information, image URL, image metadata.
- stats Returns specimen and sequence statistics. Includes public species count, public BIN count, public marker counts, public record count, specimen count, sequenced specimen count, barcode specimen count, species count, barcode species count.
- geo Returns collection site information. Includes country, collection site map.
- sequencinglabs Returns sequencing labs. Includes lab name, record count.
- depository Returns specimen depositories. Includes depository name, record count.
- thirdparty Returns information from third parties. Includes wikipedia summary, wikipedia URL, GBIF map.

Value

A list of data.frame's.

References

http://www.boldsystems.org/index.php/resources/api

Examples

```
## Not run:
# A basic example
bold_search(name="Apis")
bold_search(name="Agapostemon")
bold_search(name="Poa")

# Fuzzy search
head(bold_search(name="Po", fuzzy=TRUE))
head(bold_search(name="Aga", fuzzy=TRUE))

# Many names
bold_search(name=c("Apis","Puma concolor"))
nms <- names_list('species')
bold_search(name=nms)

# Searching by ID - dataTypes can be used, and includeTree can be used
bold_search(id=88899)
bold_search(id=88899, dataTypes="stats")
bold_search(id=88899, dataTypes="geo")
bold_search(id=88899, dataTypes="basic")
bold_search(id=88899, includeTree=TRUE)

## End(Not run)
```

---

children *Retrieve immediate children taxa for a given taxon name or ID.*

Description

This function is different from downstream in that it only collects immediate taxonomic children, while downstream collects taxonomic names down to a specified taxonomic rank, e.g., getting all species in a family.

Usage

```
children(...)

## Default S3 method:
children(x, db = NULL, rows = NA, ...)
```

```
## S3 method for class 'tsn'
children(x, db = NULL, ...)

## S3 method for class 'colid'
children(x, db = NULL, ...)

## S3 method for class 'wormsid'
children(x, db = NULL, ...)

## S3 method for class 'ids'
children(x, db = NULL, ...)

## S3 method for class 'uid'
children(x, db = NULL, ...)
```

Arguments

...	Further args passed on to col_children, hierarchy_down, ncbi_children, or wm_children See those functions for what parameters can be passed on.
x	Vector of taxa names (character) or IDs (character or numeric) to query.
db	character; database to query. One or more of itis, col, ncbi, or worms. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong db value for the identifier you could get a result, but it will likely be wrong (not what you were expecting). If using ncbi, we recommend getting an API key; see taxize-authentication
rows	(numeric) Any number from 1 to infinity. If the default NA, all rows are considered. Note that this parameter is ignored if you pass in a taxonomic id of any of the acceptable classes: tsn, colid. NCBI has a method for this function but rows doesn't work.

Value

A named list of data.frames with the children names of every supplied taxa. You get an NA if there was no match in the database.

Examples

```
## Not run:
# Plug in taxonomic IDs
children(161994, db = "itis")
children(8028, db = "ncbi")
children("578cbfd2674a9b589f19af71a33b89b6", db = "col")
## works with numeric if as character as well
children("161994", db = "itis")

# Plug in taxon names
children("Salmo", db = 'col')
children("Salmo", db = 'itis')
children("Salmo", db = 'ncbi')
```

```
children("Salmo", db = 'worms')

# Plug in IDs
(id <- get_colid("Apis"))
children(id)

(id <- get_wormsid("Platanista"))
children(id)

## Equivalently, plug in the call to get the id via e.g., get_colid
## into children
(id <- get_colid("Apis"))
children(id)
children(get_colid("Apis"))

# Many taxa
sp <- c("Tragia", "Schistocarpha", "Encalypta")
children(sp, db = 'col')
children(sp, db = 'itis')

# Two data sources
(ids <- get_ids("Apis", db = c('col','itis')))
children(ids)
## same result
children(get_ids("Apis", db = c('col','itis')))

# Use the rows parameter
children("Poa", db = 'col')
children("Poa", db = 'col', rows=1)

# use curl options
res <- children("Poa", db = 'col', rows=1, verbose = TRUE)

## End(Not run)
```

`class2tree` *Convert a list of classifications to a tree.*

Description

This function converts a list of hierarchies for individual species into a single species by taxonomic level matrix, then calculates a distance matrix based on taxonomy alone, and outputs either a phylo or dist object. See details for more information.

Usage

```
class2tree(input, varstep = TRUE, check = TRUE, ...)

## S3 method for class 'classtree'
plot(x, ...)
```

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

Arguments

input	List of classification data.frame's from the function classification
varstep	Vary step lengths between successive levels relative to proportional loss of the number of distinct classes.
check	If TRUE, remove all redundant levels which are different for all rows or constant for all rows and regard each row as a different basal taxon (species). If FALSE all levels are retained and basal taxa (species) also must be coded as variables (columns). You will get a warning if species are not coded, but you can ignore this if that was your intention.
...	Further arguments passed on to hclust.
x	Input object to print or plot - output from class2tree function.

Details

See taxa2dist. Thanks to Jari Oksanen for making the taxa2dist function and pointing it out, and Clarke & Warwick (1998, 2001), which taxa2dist was based on.

Value

An object of class "classtree" with slots:

- phylo - The resulting object, a phylo object
- classification - The classification data.frame, with taxa as rows, and different classification levels as columns
- distmat - Distance matrix
- names - The names of the tips of the phylogeny

Note that when you execute the resulting object, you only get the phylo object. You can get to the other 3 slots by calling them directly, like output$names, etc.

Examples

```
## Not run:
spnames <- c('Quercus robur', 'Iris oratoria', 'Arachis paraguariensis',
 'Helianthus annuus','Madia elegans','Lupinus albicaulis',
 'Pinus lambertiana')
out <- classification(spnames, db='itis')
tr <- class2tree(out)
plot(tr)

spnames <- c('Klattia flava', 'Trollius sibiricus', 'Arachis paraguariensis',
 'Tanacetum boreale', 'Gentiana yakushimensis','Sesamum schinzianum',
 'Pilea verrucosa','Tibouchina striphnocalyx','Lycium dasystemum',
 'Berkheya echinacea','Androcymbium villosum',
```

```
 'Helianthus annuus','Madia elegans','Lupinus albicaulis',
 'Pinus lambertiana')
out <- classification(spnames, db='ncbi')
tr <- class2tree(out)
plot(tr)

## End(Not run)
```

`classification` *Retrieve the taxonomic hierarchy for a given taxon ID.*

Description

Retrieve the taxonomic hierarchy for a given taxon ID.

Usage

```
classification(...)

## Default S3 method:
classification(x, db = NULL, callopts = list(),
  return_id = TRUE, rows = NA, ...)

## S3 method for class 'tsn'
classification(id, return_id = TRUE, ...)

## S3 method for class 'uid'
classification(id, callopts = list(), return_id = TRUE,
  ...)

## S3 method for class 'eolid'
classification(id, callopts = list(), return_id = TRUE,
  ...)

## S3 method for class 'colid'
classification(id, start = NULL, checklist = NULL,
  callopts = list(), return_id = TRUE, ...)

## S3 method for class 'tpsid'
classification(id, callopts = list(), return_id = TRUE,
  ...)

## S3 method for class 'gbifid'
classification(id, callopts = list(),
  return_id = TRUE, ...)

## S3 method for class 'nbnid'
```

```
classification(id, callopts = list(), return_id = TRUE,
  ...)

## S3 method for class 'tolid'
classification(id, callopts = list(), return_id = TRUE,
  ...)

## S3 method for class 'wormsid'
classification(id, callopts = list(),
  return_id = TRUE, ...)

## S3 method for class 'natservid'
classification(id, callopts = list(),
  return_id = TRUE, ...)

## S3 method for class 'boldid'
classification(id, callopts = list(),
  return_id = TRUE, ...)

## S3 method for class 'wiki'
classification(id, callopts = list(), return_id = TRUE,
  ...)

## S3 method for class 'pow'
classification(id, callopts = list(), return_id = TRUE,
  ...)

## S3 method for class 'ids'
classification(id, ...)

## S3 method for class 'classification'
cbind(...)

## S3 method for class 'classification'
rbind(...)

## S3 method for class 'classification_ids'
cbind(...)

## S3 method for class 'classification_ids'
rbind(...)
```

Arguments

...	For `classification`: other arguments passed to `get_tsn`, `get_uid`, `get_eolid`, `get_colid`, `get_tpsid`, `get_gbifid`, `get_wormsid`, `get_natservid`, `get_wormsid`, `get_wiki`, `get_pow`. For `rbind.classification` and `cbind.classification`: one or more objects of class `classification`

x	Vector of taxa names (character) or IDs (character or numeric) to query.
db	character; database to query. either ncbi, itis, eol, col, tropicos, gbif, nbn, worms, natserv, bold, wiki, or pow. Note that each taxonomic data source has, their own identifiers, so that if you provide the wrong db value for the identifier you could get a result, but it will likely be wrong (not what you were expecting). If using ncbi, eol, and/or tropicos, we recommend getting an API key; see taxize-authentication
callopts	Curl options passed on to verb-GET
return_id	(logical) If TRUE (default), return the taxon id as well as the name and rank of taxa in the lineage returned. Ignored for natserv as they don't return IDs in their taxonomic classification data.
rows	(numeric) Any number from 1 to infinity. If the default NA, all rows are considered. Note that this parameter is ignored if you pass in a taxonomic id instead of a name of class character.
id	character; identifiers, returned by get_tsn, get_uid, get_eolid, get_colid, get_tpsid, get_gbifid, get_tolid, get_wormsid, get_natservid, get_wormsid, get_wiki, get_pow
start	The first record to return. If omitted, the results are returned from the first record (start=0). This is useful if the total number of results is larger than the maximum number of results returned by a single Web service query (currently the maximum number of results returned by a single query is 500 for terse queries and 50 for full queries).
checklist	character; The year of the checklist to query, if you want a specific year's checklist instead of the lastest as default (numeric).

Details

If IDs are supplied directly (not from the get_* functions) you must specify the type of ID. There is a timeout of 1/3 seconds between queries to NCBI.

BEWARE: Right now, NBN doesn't return the queried taxon in the classification. But you can attach it yourself quite easily of course. This behavior is different from the other data sources.

Value

A named list of data.frames with the taxonomic classification of every supplied taxa.

Lots of results

It may happen sometimes that you get more results back from your query than will show in the data.frame on screen. Our advice is to refine your query in those cases. On a data source basis we can attempt to help make it easier to refine queries, whether it be with the data provider (unlikely to happen), or in the code in this package (more likely) - let us know if you run into too many results problem and we'll see what we can do.

Authentication

See taxize-authentication

See Also

get_tsn, get_uid, get_eolid, get_colid, get_tpsid, get_gbifid get_wormsid, get_natservid, get_boldid, get_wiki, get_pow

Examples

```
## Not run:
# Plug in taxon IDs
classification(9606, db = 'ncbi')
classification(c(9606, 55062), db = 'ncbi')
classification(129313, db = 'itis')
classification(6985636, db = 'eol')
classification(126436, db = 'worms')
classification('Helianthus annuus', db = 'pow')
classification('Helianthus', db = 'pow')
classification('Asteraceae', db = 'pow')
classification("ELEMENT_GLOBAL.2.134717", db = 'natserv')
classification(c(2704179, 2441176), db = 'gbif')
classification(25509881, db = 'tropicos')
classification("NBNSYS0000004786", db = 'nbn')
classification(as.nbnid("NBNSYS0000004786"), db = 'nbn')
classification(3930798, db = 'tol')

## works the same if IDs are in class character
classification(c("2704179", "2441176"), db = 'gbif')
classification("Agapostemon", db = "bold")

# wikispecies
classification("Malus domestica", db = "wiki")
classification("Pinus contorta", db = "wiki")
classification("Pinus contorta", db = "wiki", wiki_site = "commons")
classification("Pinus contorta", db = "wiki", wiki_site = "pedia")
classification("Pinus contorta", db = "wiki", wiki_site = "pedia",
  wiki = "fr")

classification(get_wiki("Malus domestica", "commons"))
classification(get_wiki("Malus domestica", "species"))
classification(c("Pinus contorta", "Malus domestica"), db = "wiki")

# Plug in taxon names
## in this case, we use get_*() fxns internally to first get taxon IDs
classification("Oncorhynchus mykiss", db = "eol")
classification(c("Chironomus riparius", "aaa vva"), db = 'ncbi')
classification(c("Chironomus riparius", "aaa vva"), db = 'ncbi',
  messages=FALSE)
classification(c("Chironomus riparius", "aaa vva"), db = 'itis')
classification(c("Chironomus riparius", "aaa vva"), db = 'itis',
  messages=FALSE)
classification(c("Chironomus riparius", "aaa vva"), db = 'eol')
classification(c("Chironomus riparius", "aaa vva"), db = 'col')
classification("Alopias vulpinus", db = 'nbn')
classification('Gadus morhua', db = 'worms')
```

```
classification('Aquila chrysaetos', db = 'natserv')
classification('Gadus morhua', db = 'natserv')
classification('Pomatomus saltatrix', db = 'natserv')
classification('Aquila chrysaetos', db = 'natserv')
classification(c("Chironomus riparius", "aaa vva"), db = 'col',
  messages=FALSE)
classification(c("Chironomus riparius", "asdfasdfsfdfsd"), db = 'gbif')
classification("Chironomus", db = 'tol')
classification("Poa annua", db = 'tropicos')

# Use methods for get_uid, get_tsn, get_eolid, get_colid, get_tpsid
classification(get_uid(c("Chironomus riparius", "Puma concolor")))

classification(get_uid(c("Chironomus riparius", "aaa vva")))
classification(get_tsn(c("Chironomus riparius", "aaa vva")))
classification(get_tsn(c("Chironomus riparius", "aaa vva"),
  messages = FALSE))
classification(get_eolid(c("Chironomus riparius", "aaa vva")))
classification(get_colid(c("Chironomus riparius", "aaa vva")))
classification(get_tpsid(c("Poa annua", "aaa vva")))
classification(get_gbifid(c("Poa annua", "Bison bison")))

# Pass many ids from class "ids"
(out <- get_ids(names="Puma concolor", db = c('ncbi','gbif')))
(cl <- classification(out))

# Bind width-wise from class classification_ids
cbind(cl)

# Bind length-wise
rbind(cl)

# Many names to get_ids
(out <- get_ids(names=c("Puma concolor","Accipiter striatus"),
  db = c('ncbi','itis','col')))
(cl <- classification(out))
rbind(cl)
## cbind with so many names results in some messy data
cbind(cl)
## so you can turn off return_id
cbind( classification(out, return_id=FALSE) )

# rbind and cbind on class classification (from a
# call to get_colid, get_tsn, etc. other than get_ids)
(cl_col <- classification(
  get_colid(c("Puma concolor","Accipiter striatus"))))
rbind(cl_col)
cbind(cl_col)

(cl_uid <- classification(get_uid(c("Puma concolor",
  "Accipiter striatus")), return_id=FALSE))
rbind(cl_uid)
cbind(cl_uid)
```

```
## cbind works a bit odd when there are lots of ranks without names
(cl_uid <- classification(get_uid(c("Puma concolor","Accipiter striatus")),
  return_id=TRUE))
cbind(cl_uid)

(cl_tsn <- classification(get_tsn(c("Puma concolor","Accipiter striatus"))))
rbind(cl_tsn)
cbind(cl_tsn)

(tsns <- get_tsn(c("Puma concolor","Accipiter striatus")))
(cl_tsns <- classification(tsns))
cbind(cl_tsns)

# NBN data
(res <- classification(c("Alopias vulpinus","Pinus sylvestris"),
  db = 'nbn'))
rbind(res)
cbind(res)

# Return taxonomic IDs
## the return_id parameter is logical, and you can turn it on or off.
## It's TRUE by default
classification(c("Alopias vulpinus","Pinus sylvestris"), db = 'ncbi',
  return_id = TRUE)
classification(c("Alopias vulpinus","Pinus sylvestris"), db = 'ncbi',
  return_id = FALSE)

# Use rows parameter to select certain
classification('Poa annua', db = 'tropicos')
classification('Poa annua', db = 'tropicos', rows=1:4)
classification('Poa annua', db = 'tropicos', rows=1)
classification('Poa annua', db = 'tropicos', rows=6)

## End(Not run)

## Not run:
# Fails without db param set
# classification(315576)

## End(Not run)
```

`col_children` *Search Catalogue of Life for for direct children of a particular taxon.*

Description

Search Catalogue of Life for for direct children of a particular taxon.

Usage

```
col_children(name = NULL, id = NULL, format = NULL, start = NULL,
  checklist = NULL, extant_only = FALSE, ...)
```

Arguments

name	The string to search for. Only exact matches found the name given will be returned, unless one or wildcards are included in the search string. An * (asterisk) character denotes a wildcard; a character may also be used. The name must be at least 3 characters long, not counting wildcard characters.
id	The record ID of the specific record to return (only for scientific names of species or infraspecific taxa)
format	format of the results returned. Valid values are format=xml and format=php; if the format parameter is omitted, the results are returned in the default XML format. If format=php then results are returned as a PHP array in serialized string format, which can be converted back to an array in PHP using the unserialize command
start	The first record to return. If omitted, the results are returned from the first record (start=0). This is useful if the total number of results is larger than the maximum number of results returned by a single Web service query (currently the maximum number of results returned by a single query is 500 for terse queries and 50 for full queries).
checklist	The year of the checklist to query, if you want a specific year's checklist instead of the lastest as default (numeric).
extant_only	(logical) keep extant taxa only? default: FALSE. by default we give back all taxa. set to TRUE to get only extant taxa
...	Curl options passed on to verb-GET

Details

You must provide one of name or id. The other parameters (format and start) are optional.

Value

A list of data.frame's, where each data.frame has columns:

- childtaxa_id: (character) COL identifier
- childtaxa_name: (character) taxonomic name
- childtaxa_rank: (character) rank name
- childtaxa_extinct: (logical) extinct or not

Examples

```
## Not run:
# A basic example
col_children(name="Apis")
```

```
# An example where there is no classification, results in data.frame with
# no rows
col_children(id='b2f88f382aa5568f93a97472c6be6516')

# Use a specific year's checklist
col_children(name="Apis", checklist=2012)
col_children(name="Apis", checklist=2009)

# Pass in many names or many id's
out <- col_children(name=c("Buteo","Apis","Accipiter","asdf"),
  checklist = "2012")
out$Apis # get just the output you want
library("plyr")
ldply(out) # or combine to one data.frame

# or pass many id's
ids <- c('abe977b1d27007a76dd12a5c93a637bf',
  'b2f88f382aa5568f93a97472c6be6516')
out <- col_children(id = ids, checklist=2012)
library("plyr")
ldply(out) # combine to one data.frame

# keep extant taxa only, prunes out extinct taxa
col_children(name = "Insecta")
col_children(name = "Insecta", extant_only = TRUE)

## End(Not run)
```

`col_downstream` *Use Catalogue of Life to get downstream taxa to a given taxonomic level*

Description

Use Catalogue of Life to get downstream taxa to a given taxonomic level

Usage

```
col_downstream(name = NULL, id = NULL, downto, format = NULL,
  start = NULL, checklist = NULL, verbose = TRUE,
  intermediate = FALSE, extant_only = FALSE, ...)
```

Arguments

`name`	The string to search for. Only exact matches found the name given will be returned, unless one or wildcards are included in the search string. An * (asterisk) character denotes a wildcard; a character may also be used. The name must be at least 3 characters long, not counting wildcard characters.
`id`	The record ID of the specific record to return (only for scientific names of species or infraspecific taxa)

downto	The taxonomic level you want to go down to. See examples below. The taxonomic level IS case sensitive, and you do have to spell it correctly. See data(rank_ref) for spelling.
format	The returned format (default = NULL). If NULL xml is used. Currently only xml is supported.
start	The first record to return (default = NULL). If NULL, the results are returned from the first record (start=0). This is useful if the total number of results is larger than the maximum number of results returned by a single Web service query (currently the maximum number of results returned by a single query is 500 for terse queries and 50 for full queries).
checklist	The year of the checklist to query, if you want a specific year's checklist instead of the lastest as default (numeric).
verbose	Print or suppress messages.
intermediate	(logical) If TRUE, return a list of length two with target taxon rank names, with additional list of data.frame's of intermediate taxonomic groups. Default: FALSE
extant_only	(logical) keep extant taxa only? default: FALSE. by default we give back all taxa. set to TRUE to get only extant taxa
...	Curl options passed on to verb-GET

Details

Provide only names instead of id's

Value

A list of data.frame's, where each data.frame has columns:

- childtaxa_id: (character) COL identifier
- childtaxa_name: (character) taxonomic name
- childtaxa_rank: (character) rank name
- childtaxa_extinct: (logical) extinct or not

Examples

```
## Not run:
# Some basic examples
col_downstream(name="Apis", downto="species")
col_downstream(name="Bryophyta", downto="family")

# get classes down from the kingdom Animalia
col_downstream(name="Animalia", downto="class")
col_downstream(name="Animalia", downto="class", intermediate=TRUE)

# An example that takes a bit longer
col_downstream(name=c("Plantae", "Animalia"), downto="class")

# Using a checklist from a specific year
col_downstream(name="Bryophyta", downto="family", checklist=2009)
```

```
# By id
col_downstream(id='576d098d770a39d09e2bcfa1c0896b26', downto="species",
  checklist=2012)

# keep extant taxa only, prunes out extinct taxa
col_downstream(name = "Insecta", downto = "order")
col_downstream(name = "Insecta", downto = "order", extant_only = TRUE)

## End(Not run)
```

`col_search` *Search Catalogue of Life for taxonomic IDs*

Description

Search Catalogue of Life for taxonomic IDs

Usage

```
col_search(name = NULL, id = NULL, start = NULL, checklist = NULL,
  response = "terse", ...)
```

Arguments

`name`	The string to search for. Only exact matches found the name given will be returned, unless one or wildcards are included in the search string. An * (asterisk) character denotes a wildcard; a character may also be used. The name must be at least 3 characters long, not counting wildcard characters.
`id`	The record ID of the specific record to return (only for scientific names of species or infraspecific taxa)
`start`	The first record to return. If omitted, the results are returned from the first record (start=0). This is useful if the total number of results is larger than the maximum number of results returned by a single Web service query (currently the maximum number of results returned by a single query is 500 for terse queries and 50 for full queries).
`checklist`	The year of the checklist to query, if you want a specific year's checklist instead of the lastest as default (numeric).
`response`	(character) one of "terse" or "full"
`...`	Curl options passed on to `HttpClient`

Details

You must provide one of name or id. The other parameters (format and start) are optional.

Value

A list of data.frame's, each data.frame has the attributes:

- id:
- name:
- total_number_of_results:
- number_of_results_returned:
- start:
- error_message:
- version:
- rank:

References

http://webservice.catalogueoflife.org/

Examples

```
## Not run:
# A basic example
col_search(name="Apis")
col_search(name="Agapostemon")
col_search(name="Poa")

# Get full response, i.e., more data
col_search(name="Apis", response="full")
col_search(name="Poa", response="full")

# Many names
col_search(name=c("Apis","Puma concolor"))
col_search(name=c("Apis","Puma concolor"), response = "full")

# An example where there is no data
col_search(id = "36c623ad9e3da39c2e978fa3576ad415")
col_search(id = "36c623ad9e3da39c2e978fa3576ad415", response = "full")
col_search(id = "787ce23969f5188c2467126d9a545be1")
col_search(id = "787ce23969f5188c2467126d9a545be1", response = "full")
col_search(id = c("36c623ad9e3da39c2e978fa3576ad415",
  "787ce23969f5188c2467126d9a545be1"))
## a synonym
col_search(id = "f726bdaa5924cabf8581f99889de51fc")
col_search(id = "f726bdaa5924cabf8581f99889de51fc", response = "full")

## End(Not run)
```

comm2sci *Get scientific names from common names.*

Description

Get scientific names from common names.

Usage

```
comm2sci(commnames, db = "ncbi", itisby = "search", simplify = TRUE,
  ...)
```

Arguments

commnames	One or more common names or partial names.
db	Data source, one of *"ncbi"* (default), *"itis"*, *"tropicos"*, *"eol"*, or *"worms"*. If using ncbi, we recommend getting an API key; see taxize-authentication
itisby	Search for common names across entire names (search, default), at beginning of names (begin), or at end of names (end).
simplify	(logical) If TRUE, simplify output to a vector of names. If FALSE, return variable formats from different sources, usually a data.frame.
...	Further arguments passed on to internal methods.

Details

For data sources ITIS and NCBI you can pass in common names directly, and use get_uid or get_tsn to get ids first, then pass in to this fxn.

For the other data sources, you can only pass in common names directly.

Value

If simplify=TRUE, a list of scientific names, with list labeled by your input names. If simplify=FALSE, a data.frame with columns that vary by data source

Authentication

See taxize-authentication for help on authentication

Author(s)

Scott Chamberlain

See Also

sci2comm

Examples

```
## Not run:
comm2sci(commnames='american black bear')
comm2sci(commnames='american black bear', simplify = FALSE)
comm2sci(commnames='black bear', db='itis')
comm2sci(commnames='american black bear', db='itis')
comm2sci(commnames='annual blue grass', db='tropicos')
comm2sci(commnames=c('annual blue grass','tree of heaven'), db='tropicos')
comm2sci('blue whale', db = "worms")
comm2sci(c('blue whale', 'dwarf surfclam'), db = "worms")

# Output easily converts to a data.frame with plyr::ldply
library(plyr)
ldply(comm2sci(commnames=c('annual blue grass','tree of heaven'),
  db='tropicos'))

# ncbi: pass in uid's from get_uid() directly
x <- get_uid("western capercaillie", modifier = "Common Name")
comm2sci(x)
# itis: pass in tsn's from get_tsn() directly
x <- get_tsn(c("Louisiana black bear", "american crow"),
  searchtype = "common")
comm2sci(x)

## End(Not run)
```

---

downstream *Retrieve the downstream taxa for a given taxon name or ID.*

Description

This function uses a while loop to continually collect children taxa down to the taxonomic rank that you specify in the downto parameter. You can get data from ITIS (itis), Catalogue of Life (col), GBIF (gbif), NCBI (ncbi) or WORMS (worms). There is no method exposed by these four services for getting taxa at a specific taxonomic rank, so we do it ourselves here.

Usage

```
downstream(...)

## Default S3 method:
downstream(x, db = NULL, downto = NULL,
  intermediate = FALSE, rows = NA, ...)

## S3 method for class 'tsn'
downstream(x, db = NULL, downto = NULL,
  intermediate = FALSE, ...)
```

```
## S3 method for class 'colid'
downstream(x, db = NULL, downto = NULL,
  intermediate = FALSE, ...)

## S3 method for class 'gbifid'
downstream(x, db = NULL, downto = NULL,
  intermediate = FALSE, limit = 100, start = NULL, ...)

## S3 method for class 'uid'
downstream(x, db = NULL, downto = NULL,
  intermediate = FALSE, ...)

## S3 method for class 'wormsid'
downstream(x, db = NULL, downto = NULL,
  intermediate = FALSE, ...)

## S3 method for class 'ids'
downstream(x, db = NULL, downto = NULL,
  intermediate = FALSE, ...)
```

Arguments

...	Further args passed on to itis_downstream, col_downstream, gbif_downstream, ncbi_downstream, or worms_downstream
x	Vector of taxa names (character) or IDs (character or numeric) to query.
db	character; database to query. One or more of itis, col, gbif, ncbi or worms. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong db value for the identifier you could get a result, but it will likely be wrong (not what you were expecting). If using ncbi, we recommend getting an API key; see taxize-authentication
downto	What taxonomic rank to go down to. One of: 'superkingdom', 'kingdom', 'subkingdom','infrakingdom','phylum','division','subphylum', 'subdivision','infradivision', 'superclass','class','subclass','infraclass', 'superorder','order','suborder','infraorder','superfamily','famil 'subfamily','tribe','subtribe','genus','subgenus','section','subsection', 'species group','species','subspeci 'stirp', 'morph','aberration','subform', 'unspecified', 'no rank'
intermediate	(logical) If TRUE, return a list of length two with target taxon rank names, with additional list of data.frame's of intermediate taxonomic groups. Default: FALSE
rows	(numeric) Any number from 1 to infinity. If the default NA, all rows are considered. Note that this parameter is ignored if you pass in a taxonomic id of any of the acceptable classes: tsn, colid.
limit	Number of records to return
start	Record number to start at

Value

A named list of data.frames with the downstream names of every supplied taxa. You get an NA if there was no match in the database.

Authentication

See taxize-authentication for help on authentication

Examples

```
## Not run:
# Plug in taxon IDs
downstream("015be25f6b061ba517f495394b80f108", db = "col",
  downto = "species")
downstream(125732, db = 'worms', downto = 'species')

# Plug in taxon names
downstream("Insecta", db = 'col', downto = 'order')
downstream("Apis", db = 'col', downto = 'species')
downstream("Apis", db = 'ncbi', downto = 'species')
downstream("Apis", db = 'itis', downto = 'species')
downstream("Gadus", db = 'worms', downto = 'species')
downstream(c("Apis","Epeoloides"), db = 'itis', downto = 'species')
downstream(c("Apis","Epeoloides"), db = 'col', downto = 'species')
downstream("Ursus", db = 'gbif', downto = 'species')
downstream(get_gbifid("Ursus"), db = 'gbif', downto = 'species')

# Plug in IDs
id <- get_colid("Apis")
downstream(id, downto = 'species')

## Equivalently, plug in the call to get the id via e.g., get_colid
## into downstream
identical(downstream(id, downto = 'species'),
         downstream(get_colid("Apis"), downto = 'species'))

id <- get_colid("Apis")
downstream(id, downto = 'species')
downstream(get_colid("Apis"), downto = 'species')

# Many taxa
sp <- names_list("genus", 3)
downstream(sp, db = 'col', downto = 'species')
downstream(sp, db = 'itis', downto = 'species')
downstream(sp, db = 'gbif', downto = 'species')

# Both data sources
ids <- get_ids("Apis", db = c('col','itis'))
downstream(ids, downto = 'species')
## same result
downstream(get_ids("Apis", db = c('col','itis')), downto = 'species')

# Collect intermediate names
## itis
downstream('Bangiophyceae', db="itis", downto="genus")
downstream('Bangiophyceae', db="itis", downto="genus", intermediate=TRUE)
downstream(get_tsn('Bangiophyceae'), downto="genus")
```

```
downstream(get_tsn('Bangiophyceae'), downto="genus", intermediate=TRUE)
## col
downstream(get_colid("Animalia"), downto="class")
downstream(get_colid("Animalia"), downto="class", intermediate=TRUE)

# Use the rows parameter
## note how in the second function call you don't get the prompt
downstream("Poa", db = 'col', downto="species")
downstream("Poa", db = 'col', downto="species", rows=1)

# use curl options
res <- downstream("Apis", db = 'col', downto = 'species', verbose = TRUE)

## End(Not run)
```

`eol_dataobjects` *Given the identifier for a data object, return all metadata about the object*

Description

Given the identifier for a data object, return all metadata about the object

Usage

```
eol_dataobjects(id, taxonomy = TRUE, language = NULL, usekey = TRUE,
  key = NULL, ...)
```

Arguments

`id`	(character) The EOL data object identifier
`taxonomy`	(logical) Whether to return any taxonomy details from different taxon hierarchy providers, in an array named `taxonconcepts`
`language`	(character) provides the results in the specified language. one of ms, de, en, es, fr, gl, it, nl, nb, oc, pt-BR, sv, tl, mk, sr, uk, ar, zh-Hans, zh-Hant, ko
`usekey`	(logical) use your API key or not (`TRUE` or `FALSE`)
`key`	(character) Your EOL API key; ; see `taxize-authentication` for help on authentication
`...`	Curl options passed on to `HttpClient`

Details

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

Value

A list, optionally with a data.frame if `taxonomy=TRUE`

Examples

```
## Not run:
eol_dataobjects(id = 7561533)

# curl options
eol_dataobjects(id = 7561533, verbose = TRUE)

## End(Not run)
```

`eol_pages` *Search for pages in EOL database using a taxonconceptID.*

Description

Search for pages in EOL database using a taxonconceptID.

Usage

```
eol_pages(taxonconceptID, images_per_page = NULL, images_page = NULL,
  videos_per_page = NULL, videos_page = NULL, sounds_per_page = NULL,
  sounds_page = NULL, maps_per_page = NULL, maps_page = NULL,
  texts_per_page = NULL, texts_page = NULL, subjects = "overview",
  licenses = "all", details = FALSE, common_names = FALSE,
  synonyms = FALSE, references = FALSE, taxonomy = TRUE,
  vetted = 0, cache_ttl = NULL, key = NULL, ...)
```

Arguments

`taxonconceptID`	(numeric) a taxonconceptID, which is also the page number
`images_per_page`	(integer) number of returned image objects (0-75)
`images_page`	(integer) images page
`videos_per_page`	(integer) number of returned video objects (0-75)
`videos_page`	(integer) videos page
`sounds_per_page`	(integer) number of returned sound objects (0-75)
`sounds_page`	(integer) sounds page
`maps_per_page`	(integer) number of returned map objects (0-75)
`maps_page`	(integer) maps page
`texts_per_page`	(integer) number of returned text objects (0-75)
`texts_page`	(integer) texts page

subjects	'overview' (default) to return the overview text (if exists), a pipe | delimited list of subject names from the list of EOL accepted subjects (e.g. TaxonBiology, FossilHistory), or 'all' to get text in any subject. Always returns an overview text as a first result (if one exists in the given context).
licenses	A pipe | delimited list of licenses or 'all' (default) to get objects under any license. Licenses abbreviated cc- are all Creative Commons licenses. Visit their site for more information on the various licenses they offer.
details	Include all metadata for data objects. (Default: FALSE)
common_names	Return all common names for the page's taxon (Default: FALSE)
synonyms	Return all synonyms for the page's taxon (Default: FALSE)
references	Return all references for the page's taxon (Default: FALSE)
taxonomy	(logical) Whether to return any taxonomy details from different taxon hierarchy providers, in an array named taxonconcepts (Default: TRUE)
vetted	If 'vetted' is given a value of '1', then only trusted content will be returned. If 'vetted' is '2', then only trusted and unreviewed content will be returned (untrusted content will not be returned). The default is to return all content. (Default: FALSE)
cache_ttl	The number of seconds you wish to have the response cached.
key	Your EOL API key; see taxize-authentication for help on authentication
...	Curl options passed on to HttpClient

Details

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

Value

JSON list object, or data.frame.

Examples

```
## Not run:
(pageid <- eol_search('Pomatomus')$pageid[1])
x <- eol_pages(taxonconceptID = pageid)
x
x$scinames

z <- eol_pages(taxonconceptID = pageid, synonyms = TRUE)
z$synonyms

z <- eol_pages(taxonconceptID = pageid, common_names = TRUE)
z$vernacular

## End(Not run)
```

`eol_search` *Search for terms in EOL database.*

Description

Search for terms in EOL database.

Usage

```
eol_search(terms, page = 1, exact = NULL, filter_tid = NULL,
  filter_heid = NULL, filter_by_string = NULL, cache_ttl = NULL,
  key = NULL, ...)
```

Arguments

`terms`	search terms (character)
`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)
`exact`	Will find taxon pages if the preferred name or any synonym or common name exactly matches the search term.
`filter_tid`	Given an EOL page ID, search results will be limited to members of that taxonomic group
`filter_heid`	Given a Hierarchy Entry ID, search results will be limited to members of that taxonomic group
`filter_by_string`	Given a search term, an exact search will be made and that matching page will be used as the taxonomic group against which to filter search results
`cache_ttl`	The number of seconds you wish to have the response cached.
`key`	Your EOL API key. See `taxize-authentication` for help on authentication
`...`	Curl options passed on to `HttpClient`

Details

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

Value

A data frame with four columns:

- pageid: pageid, this is the same as the eolid you can get from `get_eolid`
- name: taxonomic name, may or may not contain the taxonomic authority
- link: URL for the taxon in question
- content: a string of semi-colon separated names. it's not clear to us what these represent exactly, but figured why not give it to users in case some may find it useful

Authentication

See taxize-authentication for help on authentication

Examples

```
## Not run:
eol_search(terms='Homo')
eol_search(terms='Salix', verbose = TRUE)
eol_search(terms='Ursus americanus')
eol_search('Pinus contorta')

## End(Not run)
```

---

eubon *EUBON taxonomy search*

Description

EUBON taxonomy search

Usage

```
eubon(query, providers = "pesi", searchMode = "scientificNameExact",
  addSynonymy = FALSE, addParentTaxon = FALSE, timeout = 0,
  dedup = NULL, ...)

eubon_search(query, providers = "pesi",
  searchMode = "scientificNameExact", addSynonymy = FALSE,
  addParentTaxon = FALSE, timeout = 0, dedup = NULL, ...)
```

Arguments

query	(character) The scientific name to search for. For example: "Bellis perennis", "Prionus" or "Bolinus brandaris". This is an exact search so wildcard characters are not supported
providers	(character) A list of provider id strings concatenated by comma characters. The default : "pesi,bgbm-cdm-server[col]" will be used if this parameter is not set. A list of all available provider ids can be obtained from the '/capabilities' service end point. Providers can be nested, that is a parent provider can have sub providers. If the id of the parent provider is supplied all subproviders will be queried. The query can also be restricted to one or more subproviders by using the following syntax: parent-id[sub-id-1,sub-id2,...]
searchMode	(character) Specifies the searchMode. Possible search modes are: scientificNameExact, scientificNameLike (begins with), vernacularNameExact, vernacularNameLike (contains), findByIdentifier. If the a provider does not support the chosen searchMode it will be skipped and the status message in the tnrClientStatus will be set to 'unsupported search mode' in this case.

addSynonymy	(logical) Indicates whether the synonymy of the accepted taxon should be included into the response. Turning this option on may cause an increased response time. Default: FALSE
addParentTaxon	(logical) Indicates whether the the parent taxon of the accepted taxon should be included into the response. Turning this option on may cause a slightly increased response time. Default: FALSE
timeout	(numeric) The maximum of milliseconds to wait for responses from any of the providers. If the timeout is exceeded the service will just return the responses that have been received so far. The default timeout is 0 ms (wait for ever)
dedup	(character) Allows to deduplicate the results by making use of a deduplication strategy. The deduplication is done by comparing specific properties of the taxon: • id: compares 'taxon.identifier' • id_name: compares 'taxon.identifier' AND 'taxon.taxonName.scientificName' • name: compares 'taxon.taxonName.scientificName' Using the pure 'name' strategy is not recommended.
...	Curl options passed on to verb-GET

Details

Note that paging is not yet implemented, so you only get the first chunk of up to 50 results for methods that require paging. We will implement paging here when it is available in the EU BON API.

References

http://cybertaxonomy.eu/eu-bon/utis/1.2/doc.html

See Also

Other eubon-methods: eubon_capabilities, eubon_children, eubon_hierarchy

Examples

```
## Not run:
eubon_search("Prionus")
eubon_search("Salmo", "pesi")
eubon_search("Salmo", c("pesi", "worms"))
eubon_search("Salmo", "worms", "scientificNameLike")
eubon_search("Salmo", "worms", addSynonymy = TRUE)
eubon_search("Salmo", "worms", addParentTaxon = TRUE)

## End(Not run)
```

eubon_capabilities *EUBON capabilities*

Description

EUBON capabilities

Usage

```
eubon_capabilities(...)
```

Arguments

...	Curl options passed on to verb-GET

References

http://cybertaxonomy.eu/eu-bon/utis/1.2/doc.html

See Also

Other eubon-methods: eubon_children, eubon_hierarchy, eubon

Examples

```
## Not run:
eubon_capabilities()

## End(Not run)
```

eubon_children *EUBON children*

Description

EUBON children

Usage

```
eubon_children(id, providers = NULL, timeout = 0, ...)
```

Arguments

id	(character) identifier for the taxon. (LSID, DOI, URI, or any other identifier used by the checklist provider)
providers	(character) A list of provider id strings concatenated by comma characters. The default : "pesi,bgbm-cdm-server[col]" will be used if this parameter is not set. A list of all available provider ids can be obtained from the '/capabilities' service end point. Providers can be nested, that is a parent provider can have sub providers. If the id of the parent provider is supplied all subproviders will be queried. The query can also be restricted to one or more subproviders by using the following syntax: parent-id[sub-id-1,sub-id2,...]
timeout	(numeric) The maximum of milliseconds to wait for responses from any of the providers. If the timeout is exceeded the service will just return the responses that have been received so far. The default timeout is 0 ms (wait for ever)
...	Curl options passed on to verb-GET

Value

a data.frame or an empty list if no results found

References

http://cybertaxonomy.eu/eu-bon/utis/1.2/doc.html

See Also

Other eubon-methods: eubon_capabilities, eubon_hierarchy, eubon

Examples

```
## Not run:
x <- eubon_children(id = "urn:lsid:marinespecies.org:taxname:126141",
  providers = 'worms')
head(x)

## End(Not run)
```

---

eubon_hierarchy *EUBON hierarchy*

Description

EUBON hierarchy

Usage

```
eubon_hierarchy(id, providers = "pesi", timeout = 0, ...)
```

Arguments

id	(character) identifier for the taxon. (LSID, DOI, URI, or any other identifier used by the checklist provider)
providers	(character) A list of provider id strings concatenated by comma characters. The default : "pesi,bgbm-cdm-server[col]" will be used if this parameter is not set. A list of all available provider ids can be obtained from the '/capabilities' service end point. Providers can be nested, that is a parent provider can have sub providers. If the id of the parent provider is supplied all subproviders will be queried. The query can also be restricted to one or more subproviders by using the following syntax: parent-id[sub-id-1,sub-id2,...]
timeout	(numeric) The maximum of milliseconds to wait for responses from any of the providers. If the timeout is exceeded the service will just return the responses that have been received so far. The default timeout is 0 ms (wait for ever)
...	Curl options passed on to verb-GET

References

http://cybertaxonomy.eu/eu-bon/utis/1.2/doc.html

See Also

Other eubon-methods: eubon_capabilities, eubon_children, eubon

Examples

```
## Not run:
eubon_hierarchy(id = "urn:lsid:marinespecies.org:taxname:126141", 'worms')
eubon_hierarchy(id = "urn:lsid:marinespecies.org:taxname:274350", 'worms')

## End(Not run)
```

---

fungorum *Index Fungorum*

Description

Search for taxonomic names in Index Fungorum

Usage

```
fg_name_search(q, anywhere = TRUE, limit = 10, ...)

fg_author_search(q, anywhere = TRUE, limit = 10, ...)

fg_epithet_search(q, anywhere = TRUE, limit = 10, ...)
```

```
fg_name_by_key(key, ...)

fg_name_full_by_lsid(lsid, ...)

fg_all_updated_names(date, ...)

fg_deprecated_names(date, ...)
```

Arguments

q	(character) Query term
anywhere	(logical) Default: TRUE
limit	(integer) Number of results to return. max limit value appears to be 6000, not positive about that though
...	Curl options passed on to verb-GET
key	(character) A IndexFungorum taxon key
lsid	(character) an LSID, e.,g. "urn:lsid:indexfungorum.org:names:81085"
date	(character) Date, of the form YYYMMDD

Value

A data.frame, or NULL if no results

References

http://www.indexfungorum.org/, API docs: http://www.indexfungorum.org/ixfwebservice/fungus.asmx

Examples

```
## Not run:
# NameSearch
fg_name_search(q = "Gymnopus", limit = 2, verbose = TRUE)
fg_name_search(q = "Gymnopus")

# EpithetSearch
fg_epithet_search(q = "phalloides")

# NameByKey
fg_name_by_key(17703)

# NameFullByKey
fg_name_full_by_lsid("urn:lsid:indexfungorum.org:names:81085")

# AllUpdatedNames
fg_all_updated_names(date = gsub("-", "", Sys.Date() - 2))

# DeprecatedNames
fg_deprecated_names(date=20151001)
```

```
# AuthorSearch
fg_author_search(q = "Fayod", limit = 2)

## End(Not run)
```

---

gbif_downstream *Retrieve all taxa names downstream in hierarchy for GBIF*

Description

Retrieve all taxa names downstream in hierarchy for GBIF

Usage

```
gbif_downstream(key, downto, intermediate = FALSE, limit = 100,
  start = NULL, ...)
```

Arguments

key	A taxonomic serial number.
downto	The taxonomic level you want to go down to. See examples below. The taxonomic level IS case sensitive, and you do have to spell it correctly. See data(rank_ref) for spelling.
intermediate	(logical) If TRUE, return a list of length two with target taxon rank names, with additional list of data.frame's of intermediate taxonomic groups. Default: FALSE
limit	Number of records to return
start	Record number to start at
...	Further args passed on to gbif_name_usage

Details

Sometimes records don't have a canonicalName entry which is what we look for. In that case we grab the scientificName entry. You can see the type of name colleceted in the column name_type

Value

data.frame of taxonomic information downstream to family from e.g., Order, Class, etc., or if intermediated=TRUE, list of length two, with target taxon rank names, and intermediate names.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
## the plant class Bangiophyceae
gbif_downstream(key = 198, downto="genus")
gbif_downstream(key = 198, downto="genus", intermediate=TRUE)

# families downstream from the family Strepsiptera (twisted wing parasites)
gbif_downstream(key = 1227, "family")
## here, intermediate leads to the same result as the target
gbif_downstream(key = 1227, "family", intermediate=TRUE)

# Lepidoptera
gbif_downstream(key = 797, "family")

# get species downstream from the genus Ursus
gbif_downstream(key = 2433406, "species")

# get tribes down from the family Apidae
gbif_downstream(key = 7799978, downto="species")
gbif_downstream(key = 7799978, downto="species", intermediate=TRUE)

# names that don't have canonicalname entries for some results
key <- get_gbifid("Myosotis")
res <- gbif_downstream(key, downto = "species")
res2 <- downstream(key, db = "gbif", downto = "species")

## End(Not run)
```

---

gbif_name_usage *Lookup details for specific names in all taxonomies in GBIF.*

Description

This is a taxize version of the same function in the `rgbif` package so as to not have to import rgbif and thus require GDAL binary installation.

Usage

```
gbif_name_usage(key = NULL, name = NULL, data = "all",
  language = NULL, datasetKey = NULL, uuid = NULL, sourceId = NULL,
  rank = NULL, shortname = NULL, start = NULL, limit = 20, ...)
```

Arguments

`key`	(numeric) A GBIF key for a taxon
`name`	(character) Filters by a case insensitive, canonical namestring, e.g. 'Puma concolor'
`data`	(character) Specify an option to select what data is returned. See Description below.

language	(character) Language, default is english
datasetKey	(character) Filters by the dataset's key (a uuid)
uuid	(character) A uuid for a dataset. Should give exact same results as datasetKey.
sourceId	(numeric) Filters by the source identifier. Not used right now.
rank	(character) Taxonomic rank. Filters by taxonomic rank as one of: CLASS, CULTIVAR, CULTIVAR_GROUP, DOMAIN, FAMILY, FORM, GENUS, INFORMAL, INFRAGENERIC_NAME, INFRAORDER, INFRASPECIFIC_NAME, INFRASUBSPECIFIC_NAME, KINGDOM, ORDER, PHYLUM, SECTION, SERIES, SPECIES, STRAIN, SUBCLASS, SUBFAMILY, SUBFORM, SUBGENUS, SUBKINGDOM, SUBORDER, SUBPHYLUM, SUBSECTION, SUBSERIES, SUBSPECIES, SUBTRIBE, SUBVARIETY, SUPERCLASS, SUPERFAMILY, SUPERORDER, SUPERPHYLUM, SUPRAGENERIC_NAME, TRIBE, UNRANKED, VARIETY
shortname	(character) A short name..need more info on this?
start	Record number to start at
limit	Number of records to return
...	Curl options passed on to HttpClient

Value

A list of length two. The first element is metadata. The second is either a data.frame (verbose=FALSE, default) or a list (verbose=TRUE)

References

http://www.gbif.org/developer/summary

---

gbif_parse *Parse taxon names using the GBIF name parser.*

Description

Parse taxon names using the GBIF name parser.

Usage

```
gbif_parse(scientificname, ...)
```

Arguments

scientificname	(character) scientific names
...	Further args passed on to verb-POST

Value

A `data.frame` containing fields extracted from parsed taxon names. Fields returned are the union of fields extracted from all species names in `scientificname`.

Author(s)

John Baumgartner (johnbb@student.unimelb.edu.au)

References

https://www.gbif.org/tools/name-parser/about

See Also

`gni_parse`

Examples

```
## Not run:
gbif_parse(scientificname='x Agropogon littoralis')
gbif_parse(c('Arrhenatherum elatius var. elatius',
             'Secale cereale subsp. cereale', 'Secale cereale ssp. cereale',
             'Vanessa atalanta (Linnaeus, 1758)'))

## End(Not run)
```

---

genbank2uid *Get NCBI taxonomy UID from GenBankID*

Description

Get NCBI taxonomy UID from GenBankID

Usage

```
genbank2uid(id, batch_size = 100, key = NULL, ...)
```

Arguments

`id`	A GenBank accession alphanumeric string, or a gi numeric string.
`batch_size`	The number of queries to submit at a time.
`key`	(character) NCBI Entrez API key. optional. See Details.
`...`	Curl args passed on to `HttpClient`

Details

See `http://www.ncbi.nlm.nih.gov/Sitemap/sequenceIDs.html` for help on why there are two identifiers, and the difference between them.

Value

one or more NCBI taxonomic IDs

Authentication

See taxize-authentication for help on authentication. We recommend getting an API key.

Examples

```
## Not run:
# with accession numbers
genbank2uid(id = 'AJ748748')
genbank2uid(id = 'Y13155')
genbank2uid(id = 'X78312')
genbank2uid(id = 'KM495596')

# with gi numbers
genbank2uid(id = 62689767)
genbank2uid(id = 22775511)
genbank2uid(id = 156446673)

# pass in many accession or gi numbers
genbank2uid(c(62689767,156446673))
genbank2uid(c('X78312','KM495596'))
genbank2uid(list('X78312',156446673))

# curl options
res <- genbank2uid(id = 156446673, verbose = TRUE)

## End(Not run)
```

---

getkey *Function to get API key.*

Description

Checks first to get key from your .Rprofile or .Renviron (or similar) file

Usage

```
getkey(x = NULL, service)
```

Arguments

x	(character) An API key, defaults to NULL
service	(character) The API data provider, used to match to default guest key (for Tropicos and EOL; there's no guest key for NCBI or IUCN, for which you have to get your own)

Examples

```
## Not run:
getkey(service="tropicos")
getkey(service="eol")
getkey(service="iucn")
getkey(service="entrez")

## End(Not run)
```

---

get_boldid *Get the BOLD (Barcode of Life) code for a search term.*

Description

Get the BOLD (Barcode of Life) code for a search term.

Usage

```
get_boldid(searchterm, fuzzy = FALSE, dataTypes = "basic",
  includeTree = FALSE, ask = TRUE, verbose = TRUE, rows = NA,
  rank = NULL, division = NULL, parent = NULL, ...)

as.boldid(x, check = TRUE)

## S3 method for class 'boldid'
as.boldid(x, check = TRUE)

## S3 method for class 'character'
as.boldid(x, check = TRUE)

## S3 method for class 'list'
as.boldid(x, check = TRUE)

## S3 method for class 'numeric'
as.boldid(x, check = TRUE)

## S3 method for class 'data.frame'
as.boldid(x, check = TRUE)

## S3 method for class 'boldid'
as.data.frame(x, ...)

get_boldid_(searchterm, verbose = TRUE, fuzzy = FALSE,
  dataTypes = "basic", includeTree = FALSE, rows = NA, ...)
```

Arguments

searchterm	character; A vector of common or scientific names.
fuzzy	(logical) Whether to use fuzzy search or not (default: FALSE).
dataTypes	(character) Specifies the datatypes that will be returned. See bold_search for options.
includeTree	(logical) If TRUE (default: FALSE), returns a list containing information for parent taxa as well as the specified taxon.
ask	logical; should get_tsn be run in interactive mode? If TRUE and more than one TSN is found for teh species, the user is asked for input. If FALSE NA is returned for multiple matches.
verbose	logical; should progress be printed?
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a boldid class object with one to many identifiers. See get_boldid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
rank	(character) A taxonomic rank name. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
division	(character) A division (aka phylum) name. Optional. See Filtering below.
parent	(character) A parent name (i.e., the parent of the target search taxon). Optional. See Filtering below.
...	Curl options passed on to verb-GET
x	Input to as.boldid
check	logical; Check if ID matches any existing on the DB, only used in as.boldid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

Filtering

The parameters division, parent, and rank are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For all these parameters, you can use regex strings since we use grep internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

See Also

classification

Other taxonomic-ids: get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_boldid(searchterm = "Agapostemon")
get_boldid(searchterm = "Chironomus riparius")
get_boldid(c("Chironomus riparius","Quercus douglasii"))
splist <- names_list('species')
get_boldid(splist, verbose=FALSE)

# Fuzzy searching
get_boldid(searchterm="Osmi", fuzzy=TRUE)

# Get back a subset
get_boldid(searchterm="Osmi", fuzzy=TRUE, rows = 1)
get_boldid(searchterm="Osmi", fuzzy=TRUE, rows = 1:10)
get_boldid(searchterm=c("Osmi","Aga"), fuzzy=TRUE, rows = 1)
get_boldid(searchterm=c("Osmi","Aga"), fuzzy=TRUE, rows = 1:3)

# found
get_boldid('Epicordulia princeps')
get_boldid('Arigomphus furcifer')

# When not found
get_boldid("howdy")
get_boldid(c("Chironomus riparius", "howdy"))
get_boldid("Cordulegaster erronea")
get_boldid("Nasiaeshna pentacantha")

# Narrow down results to a division or rank, or both
## Satyrium example
### Results w/o narrowing
get_boldid("Satyrium")
### w/ phylum
get_boldid("Satyrium", division = "Plants")
get_boldid("Satyrium", division = "Animals")

## Rank example
get_boldid("Osmia", fuzzy = TRUE)
get_boldid("Osmia", fuzzy = TRUE, rank = "genus")

# Fuzzy filter on any filtering fields
## uses grep on the inside
get_boldid("Satyrium", division = "anim")
get_boldid("Aga", fuzzy = TRUE, parent = "*idae")

# Convert a boldid without class information to a boldid class
as.boldid(get_boldid("Agapostemon")) # already a boldid, returns the same
as.boldid(get_boldid(c("Agapostemon","Quercus douglasii"))) # same
as.boldid(1973) # numeric
as.boldid(c(1973,101009,98597)) # numeric vector, length > 1
as.boldid("1973") # character
as.boldid(c("1973","101009","98597")) # character vector, length > 1
as.boldid(list("1973","101009","98597")) # list, either numeric or character
```

```
## dont check, much faster
as.boldid("1973", check=FALSE)
as.boldid(1973, check=FALSE)
as.boldid(c("1973","101009","98597"), check=FALSE)
as.boldid(list("1973","101009","98597"), check=FALSE)

(out <- as.boldid(c(1973,101009,98597)))
data.frame(out)
as.boldid( data.frame(out) )

# Get all data back
get_boldid_("Osmia", fuzzy=TRUE, rows=1:5)
get_boldid_("Osmia", fuzzy=TRUE, rows=1)
get_boldid_(c("Osmi","Aga"), fuzzy=TRUE, rows = 1:3)

## End(Not run)
```

`get_colid` *Get the Catalogue of Life ID from taxonomic names*

Description

Get the Catalogue of Life ID from taxonomic names

Usage

```
get_colid(sciname, ask = TRUE, messages = TRUE, rows = NA,
  kingdom = NULL, phylum = NULL, class = NULL, order = NULL,
  family = NULL, rank = NULL, status = NULL, ...)

as.colid(x, check = TRUE)

## S3 method for class 'colid'
as.colid(x, check = TRUE)

## S3 method for class 'character'
as.colid(x, check = TRUE)

## S3 method for class 'list'
as.colid(x, check = TRUE)

## S3 method for class 'data.frame'
as.colid(x, check = TRUE)

## S3 method for class 'colid'
as.data.frame(x, ...)

get_colid_(sciname, messages = TRUE, rows = NA)
```

Arguments

sciname	character; scientific name.
ask	logical; should get_colid be run in interactive mode? If TRUE and more than one ID is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; If TRUE the actual taxon queried is printed on the console.
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a colid class object with one to many identifiers. See get_colid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
kingdom	(character) A kingdom name. Optional. See Filtering below.
phylum	(character) A phylum (aka division) name. Optional. See Filtering below.
class	(character) A class name. Optional. See Filtering below.
order	(character) An order name. Optional. See Filtering below.
family	(character) A family name. Optional. See Filtering below.
rank	(character) A taxonomic rank name. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
status	(character) A name status, e.g., "accepted name", "misapplied name", "synonym", "ambiguous synonym", "common name", and more. Optional. See Filtering below.
...	Ignored
x	Input to as.colid
check	logical; Check if ID matches any existing on the DB, only used in as.colid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

Number of results

We didn't used to, but as of **taxize** version v0.9.6 we paginate through results for any queries so that you get all results. For example, COL allows only 50 records per request for full responses that we request, so if a query results in 100 records, we make two requests to get all the data.

Filtering

The parameters kingdom, phylum, class, order, family, rank, and status are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For all these parameters, you can use regex strings since we use grep internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

See Also

classification

Other taxonomic-ids: get_boldid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_colid(sciname='Poa annua')
get_colid(sciname='Pinus contorta')
get_colid(sciname='Puma concolor')
get_colid(sciname="Abudefduf saxatilis")

get_colid(c("Poa annua", "Pinus contorta"))

# specify rows to limit choices available
get_colid(sciname='Poa annua')
get_colid(sciname='Poa annua', rows=1)
get_colid(sciname='Poa annua', rows=2)
get_colid(sciname='Poa annua', rows=1:2)

# When not found
get_colid(sciname="uaudnadndj")
get_colid(c("Chironomus riparius", "uaudnadndj"))

# Narrow down results to a division or rank, or both
## Satyrium example
### Results w/o narrowing
get_colid("Satyrium")
### w/ division
get_colid("Satyrium", kingdom = "Plantae")
get_colid("Satyrium", kingdom = "Animalia")

## Rank example
get_colid("Poa")
get_colid("Poa", kingdom = "Plantae")
get_colid("Poa", kingdom = "Animalia")

# Fuzzy filter on any filtering fields
## uses grep on the inside
get_colid("Satyrium", kingdom = "p")

# Convert a uid without class information to a uid class
as.colid(get_colid("Chironomus riparius")) # already a uid, returns the same
as.colid(get_colid(c("Chironomus riparius","Pinus contorta"))) # same
as.colid("714831352ad94741e4321eccdeb29f58") # character
# character vector, length > 1
```

```
as.colid(c("714831352ad94741e4321eccdeb29f58",
  "3b35900f74ff6e4b073ddb95c32b1f8d"))
# list, either numeric or character
as.colid(list("714831352ad94741e4321eccdeb29f58",
  "3b35900f74ff6e4b073ddb95c32b1f8d"))
## dont check, much faster
as.colid("714831352ad94741e4321eccdeb29f58", check=FALSE)
as.colid(c("714831352ad94741e4321eccdeb29f58",
 "3b35900f74ff6e4b073ddb95c32b1f8d"),
 check=FALSE)
as.colid(list("714831352ad94741e4321eccdeb29f58",
 "3b35900f74ff6e4b073ddb95c32b1f8d"),
 check=FALSE)

(out <- as.colid(c("714831352ad94741e4321eccdeb29f58",
 "3b35900f74ff6e4b073ddb95c32b1f8d")))
data.frame(out)
as.colid( data.frame(out) )

# Get all data back
get_colid_("Poa annua")
get_colid_("Poa annua", rows=2)
get_colid_("Poa annua", rows=1:2)
get_colid_(c("asdfadfasd","Pinus contorta"))

get_colid(sciname="Andropadus nigriceps fusciceps", rows=1)

# use curl options
get_colid("Quercus douglasii", verbose = TRUE)

## End(Not run)
```

`get_eolid` *Get the EOL ID from Encyclopedia of Life from taxonomic names.*

Description

Note that EOL doesn't expose an API endpoint for directly querying for EOL taxon ID's, so we first use the function `eol_search` to find pages that deal with the species of interest, then use `eol_pages` to find the actual taxon IDs.

Usage

```
get_eolid(sciname, ask = TRUE, messages = TRUE, key = NULL,
  rows = NA, rank = NULL, data_source = NULL, ...)

as.eolid(x, check = TRUE)

## S3 method for class 'eolid'
```

```
as.eolid(x, check = TRUE)

## S3 method for class 'character'
as.eolid(x, check = TRUE)

## S3 method for class 'list'
as.eolid(x, check = TRUE)

## S3 method for class 'numeric'
as.eolid(x, check = TRUE)

## S3 method for class 'data.frame'
as.eolid(x, check = TRUE)

## S3 method for class 'eolid'
as.data.frame(x, ...)

get_eolid_(sciname, messages = TRUE, key = NULL, rows = NA, ...)
```

Arguments

sciname	character; scientific name.
ask	logical; should get_eolid be run in interactive mode? If TRUE and more than one ID is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; If TRUE the actual taxon queried is printed on the console.
key	API key. passed on to eol_search and eol_pages internally. We recommend getting an API key; see taxize-authentication
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a eolid class object with one to many identifiers. See get_eolid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
rank	(character) A taxonomic rank name. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
data_source	(character) A data source inside of EOL. These are longish names like e.g., "Barcode of Life Data Systems" or "USDA PLANTS images". Optional. See Filtering below.
...	Further args passed on to eol_search
x	Input to as.eolid
check	logical; Check if ID matches any existing on the DB, only used in as.eolid

Details

EOL is a bit odd in that they have page IDs for each taxon, but then within that, they have taxon ids for various taxa within that page (e.g., GBIF and NCBI each have a taxon they refer to within

the page [i.e., taxon]). And we need the taxon ids from a particular data provider (e.g, NCBI) to do other things, like get a higher classification tree. However, humans want the page id, not the taxon id. So, the id returned from this function is the taxon id, not the page id. You can get the page id for a taxon by using eol_search and eol_pages, and the URI returned in the attributes for a taxon will lead you to the taxon page, and the ID in the URL is the page id.

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

Authentication

See taxize-authentication for help on authentication

Filtering

The parameters rank and data_source are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For all these parameters, you can use regex strings since we use grep internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_eolid(sciname='Pinus contorta')
get_eolid(sciname='Puma concolor')

get_eolid(c("Puma concolor", "Pinus contorta"))

# specify rows to limit choices available
get_eolid('Poa annua')
get_eolid('Poa annua', rows=1)
get_eolid('Poa annua', rows=2)
get_eolid('Poa annua', rows=1:2)

# When not found
```

```
get_eolid(sciname="uaudnadndj")
get_eolid(c("Chironomus riparius", "uaudnadndj"))

# filter results to a rank or data source, or both
get_eolid("Satyrium")
get_eolid("Satyrium", rank = "genus")
get_eolid("Satyrium", data_source = "INAT")
get_eolid("Satyrium", rank = "genus", data_source = "North Pacific")

# Convert a eolid without class information to a eolid class
# already a eolid, returns the same
as.eolid(get_eolid("Chironomus riparius"))
# same
as.eolid(get_eolid(c("Chironomus riparius","Pinus contorta")))
# numeric
as.eolid(10247706)
# numeric vector, length > 1
as.eolid(c(6985636,12188704,10247706))
# character
as.eolid("6985636")
# character vector, length > 1
as.eolid(c("6985636","12188704","10247706"))
# list, either numeric or character
as.eolid(list("6985636","12188704","10247706"))
## dont check, much faster
as.eolid("6985636", check=FALSE)
as.eolid(6985636, check=FALSE)
as.eolid(c("6985636","12188704","10247706"), check=FALSE)
as.eolid(list("6985636","12188704","10247706"), check=FALSE)

(out <- as.eolid(c(6985636,12188704,10247706)))
data.frame(out)
as.eolid( data.frame(out) )

# Get all data back
get_eolid_("Poa annua")
get_eolid_("Poa annua", rows=2)
get_eolid_("Poa annua", rows=1:2)
get_eolid_(c("asdfadfasd", "Pinus contorta"))

## End(Not run)
```

`get_gbifid` *Get the GBIF backbone taxon ID from taxonomic names.*

Description

Get the GBIF backbone taxon ID from taxonomic names.

Usage

```
get_gbifid(sciname, ask = TRUE, messages = TRUE, rows = NA,
  phylum = NULL, class = NULL, order = NULL, family = NULL,
  rank = NULL, method = "backbone", ...)

as.gbifid(x, check = FALSE)

## S3 method for class 'gbifid'
as.gbifid(x, check = FALSE)

## S3 method for class 'character'
as.gbifid(x, check = TRUE)

## S3 method for class 'list'
as.gbifid(x, check = TRUE)

## S3 method for class 'numeric'
as.gbifid(x, check = TRUE)

## S3 method for class 'data.frame'
as.gbifid(x, check = TRUE)

## S3 method for class 'gbifid'
as.data.frame(x, ...)

get_gbifid_(sciname, messages = TRUE, rows = NA, method = "backbone")
```

Arguments

sciname	character; scientific name.
ask	logical; should get_colid be run in interactive mode? If TRUE and more than one ID is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; If TRUE the actual taxon queried is printed on the console.
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a gbifid class object with one to many identifiers. See get_gbifid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
phylum	(character) A phylum (aka division) name. Optional. See Filtering below.
class	(character) A class name. Optional. See Filtering below.
order	(character) An order name. Optional. See Filtering below.
family	(character) A family name. Optional. See Filtering below.
rank	(character) A taxonomic rank name. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
method	(character) one of "backbone" or "lookup". See Details.

...	Ignored
x	Input to as.gbifid
check	logical; Check if ID matches any existing on the DB, only used in as.gbifid

Details

Internally in this function we use a function to search GBIF's taxonomy, and if we find an exact match we return the ID for that match. If there isn't an exact match we return the options to you to pick from.

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

method parameter

"backbone" uses the /species/match GBIF API route, matching against their backbone taxonomy. We turn on fuzzy matching by default, as the search without fuzzy against backbone is quite narrow. "lookup" uses the /species/search GBIF API route, doing a full text search of name usages covering scientific and vernacular named, species descriptions, distributions and the entire classification.

Filtering

The parameters phylum, class, order, family, and rank are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For all these parameters, you can use regex strings since we use grep internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_gbifid(sciname='Poa annua')
get_gbifid(sciname='Pinus contorta')
get_gbifid(sciname='Puma concolor')
```

```
# multiple names
get_gbifid(c("Poa annua", "Pinus contorta"))

# specify rows to limit choices available
get_gbifid(sciname='Pinus')
get_gbifid(sciname='Pinus', rows=10)
get_gbifid(sciname='Pinus', rows=1:3)

# When not found, NA given
get_gbifid(sciname="uaudnadndj")
get_gbifid(c("Chironomus riparius", "uaudnadndj"))

# Narrow down results to a division or rank, or both
## Satyrium example
### Results w/o narrowing
get_gbifid("Satyrium")
### w/ phylum
get_gbifid("Satyrium", phylum = "Tracheophyta")
get_gbifid("Satyrium", phylum = "Arthropoda")
### w/ phylum & rank
get_gbifid("Satyrium", phylum = "Arthropoda", rank = "genus")

## Rank example
get_gbifid("Poa", method = "lookup")
get_gbifid("Poa", method = "lookup", rank = "genus")
get_gbifid("Poa", method = "lookup", family = "Thripidae")

# Fuzzy filter on any filtering fields
## uses grep on the inside
get_gbifid("Satyrium", phylum = "arthropoda")
get_gbifid("A*", method = "lookup", order = "*tera")
get_gbifid("A*", method = "lookup", order = "*ales")

# Convert a uid without class information to a uid class
as.gbifid(get_gbifid("Poa annua")) # already a uid, returns the same
as.gbifid(get_gbifid(c("Poa annua","Puma concolor"))) # same
as.gbifid(2704179) # numeric
as.gbifid(c(2704179,2435099,3171445)) # numeric vector, length > 1
as.gbifid("2704179") # character
as.gbifid(c("2704179","2435099","3171445")) # character vector, length > 1
as.gbifid(list("2704179","2435099","3171445")) # list, either numeric or character
## dont check, much faster
as.gbifid("2704179", check=FALSE)
as.gbifid(2704179, check=FALSE)
as.gbifid(2704179, check=FALSE)
as.gbifid(c("2704179","2435099","3171445"), check=FALSE)
as.gbifid(list("2704179","2435099","3171445"), check=FALSE)

(out <- as.gbifid(c(2704179,2435099,3171445)))
data.frame(out)
as.uid( data.frame(out) )
```

```
# Get all data back
get_gbifid_("Puma concolor")
get_gbifid_(c("Pinus", "uaudnadndj"))
get_gbifid_(c("Pinus", "Puma"), rows=5)
get_gbifid_(c("Pinus", "Puma"), rows=1:5)

# use curl options
invisible(get_gbifid("Quercus douglasii", verbose = TRUE))

## End(Not run)
```

`get_ids` *Retrieve taxonomic identifiers for a given taxon name.*

Description

This is a convenience function to get identifiers across all data sources. You can use other `get_*` functions to get identifiers from specific sources if you like.

Usage

```
get_ids(names, db = c("itis", "ncbi", "eol", "col", "tropicos", "gbif",
  "nbn", "pow"), ...)

get_ids_(names, db = get_ids_dbs, rows = NA, ...)
```

Arguments

`names`	character; Taxonomic name to query.
`db`	character; database to query. One or more of `ncbi`, `itis`, `eol`, `col`, `tropicos`, `gbif`, `nbn`, or `pow`. By default db is set to search all data sources. Note that each taxonomic data source has their own identifiers, so that if you give the wrong db value for the identifier you could get a result, it will likely be wrong (not what you were expecting). If using ncbi, eol, and/or tropicos we recommend getting API keys; see `taxize-authentication`
`...`	Other arguments passed to `get_tsn`, `get_uid`, `get_eolid`, `get_colid`, `get_tpsid`, `get_gbifid`, `get_nbnid`.
`rows`	numeric; Any number from 1 to infinity. If the default NA, all rows are returned. When used in `get_ids` this function still only gives back a ids class object with one to many identifiers. See `get_ids_` to get back all, or a subset, of the raw data that you are presented during the ask process.

Value

A vector of taxonomic identifiers, each retaining their respective S3 classes so that each element can be passed on to another function (see e.g.'s).

Authentication

See taxize-authentication for help on authentication

Note

There is a timeout of 1/3 seconds between queries to NCBI.

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
# Plug in taxon names directly
## By default you get ids for all data sources
get_ids(names="Chironomus riparius")

# specify rows to limit choices available
get_ids(names="Poa annua", db=c("col", "eol"), rows=1)
get_ids(names="Poa annua", db=c("col", "eol"), rows=1:2)

## Or you can specify which source you want via the db parameter
get_ids(names="Chironomus riparius", db = 'ncbi')
get_ids(names="Salvelinus fontinalis", db = 'nbn')

get_ids(names=c("Chironomus riparius", "Pinus contorta"), db = 'ncbi')
get_ids(names=c("Chironomus riparius", "Pinus contorta"),
  db = c('ncbi','itis'))
get_ids(names=c("Chironomus riparius", "Pinus contorta"),
  db = c('ncbi','itis','col'))
get_ids(names="Pinus contorta",
  db = c('ncbi','itis','col','eol','tropicos'))
get_ids(names="ava avvva", db = c('ncbi','itis','col','eol','tropicos'))

# Pass on to other functions
out <- get_ids(names="Pinus contorta",
 db = c('ncbi','itis','col','eol','tropicos'))
classification(out$itis)
synonyms(out$tropicos)

# Get all data back
get_ids_(c("Chironomus riparius", "Pinus contorta"), db = 'nbn',
  rows=1:10)
get_ids_(c("Chironomus riparius", "Pinus contorta"), db = c('nbn','gbif'),
  rows=1:10)

# use curl options
get_ids("Agapostemon", db = "ncbi", verbose = TRUE)
```

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

`get_id_details` *Details on* `get_*()` *functions*

Description

Including outputs from `get_*()` functions, as well as their attributes, and all exception behaviors.

Details

This document applies to the following functions:

- `get_boldid`
- `get_colid`
- `get_eolid`
- `get_gbifid`
- `get_ids`
- `get_iucn`
- `get_natservid`
- `get_nbnid`
- `get_tolid`
- `get_tpsid`
- `get_tsn`
- `get_ubioid`
- `get_uid`
- `get_wiki`
- `get_wormsid`

attributes

Each output from `get_*()` functions have the following attributes:

- *match* (character) - the reason for NA, either 'not found', 'found' or if `ask = FALSE` then 'NA due to ask=FALSE')
- *multiple_matches* (logical) - Whether multiple matches were returned by the data source. This can be `TRUE`, even if you get 1 name back because we try to pattern match the name to see if there's any direct matches. So sometimes this attribute is `TRUE`, as well as `pattern_match`, which then returns 1 resulting name without user prompt.
- *pattern_match* (logical) - Whether a pattern match was made. If `TRUE` then `multiple_matches` must be `TRUE`, and we found a perfect match to your name, ignoring case. If `FALSE`, there wasn't a direct match, and likely you need to pick from many choices or further parameters can be used to limit results
- *uri* (character) - The URI where more information can be read on the taxon - includes the taxonomic identifier in the URL somewhere. This may be missing if the value returned is `NA`

exceptions

The following are the various ways in which `get_*()` functions behave:

- success - the value returned is a character string or numeric
- no matches found - you'll get an NA, refine your search or possible the taxon searched for does not exist in the database you're using
- more than on match and ask = FALSE - if there's more than one matching result, and you have set `ask = FALSE`, then we can't determine the single match to return, so we give back `NA`. However, in this case we do set the `match` attribute to say `NA due to ask=FALSE & > 1 result` so it's very clear what happened - and you can even programatically check this as well
- NA due to some other reason - some `get_*()` functions have additional parameters for filtering taxa. It's possible that even though there's results (that is, `found` will say `TRUE`), you can get back an NA. This is most likely if the parameter filters taxa after they are returned from the data provider and the value passed to the parameter leads to no matches.

`get_iucn` *Get a IUCN Redlist taxon*

Description

Get a IUCN Redlist taxon

Usage

```
get_iucn(x, messages = TRUE, key = NULL, ...)

as.iucn(x, check = TRUE, key = NULL)

## S3 method for class 'iucn'
as.iucn(x, check = TRUE, key = NULL)

## S3 method for class 'character'
as.iucn(x, check = TRUE, key = NULL)

## S3 method for class 'list'
as.iucn(x, check = TRUE, key = NULL)

## S3 method for class 'numeric'
as.iucn(x, check = TRUE, key = NULL)

## S3 method for class 'data.frame'
as.iucn(x, check = TRUE, key = NULL)

## S3 method for class 'iucn'
as.data.frame(x, ...)
```

Arguments

x	(character) A vector of common or scientific names
messages	logical; should progress be printed?
key	(character) required. you IUCN Redlist API key. See rredlist-package for help on authenticating with IUCN Redlist
...	Ignored
check	(logical) Check if ID matches any existing on the DB, only used in as.iucn

Details

There is no underscore method, because there's no real search for IUCN, that is, where you search for a string, and get back a bunch of results due to fuzzy matching. If that exists in the future we'll add an underscore method here.

IUCN ids only work with synonyms and sci2comm methods.

Value

A vector of taxonomic identifiers as an S3 class.

Comes with the following attributes:

- *match* (character) - the reason for NA, either 'not found', 'found' or if ask = FALSE then 'NA due to ask=FALSE')
- *name* (character) - the taxonomic name, which is needed in synonyms and sci2comm methods since they internally use **rredlist** functions which require the taxonomic name, and not the taxonomic identifier
- *uri* (character) - The URI where more information can be read on the taxon - includes the taxonomic identifier in the URL somewhere

multiple_matches and *pattern_match* do not apply here as in other get_* methods since there is no IUCN Redlist search, so you either get a match or you do not get a match.

See Also

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_iucn(x = "Branta canadensis")
get_iucn(x = "Branta bernicla")
get_iucn(x = "Panthera uncia")

# as coercion
as.iucn(22732)
as.iucn("22732")
(res <- as.iucn(c(22679946, 22732, 22679935)))
data.frame(res)
```

```
as.iucn(data.frame(res))

## End(Not run)
```

`get_natservid` *Get NatureServe taxonomic ID for a taxon name*

Description

Get NatureServe taxonomic ID for a taxon name

Usage

```
get_natservid(query, searchtype = "scientific", ask = TRUE,
  verbose = TRUE, rows = NA, key = NULL, ...)

as.natservid(x, check = TRUE)

## S3 method for class 'natservid'
as.natservid(x, check = TRUE)

## S3 method for class 'character'
as.natservid(x, check = TRUE)

## S3 method for class 'list'
as.natservid(x, check = TRUE)

## S3 method for class 'numeric'
as.natservid(x, check = TRUE)

## S3 method for class 'data.frame'
as.natservid(x, check = TRUE)

## S3 method for class 'natservid'
as.data.frame(x, ...)

get_natservid_(query, verbose = TRUE, rows = NA, key = NULL, ...)
```

Arguments

`query`	character; A vector of common or scientific names.
`searchtype`	character; One of 'scientific' (default) or 'common'. This doesn't affect the query to NatureServe - but rather affects what column of data is targeted in name filtering post data request.
`ask`	logical; should get_natservid be run in interactive mode? If `TRUE` and more than one wormsid is found for the species, the user is asked for input. If `FALSE` NA is returned for multiple matches.

verbose	logical; should progress be printed?
rows	numeric; Any number from 1 to infinity. If the default NaN, all rows are considered. Note that this function still only gives back a natservid class object with one to many identifiers. See get_natservid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
key	(character) your NatureServe API key. Required. See **Authentication** below for more.
...	Ignored
x	Input to as.natservid
check	logical; Check if ID matches any existing on the DB, only used in as.natservid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

Authentication

Get an API key from NatureServe at https://services.natureserve.org/developer/index.jsp. You can pass your token in as an argument or store it one of two places:

- your .Rprofile file with an entry like options(NatureServeKey = "your-natureserve-key")
- your .Renviron file with an entry like NATURE_SERVE_KEY=your-natureserve-key

See Startup for information on how to create/find your .Rprofile and .Renviron files

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
(x <- get_natservid("Helianthus annuus"))
attributes(x)
attr(x, "match")
attr(x, "multiple_matches")
attr(x, "pattern_match")
attr(x, "uri")

get_natservid('Gadus morhua')
get_natservid(c("Helianthus annuus", 'Gadus morhua'))
```

```
# specify rows to limit choices available
get_natservid('Ruby Quaker Moth', 'common')
get_natservid('Ruby*', 'common')
get_natservid('Ruby*', 'common', rows=1)
get_natservid('Ruby*', 'common', rows=1:2)

# When not found
get_natservid("howdy")
get_natservid(c('Gadus morhua', "howdy"))

# Convert a natservid without class information to a natservid class
# already a natservid, returns the same
as.natservid(get_natservid('Gadus morhua'))
# same
as.natservid(get_natservid(c('Gadus morhua', 'Pomatomus saltatrix')))
# character
as.natservid("ELEMENT_GLOBAL.2.101905")
# character vector, length > 1
as.natservid(c("ELEMENT_GLOBAL.2.101905", "ELEMENT_GLOBAL.2.101998"))
# list, either numeric or character
as.natservid(list("ELEMENT_GLOBAL.2.101905", "ELEMENT_GLOBAL.2.101998"))
## dont check, much faster
as.natservid("ELEMENT_GLOBAL.2.101905", check = FALSE)
as.natservid(c("ELEMENT_GLOBAL.2.101905", "ELEMENT_GLOBAL.2.101998"),
  check = FALSE)
as.natservid(list("ELEMENT_GLOBAL.2.101905", "ELEMENT_GLOBAL.2.101998"),
  check = FALSE)

(out <- as.natservid(
  c("ELEMENT_GLOBAL.2.101905", "ELEMENT_GLOBAL.2.101998")))
data.frame(out)
as.natservid( data.frame(out) )

# Get all data back
get_natservid_("Ruby*")
get_natservid_("Ruby*", rows=1:3)

## End(Not run)
```

---

get_nbnid *Get the UK National Biodiversity Network ID from taxonomic names.*

Description

Get the UK National Biodiversity Network ID from taxonomic names.

Usage

```
get_nbnid(name, ask = TRUE, messages = TRUE, rec_only = FALSE,
  rank = NULL, rows = NA, ...)
```

```
as.nbnid(x, check = TRUE)

## S3 method for class 'nbnid'
as.nbnid(x, check = TRUE)

## S3 method for class 'character'
as.nbnid(x, check = TRUE)

## S3 method for class 'list'
as.nbnid(x, check = TRUE)

## S3 method for class 'data.frame'
as.nbnid(x, check = TRUE)

## S3 method for class 'nbnid'
as.data.frame(x, ...)

get_nbnid_(name, messages = TRUE, rec_only = FALSE, rank = NULL,
  rows = NA, ...)
```

Arguments

name	character; scientific name.
ask	logical; should get_nbnid be run in interactive mode? If TRUE and more than one ID is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; If TRUE the actual taxon queried is printed on the console.
rec_only	(logical) If TRUE ids of recommended names are returned (i.e. synonyms are removed). Defaults to FALSE. Remember, the id of a synonym is a taxa with 'recommended' name status.
rank	(character) If given, we attempt to limit the results to those taxa with the matching rank.
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a nbnid class object with one to many identifiers. See get_nbnid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
...	Further args passed on to nbn_search
x	Input to as.nbnid
check	logical; Check if ID matches any existing on the DB, only used in as.nbnid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

an object of class nbnid, a light wrapper around a character string that is the taxonomic ID - includes attributes with relavant metadata

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

References

<https://api.nbnatlas.org/>

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Other nbn: nbn_classification, nbn_search, nbn_synonyms

Examples

```
## Not run:
get_nbnid(name='Poa annua')
get_nbnid(name='Poa annua', rec_only=TRUE)
get_nbnid(name='Poa annua', rank='Species')
get_nbnid(name='Poa annua', rec_only=TRUE, rank='Species')
get_nbnid(name='Pinus contorta')

# The NBN service handles common names too
get_nbnid(name='red-winged blackbird')

# specify rows to limit choices available
get_nbnid('Poa annua')
get_nbnid('Poa annua', rows=1)
get_nbnid('Poa annua', rows=25)
get_nbnid('Poa annua', rows=1:2)

# When not found
get_nbnid(name="uaudnadndj")
get_nbnid(c("Zootoca vivipara", "uaudnadndj"))
get_nbnid(c("Zootoca vivipara","Chironomus riparius", "uaudnadndj"))

# Convert an nbnid without class information to a nbnid class
as.nbnid(get_nbnid("Zootoca vivipara")) # already a nbnid, returns the same
as.nbnid(get_nbnid(c("Zootoca vivipara","Pinus contorta"))) # same
as.nbnid('NHMSYS0001706186') # character
# character vector, length > 1
as.nbnid(c("NHMSYS0001706186","NHMSYS0000494848","NBNSYS0000010867"))
# list
as.nbnid(list("NHMSYS0001706186","NHMSYS0000494848","NBNSYS0000010867"))
## dont check, much faster
```

```
as.nbnid('NHMSYS0001706186', check=FALSE)
as.nbnid(list("NHMSYS0001706186","NHMSYS0000494848","NBNSYS0000010867"),
  check=FALSE)

(out <- as.nbnid(c("NHMSYS0001706186","NHMSYS0000494848",
  "NBNSYS0000010867")))
data.frame(out)
as.nbnid( data.frame(out) )

# Get all data back
get_nbnid_("Zootoca vivipara")
get_nbnid_("Poa annua", rows=2)
get_nbnid_("Poa annua", rows=1:2)
get_nbnid_(c("asdfadfasd","Pinus contorta"), rows=1:5)

# use curl options
invisible(get_nbnid("Quercus douglasii", verbose = TRUE))

## End(Not run)
```

---

get_pow *Get Kew's Plants of the World code for a taxon*

Description

Get Kew's Plants of the World code for a taxon

Usage

```
get_pow(x, accepted = FALSE, ask = TRUE, messages = TRUE,
  rows = NA, family_filter = NULL, rank_filter = NULL, ...)

as.pow(x, check = TRUE)

## S3 method for class 'pow'
as.pow(x, check = TRUE)

## S3 method for class 'character'
as.pow(x, check = TRUE)

## S3 method for class 'list'
as.pow(x, check = TRUE)

## S3 method for class 'data.frame'
as.pow(x, check = TRUE)

## S3 method for class 'pow'
as.data.frame(x, ...)

get_pow_(x, messages = TRUE, rows = NA, ...)
```

Arguments

x	character; A vector of common or scientific names
accepted	logical; If TRUE, removes names that are not accepted valid names by ITIS. Set to FALSE (default) to give back both accepted and unaccepted names.
ask	logical; should get_pow be run in interactive mode? If TRUE and more than one pow is found for teh species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; should progress be printed?
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a pow class object with one to many identifiers. See get_pow_ to get back all, or a subset, of the raw data that you are presented during the ask process.
family_filter	(character) A division (aka phylum) name to filter data after retrieved from NCBI. Optional. See Filtering below.
rank_filter	(character) A taxonomic rank name to filter data after retrieved from NCBI. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
...	Curl options passed on to HttpClient
check	logical; Check if ID matches any existing on the DB, only used in as.pow

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

Filtering

The parameters family_filter and rank_filter are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For these two parameters, you can use regex strings since we use grep internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

See Also

classification

Other pow: pow_lookup, pow_search

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_pow(x = "Helianthus")
get_pow(c("Helianthus","Quercus douglasii"))

# Get back a subset
get_pow(x="Helianthus", rows = 1)
get_pow(x="Helianthus", rows = 1:10)

# When not found
get_pow("howdy")
get_pow(c("Helianthus annuus", "howdy"))

# Narrow down results
# to accepted names
get_pow("Helianthus", accepted = TRUE)
# to a kingom
get_pow("Helianthus", rank_filter = "genus")
# to accepted names and rank
get_pow("Helianthus annuus", accepted = TRUE, rank_filter = "species")
# to a family
get_pow("flower", family_filter = "Acanthaceae")

# Convert a pow without class information to a pow class
z <- get_pow("Helianthus annuus", accepted = TRUE, rank_filter = "species")
# already a pow, returns the same
as.pow(z)
as.pow("urn:lsid:ipni.org:names:119003-2")
# character vector, length > 1
ids <- c("urn:lsid:ipni.org:names:119003-2","urn:lsid:ipni.org:names:328247-2")
as.pow(ids)
# list, with character strings
as.pow(as.list(ids))
## dont check, much faster
as.pow("urn:lsid:ipni.org:names:119003-2", check=FALSE)
as.pow(ids, check=FALSE)
as.pow(as.list(ids), check=FALSE)

(out <- as.pow(ids))
data.frame(out)
as.pow( data.frame(out) )

# Get all data back
get_pow_("Quercus", rows=1:5)
get_pow_("Quercus", rows=1)
get_pow_(c("Pinus", "Abies"), rows = 1:3)

## End(Not run)
```

`get_tolid` *Get the OTT id for a search term*

Description

Retrieve the Open Tree of Life Taxonomy (OTT) id of a taxon from OpenTreeOfLife

Usage

```
get_tolid(sciname, ask = TRUE, verbose = TRUE, rows = NA, ...)

as.tolid(x, check = TRUE)

## S3 method for class 'tolid'
as.tolid(x, check = TRUE)

## S3 method for class 'character'
as.tolid(x, check = TRUE)

## S3 method for class 'list'
as.tolid(x, check = TRUE)

## S3 method for class 'numeric'
as.tolid(x, check = TRUE)

## S3 method for class 'data.frame'
as.tolid(x, check = TRUE)

## S3 method for class 'tolid'
as.data.frame(x, ...)

get_tolid_(sciname, verbose = TRUE, rows = NA)
```

Arguments

sciname	character; scientific name.
ask	logical; should get_tolid be run in interactive mode? If TRUE and more than one TOL is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
verbose	logical; should progress be printed?
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a tol class object with one to many identifiers. See get_tolid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
...	Ignored
x	Input to as.tolid
check	logical; Check if ID matches any existing on the DB, only used in as.tolid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tpsid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_tolid(sciname = "Quercus douglasii")
get_tolid(sciname = "Chironomus riparius")
get_tolid(c("Chironomus riparius","Quercus douglasii"))
splist <- c("annona cherimola", 'annona muricata', "quercus robur",
"shorea robusta", "pandanus patina", "oryza sativa", "durio zibethinus")
get_tolid(splist, verbose=FALSE)

# specify rows to limit choices available
get_tolid('Arni')
get_tolid('Arni', rows=1)
get_tolid('Arni', rows=1:2)

# When not found
get_tolid("howdy")
get_tolid(c("Chironomus riparius", "howdy"))

# Convert a tol without class information to a tol class
as.tolid(get_tolid("Quercus douglasii")) # already a tol, returns the same
as.tolid(get_tolid(c("Chironomus riparius","Pinus contorta"))) # same
as.tolid(3930798) # numeric
as.tolid(c(3930798,515712,872577)) # numeric vector, length > 1
as.tolid("3930798") # character
as.tolid(c("3930798","515712","872577")) # character vector, length > 1
as.tolid(list("3930798","515712","872577")) # list, either numeric or character
## dont check, much faster
as.tolid("3930798", check=FALSE)
as.tolid(3930798, check=FALSE)
as.tolid(c("3930798","515712","872577"), check=FALSE)
as.tolid(list("3930798","515712","872577"), check=FALSE)

(out <- as.tolid(c(3930798,515712,872577)))
data.frame(out)
as.tolid( data.frame(out) )

# Get all data back
```

```
get_tolid_(sciname="Arni")
get_tolid_("Arni", rows=1)
get_tolid_("Arni", rows=1:2)
get_tolid_(c("asdfadfasd","Pinus contorta"))

## End(Not run)
```

`get_tpsid` *Get the NameID codes from Tropicos for taxonomic names.*

Description

Get the NameID codes from Tropicos for taxonomic names.

Usage

```
get_tpsid(sciname, ask = TRUE, messages = TRUE, key = NULL,
  rows = NA, family = NULL, rank = NULL, ...)

as.tpsid(x, check = TRUE)

## S3 method for class 'tpsid'
as.tpsid(x, check = TRUE)

## S3 method for class 'character'
as.tpsid(x, check = TRUE)

## S3 method for class 'list'
as.tpsid(x, check = TRUE)

## S3 method for class 'numeric'
as.tpsid(x, check = TRUE)

## S3 method for class 'data.frame'
as.tpsid(x, check = TRUE)

## S3 method for class 'tpsid'
as.data.frame(x, ...)

get_tpsid_(sciname, messages = TRUE, key = NULL, rows = NA, ...)
```

Arguments

`sciname`	(character) One or more scientific name's as a vector or list.
`ask`	logical; should get_tpsid be run in interactive mode? If TRUE and more than one ID is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.

messages	logical; If TRUE the actual taxon queried is printed on the console.
key	Your API key; see taxize-authentication
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a tpsid class object with one to many identifiers. See get_tpsid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
family	(character) A family name. Optional. See Filtering below.
rank	(character) A taxonomic rank name. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
...	Other arguments passed to tp_search.
x	Input to as.tpsid
check	logical; Check if ID matches any existing on the DB, only used in as.tpsid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

Filtering

The parameters family and rank are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For all these parameters, you can use regex strings since we use grep internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tsn, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_tpsid(sciname='Poa annua')
get_tpsid(sciname='Pinus contorta')

get_tpsid(c("Poa annua", "Pinus contorta"))
```

```
# specify rows to limit choices available
get_tpsid('Poa annua')
get_tpsid('Poa annua', rows=1)
get_tpsid('Poa annua', rows=25)
get_tpsid('Poa annua', rows=1:2)

# When not found, NA given (howdy is not a species name, and Chrinomus is a fly)
get_tpsid("howdy")
get_tpsid(c("Chironomus riparius", "howdy"))

# Narrow down results to a division or rank, or both
## Satyrium example
### Results w/o narrowing
get_tpsid("Satyrium")
### w/ rank
get_tpsid("Satyrium", rank = "var.")
get_tpsid("Satyrium", rank = "sp.")

## w/ family
get_tpsid("Poa")
get_tpsid("Poa", family = "Iridaceae")
get_tpsid("Poa", family = "Orchidaceae")
get_tpsid("Poa", family = "Orchidaceae", rank = "gen.")

# Fuzzy filter on any filtering fields
## uses grep on the inside
get_tpsid("Poa", family = "orchidaceae")
get_tpsid("Aga", fuzzy = TRUE, parent = "*idae")

# pass to classification function to get a taxonomic hierarchy
classification(get_tpsid(sciname='Poa annua'))

# factor class names are converted to character internally
spnames <- as.factor(c("Poa annua", "Pinus contorta"))
class(spnames)
get_tpsid(spnames)

# pass in a list, works fine
get_tpsid(list("Poa annua", "Pinus contorta"))

# Convert a tpsid without class information to a tpsid class
as.tpsid(get_tpsid("Pinus contorta")) # already a tpsid, returns the same
as.tpsid(get_tpsid(c("Chironomus riparius","Pinus contorta"))) # same
as.tpsid(24900183) # numeric
as.tpsid(c(24900183,50150089,50079838)) # numeric vector, length > 1
as.tpsid("24900183") # character
as.tpsid(c("24900183","50150089","50079838")) # character vector, length > 1
as.tpsid(list("24900183","50150089","50079838")) # list, either numeric or character
## dont check, much faster
as.tpsid("24900183", check=FALSE)
as.tpsid(24900183, check=FALSE)
as.tpsid(c("24900183","50150089","50079838"), check=FALSE)
as.tpsid(list("24900183","50150089","50079838"), check=FALSE)
```

```
(out <- as.tpsid(c(24900183,50150089,50079838)))
data.frame(out)
as.tpsid( data.frame(out) )

# Get all data back
get_tpsid_("Poa annua")
get_tpsid_("Poa annua", rows=2)
get_tpsid_("Poa annua", rows=1:2)
get_tpsid_(c("asdfadfasd","Pinus contorta"), rows=1:5)

# use curl options
invisible(get_tpsid("Quercus douglasii", verbose = TRUE))

## End(Not run)
```

---

get_tsn *Get the TSN code for a search term.*

Description

Retrieve the taxonomic serial numbers (TSN) of a taxon from ITIS.

Usage

```
get_tsn(searchterm, searchtype = "scientific", accepted = FALSE,
  ask = TRUE, messages = TRUE, rows = NA, ...)

as.tsn(x, check = TRUE)

## S3 method for class 'tsn'
as.tsn(x, check = TRUE)

## S3 method for class 'character'
as.tsn(x, check = TRUE)

## S3 method for class 'list'
as.tsn(x, check = TRUE)

## S3 method for class 'numeric'
as.tsn(x, check = TRUE)

## S3 method for class 'data.frame'
as.tsn(x, check = TRUE)

## S3 method for class 'tsn'
as.data.frame(x, ...)
```

```
get_tsn_(searchterm, messages = TRUE, searchtype = "scientific",
  accepted = TRUE, rows = NA, ...)
```

Arguments

searchterm	character; A vector of common or scientific names.
searchtype	character; One of 'scientific' or 'common', or any unique abbreviation
accepted	logical; If TRUE, removes names that are not accepted valid names by ITIS. Set to FALSE (default) to give back both accepted and unaccepted names.
ask	logical; should get_tsn be run in interactive mode? If TRUE and more than one TSN is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; should progress be printed?
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a tsn class object with one to many identifiers. See get_tsn_ to get back all, or a subset, of the raw data that you are presented during the ask process.
...	Ignored
x	Input to as.tsn
check	logical; Check if ID matches any existing on the DB, only used in as.tsn

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_uid, get_wiki, get_wormsid

Examples

```
## Not run:
get_tsn("Quercus douglasii")
get_tsn("Chironomus riparius")
get_tsn(c("Chironomus riparius","Quercus douglasii"))
splist <- c("annona cherimola", 'annona muricata', "quercus robur",
"shorea robusta", "pandanus patina", "oryza sativa", "durio zibethinus")
get_tsn(splist, verbose=FALSE)

# specify rows to limit choices available
get_tsn('Arni')
```

```
get_tsn('Arni', rows=1)
get_tsn('Arni', rows=1:2)

# When not found
get_tsn("howdy")
get_tsn(c("Chironomus riparius", "howdy"))

# Using common names
get_tsn(searchterm="black bear", searchtype="common")

# Convert a tsn without class information to a tsn class
as.tsn(get_tsn("Quercus douglasii")) # already a tsn, returns the same
as.tsn(get_tsn(c("Chironomus riparius","Pinus contorta"))) # same
as.tsn(19322) # numeric
as.tsn(c(19322,129313,506198)) # numeric vector, length > 1
as.tsn("19322") # character
as.tsn(c("19322","129313","506198")) # character vector, length > 1
as.tsn(list("19322","129313","506198")) # list, either numeric or character
## dont check, much faster
as.tsn("19322", check=FALSE)
as.tsn(19322, check=FALSE)
as.tsn(c("19322","129313","506198"), check=FALSE)
as.tsn(list("19322","129313","506198"), check=FALSE)

(out <- as.tsn(c(19322,129313,506198)))
data.frame(out)
as.tsn( data.frame(out) )

# Get all data back
get_tsn_("Arni")
get_tsn_("Arni", rows=1)
get_tsn_("Arni", rows=1:2)
get_tsn_(c("asdfadfasd","Pinus contorta"), rows=1:5)

## End(Not run)
```

`get_ubioid` *Get the uBio id for a search term*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
get_ubioid(searchterm, searchtype = "scientific", ask = TRUE,
  verbose = TRUE, rows = NA, family = NULL, rank = NULL, ...)

as.ubioid(x, check = TRUE)
```

```
## S3 method for class 'ubioid'
as.ubioid(x, check = TRUE)

## S3 method for class 'character'
as.ubioid(x, check = TRUE)

## S3 method for class 'list'
as.ubioid(x, check = TRUE)

## S3 method for class 'numeric'
as.ubioid(x, check = TRUE)

## S3 method for class 'data.frame'
as.ubioid(x, check = TRUE)

## S3 method for class 'ubioid'
as.data.frame(x, ...)

get_ubioid_(searchterm, verbose = TRUE, searchtype = "scientific",
  rows = NA)
```

Arguments

searchterm	character; A vector of common or scientific names.
searchtype	character; One of 'scientific' or 'common', or any unique abbreviation
ask	logical; should get_tsn be run in interactive mode? If TRUE and more than one TSN is found for teh species, the user is asked for input. If FALSE NA is returned for multiple matches.
verbose	logical; should progress be printed?
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a ubioid class object with one to many identifiers. See get_ubioid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
family	(character) A family name. Optional. See Filtering below.
rank	(character) A taxonomic rank name. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
...	Ignored
x	Input to as.ubioid
check	logical; Check if ID matches any existing on the DB, only used in as.ubioid

Value

A vector of uBio ids. If a taxon is not found NA is given. If more than one uBio id is found the function asks for user input (if ask = TRUE), otherwise returns NA. Comes with an attribute *match* to investigate the reason for NA (either 'not found', 'found' or if ask = FALSE 'NA due to ask=FALSE')

Filtering

The parameters `family` and `rank` are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For all these parameters, you can use regex strings since we use `grep` internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

See Also

`get_uid`, `ubio_search`

`get_uid` *Get the UID codes from NCBI for taxonomic names.*

Description

Retrieve the Unique Identifier (UID) of a taxon from NCBI taxonomy browser.

Usage

```
get_uid(sciname, ask = TRUE, messages = TRUE, rows = NA,
  modifier = NULL, rank_query = NULL, division_filter = NULL,
  rank_filter = NULL, key = NULL, ...)

as.uid(x, check = TRUE)

## S3 method for class 'uid'
as.uid(x, check = TRUE)

## S3 method for class 'character'
as.uid(x, check = TRUE)

## S3 method for class 'list'
as.uid(x, check = TRUE)

## S3 method for class 'numeric'
as.uid(x, check = TRUE)

## S3 method for class 'data.frame'
as.uid(x, check = TRUE)

## S3 method for class 'uid'
as.data.frame(x, ...)

get_uid_(sciname, messages = TRUE, rows = NA, key = NULL, ...)
```

Arguments

sciname	character; scientific name.
ask	logical; should get_uid be run in interactive mode? If TRUE and more than one TSN is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; If 'TRUE' (default) the actual taxon queried is printed on the console.
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a uid class object with one to many identifiers. See get_uid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
modifier	(character) A modifier to the sciname given. Options include: Organism, Scientific Name, Common Name, All Names, Division, Filter, Lineage, GC, MGC, Name Tokens, Next Level, PGC, Properties, Rank, Subtree, Synonym, Text Word. These are not checked, so make sure they are entered correctly, as is.
rank_query	(character) A taxonomic rank name to modify the query sent to NCBI. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Querying below.
division_filter	(character) A division (aka phylum) name to filter data after retrieved from NCBI. Optional. See Filtering below.
rank_filter	(character) A taxonomic rank name to filter data after retrieved from NCBI. See rank_ref for possible options. Though note that some data sources use atypical ranks, so inspect the data itself for options. Optional. See Filtering below.
key	(character) NCBI Entrez API key. optional. See Details.
...	Ignored
x	Input to as.uid
check	logical; Check if ID matches any existing on the DB, only used in as.uid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

Querying

The parameter rank_query is used in the search sent to NCBI, whereas rank_filter filters data after it comes back. The parameter modifier adds modifiers to the name. For example, modifier="Organism" adds that to the name, giving e.g., Helianthus[Organism].

Filtering

The parameters `division_filter` and `rank_filter` are not used in the search to the data provider, but are used in filtering the data down to a subset that is closer to the target you want. For all these parameters, you can use regex strings since we use `grep` internally to match. Filtering narrows down to the set that matches your query, and removes the rest.

Beware

NCBI does funny things sometimes. E.g., if you search on Fringella morel, a slight misspelling of the genus name, and a non-existent epithet, NCBI gives back a morel fungal species. In addition, NCBI doesn't really do fuzzy searching very well, so if there is a slight mis-spelling in your names, you likely won't get what you are expecting. The lesson: clean your names before using this function. Other data sources are better about fuzzy matching.

Authentication

See `taxize-authentication` for help on authentication

Note that even though you can't pass in your key to 'as.uid' functions, we still use your Entrez API key if you have it saved as an R option or environment variable.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

See Also

`classification`

Other taxonomic-ids: `get_boldid`, `get_colid`, `get_eolid`, `get_gbifid`, `get_ids`, `get_iucn`, `get_natservid`, `get_nbnid`, `get_pow`, `get_tolid`, `get_tpsid`, `get_tsn`, `get_wiki`, `get_wormsid`

Examples

```
## Not run:
get_uid(c("Chironomus riparius", "Chaetopteryx"))
get_uid(c("Chironomus riparius", "aaa vva"))

# When not found
get_uid("howdy")
get_uid(c("Chironomus riparius", "howdy"))

# Narrow down results to a division or rank, or both
## By modifying the query
### w/ modifiers to the name
get_uid(sciname = "Aratinga acuticauda", modifier = "Organism")
get_uid(sciname = "bear", modifier = "Common Name")

### w/ rank query
get_uid(sciname = "Pinus", rank_query = "genus")
get_uid(sciname = "Pinus", rank_query = "subgenus")
### division query doesn't really work, for unknown reasons, so not available
```

```
## By filtering the result
## Echinacea example
### Results w/o narrowing
get_uid("Echinacea")
### w/ division
get_uid(sciname = "Echinacea", division_filter = "eudicots")
get_uid(sciname = "Echinacea", division_filter = "sea urchins")

## Satyrium example
### Results w/o narrowing
get_uid(sciname = "Satyrium")
### w/ division
get_uid(sciname = "Satyrium", division_filter = "monocots")
get_uid(sciname = "Satyrium", division_filter = "butterflies")

## Rank example
get_uid(sciname = "Pinus")
get_uid(sciname = "Pinus", rank_filter = "genus")
get_uid(sciname = "Pinus", rank_filter = "subgenus")

# Fuzzy filter on any filtering fields
## uses grep on the inside
get_uid("Satyrium", division_filter = "m")

# specify rows to limit choices available
get_uid('Dugesia') # user prompt needed
get_uid('Dugesia', rows=1) # 2 choices, so returns only 1 row, so no choices
get_uid('Dugesia', ask = FALSE) # returns NA for multiple matches

# Go to a website with more info on the taxon
res <- get_uid("Chironomus riparius")
browseURL(attr(res, "uri"))

# Convert a uid without class information to a uid class
as.uid(get_uid("Chironomus riparius")) # already a uid, returns the same
as.uid(get_uid(c("Chironomus riparius","Pinus contorta"))) # same
as.uid(315567) # numeric
as.uid(c(315567,3339,9696)) # numeric vector, length > 1
as.uid("315567") # character
as.uid(c("315567","3339","9696")) # character vector, length > 1
as.uid(list("315567","3339","9696")) # list, either numeric or character
## dont check, much faster
as.uid("315567", check=FALSE)
as.uid(315567, check=FALSE)
as.uid(c("315567","3339","9696"), check=FALSE)
as.uid(list("315567","3339","9696"), check=FALSE)

(out <- as.uid(c(315567,3339,9696)))
data.frame(out)
as.uid( data.frame(out) )

# Get all data back
```

```
get_uid_("Puma concolor")
get_uid_("Dugesia")
get_uid_("Dugesia", rows=2)
get_uid_("Dugesia", rows=1:2)
get_uid_(c("asdfadfasd","Pinus contorta"))

# use curl options
get_uid("Quercus douglasii", verbose = TRUE)

## End(Not run)
```

---

get_wiki *Get the page name for a Wiki taxon*

Description

Get the page name for a Wiki taxon

Usage

```
get_wiki(x, wiki_site = "species", wiki = "en", ask = TRUE,
  verbose = TRUE, limit = 100, rows = NA, ...)

as.wiki(x, check = TRUE, wiki_site = "species", wiki = "en")

## S3 method for class 'wiki'
as.wiki(x, check = TRUE, wiki_site = "species",
  wiki = "en")

## S3 method for class 'character'
as.wiki(x, check = TRUE, wiki_site = "species",
  wiki = "en")

## S3 method for class 'list'
as.wiki(x, check = TRUE, wiki_site = "species",
  wiki = "en")

## S3 method for class 'numeric'
as.wiki(x, check = TRUE, wiki_site = "species",
  wiki = "en")

## S3 method for class 'data.frame'
as.wiki(x, check = TRUE, wiki_site = "species",
  wiki = "en")

## S3 method for class 'wiki'
as.data.frame(x, ...)
```

```
get_wiki_(x, verbose = TRUE, wiki_site = "species", wiki = "en",
  limit = 100, rows = NA, ...)
```

Arguments

x	(character) A vector of common or scientific names.
wiki_site	(character) Wiki site. One of species (default), pedia, commons
wiki	(character) language. Default: en
ask	logical; should get_wiki be run in interactive mode? If TRUE and more than one wiki is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
verbose	logical; should progress be printed?
limit	(integer) number of records to return
rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. Note that this function still only gives back a wiki class object with one to many identifiers. See get_wiki_ to get back all, or a subset, of the raw data that you are presented during the ask process.
...	Ignored
check	logical; Check if ID matches any existing on the DB, only used in as.wiki

Details

For type = pedia, we use the english language site by default. Set the language parameter for a different language site.

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wormsid

Examples

```
## Not run:
get_wiki(x = "Quercus douglasii")
get_wiki(x = "Quercu")
get_wiki(x = "Quercu", "pedia")
get_wiki(x = "Quercu", "commons")
```

```
# diff. wikis with wikipedia
get_wiki("Malus domestica", "pedia")
get_wiki("Malus domestica", "pedia", "fr")

# as coercion
as.wiki("Malus_domestica")
as.wiki("Malus_domestica", wiki_site = "commons")
as.wiki("Malus_domestica", wiki_site = "pedia")
as.wiki("Malus_domestica", wiki_site = "pedia", wiki = "fr")
as.wiki("Malus_domestica", wiki_site = "pedia", wiki = "da")

## End(Not run)
```

`get_wormsid` *Get Worms ID for a taxon name*

Description

Retrieve Worms ID of a taxon from World Register of Marine Species (WORMS).

Usage

```
get_wormsid(query, searchtype = "scientific", accepted = FALSE,
  ask = TRUE, messages = TRUE, rows = NA, ...)

as.wormsid(x, check = TRUE)

## S3 method for class 'wormsid'
as.wormsid(x, check = TRUE)

## S3 method for class 'character'
as.wormsid(x, check = TRUE)

## S3 method for class 'list'
as.wormsid(x, check = TRUE)

## S3 method for class 'numeric'
as.wormsid(x, check = TRUE)

## S3 method for class 'data.frame'
as.wormsid(x, check = TRUE)

## S3 method for class 'wormsid'
as.data.frame(x, ...)

get_wormsid_(query, messages = TRUE, searchtype = "scientific",
  accepted = TRUE, rows = NA, ...)
```

Arguments

query	character; A vector of common or scientific names.
searchtype	character; One of 'scientific' or 'common', or any unique abbreviation
accepted	logical; If TRUE, removes names that are not accepted valid names by WORMS. Set to FALSE (default) to give back both accepted and unaccepted names.
ask	logical; should get_wormsid be run in interactive mode? If TRUE and more than one wormsid is found for the species, the user is asked for input. If FALSE NA is returned for multiple matches.
messages	logical; should progress be printed?
rows	numeric; Any number from 1 to infinity. If the default NaN, all rows are considered. Note that this function still only gives back a wormsid class object with one to many identifiers. See get_wormsid_ to get back all, or a subset, of the raw data that you are presented during the ask process.
...	Ignored
x	Input to as.wormsid
check	logical; Check if ID matches any existing on the DB, only used in as.wormsid

Value

A vector of taxonomic identifiers as an S3 class. If a taxon is not found an NA is given. If more than one identifier is found the function asks for user input if ask = TRUE, otherwise returns NA. If ask=FALSE and rows does not equal NA, then a data.frame is given back, but not of the uid class, which you can't pass on to other functions as you normally can.

See get_id_details for further details including attributes and exceptions

See Also

classification

Other taxonomic-ids: get_boldid, get_colid, get_eolid, get_gbifid, get_ids, get_iucn, get_natservid, get_nbnid, get_pow, get_tolid, get_tpsid, get_tsn, get_uid, get_wiki

Examples

```
## Not run:
(x <- get_wormsid('Platanista gangetica'))
attributes(x)
attr(x, "match")
attr(x, "multiple_matches")
attr(x, "pattern_match")
attr(x, "uri")

get_wormsid('Gadus morhua')
get_wormsid('Pomatomus saltatrix')
get_wormsid(c("Platanista gangetica", "Lichenopora neapolitana"))

# by common name
get_wormsid("dolphin", 'common')
```

```
get_wormsid("clam", 'common')

# specify rows to limit choices available
get_wormsid('Plat')
get_wormsid('Plat', rows=1)
get_wormsid('Plat', rows=1:2)

# When not found
get_wormsid("howdy")
get_wormsid(c('Gadus morhua', "howdy"))

# Convert a wormsid without class information to a wormsid class
# already a wormsid, returns the same
as.wormsid(get_wormsid('Gadus morhua'))
# same
as.wormsid(get_wormsid(c('Gadus morhua', 'Pomatomus saltatrix')))
# numeric
as.wormsid(126436)
# numeric vector, length > 1
as.wormsid(c(126436,151482))
# character
as.wormsid("126436")
# character vector, length > 1
as.wormsid(c("126436","151482"))
# list, either numeric or character
as.wormsid(list("126436","151482"))
## dont check, much faster
as.wormsid("126436", check=FALSE)
as.wormsid(126436, check=FALSE)
as.wormsid(c("126436","151482"), check=FALSE)
as.wormsid(list("126436","151482"), check=FALSE)

(out <- as.wormsid(c(126436,151482)))
data.frame(out)
as.wormsid( data.frame(out) )

# Get all data back
get_wormsid_("Plat")
get_wormsid_("Plat", rows=1)
get_wormsid_("Plat", rows=1:2)
get_wormsid_("Plat", rows=1:75)
# get_wormsid_(c("asdfadfasd","Plat"), rows=1:5)

## End(Not run)
```

---

gni_details *Search for taxonomic name details using the Global Names Index*

Description

Uses the Global Names Index, see http://gni.globalnames.org/

Usage

```
gni_details(id, all_records = 1, ...)
```

Arguments

id	Name id. Required.
all_records	If all_records is 1, GNI returns all records from all repositories for the name string (takes 0, or 1 [default]).
...	Curl options passed on to verb-GET

Value

Data.frame of results.

Author(s)

Scott Chamberlain myrmecocystus@gmail.com

See Also

gnr_datasources, gni_search.

Examples

```
## Not run:
gni_details(id = 17802847)
library("plyr")
ldply(list(1265133, 17802847), gni_details)

# pass on curl options
gni_details(id = 17802847, verbose = TRUE)

## End(Not run)
```

---

gni_parse *Parse scientific names using EOL's name parser.*

Description

Parse scientific names using EOL's name parser.

Usage

```
gni_parse(names, ...)
```

Arguments

names	A vector of length 1 or more of taxonomic names
...	Curl options passed on to verb-GET

Value

A data.frame with results, the submitted names, and the parsed names with additional information.

References

http://gni.globalnames.org/

See Also

gbif_parse

Examples

```
## Not run:
gni_parse("Cyanistes caeruleus")
gni_parse("Plantago minor")
gni_parse("Plantago minor minor")
gni_parse(c("Plantago minor minor","Helianthus annuus texanus"))

# pass on curl options
gni_parse("Cyanistes caeruleus", verbose = TRUE)

## End(Not run)
```

---

gni_search *Search for taxonomic names using the Global Names Index*

Description

Uses the Global Names Index, see http://gni.globalnames.org

Usage

```
gni_search(search_term = NULL, per_page = NULL, page = NULL,
  justtotal = FALSE, parse_names = FALSE, ...)
```

Arguments

search_term	Name pattern you want to search for. WARNING: Does not work for vernacular/common names. Search term may include following options (Note: can, uni, gen, sp, ssp, au, yr work only for parsed names • * wild card - Search by part of a word (E.g.: planta*)

- exact exact match - Search for exact match of a literal string (E.g.: exact:Parus major)
- ns name string- Search for literal string from its beginning (other modifiers will be ignored) (E.g.: ns:parus maj*)
- can canonical form- Search name without authors (other modifiers will be ignored) (E.g.: can:parus major)
- uni uninomial- Search for higher taxa (E.g.: uni:parus)
- gen genus - Search by genus epithet of species name (E.g.: gen:parus)
- sp species - Search by species epithet (E.g.: sp:major)
- ssp subspecies - Search by infraspecies epithet (E.g.: ssp:major)
- au author - Search by author word (E.g.: au:Shipunov)
- yr year - Search by year (E.g.: yr:2005)

per_page	Number of items per one page (numbers larger than 1000 will be decreased to 1000) (default is 30).
page	Page number you want to see (default is 1).
justtotal	Return only the total results found.
parse_names	If TRUE, use gni_parse to parse names. Default: FALSE
...	Curl options passed on to verb-GET

Details

Note that you can use fuzzy searching, e.g., by attaching an asterisk to the end of a search term. See the first two examples below

Value

data.frame of results.

Author(s)

Scott Chamberlain myrmecocystus@gmail.com

References

http://gni.globalnames.org/, https://github.com/dimus/gni/wiki/api

See Also

gnr_datasources, gni_search

Examples

```
## Not run:
gni_search(search_term = "ani*")
gni_search(search_term = "ama*", per_page = 3, page = 21)
gni_search(search_term = "animalia", per_page = 8, page = 1)
gni_search(search_term = "animalia", per_page = 8, page = 1, justtotal=TRUE)
```

```
gni_search(search_term = "Cyanistes caeruleus", parse_names=TRUE)

# pass on curl options
gni_search(search_term = "ani*", verbose = TRUE)

## End(Not run)
```

---

gnr_datasources *Global Names Resolver Data Sources*

Description

Retrieve data sources used in the Global Names Resolver

Usage

```
gnr_datasources(..., todf)
```

Arguments

...	Curl options passed on to HttpClient
todf	defunct, always get a data.frame back now

Value

data.frame/tibble

References

https://resolver.globalnames.org/data_sources

See Also

gnr_resolve, gni_search

Examples

```
## Not run:
# all data sources
gnr_datasources()

# give me the id for EOL
out <- gnr_datasources()
out[out$title == "EOL", "id"]

# Fuzzy search for sources with the word zoo
out <- gnr_datasources()
out[agrep("zoo", out$title, ignore.case = TRUE), ]

## End(Not run)
```

gnr_resolve *Resolve names using Global Names Resolver*

Description

See section **Age of datasets in the Global Names Resolver**

Usage

```
gnr_resolve(names, data_source_ids = NULL, resolve_once = FALSE,
  with_context = FALSE, canonical = FALSE, highestscore = TRUE,
  best_match_only = FALSE, preferred_data_sources = NULL,
  with_canonical_ranks = FALSE, http = "get", cap_first = TRUE,
  fields = "minimal", ...)
```

Arguments

names	character; taxonomic names to be resolved. Doesn't work for vernacular/common names.
data_source_ids	character; IDs to specify what data source is searched. See gnr_datasources.
resolve_once	logical; Find the first available match instead of matches across all data sources with all possible renderings of a name. When TRUE, response is rapid but incomplete.
with_context	logical; Reduce the likelihood of matches to taxonomic homonyms. When TRUE a common taxonomic context is calculated for all supplied names from matches in data sources that have classification tree paths. Names out of determined context are penalized during score calculation.
canonical	logical; If FALSE (default), gives back names with taxonomic authorities. If TRUE, returns canocial names (without tax. authorities and abbreviations).
highestscore	logical; Return those names with the highest score for each searched name? Defunct
best_match_only	(logical) If TRUE, best match only returned. Default: FALSE
preferred_data_sources	(character) A vector of one or more data source IDs.
with_canonical_ranks	(logical) Returns names with infraspecific ranks, if present. If TRUE, we force canonical=TRUE, otherwise this parameter would have no effect. Default: FALSE
http	The HTTP method to use, one of "get" or "post". Default: "get". Use http="post" with large queries. Queries with > 300 records use "post" automatically because "get" would fail
cap_first	(logical) For each name, fix so that the first name part is capitalized, while others are not. This web service is sensitive to capitalization, so you'll get different results depending on capitalization. First name capitalized is likely what you'll want and is the default. If FALSE, names are not modified. Default: TRUE

`fields`	(character) One of minimal (default) or all. Minimal gives back just four fields, whereas all gives all fields back.
`...`	Curl options passed on to `HttpClient`

Value

A data.frame with one attribute `not_known`: a character vector of taxa unknown to the Global Names Index. Access like `attr(output, "not_known")`, or `attributes(output)$not_known`.

Columns of the output data.frame:

- user_supplied_name (character) - the name you passed in to the `names` parameter, unchanged.
- submitted_name (character) - the actual name submitted to the GNR service
- data_source_id (integer/numeric) - data source ID
- data_source_title (character) - data source name
- gni_uuid (character) - Global Names Index UUID (aka identifier)
- matched_name (character) - the matched name in the GNR service
- matched_name2 (character) - returned if `canonical=TRUE`, in which case *matched_name* is not returned
- classification_path (character) - names of the taxonomic classification tree, with names separated by pipes (|)
- classification_path_ranks (character) - ranks of the taxonomic classification tree, with names separated by pipes (|)
- classification_path_ids (character) - identifiers of the taxonomic classification tree, with names separated by pipes (|)
- taxon_id (character) - taxon identifier
- edit_distance (integer/numeric) - edit distance
- imported_at (character) - date imported
- match_type (integer/numeric) - match type
- match_value (character) - description of match type
- prescore (character) - pre score
- score (numeric) - score
- local_id (character) - local identifier
- url (character) - URL for taxon
- global_id (character) - global identifier
- current_taxon_id (character) - current taxon id
- current_name_string (character) - current name string

Note that names (i.e. rows) are dropped that are NA, are zero length strings, are not character vectors, or are not found by the API.

Age of datasets in the Global Names Resolver

IMPORTANT: Datasets used in the Global Names Resolver vary in how recently they've been updated. See the `updated_at` field in the output of `gnr_datasources` for dates when each dataset was last updated.

preferred_data_sources

If `preferred_data_sources` is used, only the preferred data is returned - if it has any results.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

References

http://gnrd.globalnames.org/api http://gnrd.globalnames.org/

See Also

`gnr_datasources` `tnrs`

Examples

```
## Not run:
gnr_resolve(names = c("Helianthus annuus", "Homo sapiens"))
gnr_resolve(names = c("Asteraceae", "Plantae"))

# Using data source 12 (Encyclopedia of Life)
sources <- gnr_datasources()
sources
eol <- sources$id[sources$title == 'EOL']
gnr_resolve(names=c("Helianthos annuus","Homo sapians"), data_source_ids=eol)

# Two species in the NE Brazil catalogue
sps <- c('Justicia brasiliana','Schinopsis brasiliensis')
gnr_resolve(names = sps, data_source_ids = 145)

# Best match only, compare the two
gnr_resolve(names = "Helianthus annuus", best_match_only = FALSE)
gnr_resolve(names = "Helianthus annuus", best_match_only = TRUE)

# Preferred data source
gnr_resolve(names = "Helianthus annuus", preferred_data_sources = c(3,4))

# Return canonical names - default is canonical=FALSE
head(gnr_resolve(names = "Helianthus annuus"))
head(gnr_resolve(names = "Helianthus annuus", canonical=TRUE))

# Return canonical names with authority stripped but
# ranks still present
gnr_resolve("Scorzonera hispanica L. subsp. asphodeloides Wallr.")
```

```
## vs.
gnr_resolve("Scorzonera hispanica L. subsp. asphodeloides Wallr.",
   with_canonical_ranks = TRUE)

## End(Not run)
```

`id2name` *Taxonomic IDs to taxonomic names*

Description

Taxonomic IDs to taxonomic names

Usage

```
id2name(x, db = NULL, ...)

## Default S3 method:
id2name(x, db = NULL, ...)

## S3 method for class 'tolid'
id2name(x, ...)

## S3 method for class 'tsn'
id2name(x, ...)

## S3 method for class 'uid'
id2name(x, ...)

## S3 method for class 'wormsid'
id2name(x, ...)

## S3 method for class 'gbifid'
id2name(x, ...)

## S3 method for class 'colid'
id2name(x, ...)

## S3 method for class 'boldid'
id2name(x, ...)
```

Arguments

x	vector of taxonomic IDs (character or numeric)
db	(character) database to query. One or more of `tol`, `itis`, `ncbi`, `worms`, `gbif`, `col`, or `bold`. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong `db` value for the identifier you could get a result,

but it will likely be wrong (not what you were expecting). If using ncbi we recommend getting API keys; see taxize-authentication

... Further args passed on to tol_id2name or itis_getrecord, or other internal functions. See those functions for what parameters can be passed on.

Value

A named list of data.frames, named by the input taxonomic ids

Examples

```
## Not run:
# ITIS
id2name(19322, db = "itis")

# TOL
id2name(515698, db = "tol")
# get NCBI ID and pass to classification()
x <- id2name(515698, db = "tol")
classification(as.uid(x[[1]]$tax_sources_ncbi))

# NCBI
id2name(315567, db = "ncbi")
id2name(3339, db = "ncbi")
id2name(9696, db = "ncbi")
id2name(c(9695, 9696), db = "ncbi")

# WORMS
id2name(105706, db = "worms")

# GBIF
id2name(2441176, db = "gbif")

# COL
id2name("36c623ad9e3da39c2e978fa3576ad415", db = "col")

# BOLD
id2name(88899, db = "bold")

## End(Not run)
```

---

ion *ION - Index to Organism Names*

Description

ION - Index to Organism Names

Usage

```
ion(x, ...)
```

Arguments

x	An LSID number. Required.
...	Curl options passed on to verb-GET

Value

A data.frame

References

http://www.organismnames.com

Examples

```
## Not run:
ion(155166)
ion(298678)
ion(4796748) # ursus americanus
ion(1280626) # puma concolor

## End(Not run)
```

---

iplant_resolve *iPlant name resolution*

Description

iPlant name resolution

Usage

```
iplant_resolve(query, retrieve = "all", ...)
```

Arguments

query	Vector of one or more taxonomic names (no common names)
retrieve	Specifies whether to retrieve all matches for the names submitted. One of 'best' (retrieves only the single best match for each name submitted) or 'all' (retrieves all matches)
...	Curl options passed on to verb-GET

Value

A data.frame

Examples

```
## Not run:
iplant_resolve(query=c("Helianthus annuus", "Homo sapiens"))
iplant_resolve("Helianthusss")
iplant_resolve("Pooa")
iplant_resolve("Helianthusss", verbose = TRUE)

## End(Not run)
```

---

ipni_search *Search for names in the International Plant Names Index (IPNI).*

Description

Note: This data source is also provided in the Global Names Index (GNI) (http://gni.globalnames.org/data_sources). The interface to the data is different among the two services though.

Usage

```
ipni_search(family = NULL, infrafamily = NULL, genus = NULL,
  infragenus = NULL, species = NULL, infraspecies = NULL,
  publicationtitle = NULL, authorabbrev = NULL,
  includepublicationauthors = NULL, includebasionymauthors = NULL,
  geounit = NULL, addedsince = NULL, modifiedsince = NULL,
  isapnirecord = NULL, isgcirecord = NULL, isikrecord = NULL,
  ranktoreturn = NULL, output = "minimal", ...)
```

Arguments

family	Family name to search on (Optional)
infrafamily	Infrafamilial name to search on (Optional)
genus	Genus name to search on (Optional)
infragenus	Infrageneric name to search on (Optional)
species	Species name to search on (Optional) - Note, this is the epithet, not the full genus - epithet name combination.
infraspecies	Infraspecies name to search on (Optional)
publicationtitle	Publication name or abbreviation to search on. Again, replace any spaces with a '+' (e.g. 'J.+Bot.') (Optional)
authorabbrev	Author standard form to search on (publishing author, basionym author or both - see below) (Optional)
includepublicationauthors	TRUE (default) to include the taxon author in the search or FALSE to exclude it

includebasionymauthors
: TRUE (default) to include the basionum author in the search or FALSE to exclude it

geounit
: Country name or other geographical unit to search on (see the help pages for more information and warnings about the use of this option) (Optional)

addedsince
: Date to search on in the format 'yyyy-mm-dd', e.g. 2005-08-01 for all records added since the first of August, 2005. (see the help pages for more information and warnings about the use of this option) (Optional. If supplied must be in format YYYY-MM-DD and must be greater than or equal to 1984-01-01.)

modifiedsince
: Date to search on in the format 'yyyy-mm-dd', e.g. 2005-08-01 for all records edited since the first of August, 2005. (See the help pages for more information about the use of this option) (Optional. If supplied must be in format YYYY-MM-DD and must be greater than or equal to 1993-01-01.)

isapnirecord
: FALSE (default) to exclude records from the Australian Plant Name Index

isgcirecord
: FALSE (default) to exclude records from the Gray Cards Index

isikrecord
: FALSE (default) to exclude records from the Index Kewensis

ranktoreturn
: One of a few options to choose the ranks returned. See details.

output
: One of minimal (default), classic, short, or extended

...
: Curl options passed on to verb-GET (Optional). Default: returns all ranks.

Details

rankToReturn options:

-
- "all" - all records
- "fam" - family records
- "infrafam" - infrafamilial records
- "gen" - generic records
- "infragen" - infrageneric records
- "spec" - species records
- "infraspec" - infraspecific records

Value

A data frame

References

http://www.ipni.org/link_to_ipni.html

Examples

```
## Not run:
ipni_search(genus='Brintonia', isapnirecord=TRUE, isgcirecord=TRUE,
  isikrecord=TRUE)
head(ipni_search(genus='Ceanothus'))
head(ipni_search(genus='Pinus', species='contorta'))

# Different output formats
head(ipni_search(genus='Ceanothus'))
head(ipni_search(genus='Ceanothus', output='short'))
head(ipni_search(genus='Ceanothus', output='extended'))

## End(Not run)
```

---

itis_acceptname *Retrieve accepted TSN and name*

Description

Retrieve accepted TSN and name

Usage

```
itis_acceptname(searchtsn, ...)
```

Arguments

searchtsn	One or more TSN for a taxon (numeric/integer)
...	Curl options passed on to verb-GET

Value

data.frame with with row number equal to input vector length, and with three columns:

- submittedtsn (numeric) - The submitted TSN
- acceptedname (character) - The accepted name - if the submitted TSN is the accepted TSN, then this is NA_character_ because ITIS does not return a name along with the TSN if it's an accepted name. We could make an extra HTTP request to ITIS, but that means additional time.
- acceptedtsn (numeric) - The accepted TSN
- author (character) - taxonomic authority

Examples

```
## Not run:
# TSN accepted - good name
itis_acceptname(searchtsn = 208527)

# TSN not accepted - input TSN is old
itis_acceptname(searchtsn = 504239)

# many accepted names
ids <- c(18161, 18162, 18163, 18164, 18165, 18166, 46173, 46174,
46178, 46181, 46186, 46193, 46196, 46197, 46200, 46201, 46204,
46207, 46867, 46868)
itis_acceptname(searchtsn = ids)

# many unaccepted names
ids <- c(39087, 46208, 46973, 46976, 46978, 46980, 47295, 47445,
47448, 47512, 47515, 47527, 47546, 47622, 47783, 47786, 47787,
47788, 47835, 47839)
itis_acceptname(searchtsn = ids)

# many: mix of accepted and unaccepted names
ids <- c(18161, 18162, 47527, 47546, 47622, 46200)
itis_acceptname(searchtsn = ids)

## End(Not run)
```

`itis_downstream` *Retrieve all taxa names or TSNs downstream in hierarchy from given TSN.*

Description

Retrieve all taxa names or TSNs downstream in hierarchy from given TSN.

Usage

```
itis_downstream(tsns, downto, intermediate = FALSE, ...)
```

Arguments

`tsns`	A taxonomic serial number.
`downto`	The taxonomic level you want to go down to. See examples below. The taxonomic level IS case sensitive, and you do have to spell it correctly. See `data(rank_ref)` for spelling.
`intermediate`	(logical) If TRUE, return a list of length two with target taxon rank names, with additional list of data.frame's of intermediate taxonomic groups. Default: FALSE
`...`	Further args passed on to `rank_name` and `hierarchy_down`

Value

Data.frame of taxonomic information downstream to family from e.g., Order, Class, etc., or if `intermediated=TRUE`, list of length two, with target taxon rank names, and intermediate names.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
## the plant class Bangiophyceae, tsn 846509
itis_downstream(tsns = 846509, downto="genus")
itis_downstream(tsns = 846509, downto="genus", intermediate=TRUE)

# get families downstream from Acridoidea
itis_downstream(tsns = 650497, "family")
## here, intermediate leads to the same result as the target
itis_downstream(tsns = 650497, "family", intermediate=TRUE)

# get species downstream from Ursus
itis_downstream(tsns = 180541, "species")

# get orders down from the Division Rhodophyta (red algae)
itis_downstream(tsns = 660046, "order")
itis_downstream(tsns = 660046, "order", intermediate=TRUE)

# get tribes down from the family Apidae
itis_downstream(tsns = 154394, downto="tribe")
itis_downstream(tsns = 154394, downto="tribe", intermediate=TRUE)

## End(Not run)
```

---

itis_getrecord *Get full ITIS record for one or more ITIS TSN's or lsid's.*

Description

Get full ITIS record for one or more ITIS TSN's or lsid's.

Usage

```
itis_getrecord(values, by = "tsn", ...)
```

Arguments

values	(character) One or more TSN's (taxonomic serial number) or lsid's for a taxonomic group
by	(character) By "tsn" (default) or "lsid"
...	Further arguments passed on to `full_record`

Details

You can only enter values in tsn parameter or lsid, not both.

Examples

```
## Not run:
# by TSN
itis_getrecord(202385)
itis_getrecord(c(202385,70340))

# by lsid
itis_getrecord("urn:lsid:itis.gov:itis_tsn:202385", "lsid")

## End(Not run)
```

---

itis_hierarchy *ITIS hierarchy*

Description

Get hierarchies from TSN values, full, upstream only, or immediate downstream only

Usage

```
itis_hierarchy(tsn, what = "full", ...)
```

Arguments

tsn	One or more TSN's (taxonomic serial number). Required.
what	One of full (full hierarchy), up (immediate upstream), or down (immediate downstream)
...	Further arguments passed on to hierarchy_full, hierarchy_up, or hierarchy_down

Details

Note that itis_downstream gets taxa downstream to a particular rank, while this function only gets immediate names downstream.

See Also

itis_downstream

Examples

```
## Not run:
# Get full hierarchy
itis_hierarchy(tsn=180543)

# Get hierarchy upstream
itis_hierarchy(tsn=180543, "up")

# Get hierarchy downstream
itis_hierarchy(tsn=180543, "down")

# Many tsn's
itis_hierarchy(tsn=c(180543,41074,36616))

## End(Not run)
```

---

itis_kingdomnames *Get kingdom names*

Description

Get kingdom names

Usage

```
itis_kingdomnames(tsn = NULL, ...)
```

Arguments

tsn	One or more TSN's (taxonomic serial number)
...	Further arguments passed on to getkingdomnamefromtsn

Examples

```
## Not run:
itis_kingdomnames(202385)
itis_kingdomnames(tsn=c(202385,183833,180543))

## End(Not run)
```

itis_lsid *Get TSN from LSID*

Description

Get TSN from LSID

Usage

```
itis_lsid(lsid = NULL, what = "tsn", ...)
```

Arguments

lsid	One or more lsid's
what	What to retrieve. One of tsn, record, or fullrecord
...	Further arguments passed on to lsid2tsn, record, or full_record

Examples

```
## Not run:
# Get TSN
itis_lsid("urn:lsid:itis.gov:itis_tsn:180543")
itis_lsid(lsid=c("urn:lsid:itis.gov:itis_tsn:180543","urn:lsid:itis.gov:itis_tsn:28726"))

# Get partial record
itis_lsid("urn:lsid:itis.gov:itis_tsn:180543", "record")

# Get full record
itis_lsid("urn:lsid:itis.gov:itis_tsn:180543", "fullrecord")

# An invalid lsid (a tsn actually)
itis_lsid(202385)

## End(Not run)
```

itis_name *Get taxonomic names for a given taxonomic name query.*

Description

Get taxonomic names for a given taxonomic name query.

Usage

```
itis_name(query = NULL, get = NULL)
```

Arguments

query	TSN number (taxonomic serial number).
get	The rank of the taxonomic name to get.

Value

Taxonomic name for the searched taxon.

Examples

```
## Not run:
itis_name(query="Helianthus annuus", get="family")

## End(Not run)
```

---

itis_native *Get jurisdiction data, i.e., native or not native in a region.*

Description

Get jurisdiction data, i.e., native or not native in a region.

Usage

```
itis_native(tsn = NULL, what = "bytsn", ...)
```

Arguments

tsn	One or more TSN's (taxonomic serial number)
what	One of bytsn, values, or originvalues
...	Further arguments passed on to jurisdictional_origin, jurisdiction_values, or jurisdiction_origin_values

Examples

```
## Not run:
# Get values
itis_native(what="values")

# Get origin values
itis_native(what="originvalues")

# Get values by tsn
itis_native(tsn=180543)
itis_native(tsn=c(180543,41074,36616))

## End(Not run)
```

itis_refs *Get references related to a ITIS TSN.*

Description

Get references related to a ITIS TSN.

Usage

```
itis_refs(tsn, ...)
```

Arguments

tsn	One or more TSN's (taxonomic serial number) for a taxonomic group (numeric)
...	Further arguments passed on to getpublicationsfromtsn

Examples

```
## Not run:
itis_refs(202385)
itis_refs(c(202385, 70340))

## End(Not run)
```

itis_taxrank *Retrieve taxonomic rank name from given TSN.*

Description

Retrieve taxonomic rank name from given TSN.

Usage

```
itis_taxrank(query = NULL, ...)
```

Arguments

query	TSN for a taxonomic group (numeric). If query is left as default (NULL), you get all possible rank names, and their TSN's (using function rank_names. There is slightly different terminology for Monera vs. Plantae vs. Fungi vs. Animalia vs. Chromista, so there are separate terminologies for each group.
...	Further arguments passed on to rank_name

Details

You can print messages by setting verbose=FALSE.

Value

Taxonomic rank names or data.frame of all ranks.

Examples

```
## Not run:
# All ranks
itis_taxrank()

# A single TSN
itis_taxrank(query=202385)

# Many TSN's
itis_taxrank(query=c(202385,183833,180543))

## End(Not run)
```

`itis_terms` *Get ITIS terms, i.e., tsn's, authors, common names, and scientific names.*

Description

Get ITIS terms, i.e., tsn's, authors, common names, and scientific names.

Usage

```
itis_terms(query, what = "both", ...)
```

Arguments

`query`	One or more common or scientific names, or partial names
`what`	One of both (search common and scientific names), common (search just common names), or scientific (search just scientific names)
`...`	Further arguments passed on to `terms`

Examples

```
## Not run:
# Get terms searching both common and scientific names
itis_terms(query='bear')

# Get terms searching just common names
itis_terms(query='tarweed', "common")

# Get terms searching just scientific names
itis_terms(query='Poa annua', "scientific")

## End(Not run)
```

iucn_getname *Get any matching IUCN species names*

Description

Get any matching IUCN species names

Usage

```
iucn_getname(name, verbose = TRUE, ...)
```

Arguments

name	character; taxon name
verbose	logical; should messages be printed?
...	Further arguments passed on to iucn_summary, note that you'll need an API key.

Details

Beware: IUCN functions can give back incorrect data. This isn't our fault. We do our best to get you the correct data quickly, but sometimes IUCN gives back the wrong data, and sometimes Global Names gives back the wrong data. We will fix these as soon as possible. In the meantime, just make sure that the data you get back is correct.

Value

Character vector of names that matched in IUCN

See Also

iucn_summary iucn_status

Examples

```
## Not run:
iucn_getname(name = "Cyanistes caeruleus")
iucn_getname(name = "Panthera uncia")

# not found in global names
# iucn_getname(name = "Abronia pinsapo")

# not found in IUCN search
iucn_getname(name = "Acacia allenii")

## End(Not run)
```

iucn_id *Get an ID for a IUCN listed taxon*

Description

Get an ID for a IUCN listed taxon

Usage

```
iucn_id(sciname, key = NULL, ...)
```

Arguments

sciname	character; Scientific name. Should be cleand and in the format <*Genus*> <*Species*>. One or more.
key	(character) required. you IUCN Redlist API key. See rredlist-package for help on authenticating with IUCN Redlist
...	Curl options passed on to `crul::HttpClient`

Value

A named list (names are input taxa names) of one or more IUCN IDs. Taxa that aren't found are silently dropped.

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

Examples

```
## Not run:
iucn_id("Branta canadensis")
iucn_id("Branta bernicla")
iucn_id("Panthera uncia")
iucn_id("Lynx lynx")

# many names
iucn_id(c("Panthera uncia", "Lynx lynx"))

# many names, some not found
iucn_id(c("Panthera uncia", "Lynx lynx", "foo bar", "hello world"))

# a name not found
iucn_id("Foo bar")

## End(Not run)
```

iucn_status *Extractor functions for* iucn*-class.*

Description

Extractor functions for iucn-class.

Usage

```
iucn_status(x, ...)
```

Arguments

x	an iucn-object as returned by iucn_summary
...	Currently not used

Value

A character vector with the status.

See Also

iucn_summary

Examples

```
## Not run:
ia <- iucn_summary(c("Panthera uncia", "Lynx lynx"))
iucn_status(ia)
## End(Not run)
```

iucn_summary *Get a summary from the IUCN Red List*

Description

Get a summary from the IUCN Red List (http://www.iucnredlist.org/).

Usage

```
iucn_summary(x, parallel = FALSE, distr_detail = FALSE, key = NULL,
  ...)
```

Arguments

x	character; Scientific name. Should be cleaned and in the format <*Genus*> <*Species*>.
parallel	logical; Search in parallel to speed up search. You have to register a parallel backend if TRUE. See e.g., doMC, doSNOW, etc.
distr_detail	logical; If TRUE, the geographic distribution is returned as a list of vectors corresponding to the different range types: native, introduced, etc.
key	a Redlist API key, get one from http://apiv3.iucnredlist.org/api/v3/token. Required for iucn_summary. Defaults to NULL in case you have your key stored (see Redlist Authentication below).
...	curl options passed on to [crul::verb-GET]

Details

Beware: IUCN functions can give back incorrect data. This isn't our fault. We do our best to get you the correct data quickly, but sometimes IUCN gives back the wrong data, and sometimes Global Names gives back the wrong data. We will fix these as soon as possible. In the meantime, just make sure that the data you get back is correct.

iucn_summary has a default method that errors when anything's passed in that's not character or iucn class - a iucn_summary.character method for when you pass in taxon names - and a iucn_summary.iucn method so you can pass in iucn class objects as output from get_iucn or as.iucn. If you already have IUCN IDs, coerce them to iucn class via as.iucn(..., check = FALSE)

Value

A list (for every species one entry) of lists with the following items:

status	Red List Category.
history	History of status, if available.
distr	Geographic distribution, if available.
trend	Trend of population size, if available.

Redlist Authentication

iucn_summary uses the new Redlist API for searching for a IUCN ID, so we use the rl_search function internally. This function requires an API key. Get the key at http://apiv3.iucnredlist.org/api/v3/token, and pass it to the key parameter, or store in your .Renviron file like IUCN_REDLIST_KEY=yourkey or in your .Rprofile file like options(iucn_redlist_key="yourkey". We strongly encourage you to not pass the key in the function call but rather store it in one of those two files. This key will also set you up to use the **rredlist** package.

Note

Not all entries (history, distr, trend) are available for every species and NA is returned. iucn_status is an extractor function to easily extract status into a vector.

Author(s)

Eduard Szoecs, <eduardszoecs@gmail.com>

Philippe Marchand, <marchand.philippe@gmail.com>

Scott Chamberlain, <myrmecocystus@gmail.com>

See Also

iucn_status

Examples

```
## Not run:
# if you send a taxon name, an IUCN API key is required
## here, the key is being detected from a .Rprofile file
## or .Renviron file, See "Redlist Authentication" above
iucn_summary("Lutra lutra")

ia <- iucn_summary(c("Panthera uncia", "Lynx lynx"))
ia <- iucn_summary(c("Panthera uncia", "Lynx lynx", "aaa"))

## get detailed distribution
iac <- iucn_summary(x="Ara chloropterus", distr_detail = TRUE)
iac[[1]]$distr


# If you pass in an IUCN ID, you don't need to pass in a Redlist API Key
# extract status
iucn_status(iac)

# using parallel, e.g., with doMC package, register cores first
# library(doMC)
# registerDoMC(cores = 2)
# nms <- c("Panthera uncia", "Lynx lynx", "Ara chloropterus", "Lutra lutra")
# (res <- iucn_summary(nms, parallel = TRUE))

## End(Not run)
```

---

key_helpers *Helpers to set up authentication for the different providers.*

Description

Sets up authentication to diverse providers by providing the user a detailed prompt.

Usage

```
use_tropicos()

use_eol()

use_entrez()

use_iucn()
```

Details

Key helpers

'use_tropicos()'

Browses to Tropicos API key request URL and provides instruction on how to store the key. After filling the form you will get the key soon, but not immediately.

'use_eol()'

Browse EOL to help make an API key request and provides instruction on how to store the key. There's no direct URL to request a key, one first needs to log in or register and then to generate a key from one's Preferences page.

'use_entrez()'

Browse NCBI Entrez to help make an API key request and provides instruction on how to store the key. There's no direct URL to request a key, one first needs to log in or register and then to generate a key from one's account.

Note that NCBI Entrez doesn't require that you use an API key, but you should get higher rate limit with a key, so do get one.

'use_iucn()'

Browse IUCN Red List API key request URL and provides instruction on how to store the key. This function wraps `rl_use_iucn` from the `rredlist` package. After filling the form you will get the key soon, but not immediately.

See Also

`taxize-authentication`

lowest_common *Retrieve the lowest common taxon and rank for a given taxon name or ID*

Description

Retrieve the lowest common taxon and rank for a given taxon name or ID

Usage

```
lowest_common(...)

## Default S3 method:
lowest_common(x, db = NULL, rows = NA,
  class_list = NULL, low_rank = NULL, ...)

## S3 method for class 'uid'
lowest_common(x, class_list = NULL, low_rank = NULL, ...)

## S3 method for class 'tsn'
lowest_common(x, class_list = NULL, low_rank = NULL, ...)

## S3 method for class 'gbifid'
lowest_common(x, class_list = NULL, low_rank = NULL,
  ...)

## S3 method for class 'colid'
lowest_common(x, class_list = NULL, low_rank = NULL,
  ...)

## S3 method for class 'tolid'
lowest_common(x, class_list = NULL, low_rank = NULL,
  ...)
```

Arguments

...	Other arguments passed to get_tsn, get_uid, get_colid, get_gbifid, get_tolid
x	Vector of taxa names (character) or id (character or numeric) to query.
db	character; database to query. either ncbi, itis, gbif, col, or tol. If using ncbi, we recommend getting an API key; see taxize-authentication
rows	(numeric) Any number from 1 to infinity. If the default NA, all rows are considered. Note that this parameter is ignored if you pass in a taxonomic id of any of the acceptable classes: tsn, colid, gbifid, tolid. NCBI has a method for this function but rows doesn't work.
class_list	(list) A list of classifications, as returned from classification
low_rank	(character) taxonomic rank to return, of length 1

Value

NA when no match, or a data.frame with columns

- name
- rank
- id

Authentication

See taxize-authentication for help on authentication

Author(s)

Jimmy O'Donnell <jodonnellbio@gmail.com> Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
id <- c("9031", "9823", "9606", "9470")
id_class <- classification(id, db = 'ncbi')
lowest_common(id[2:4], db = "ncbi")
lowest_common(id[2:4], db = "ncbi", low_rank = 'class')
lowest_common(id[2:4], db = "ncbi", low_rank = 'family')
lowest_common(id[2:4], class_list = id_class)
lowest_common(id[2:4], class_list = id_class, low_rank = 'class')
lowest_common(id[2:4], class_list = id_class, low_rank = 'family')

# COL
taxa <- c('Nycticebus coucang', 'Homo sapiens', 'Sus scrofa')
cls <- classification(taxa, db = "col")
lowest_common(taxa, class_list = cls, db = "col")
lowest_common(get_colid(taxa), class_list = cls)
xx <- get_colid(taxa)
lowest_common(xx, class_list = cls)

# TOL
taxa <- c("Angraecum sesquipedale", "Dracula vampira",
  "Masdevallia coccinea")
(cls <- classification(taxa, db = "tol"))
lowest_common(taxa, db = "tol", class_list = cls)
lowest_common(get_tolid(taxa), class_list = cls)
xx <- get_tolid(taxa)
lowest_common(xx, class_list = cls)

spp <- c("Sus scrofa", "Homo sapiens", "Nycticebus coucang")
lowest_common(spp, db = "ncbi")
lowest_common(get_uid(spp))

lowest_common(spp, db = "itis")
lowest_common(get_tsn(spp))
```

```
gbifid <- c("2704179", "3119195")
lowest_common(gbifid, db = "gbif")

spp <- c("Poa annua", "Helianthus annuus")
lowest_common(spp, db = "gbif")
lowest_common(get_gbifid(spp))

cool_orchid <- c("Angraecum sesquipedale", "Dracula vampira",
  "Masdevallia coccinea")
orchid_ncbi <- get_uid(cool_orchid)
orchid_gbif <- get_gbifid(cool_orchid)

cool_orchids2 <- c("Domingoa haematochila", "Gymnadenia conopsea",
  "Masdevallia coccinea")
orchid_itis <- get_tsn(cool_orchids2)

orchid_hier_ncbi <- classification(orchid_ncbi, db = 'ncbi')
orchid_hier_gbif <- classification(orchid_gbif, db = 'gbif')
orchid_hier_itis <- classification(orchid_itis, db = 'itis')

lowest_common(orchid_ncbi, low_rank = 'class')
lowest_common(orchid_ncbi, class_list = orchid_hier_ncbi,
  low_rank = 'class')
lowest_common(orchid_gbif, low_rank = 'class')
lowest_common(orchid_gbif, orchid_hier_gbif, low_rank = 'class')
lowest_common(get_uid(cool_orchid), low_rank = 'class')
lowest_common(get_uid(cool_orchid), low_rank = 'family')

lowest_common(orchid_ncbi, class_list = orchid_hier_ncbi,
  low_rank = 'subfamily')
lowest_common(orchid_gbif, class_list = orchid_hier_gbif,
  low_rank = 'subfamily')

lowest_common(orchid_itis, class_list = orchid_hier_itis,
  low_rank = 'class')

## Pass in sci. names
nms <- c("Angraecum sesquipedale", "Dracula vampira", "Masdevallia coccinea")
lowest_common(x = nms, db = "ncbi")
lowest_common(x = nms, db = "gbif")
# lowest_common(x = nms, db = "itis")

## NAs due to taxon not found, stops with error message
# lowest_common(orchid_itis, db = "itis")
# lowest_common(get_tsn(cool_orchid))

## End(Not run)
```

---

names_list *Get a random vector of species names.*

Description

Family and order names come from the APG plant names list. Genus and species names come from Theplantlist.org.

Usage

```
names_list(rank = "genus", size = 10)
```

Arguments

rank	Taxonomic rank, one of species, genus (default), family, order.
size	Number of names to get. Maximum depends on the rank.

Value

Vector of taxonomic names.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
names_list()
names_list('species')
names_list('genus')
names_list('family')
names_list('order')
names_list('order', '2')
names_list('order', '15')

# You can get a lot of genus or species names if you want
nrow(theplantlist)
names_list('genus', 500)
```

---

nbn_classification *Search UK National Biodiversity Network database for taxonomic classification*

Description

Search UK National Biodiversity Network database for taxonomic classification

Usage

```
nbn_classification(id, ...)
```

Arguments

id	(character) An NBN identifier.
...	Further args passed on to verb-GET

Value

A data.frame

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

References

<https://api.nbnatlas.org/>

See Also

Other nbn: get_nbnid, nbn_search, nbn_synonyms

Examples

```
## Not run:
nbn_classification(id="NHMSYS0000376773")

# get id first, then pass to this fxn
id <- get_nbnid("Zootoca vivipara", rec_only = TRUE, rank = "Species")
nbn_classification(id)

nbn_classification(id="NHMSYS0000502940", verbose = TRUE)

## End(Not run)
```

---

nbn_search *Search UK National Biodiversity Network*

Description

Search UK National Biodiversity Network

Usage

```
nbn_search(q, fq = NULL, order = NULL, sort = NULL, start = 0,
  rows = 25, facets = NULL, ...)
```

Arguments

q	(character) The query terms(s)
fq	(character) Filters to be applied to the original query. These are additional params of the form fq=INDEXEDFIELD:VALUE e.g. fq=rank:kingdom. See <https://species-ws.nbnatlas.org/indexFields> for all the fields that are queryable.
order	(character) Supports "asc" or "desc"
sort	(character) The indexed field to sort by
start	(integer) Record offset, to enable paging
rows	(integer) Number of records to return
facets	(list) Comma separated list of the fields to create facets on e.g. facets=basis_of_record.
...	Further args passed on to HttpClient.

Value

a list with slots for metadata (‘meta‘) with list of response attributes, and data (‘data“) with a data.frame of results

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

References

<https://api.nbnatlas.org/>

See Also

Other nbn: get_nbnid, nbn_classification, nbn_synonyms

Examples

```
## Not run:
x <- nbn_search(q = "Vulpes")
x$meta$totalRecords
x$meta$pageSize
x$meta$urlParameters
x$meta$queryTitle
head(x$data)

nbn_search(q = "blackbird", start = 4)

# debug curl stuff
nbn_search(q = "blackbird", verbose = TRUE)

## End(Not run)
```

nbn_synonyms *Return all synonyms for a taxon name with a given id from NBN*

Description

Return all synonyms for a taxon name with a given id from NBN

Usage

```
nbn_synonyms(id, ...)
```

Arguments

id	the taxon identifier code
...	Further args passed on to verb-GET

Value

A data.frame

References

<https://api.nbnatlas.org/>

See Also

Other nbn: get_nbnid, nbn_classification, nbn_search

Examples

```
## Not run:
nbn_synonyms(id = 'NHMSYS0001501147')
nbn_synonyms(id = 'NHMSYS0000456036')

# none
nbn_synonyms(id = 'NHMSYS0000502940')

## End(Not run)
```

ncbi_children *Search NCBI for children of a taxon*

Description

Search the NCBI Taxonomy database for uids of children of taxa. Taxa can be referenced by name or uid. Referencing by name is faster

In a few cases, different taxa have the same name (e.g. Satyrium; see examples). If one of these are searched for then the children of both taxa will be returned. This can be avoided by using a uid instead of the name or specifying an ancestor. If an ancestor is provided, only children of both the taxon and its ancestor are returned. This will only fail if there are two taxa with the same name and the same specified ancestor.

Usage

```
ncbi_children(name = NULL, id = NULL, start = 0, max_return = 1000,
  ancestor = NULL, out_type = c("summary", "uid"), ambiguous = FALSE,
  key = NULL, ...)
```

Arguments

name	(character) The string to search for. Only exact matches found the name given will be returned. Not compatible with id.
id	(character) The uid to search for. Not compatible with name.
start	The first record to return. If omitted, the results are returned from the first record (start=0).
max_return	(numeric; length=1) The maximum number of children to return.
ancestor	(character) The ancestor of the taxon being searched for. This is useful if there could be more than one taxon with the same name. Has no effect if id is used.
out_type	(character) Currently either "summary" or "uid": **summary** The output is a list of data.frame with children uid, name, and rank. **uid** A list of character vectors of children uids
ambiguous	logical; length 1 If FALSE, children taxa with words like "unclassified", "unknown", "uncultured", or "sp." are removed from the output. NOTE: This option only applies when out_type = "summary".
key	(character) NCBI Entrez API key. optional. See Details.
...	Curl options passed on to HttpClient

Value

The output type depends on the value of the out_type parameter. Taxa that cannot be found will result in NAs and a lack of children results in an empty data structure.

Authentication

See taxize-authentication for help on authentication. We strongly recommend getting an API key

Author(s)

Zachary Foster <zacharyfoster1989@gmail.com>

See Also

ncbi_get_taxon_summary, children

Examples

```
## Not run:
ncbi_children(name="Satyrium") #Satyrium is the name of two different genera
ncbi_children(name="Satyrium", ancestor="Eumaeini") # A genus of butterflies
ncbi_children(name="Satyrium", ancestor="Orchidaceae") # A genus of orchids
ncbi_children(id="266948") #"266948" is the uid for the butterfly genus
ncbi_children(id="62858") #"62858" is the uid for the orchid genus

# use curl options
ncbi_children(name="Satyrium", ancestor="Eumaeini", verbose = TRUE)

## End(Not run)
```

---

ncbi_downstream *Retrieve all taxa names downstream in hierarchy for NCBI*

Description

Retrieve all taxa names downstream in hierarchy for NCBI

Usage

```
ncbi_downstream(id, downto, intermediate = FALSE, ...)
```

Arguments

id	(numeric/integer) An NCBI taxonomic identifier
downto	The taxonomic level you want to go down to. See examples below. The taxonomic level IS case sensitive, and you do have to spell it correctly. See data(rank_ref) for spelling.
intermediate	(logical) If TRUE, return a list of length two with target taxon rank names, with additional list of data.frame's of intermediate taxonomic groups. Default: FALSE
...	Further args passed on to ncbi_children

Value

Data.frame of taxonomic information downstream to family from e.g., Order, Class, etc., or if `intermediate=TRUE`, list of length two, with target taxon rank names, and intermediate names.

No Rank

A sticky point with NCBI is that they can have designation for taxonomic rank of "No Rank". So we have no way of programatically knowing what to do with that taxon. Of course one can manually look at a name and perhaps know what it is, or look it up on the web - but we can't do anything programatically. So, no rank things will sometimes be missing.

Authentication

See `taxize-authentication` for help on authentication. We strongly recommend getting an API key

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
## genus Apis
ncbi_downstream(id = 7459, downto="species")

## get intermediate taxa as a separate object
ncbi_downstream(id = 7459, downto="species", intermediate = TRUE)

## get intermediate taxa as a separate object
ncbi_downstream(id = 7459, downto="species", intermediate = TRUE)

## Lepidoptera
ncbi_downstream(id = 7088, downto="superfamily")

## families in the ferns (Moniliformopses)
(id <- get_uid("Moniliformopses"))
ncbi_downstream(id = id, downto = "order")

## End(Not run)
```

`ncbi_get_taxon_summary`

NCBI taxon information from uids

Description

Downloads summary taxon information from the NCBI taxonomy databases for a set of taxonomy UIDs using eutils esummary.

Usage

```
ncbi_get_taxon_summary(id, key = NULL, ...)
```

Arguments

id	(character) NCBI taxonomy uids to retrieve information for. See Details.
key	(character) NCBI Entrez API key. optional. See Details.
...	Curl options passed on to verb-GET

Details

If your input vector or list of NCBI IDs is longer than about 2500 characters (use `nchar(paste(ids, collapse = "+"))`), split the list up into chunks since at about that number of characters you will run into the HTTP 414 error "Request-URI Too Long".

Value

A `data.frame` with the following columns:

uid The uid queried for

name The name of the taxon; a binomial name if the taxon is of rank species

rank The taxonomic rank (e.g. 'Genus')

Authentication

See taxize-authentication for help on authentication. We strongly recommend getting an API key

Author(s)

Zachary Foster <zacharyfoster1989@Sgmail.com>

Examples

```
## Not run:
ncbi_get_taxon_summary(c(1430660, 4751))

# use curl options
ncbi_get_taxon_summary(c(1430660, 4751), verbose = TRUE)

## End(Not run)
```

phylomatic_format *Get family names to make Phylomatic input object, and output input string to Phylomatic for use in the function phylomatic_tree.*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
phylomatic_format(...)
```

Arguments

...	Parameters, ignored

phylomatic_tree *Query Phylomatic for a phylogenetic tree.*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
phylomatic_tree(...)
```

Arguments

...	Parameters, ignored

ping *Ping an API used in taxize to see if it's working.*

Description

Ping an API used in taxize to see if it's working.

Usage

```
col_ping(what = "status", ...)

eol_ping(what = "status", ...)

itis_ping(what = "status", ...)

ncbi_ping(what = "status", key = NULL, ...)

tropicos_ping(what = "status", ...)

nbn_ping(what = "status", ...)

gbif_ping(what = "status", ...)

bold_ping(what = "status", ...)

ipni_ping(what = "status", ...)

vascan_ping(what = "status", ...)

fg_ping(what = "status", ...)
```

Arguments

what	(character) One of status (default), content, or an HTTP status code. If status, we just check that the HTTP status code is 200, or similar signifying the service is up. If content, we do a simple, quick check to determine if returned content matches what's expected. If an HTTP status code, it must match an appropriate code. See status_codes.
...	Curl options passed on to verb-GET
key	(character) NCBI Entrez API key. optional. See get_uid

Details

For ITIS, see description, which provides number of scientific and common names in a character string.

Value

A logical, TRUE or FALSE

Examples

```
## Not run:
col_ping()
col_ping("content")
col_ping(200)
col_ping("200")
```

```
col_ping(204)

itis_ping()
eol_ping()
ncbi_ping()
tropicos_ping()
nbn_ping()

gbif_ping()
gbif_ping(200)

bold_ping()
bold_ping(200)
bold_ping("content")

ipni_ping()
ipni_ping(200)
ipni_ping("content")

vascan_ping()
vascan_ping(200)
vascan_ping("content")

# curl options
vascan_ping(verbose = TRUE)
eol_ping(500, verbose = TRUE)

## End(Not run)
```

`plantGenusNames` *Vector of plant genus names from ThePlantList*

Description

These names are from http://www.theplantlist.org, and are a randomly chosen subset of genera names for the purpose of having some names to play with for examples in this package.

Format

A vector of length 793

Source

http://www.theplantlist.org

plantminer *Search for taxonomy data from Plantminer.com*

Description

Search for taxonomy data from Plantminer.com

Usage

```
plantminer(plants, from = "tpl", messages = TRUE, ...)
```

Arguments

plants	(character) Vector of plant species names. Required.
from	(character) One of tpl (for theplantlist.com data), or flora (for Brazilian Flora Checklist). Required. Default: tpl
messages	(logical) informative messages or not. Default: TRUE
...	curl options passed on to HttpClient

Value

data.frame of results.

Note

you used to need an API key for Plantminer; it's no longer needed

Examples

```
## Not run:
# A single taxon
plantminer("Ocotea pulchella")

# Many taxa
plants <- c("Myrcia lingua", "Myrcia bella", "Ocotea pulchella",
"Miconia", "Coffea arabica var. amarella", "Bleh")
plantminer(plants)

# By deafult, tpl is used, for Theplantlist data,
# toggle the from parameter here
plantminer("Ocotea pulchella", from = "flora")

## End(Not run)
```

plantNames — *Vector of plant species (genus - specific epithet) names from ThePlantList*

Description

These names are from http://www.theplantlist.org, and are a randomly chosen subset of names of the form genus/specific epithet for the purpose of having some names to play with for examples in this package.

Format

A vector of length 1182

Source

http://www.theplantlist.org

pow_lookup — *Lookup taxa in Kew's Plants of the World*

Description

Lookup taxa in Kew's Plants of the World

Usage

```
pow_lookup(id, include = NULL, ...)
```

Arguments

id	(character) taxon id. required
include	(character) vector of additional fields to include in results. options include 'distribution' and 'descriptions'. optional
...	Further args passed on to HttpClient.

See Also

Other pow: get_pow, pow_search

Examples

```
## Not run:
pow_lookup(id = 'urn:lsid:ipni.org:names:320035-2')
pow_lookup(id = 'urn:lsid:ipni.org:names:320035-2',
  include = "distribution")
pow_lookup(id = 'urn:lsid:ipni.org:names:320035-2',
  include = c("distribution", "descriptions"))

## End(Not run)
```

`pow_search` *Search Kew's Plants of the World*

Description

Search Kew's Plants of the World

Usage

```
pow_search(q, limit = 100, cursor = "*", sort = NULL, ...)
```

Arguments

`q`	(character) query terms
`limit`	(integer) Number of records to return. default: 100
`cursor`	(character) cursor string
`sort`	(character) The field to sort by and sort order separted with underscore, e.g., `sort="name_desc"`
`...`	Further args passed on to `HttpClient`.

Value

a list with slots for metadata (‘meta‘) with list of response attributes, and data (‘data“) with a data.frame of results

Author(s)

Scott Chamberlain, <myrmecocystus@gmail.com>

References

<http://powo.science.kew.org/>

See Also

Other pow: `get_pow`, `pow_lookup`

Examples

```
## Not run:
x <- pow_search(q = "Quercus")
x$meta
x$meta$totalResults
x$meta$perPage
x$meta$totalPages
x$meta$page
x$meta$cursor
head(x$data)

# pagination
pow_search(q = "sunflower", limit = 2)

# debug curl stuff
invisible(pow_search(q = "Helianthus annuus", verbose = TRUE))

# sort
desc <- pow_search(q = "Helianthus", sort = "name_desc")
desc$data$name
asc <- pow_search(q = "Helianthus", sort = "name_asc")
asc$data$name

## End(Not run)
```

---

rankagg *Aggregate data by given taxonomic rank*

Description

Aggregate data by given taxonomic rank

Usage

```
rankagg(data = NULL, datacol = NULL, rank = NULL, fxn = "sum")
```

Arguments

data	A data.frame. Column headers must have capitalized ranks (e.g., Genus, Tribe, etc.) (data.frame)
datacol	The data column (character)
rank	Taxonomic rank to aggregate by (character)
fxn	Arithmetic function or vector or functions (character)

Examples

```
library("vegan")
data(dune.taxon, package='vegan')
dat <- dune.taxon
set.seed(1234)
dat$abundance <- round(rlnorm(n=nrow(dat),meanlog=5,sdlog=2),0)
rankagg(data=dat, datacol="abundance", rank="Genus")
rankagg(data=dat, "abundance", rank="Family")
rankagg(data=dat, "abundance", rank="Genus", fxn="mean")
rankagg(data=dat, "abundance", rank="Subclass")
rankagg(data=dat, "abundance", rank="Subclass", fxn="sd")
```

---

rank_ref *Lookup-table for IDs of taxonomic ranks*

Description

data.frame of 43 rows, with 2 columns:

- rankid - a numeric rank id, consecutive
- ranks - a comma separated vector of names that are considered equal to one another within the row

Details

We use this data.frame to do data sorting/filtering based on the ordering of ranks.

Please let us know if there is a rank that occurs from one of the data sources **taxize** that we don't have in `rank_ref` dataset.

Let us know if you disagree with the ordering of ranks.

---

resolve *Resolve names from different data sources*

Description

Resolve names from iPlant's name resolver, the Taxonomic Name Resolution Service (TNRS), and the Global Names Resolver (GNR)

Usage

```
resolve(query, db = "gnr", ...)
```

Arguments

query	Vector of one or more taxonomic names (common names not supported)
db	Source to check names against. One of iplant, tnrs, or gnr. Default: gnr. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong db value for the identifier you could get a result, but it will likely be wrong (not what you were expecting).
...	Curl options passed on to verb-GET or verb-POST. In addition, further named args passed on to each respective function. See examples

Value

A list with length equal to length of the db parameter (number of sources requested), with each element being a data.frame or list with results from that source.

Examples

```
## Not run:
resolve(query=c("Helianthus annuus", "Homo sapiens"))
resolve(query="Quercus keloggii", db='gnr')
resolve(query=c("Helianthus annuus", "Homo sapiens"), db='tnrs')
resolve(query=c("Helianthus annuus", "Homo sapiens"), db=c('iplant', 'gnr'))
resolve(query="Quercus keloggii", db=c('iplant', 'gnr'))
resolve(query="Quercus keloggii", db=c('iplant', 'gnr', 'tnrs'))

# pass in options specific to each source
resolve("Helianthus annuus", db = 'gnr', preferred_data_sources = c(3, 4))
resolve("Helianthus annuus", db = 'iplant', retrieve = 'best')
identical(
 resolve("Helianthus annuus", db = 'iplant', retrieve = 'best')$iplant,
 iplant_resolve("Helianthus annuus", retrieve = 'best')
)
mynames <- c("Helianthus annuus", "Pinus contorta", "Poa annua",
   "Abies magnifica", "Rosa california")
resolve(mynames, db = 'tnrs', source = "NCBI")
resolve(mynames, db = 'tnrs', source = "iPlant_TNRS")
identical(
 resolve(mynames, db = 'tnrs', source = "iPlant_TNRS")$tnrs,
 tnrs(mynames, source = "iPlant_TNRS")
)

# pass in curl options
resolve(query="Qercuss", db = "iplant", verbose = TRUE)

## End(Not run)
```

sci2comm *Get common names from scientific names.*

Description

Get common names from scientific names.

Usage

```
sci2comm(...)

## Default S3 method:
sci2comm(scinames, db = "ncbi", simplify = TRUE, ...)

## S3 method for class 'uid'
sci2comm(id, ...)

## S3 method for class 'tsn'
sci2comm(id, simplify = TRUE, ...)

## S3 method for class 'wormsid'
sci2comm(id, simplify = TRUE, ...)

## S3 method for class 'iucn'
sci2comm(id, simplify = TRUE, ...)
```

Arguments

...	Further arguments passed on to functions get_uid, get_tsn.
scinames	character; One or more scientific names or partial names.
db	character; Data source, one of *"ncbi"* (default), *"itis"* *"eol"*, *"worms"*, or *"iucn"*. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong db value for the identifier you could get a result, but it will likely be wrong (not what you were expecting). If using ncbi, eol or iucn we recommend getting an API key; see taxize-authentication
simplify	(logical) If TRUE, simplify output to a vector of names. If FALSE, return variable formats from different sources, usually a data.frame. Only applies to eol and itis. Specify FALSE to obtain the language of each vernacular in the output for eol and itis.
id	character; identifiers, as returned by get_tsn, get_uid.

Value

List of character vectors, named by input taxon name, or taxon ID

Authentication

See taxize-authentication for help on authentication

Author(s)

Scott Chamberlain (myrmecocystus@gmail.com)

See Also

comm2sci

Examples

```
## Not run:
sci2comm(scinames='Helianthus annuus')
sci2comm(scinames='Helianthus annuus', db='eol')
sci2comm(scinames='Helianthus annuus', db='itis')
sci2comm(scinames=c('Helianthus annuus', 'Poa annua'))
sci2comm(scinames='Puma concolor', db='ncbi')
sci2comm('Gadus morhua', db='worms')
sci2comm('Pomatomus saltatrix', db='worms')
sci2comm('Loxodonta africana', db='iucn')

# Passing id in, works for sources: itis and ncbi, not eol
sci2comm(get_tsn('Helianthus annuus'))
sci2comm(get_uid('Helianthus annuus'))
sci2comm(get_wormsid('Gadus morhua'))
sci2comm(get_iucn('Loxodonta africana'))

# Don't simplify returned
sci2comm(get_tsn('Helianthus annuus'), simplify=FALSE)
sci2comm(get_iucn('Loxodonta africana'), simplify=FALSE)

# Use curl options
sci2comm('Helianthus annuus', db="ncbi", verbose = TRUE)

## End(Not run)
```

---

scrapenames *Resolve names using Global Names Recognition and Discovery.*

Description

Uses the Global Names Recognition and Discovery service, see http://gnrd.globalnames.org/.

Note: this function sometimes gives data back and sometimes not. The API that this function is extremely buggy.

Usage

```
scrapenames(url = NULL, file = NULL, text = NULL, engine = NULL,
  unique = NULL, verbatim = NULL, detect_language = NULL,
  all_data_sources = NULL, data_source_ids = NULL,
  return_content = FALSE, ...)
```

Arguments

`url`	An encoded URL for a web page, PDF, Microsoft Office document, or image file, see examples
`file`	When using multipart/form-data as the content-type, a file may be sent. This should be a path to your file on your machine.
`text`	Type: string. Text content; best used with a POST request, see examples
`engine`	(optional) (integer) Default: 0. Either 1 for TaxonFinder, 2 for NetiNeti, or 0 for both. If absent, both engines are used.
`unique`	(optional) (logical) If `TRUE` (default), response has unique names without offsets.
`verbatim`	(optional) Type: boolean, If `TRUE` (default to `FALSE`), response excludes verbatim strings.
`detect_language`	(optional) Type: boolean, When `TRUE` (default), NetiNeti is not used if the language of incoming text is determined not to be English. When `FALSE`, NetiNeti will be used if requested.
`all_data_sources`	(optional) Type: boolean. Resolve found names against all available Data Sources.
`data_source_ids`	(optional) Type: string. Pipe separated list of data source ids to resolve found names against. See list of Data Sources http://resolver.globalnames.org/data_sources.
`return_content`	(logical) return OCR'ed text. returns text string in `x$meta$content` slot. Default: `FALSE`
`...`	Further args passed to verb-GET

Details

One of url, file, or text must be specified - and only one of them.

Value

A list of length two, first is metadata, second is the data as a data.frame.

Author(s)

Scott Chamberlain myrmecocystus@gmail.com

Examples

```
## Not run:
# Get data from a website using its URL
scrapenames('http://en.wikipedia.org/wiki/Araneae')
scrapenames('http://en.wikipedia.org/wiki/Animalia')
scrapenames('http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0095068')
scrapenames('http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0080498')
scrapenames('http://ucjeps.berkeley.edu/cgi-bin/get_JM_treatment.pl?CARYOPHYLLACEAE')

# Scrape names from a pdf at a URL
url <- 'http://www.plosone.org/article/fetchObject.action?uri=
info%3Adoi%2F10.1371%2Fjournal.pone.0058268&representation=PDF'
scrapenames(url = sub('\n', '', url))

# With arguments
scrapenames(url = 'http://www.mapress.com/zootaxa/2012/f/z03372p265f.pdf',
  unique=TRUE)
scrapenames(url = 'http://en.wikipedia.org/wiki/Araneae',
  data_source_ids=c(1, 169))

# Get data from a file
speciesfile <- system.file("examples", "species.txt", package = "taxize")
scrapenames(file = speciesfile)

nms <- paste0(names_list("species"), collapse="\n")
file <- tempfile(fileext = ".txt")
writeLines(nms, file)
scrapenames(file = file)

# Get data from text string
scrapenames(text='A spider named Pardosa moesta Banks, 1892')

# return OCR content
scrapenames(url='http://www.mapress.com/zootaxa/2012/f/z03372p265f.pdf',
  return_content = TRUE)

## End(Not run)
```

---

species_plantarum_binomials

Species names from Species Plantarum

Description

These names have been compiled from *Species Plantarum* by Carl Linnaeus originally published in 1753. It is the first work to consistently apply binomial names and was the starting point for the naming of plants. The book lists every species of plant known at the time, classified into genera. The dataset provides a useful reference point to see how taxonomic names have changed since their inception. The names were transcribed by Robert W. Kiger.

Format

A data frame with 5940 rows and 3 variables:

genus First part of the binomial species name for each species within the genus

epithet specific epithet or second part of the binomial species name for each species

page_number The following abbreviations sometimes are used in the page_number field.

- "add." refers to addenda that appear on the unnumbered last page of the index in volume two.
- "err."refers to the unnumbered page of errata that appears following the index in volume two.
- "canc."following a page number indicates that the binomial appeared on the cancelled version of that page and does not appear on its replacement (as in the 1957-1959 facsimile edition).

Author(s)

Carl Linnaeus

Source

Hunt Institute for Botanical Documentation

References

Linnaeus, C. 1753. Species Plantarum. 2 vols. Salvius, Stockholm. [Facsimile edition, 1957-1959, Ray Society, London.]

`status_codes` *Get HTTP status codes*

Description

Get HTTP status codes

Usage

```
status_codes()
```

See Also

`ping`

Examples

```
status_codes()
```

synonyms *Retrieve synonyms from various sources given input taxonomic names or identifiers*

Description

Retrieve synonyms from various sources given input taxonomic names or identifiers

Usage

```
synonyms(...)

## Default S3 method:
synonyms(x, db = NULL, rows = NA, ...)

## S3 method for class 'tsn'
synonyms(id, ...)

## S3 method for class 'colid'
synonyms(id, ...)

## S3 method for class 'tpsid'
synonyms(id, ...)

## S3 method for class 'nbnid'
synonyms(id, ...)

## S3 method for class 'wormsid'
synonyms(id, ...)

## S3 method for class 'iucn'
synonyms(id, ...)

## S3 method for class 'ids'
synonyms(id, ...)

synonyms_df(x)
```

Arguments

...	Other passed arguments to internal functions `get_*()` and functions to gather synonyms.
x	Vector of taxa names (character) or IDs (character or numeric) to query.
db	character; database to query. either `itis`, `tropicos`, `col`, `nbn`, or `worms`. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong `db` value for the identifier you could get a result, but it will likely be

wrong (not what you were expecting). If using tropicos, we recommend getting an API key; see taxize-authentication

rows (numeric) Any number from 1 to infinity. If the default NA, all rows are considered. Note that this parameter is ignored if you pass in a taxonomic id of any of the acceptable classes: tsn, tpsid, nbnid, ids.

id character; identifiers, returned by get_tsn, get_tpsid, get_nbnid, get_colid, get_wormsid

Details

If IDs are supplied directly (not from the get_* functions) you must specify the type of ID.

For db = "itis" you can pass in a parameter accepted to toggle whether only accepted names are used accepted = TRUE, or if all are used accepted = FALSE. The default is accepted = FALSE

Note that IUCN requires an API key. See rredlist-package for help on authentiating with IUCN Redlist

Value

A named list of data.frames with the synonyms of every supplied taxa.

See Also

get_tsn, get_tpsid, get_nbnid, get_colid, get_wormsid, get_iucn

Examples

```
## Not run:
# Plug in taxon IDs
synonyms(183327, db="itis")
synonyms("25509881", db="tropicos")
synonyms("NBNSYS0000004629", db='nbn')
# synonyms("87e986b0873f648711900866fa8abde7", db='col') # FIXME
synonyms(105706, db='worms')
synonyms(12392, db='iucn')

# Plug in taxon names directly
synonyms("Pinus contorta", db="itis")
synonyms("Puma concolor", db="itis")
synonyms(c("Poa annua",'Pinus contorta','Puma concolor'), db="itis")
synonyms("Poa annua", db="tropicos")
synonyms("Pinus contorta", db="tropicos")
synonyms(c("Poa annua",'Pinus contorta'), db="tropicos")
synonyms("Pinus sylvestris", db='nbn')
synonyms("Puma concolor", db='col')
synonyms("Ursus americanus", db='col')
synonyms("Amblyomma rotundatum", db='col')
synonyms('Pomatomus', db='worms')
synonyms('Pomatomus saltatrix', db='worms')

# not accepted names, with ITIS
```

```
## looks for whether the name given is an accepted name,
## and if not, uses the accepted name to look for synonyms
synonyms("Acer drummondii", db="itis")
synonyms("Spinus pinus", db="itis")

# Use get_* methods
synonyms(get_tsn("Poa annua"))
synonyms(get_tpsid("Poa annua"))
synonyms(get_nbnid("Carcharodon carcharias"))
synonyms(get_colid("Ornithodoros lagophilus"))
synonyms(get_iucn('Loxodonta africana'))

# Pass many ids from class "ids"
out <- get_ids(names="Poa annua", db = c('itis','tropicos'))
synonyms(out)

# Use the rows parameter to select certain rows
synonyms("Poa annua", db='tropicos', rows=1)
synonyms("Poa annua", db='tropicos', rows=1:3)
synonyms("Pinus sylvestris", db='nbn', rows=1:3)
synonyms("Amblyomma rotundatum", db='col', rows=2)
synonyms("Amblyomma rotundatum", db='col', rows=2:3)

# Use curl options
synonyms("Poa annua", db='tropicos', rows=1, verbose = TRUE)
synonyms("Poa annua", db='itis', rows=1, verbose = TRUE)
synonyms("Poa annua", db='col', rows=1, verbose = TRUE)


# combine many outputs together
x <- synonyms(c("Osmia bicornis", "Osmia rufa", "Osmia"), db = "itis")
synonyms_df(x)

## note here how Pinus contorta is dropped due to no synonyms found
x <- synonyms(c("Poa annua",'Pinus contorta','Puma concolor'), db="col")
synonyms_df(x)

## note here that ids are taxon identifiers b/c you start with them
x <- synonyms(c(25509881, 13100094), db="tropicos")
synonyms_df(x)

## NBN
x <- synonyms(c('Aglais io', 'Usnea hirta', 'Arctostaphylos uva-ursi'),
  db="nbn")
synonyms_df(x)

## End(Not run)
```

---

taxize-authentication *taxize authentication*

Description

Help on authentication

What is an API?

An API is an Application Programming Interface. The term "API" can be used for lots of scenarios, but in this case we're talking about web APIs, or APIs (interfaces) to web resources. **taxize** interacts with remote databases on the web via their APIs. You don't need to worry about the details of how that all works; just know that some of them require authentication and some do not.

What are API keys?

For those APIs that require authentication, the way that's typically done is through API keys: alphanumeric strings of variable lengths that are supplied with a request to an API.

taxize won't get these keys for you; rather, you have to go get a key for each service, but we do provide information on how to get those keys. See `key_helpers` for help on how to obtain keys for this package.

Using API keys

You can store API keys as R options in your `.Rprofile` file, or as environment variables in either your `.Renviron` file or `.bash_profile` file, or `.zshrc` file (if you use oh-my-zsh) or similar. See `Startup` for help on R options and environment variables.

Save your API keys with the following names:

- Tropicos: R option or env var as 'TROPICOS_KEY'
- EOL: R option or env var as 'EOL_KEY'
- IUCN: R option or env var as 'IUCN_REDLIST_KEY'
- ENTREZ: R option or env var as 'ENTREZ_KEY'

If you save in .Renviron it looks like: `ENTREZ_KEY=somekey`

If you save in a .bash_profile, .zshrc, or similar file it looks like: `export ENTREZ_KEY=somekey`

If you save in a .Rprofile it looks like: `options(ENTREZ_KEY = "somekey")`

Remember to restart your R session (and to start a new shell window/tab if you're using the shell) to take advantage of the new R options or environment variables.

We strongly recommend using environment variables (https://en.wikipedia.org/wiki/Environment_variable) over R options because environment variables are widely used across programming languages, operating systems, and computing environments; whereas R options are specific to R.

Note that NCBI Entrez doesn't require that you use an API key, but you do get a higher rate limit with a key (more requests per time period), from 3 to 10 requests per second, so do get one.

See Also

`key_helpers`

taxize-defunct *Defunct functions in taxize*

Description

The following functions are now defunct (no longer available):

Details

- col_classification: See classification
- eol_hierarchy: See classification
- tp_classification: See classification
- tpl_search: Use the **Taxonstand** functions TPL or TPLck directly.
- get_seqs: This function changed name to ncbi_getbyname.
- get_genes: This function changed name to ncbi_getbyid.
- get_genes_avail: This function changed name to ncbi_search.
- ncbi_getbyname: See ncbi_byname in the **traits** package.
- ncbi_getbyid: See ncbi_byid in the **traits** package.
- ncbi_search: See ncbi_searcher in the **traits** package.
- eol_invasive: See eol in the **originr** package.
- gisd_isinvasive: See gisd in the **originr** package.
- ubio_classification: The uBio web services was down for quite a while, is now (as of 2016-05-09) back up, but we don't trust that it will stay up and available.
- ubio_classification_search: The uBio web services was down for quite a while, is now (as of 2016-05-09) back up, but we don't trust that it will stay up and available.
- ubio_id: The uBio web services was down for quite a while, is now (as of 2016-05-09) back up, but we don't trust that it will stay up and available.
- ubio_ping: The uBio web services was down for quite a while, is now (as of 2016-05-09) back up, but we don't trust that it will stay up and available.
- ubio_search: The uBio web services was down for quite a while, is now (as of 2016-05-09) back up, but we don't trust that it will stay up and available.
- ubio_synonyms: The uBio web services was down for quite a while, is now (as of 2016-05-09) back up, but we don't trust that it will stay up and available.
- get_ubioid: The uBio web services are apparently down indefinitely.
- phylomatic_tree: This function is defunct. See phylomatic in the package **brranching**
- phylomatic_format: This function is defunct. See phylomatic_names in the package **brranching**
- iucn_summary_id: This function is defunct. Use iucn_summary

taxize_capwords *Capitalize the first letter of a character string.*

Description

Capitalize the first letter of a character string.

Usage

```
taxize_capwords(s, strict = FALSE, onlyfirst = FALSE)
```

Arguments

s	A character string
strict	Should the algorithm be strict about capitalizing. Defaults to FALSE.
onlyfirst	Capitalize only first word, lowercase all others. Useful for taxonomic names.

Examples

```
taxize_capwords(c("using AIC for model selection"))
taxize_capwords(c("using AIC for model selection"), strict=TRUE)
```

taxize_cite *Get citations and licenses for data sources used in taxize*

Description

Get citations and licenses for data sources used in taxize

Usage

```
taxize_cite(fxn = "itis", what = "citation")
```

Arguments

fxn	Function to search on. A special case is the package name 'taxize' that will give the citations for the package.
what	One of citation (default), license, or both.

Examples

```
taxize_cite(fxn='eol_search')
taxize_cite(fxn='itis_hierarchy')
taxize_cite(fxn='tp_classification')
taxize_cite(fxn='gbif_ping')
taxize_cite(fxn='plantminer')
taxize_cite(fxn='get_natservid_')
taxize_cite(fxn='as.natservid')
taxize_cite(fxn='get_wormsid')
taxize_cite(fxn='as.wormsid')

# Functions that use many data sources
taxize_cite(fxn='synonyms')
taxize_cite(fxn='classification')

# Get the taxize citation
taxize_cite(fxn='taxize')

# Get license information
taxize_cite(fxn='taxize', "license")
```

---

tax_agg *Aggregate species data to given taxonomic rank*

Description

Aggregate species data to given taxonomic rank

Usage

```
tax_agg(x, rank, db = "ncbi", verbose = FALSE, ...)

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

Arguments

x	Community data matrix. Taxa in columns, samples in rows.
rank	character; Taxonomic rank to aggregate by.
db	character; taxonomic API to use, 'ncbi, 'itis' or both, see tax_name. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong db value for the identifier you could get a result, but it will likely be wrong (not what you were expecting). If using ncbi we recommend getting an API key; see taxize-authentication
verbose	(logical) If FALSE (Default) suppress messages
...	Other arguments passed to get_tsn or get_uid.

Details

tax_agg aggregates (sum) taxa to a specific taxonomic level. If a taxon is not found in the database (ITIS or NCBI) or the supplied taxon is on higher taxonomic level this taxon is not aggregated.

Value

A list of class tax_agg with the following items:

x	Community data matrix with aggregated data.
by	A lookup-table showing which taxa were aggregated.
n_pre	Number of taxa before aggregation.
rank	Rank at which taxa have been aggregated.

See Also

tax_name

Examples

```
## Not run:
if (requireNamespace("vegan", quietly = TRUE)) {
  # use dune dataset
  library("vegan")
  data(dune, package='vegan')
  species <- c("Bellis perennis", "Empetrum nigrum", "Juncus bufonius",
  "Juncus articulatus",
  "Aira praecox", "Eleocharis parvula", "Rumex acetosa", "Vicia lathyroides",
  "Brachythecium rutabulum", "Ranunculus flammula", "Cirsium arvense",
  "Hypochaeris radicata", "Leontodon autumnalis", "Potentilla palustris",
  "Poa pratensis", "Calliergonella cuspidata", "Trifolium pratense",
  "Trifolium repens", "Anthoxanthum odoratum", "Salix repens", "Achillea
  millefolium",
  "Poa trivialis", "Chenopodium album", "Elymus repens", "Sagina procumbens",
  "Plantago lanceolata", "Agrostis stolonifera", "Lolium perenne", "Alopecurus
  geniculatus", "Bromus hordeaceus")
  colnames(dune) <- species

  # aggregate sample to families
  (agg <- tax_agg(dune, rank = 'family', db = 'ncbi'))

  # extract aggregated community data matrix for further usage
  agg$x
  # check which taxa have been aggregated
  agg$by
}

# A use case where there are different taxonomic levels in the same dataset
spnames <- c('Puma','Ursus americanus','Ursidae')
df <- data.frame(c(1,2,3), c(11,12,13), c(1,4,50))
names(df) <- spnames
out <- tax_agg(df, rank = 'family', db='itis')
```

```
out$x

# You can input a matrix too
mat <- matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3,
 dimnames=list(NULL, c('Puma concolor','Ursus americanus','Ailuropoda melanoleuca')))
tax_agg(mat, rank = 'family', db='itis')

## End(Not run)
```

`tax_name` *Get taxonomic names for a given rank*

Description

Get taxonomic names for a given rank

Usage

```
tax_name(query, get, db = "itis", pref = "ncbi", messages = TRUE,
  ...)
```

Arguments

`query`	(character) Vector of taxonomic names to query. required.
`get`	(character) The ranks of the taxonomic name to get, see `rank_ref`. required.
`db`	(character) The database to search from: 'itis', 'ncbi' or 'both'. If 'both' both NCBI and ITIS will be queried. Result will be the union of both. If using ncbi, we recommend getting an API key; see `taxize-authentication`
`pref`	(character) If db = 'both', sets the preference for the union. Either 'ncbi' (default) or 'itis'. Currently not implemented.
`messages`	(logical) If `TRUE` the actual taxon queried is printed on the console.
`...`	Other arguments passed to `get_tsn` or `get_uid`.

Value

A data.frame with one column for every queried rank, in addition to a column for db and queried term.

Authentication

See `taxize-authentication` for help on authentication

Note

While `tax_rank` returns the actual rank of a taxon, `tax_name` searches and returns any specified rank higher in taxonomy.

See Also

classification

Examples

```
## Not run:
# A case where itis and ncbi use the same names
tax_name(query = "Helianthus annuus", get = "family", db = "itis")
tax_name(query = "Helianthus annuus", get = "family", db = "ncbi")
tax_name(query = "Helianthus annuus", get = c("genus","family","order"),
  db = "ncbi")

# Case where itis and ncbi use different names
tax_name(query = "Helianthus annuus", get = "kingdom", db = "itis")
tax_name(query = "Helianthus annuus", get = "kingdom", db = "ncbi")

# multiple rank arguments
tax_name(query = c("Helianthus annuus","Baetis rhodani"), get = c("genus",
"kingdom"), db = "ncbi")
tax_name(query = c("Helianthus annuus","Baetis rhodani"), get = c("genus",
"kingdom"), db = "itis")

# query both sources
tax_name(query=c("Helianthus annuus", 'Baetis rhodani'), get=c("genus",
"kingdom"), db="both")

## End(Not run)
```

---

tax_rank *Get rank for a given taxonomic name.*

Description

Get rank for a given taxonomic name.

Usage

```
tax_rank(x, db = NULL, rows = NA, ...)
```

Arguments

x	(character) Vector of one or more taxon names (character) or IDs (character or numeric) to query. Or objects returned from `get_*()` functions like `get_tsn`
db	(character) database to query. either `ncbi`, `itis`, `eol`, `col`, `tropicos`, `gbif`, `nbn`, `worms`, `natserv`, `bold`. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong db value for the identifier you may get a result, but it will likely be wrong (not what you were expecting). If using ncbi or eol we recommend getting an API key; see `taxize-authentication`

rows	numeric; Any number from 1 to infinity. If the default NA, all rows are considered. passed down to get_* functions.
...	Additional arguments to classification

Value

A named list of character vectors with ranks (all lower-cased)

Note

While tax_name returns the name of a specified rank, tax_rank returns the actual rank of the taxon.

See Also

classification, tax_name

Examples

```
## Not run:
tax_rank(x = "Helianthus annuus", db = "itis")
tax_rank(get_tsn("Helianthus annuus"))
tax_rank(c("Helianthus", "Pinus", "Poa"), db = "itis")

tax_rank(get_boldid("Helianthus annuus"))
tax_rank("421377", db = "bold")
tax_rank(421377, db = "bold")

tax_rank(c("Plantae", "Helianthus annuus",
  "Puma", "Homo sapiens"), db = 'itis')
tax_rank(c("Helianthus annuus", "Quercus", "Fabaceae"), db = 'tropicos')

tax_rank(names_list("species"), db = 'gbif')
tax_rank(names_list("family"), db = 'gbif')

tax_rank(c("Platanista gangetica", "Lichenopora neapolitana"),
  db = "worms")

## End(Not run)
```

---

theplantlist *Lookup-table for family, genus, and species names for ThePlantList*

Description

These names are from http://www.theplantlist.org, and are from version 1.1 of their data. This data is used in the function names_list. This is a randomly selected subset of the ~350K accepted species names in Theplantlist.

Format

A data frame with 10,000 rows and 3 variables:

family family name

genus genus name

species specific epithet name

Source

http://www.theplantlist.org

`tnrs` *Phylotastic Taxonomic Name Resolution Service.*

Description

Match taxonomic names using the Taxonomic Name Resolution Service (TNRS). Returns score of the matched name, and whether it was accepted or not.

Usage

```
tnrs(query = NA, source = NULL, code = NULL, getpost = "POST",
  sleep = 0, splitby = 30, messages = TRUE, ...)
```

Arguments

`query`	Vector of quoted taxonomic names to search (character).
`source`	Specify the source you want to match names against. Defaults to just retrieve data from all sources. Options: NCBI, iPlant_TNRS, or MSW3. Only available when using getpost="POST".
`code`	Nomenclatural code. One of: ICZN (zoological), ICN (algae, fungi, and plants), ICNB (bacteria), ICBN (botanical), ICNCP (cultivated plants), ICTV (viruses). Only available when using getpost="POST".
`getpost`	Use GET or POST method to send the query. If you have more than say 50 species or so in your query, you should probably use POST. IMPORTANT!!!!! -> POST is the only option for this parameter if you want to use source or code parameters.
`sleep`	Number of seconds by which to pause between calls. Defaults to 0 seconds. Use when doing many calls in a for loop ar lapply type call.
`splitby`	Number by which to split species list for querying the TNRS.
`messages`	Verbosity or not (default `TRUE`)
`...`	Curl options to pass in `verb-GET` or `verb-POST`

Details

If there is no match in the Taxosaurus database, nothing is returned, so you will not get anything back for non-matches.

TNRS doesn't provide any advice about the occurrence of homonyms when queries have no indication of a taxonomic name's authority. So if there is any chance of a homonym, you probably want to send the authority as well, or use gnr_resolve. For example, `tnrs(query="Jussiaea linearis", source="iPlant_TNRS` gives result of *Jussiaea linearis (Willd.) Oliv. ex Kuntze*, but there is a homonym. If you do `tnrs(query="Jussiaea linearis Hochst.", source="iPlant_TNRS")` you get a direct match for that name. So, beware that there's no indication of homonyms.

Value

data.frame of results from TNRS plus the name submitted, with rows in order of user supplied names, though those with no matches are dropped

References

http://taxosaurus.org/

See Also

gnr_resolve

Examples

```
## Not run:
mynames <- c("Helianthus annuus", "Poa annua", "Mimulus bicolor")
tnrs(query = mynames, source = "iPlant_TNRS")

# Specifying the nomenclatural code to match against
mynames <- c("Helianthus annuus", "Poa annua")
tnrs(query = mynames, code = "ICBN")

# You can specify multiple sources, by comma-separating them
mynames <- c("Panthera tigris", "Eutamias minimus", "Magnifera indica",
"Humbert humbert")
tnrs(query = mynames, source = "NCBI,MSW3")

mynames <- c("Panthera tigris", "Eutamias minimus", "Magnifera indica",
   "Humbert humbert", "Helianthus annuus", "Pinus contorta", "Poa annua",
   "Abies magnifica", "Rosa california", "Festuca arundinace",
   "Mimulus bicolor", "Sorbus occidentalis","Madia sativa", "Thymopsis thymodes",
   "Bartlettia scaposa")
tnrs(mynames, source = "NCBI")

# Pass on curl options
mynames <- c("Helianthus annuus", "Poa annua", "Mimulus bicolor")
tnrs(query = mynames, source = "iPlant_TNRS", verbose = TRUE)

## End(Not run)
```

tnrs_sources *TNRS sources*

Description

Get sources for the Phylotastic Taxonomic Name Resolution Service

Usage

```
tnrs_sources(source = NULL, ...)
```

Arguments

source	The source to get information on, one of "iPlant_TNRS", "NCBI", or "MSW3".
...	Curl options to pass in verb-GET

Value

Sources for the TNRS API in a vector or list

Examples

```
## Not run:
# All
tnrs_sources()

# A specific source
tnrs_sources(source="NCBI")
tnrs_sources(source="MSW3")
tnrs_sources(source="iPlant_TNRS")

## End(Not run)
```

tol_resolve *Resolve names using Open Tree of Life (OTL) resolver*

Description

Resolve names using Open Tree of Life (OTL) resolver

Usage

```
tol_resolve(names = NULL, context_name = NULL,
  do_approximate_matching = TRUE, ids = NULL,
  include_suppressed = FALSE, ...)
```

Arguments

names	(character vector) taxon names to be queried
context_name	name of the taxonomic context to be searched (length-one character vector). Must match (case sensitive) one of the values returned by tnrs_contexts.
do_approximate_matching	(logical) A logical indicating whether or not to perform approximate string (a.k.a. "fuzzy") matching. Using FALSE will greatly improve speed. Default: TRUE
ids	An array of OTL ids to use for identifying names. These will be assigned to each name in the names array. If ids is provided, then ids and names must be identical in length.
include_suppressed	(logical) Ordinarily, some quasi-taxa, such as incertae sedis buckets and other non-OTUs, are suppressed from TNRS results. If this parameter is true, these quasi-taxa are allowed as possible TNRS results. Default: FALSE
...	Curl options passed on to httr::POST within tnrs_match_names

Value

A data frame summarizing the results of the query. The original query output is appended as an attribute to the returned object (and can be obtained using attr(object,"original_response")).

Author(s)

Francois Michonneau <francois.michonneau@gmail.com> Scott Chamberlain <myrmecocystus@gmail.com>

References

https://github.com/OpenTreeOfLife/germinator/wiki/TNRS-API-v3#match_names

See Also

gnr_resolve, tnrs

Examples

```
## Not run:
tol_resolve(names=c("echinodermata", "xenacoelomorpha",
  "chordata", "hemichordata"))
tol_resolve(c("Hyla", "Salmo", "Diadema", "Nautilus"))
tol_resolve(c("Hyla", "Salmo", "Diadema", "Nautilus"),
  context_name = "Animals")

turducken_spp <- c("Meleagris gallopavo", "Anas platyrhynchos",
  "Gallus gallus")
tol_resolve(turducken_spp, context_name="Animals")

## End(Not run)
```

`tpl_families` *Get The Plant List families.*

Description

Get The Plant List families.

Usage

```
tpl_families(...)
```

Arguments

`...`	(list) Curl options passed on to `verb-GET`

Details

Requires an internet connection in order to connect to www.theplantlist.org.

Value

Returns a `data.frame` including the names of all families indexed by The Plant List, and the major groups into which they fall (i.e. Angiosperms, Gymnosperms, Bryophytes and Pteridophytes).

Author(s)

John Baumgartner (johnbb@student.unimelb.edu.au)

See Also

`tpl_get`

Examples

```
## Not run:
# Get a data.frame of plant families, with the group name
# (Angiosperms, etc.)
head(tpl_families())

## End(Not run)
```

tpl_get *Get The Plant List csv files.*

Description

Get The Plant List csv files.

Usage

```
tpl_get(x, family = NULL, ...)
```

Arguments

x	Directory to write csv files to.
family	If you want just one, or >1 family, but not all, list them in a vector.
...	(list) Curl options passed on to verb-GET

Details

Throws a warning if you already have a directory of the one provided, but still works. Writes to your home directory, change x as needed.

Value

Returns nothing to console, except a message and progress bar. Writes csv files to x.

Author(s)

John Baumgartner (johnbb@student.unimelb.edu.au)

References

The Plant List http://www.theplantlist.org

See Also

tpl_families

Examples

```
## Not run:
# Get a few families
dir <- file.path(tempdir(), "abc")
tpl_get(dir, family = c("Platanaceae","Winteraceae"))
readLines(file.path(dir, "Platanaceae.csv"), n = 5)

# You can now get Gymnosperms as well
dir1 <- file.path(tempdir(), "def")
```

```
tpl_get(dir1, family = c("Pinaceae","Taxaceae"))

# You can get mosses too!
dir2 <- file.path(tempdir(), "ghi")
tpl_get(dir2, family = "Echinodiaceae")

# Get all families
## Beware, will take a while
## dir3 <- file.path(tempdir(), "jkl")
## tpl_get("dir3)

## End(Not run)
```

`tpl_search` *A light wrapper around the taxonstand fxn to call Theplantlist.org database.*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
tpl_search()
```

`tp_accnames` *Return all accepted names for a taxon name with a given id.*

Description

Return all accepted names for a taxon name with a given id.

Usage

```
tp_accnames(id, key = NULL, ...)
```

Arguments

`id`	the taxon identifier code
`key`	Your Tropicos API key; See `taxize-authentication` for help on authentication
`...`	Curl options passed on to `verb-GET`

Value

List or dataframe.

Examples

```
## Not run:
tp_accnames(id = 25503923)
tp_accnames(id = 25538750)

# No accepted names found
tp_accnames(id = 25509881)

## End(Not run)
```

`tp_dist` *Return all distribution records for for a taxon name with a given id.*

Description

Return all distribution records for for a taxon name with a given id.

Usage

```
tp_dist(id, key = NULL, ...)
```

Arguments

`id`	the taxon identifier code
`key`	Your Tropicos API key; See `taxize-authentication` for help on authentication
`...`	Curl options passed on to `HttpClient`

Value

List of two data.frame's, one named "location", and one "reference".

References

http://services.tropicos.org/help?method=GetNameDistributionsXml

Examples

```
## Not run:
# Query using a taxon name Id
out <- tp_dist(id = 25509881)
## just location data
head(out[['location']])
## just reference data
head(out[['reference']])

## End(Not run)
```

tp_refs *Return all reference records for for a taxon name with a given id.*

Description

Return all reference records for for a taxon name with a given id.

Usage

```
tp_refs(id, key = NULL, ...)
```

Arguments

id	the taxon identifier code
key	Your Tropicos API key; See taxize-authentication for help on authentication
...	Curl options passed on to HttpClient

Value

List or dataframe.

Examples

```
## Not run:
tp_refs(id = 25509881)

## End(Not run)
```

tp_search *Search Tropicos by scientific name, common name, or Tropicos ID.*

Description

Search Tropicos by scientific name, common name, or Tropicos ID.

Usage

```
tp_search(name = NULL, commonname = NULL, nameid = NULL,
  orderby = NULL, sortorder = NULL, pagesize = NULL,
  startrow = NULL, type = NULL, key = NULL, ...)
```

Arguments

name	Your search string. For instance "poa annua". See Details.
commonname	Your search string. For instance "annual blue grass"
nameid	Your search string. For instance "25509881"
orderby	Your search string. For instance "1"
sortorder	Your search string. For instance "ascending"
pagesize	Your search string. For instance "100"
startrow	Your search string. For instance "1"
type	Type of search, "wildcard" (default) will add a wildcard to the end of your search string. "exact" will use your search string exactly.
key	Your Tropicos API key; See taxize-authentication for help on authentication
...	Further args passed on to HttpClient

Details

More details on the name parameter: Tropicos will fail if you include a period (.) in your name string, e.g., var., so we replace periods before the request is made to the Tropicos web service. In addition, Tropicos for some reason doesn't want to see sub-specific rank names like var/subsp, so remove those from your query.

Value

List or dataframe.

References

http://services.tropicos.org/help?method=SearchNameXml

Examples

```
## Not run:
tp_search(name = 'Poa annua')
tp_search(name = 'Poa annua subsp. annua')
tp_search(name = 'Poa annua var. annua')
tp_search(name = 'Poa annua var annua')
tp_search(name = 'Poa annua annua')

## End(Not run)
```

tp_summary *Return summary data a taxon name with a given id.*

Description

Return summary data a taxon name with a given id.

Usage

```
tp_summary(id, key = NULL, ...)
```

Arguments

id	the taxon identifier code
key	Your Tropicos API key; See taxize-authentication for help on authentication
...	Curl options passed on to verb-GET

Value

A data.frame.

Examples

```
## Not run:
tp_summary(id = 25509881)
tp_summary(id = 2700851)
tp_summary(id = 24900183)

## End(Not run)
```

tp_synonyms *Return all synonyms for a taxon name with a given id.*

Description

Return all synonyms for a taxon name with a given id.

Usage

```
tp_synonyms(id, key = NULL, ...)
```

Arguments

id	the taxon identifier code
key	Your Tropicos API key; See taxize-authentication for help on authentication
...	Curl options passed on to HttpClient

Value

List or dataframe.

Examples

```
## Not run:
tp_synonyms(id = 25509881)

## End(Not run)
```

---

ubio_classification *uBio classification*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
ubio_classification(...)
```

Arguments

...	Parameters, ignored

---

ubio_classification_search

This function will return ClassificationBankIDs (hierarchiesIDs) that refer to the given NamebankID

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
ubio_classification_search(...)
```

Arguments

...	Parameters, ignored

ubio_id *Search uBio by namebank ID.*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
ubio_id(...)
```

Arguments

...	Parameters, ignored

ubio_ping *uBio ping*

Description

uBio ping

Usage

```
ubio_ping()
```

ubio_search *This function will return NameBankIDs that match given search terms*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
ubio_search(...)
```

Arguments

...	Parameters, ignored

ubio_synonyms *Search uBio for taxonomic synonyms by hierarchiesID.*

Description

THIS FUNCTION IS DEFUNCT.

Usage

```
ubio_synonyms(...)
```

Arguments

...	Parameters, ignored

upstream *Retrieve the upstream taxa for a given taxon name or ID.*

Description

This function uses a while loop to continually collect taxa up to the taxonomic rank that you specify in the `upto` parameter. You can get data from ITIS (itis) or Catalogue of Life (col). There is no method exposed by itis or col for getting taxa at a specific taxonomic rank, so we do it ourselves inside the function.

Usage

```
upstream(...)

## Default S3 method:
upstream(x, db = NULL, upto = NULL, rows = NA, ...)

## S3 method for class 'tsn'
upstream(x, db = NULL, upto = NULL, ...)

## S3 method for class 'colid'
upstream(x, db = NULL, upto = NULL, ...)

## S3 method for class 'ids'
upstream(x, db = NULL, upto = NULL, ...)
```

Arguments

...	Further args passed on to itis_downstream or col_downstream
x	Vector of taxa names (character) or IDs (character or numeric) to query.
db	character; database to query. One or both of itis, col. Note that each taxonomic data source has their own identifiers, so that if you provide the wrong db value for the identifier you could get a result, but it will likely be wrong (not what you were expecting).
upto	What taxonomic rank to go down to. One of: 'superkingdom', 'kingdom', 'subkingdom','infrakingdom','phylum','division','subphylum', 'subdivision','infradivision', 'superclass','class','subclass','infraclass', 'superorder','order','suborder','infraorder','superfamily','famil 'subfamily','tribe','subtribe','genus','subgenus', 'section','subsection', 'species','subspecies','variety','fo 'stirp', 'morph','aberration','subform', or 'unspecified'
rows	(numeric) Any number from 1 to infinity. If the default NA, all rows are considered. Note that this parameter is ignored if you pass in a taxonomic id of any of the acceptable classes: tsn, colid.

Value

A named list of data.frames with the upstream names of every supplied taxa. You get an NA if there was no match in the database.

Examples

```
## Not run:
## col
### get all genera at one level up
upstream("Pinus contorta", db = 'col', upto = 'genus')
### goes to same level, Abies is a genus
upstream("Abies", db = 'col', upto = 'genus')
upstream('Pinus contorta', db = 'col', upto = 'family')
upstream('Poa annua', db = 'col', upto = 'family')
upstream('Poa annua', db = 'col', upto = 'order')

## itis
upstream(x='Pinus contorta', db = 'itis', upto = 'genus')

## both
upstream(get_ids('Pinus contorta', db = c('col','itis')), upto = 'genus')

# Use rows parameter to select certain
upstream('Poa annua', db = 'col', upto = 'genus')
upstream('Poa annua', db = 'col', upto = 'genus', rows=1)

# use curl options
res <- upstream('Poa annua', db = 'col', upto = 'genus', verbose = TRUE)

## End(Not run)
```

vascan_search *Search the CANADENSYS Vascan API.*

Description

Search the CANADENSYS Vascan API.

Usage

```
vascan_search(q, format = "json", raw = FALSE, ...)
```

Arguments

q	(character) Can be a scientific name, a vernacular name or a VASCAN taxon identifier (e.g. 861)
format	(character) One of json (default) or xml.
raw	(logical) If TRUE, raw json or xml returned, if FALSE, parsed data returned.
...	(list) Further args passed on to verb-GET

Details

Note that we lowercase all outputs in data.frame's, but when a list is given back, we don't touch the list names.

Value

json, xml or a list.

Author(s)

Scott Chamberlain myrmecocystus@gmail.com

References

API docs http://data.canadensys.net/vascan/api

Examples

```
## Not run:
vascan_search(q = "Helianthus annuus")
vascan_search(q = "Helianthus annuus", raw=TRUE)
vascan_search(q = c("Helianthus annuus", "Crataegus dodgei"), raw=TRUE)

# format type
## json
c <- vascan_search(q = "Helianthus annuus", format="json", raw=TRUE)
library("jsonlite")
fromJSON(c, FALSE)
```

```
## xml
d <- vascan_search(q = "Helianthus annuus", format="xml", raw=TRUE)
library("xml2")
xml2::read_xml(d)

# lots of names, in this case 50
splist <- names_list(rank='species', size=50)
vascan_search(q = splist)

# Curl options
invisible(vascan_search(q = "Helianthus annuus", verbose = TRUE))

## End(Not run)
```

`worms_downstream` *Retrieve all taxa names downstream in hierarchy for WORMS*

Description

Retrieve all taxa names downstream in hierarchy for WORMS

Usage

```
worms_downstream(id, downto, intermediate = FALSE, start = 1, ...)
```

Arguments

`id`	(integer) One or more AphiaID's
`downto`	(character) The taxonomic level you want to go down to. See examples below. The taxonomic level IS case sensitive, and you do have to spell it correctly. See `data(rank_ref)` for spelling.
`intermediate`	(logical) If `TRUE`, return a list of length two with target taxon rank names, with additional list of data.frame's of intermediate taxonomic groups. Default: `FALSE`
`start`	(integer) Record number to start at
`...`	crul options passed on to `verb-GET`

Value

data.frame of taxonomic information downstream to family from e.g., Order, Class, etc., or if `intermediated=TRUE`, list of length two, with target taxon rank names, and intermediate names.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Examples

```
## Not run:
## the genus Gadus
worms_downstream(id = 125732, downto="species")
worms_downstream(id = 125732, downto="species", intermediate=TRUE)

worms_downstream(id = 51, downto="class")
worms_downstream(id = 51, downto="subclass", intermediate=TRUE)

worms_downstream(id = 105, downto="subclass")

## End(Not run)
```

Index