Package 'traits'

May 18, 2024

Title Species Trait Data from Around the Web

Description Species trait data from many different sources, including sequence data from 'NCBI' (<https://www.ncbi.nlm.nih.gov/>), plant trait data from 'BETYdb', data from 'EOL' 'Traitbank', 'Birdlife' International, and more.

Version 0.5.1

License MIT + file LICENSE

URL https://docs.ropensci.org/traits/, https://github.com/ropensci/traits

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

LazyData true

Encoding UTF-8

VignetteBuilder knitr

Depends R (>= 2.10)

Imports jsonlite (>= 0.9.19), httr (>= 1.1.0), crul (>= 0.6.0), tibble (>= 1.3.4), data.table (>= 1.9.6), readr (>= 1.1.1), taxize (>= 0.7.4), xml2 (>= 0.1.2), rvest (>= 0.3.1), hoardr

Suggests knitr, rmarkdown, testthat, dplyr, plyr

RoxygenNote 7.3.1

X-schema.org-applicationCategory Biodiversity

X-schema.org-keywords traits, API, web-services, species, taxonomy

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

NeedsCompilation no

Author David LeBauer [aut, cre] (<https://orcid.org/0000-0001-7228-053X>), Scott Chamberlain [aut] (<https://orcid.org/0000-0003-1444-9135>), Zachary Foster [aut], Ignasi Bartomeus [aut], Chris Black [aut], David Harris [aut], Rupert Collins [ctb]

Maintainer David LeBauer <dlebauer@gmail.com>

Repository CRAN

Date/Publication 2024-05-17 23:50:03 UTC

R topics documented:

`betydb` *Search for traits from BETYdb*

Description

Search for traits from BETYdb

Get details about a single item from a table

Usage

```
betydb_record(
  id,
  table,
  api_version = NULL,
  betyurl = NULL,
  fmt = NULL,
  key = NULL,
  user = NULL,
  pwd = NULL,
  progress = TRUE,
  ...
)
```

```
betydb_trait(
  id,
  genus = NULL,
  species = NULL,
  api_version = NULL,
  betyurl = NULL,
  fmt = "json",
  key = NULL,
  user = NULL,
  pwd = NULL,
  progress = TRUE,
  ...
)

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

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

betydb_site(
  id,
  api_version = NULL,
  betyurl = NULL,
  fmt = "json",
```

```
  key = NULL,
  user = NULL,
  pwd = NULL,
  progress = TRUE,
  ...
)

betydb_experiment(
  id,
  api_version = NULL,
  betyurl = NULL,
  fmt = "json",
  key = NULL,
  user = NULL,
  pwd = NULL,
  progress = TRUE,
  ...
)
```

Arguments

id	(integer) One or more ids for a species, site, variable, etc.
table	(character) Name of the database table with which this ID is associated.
api_version	(character) Which version of the BETY API should we query? One of "v0" or "beta". Default is `options("betydb_api_version")` if set, otherwise "v0".
betyurl	(string) url to target instance of betydb. Default is `options("betydb_url")` if set, otherwise "https:/www.betydb.org/"
fmt	(character) Format to return data in, one of json, xml, csv. Only json currently supported.
key	(character) An API key. Use this or user/pwd combo. Save in your `.Rprofile` file as `options(betydb_key = "your40digitkey")`. Optional
user, pwd	(character) A user name and password. Use a user/pwd combo or an API key. Save in your `.Rprofile` file as `options(betydb_user = "yournamehere")` and `options(betydb_pwd = "yourpasswordhere")`. Optional
progress	show progress bar? default: `TRUE`
...	Curl options passed on to `GET`. Optional
genus	(character) A genus name. Optional
species	(character) A specific epithet. Optional

Details

BETYdb includes a primary home page (betydb.org) focused on bioenergy crops as well as a network of harmonized databases that support and share data among more focused research programs.

For a list of publicly accessible instances of BETYdb and the urls that can be queried, see https://pecan.gitbooks.io/betydb-documentation/content/distributed_betydb.html

This package queries plant traits, phenotypes, biomass yields, and ecosystem functions. It does not currently interface with the workflow and provenance data that support PEcAn Project (pecanproject.org) and TERRA REF (terraref.org) software.

API documentation: https://pecan.gitbooks.io/betydb-data-access/content/API.html API endpoints are here: https://www.betydb.org/api/docs This package currently uses the original 'v0' API by default. To use a newer version, set `api_version`. Newer versions of the API will support database inserts.

Authentication

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

Functions

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

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

`betydb_search("Search terms", ...)` is a convenience wrapper that passes all further arguments to `betydb_query(table = "search", search = "Search terms", ...)`. See there for details on possible arguments.

References

API documentation https://pecan.gitbooks.io/betydb-data-access/content/API.html and https://www.betydb.org/api/docs

See Also

betydb_query

Examples

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

```
betydb_site(id = 795)

## End(Not run)
```

`betydb_query` *Query a BETY table*

Description

Query a BETY table

Usage

```
betydb_query(
  ...,
  table = "search",
  key = NULL,
  api_version = NULL,
  betyurl = NULL,
  user = NULL,
  pwd = NULL,
  progress = TRUE
)

betydb_search(
  query = "Maple SLA",
  ...,
  include_unchecked = NULL,
  progress = TRUE
)
```

Arguments

`...`	(named character) Columns to query, as key="value" pairs. Note that betydb_query passes these along to BETY with no check whether the requested keys exist in the specified table.
`table`	(character) The name of the database table to query, or "search" (the default) for the traits and yields view
`key`	(character) An API key. Use this or user/pwd combo. Save in your `.Rprofile` file as `options(betydb_key = "your40digitkey")`. Optional
`api_version`	(character) Which version of the BETY API should we query? One of "v0" or "beta". Default is `options("betydb_api_version")` if set, otherwise "v0".
`betyurl`	(string) url to target instance of betydb. Default is `options("betydb_url")` if set, otherwise "https:/www.betydb.org/"
`user, pwd`	(character) A user name and password. Use a user/pwd combo or an API key. Save in your `.Rprofile` file as `options(betydb_user = "yournamehere")` and `options(betydb_pwd = "yourpasswordhere")`. Optional

progress — show progress bar? default: TRUE

query — (character) A string containing one or more words to be queried across all columns of the "search" table.

include_unchecked — (logical) Include results that have not been quality checked? Applies only to tables with a "checked" column: "search", "traits", "yields". Default is to exclude unchecked values.

Details

Use betydb_query to retrieve records from a table that match on all the column filters specified in '...'. If no filters are specified, retrieves the whole table. In API versions that support it (i.e. not in v0), filter strings beginning with "~" are treated as regular expressions.

Value

A data.frame with attributes containing request metadata, or NULL if the query produced no results

Examples

```
## Not run:
# literal vs regular expression vs anchored regular expression:
betydb_query(units = "Mg", table = "variables")
# NULL
betydb_query(units = "Mg/ha", table = "variables") %>% select(name) %>% c()
# $name
# [1] "a_biomass"                  "root_live_biomass"
# [3] "leaf_dead_biomass_in_Mg_ha" "SDM"

betydb_query(genus = "Miscanthus", table = "species") %>% nrow()
# [1] 10
(betydb_query(genus = "~misc", table = "species", api_version = "beta")
 %>% select(genus)
 %>% unique() %>% c())
# $genus
# [1] "Platymiscium" "Miscanthus"   "Dermiscellum"

(betydb_query(genus = "~^misc", table = "species", api_version = "beta")
 %>% select(genus)
 %>% unique() %>% c())
# $genus
# [1] "Miscanthus"

## End(Not run)
```

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

Description

Get bird habitat information from BirdLife/IUCN

Usage

```
birdlife_habitat(id)
```

Arguments

`id`	A single IUCN species ID

Value

a `data.frame` with level 1 and level 2 habitat classes, as well as importance ratings and occurrence type (e.g. breeding or non-breeding). The habitat classification scheme is described at https://www.iucnredlist.org/resources/classification-schemes

Author(s)

David J. Harris <harry491@gmail.com>

See Also

Other birdlife: `birdlife_threats()`

Examples

```
## Not run:
# Setophaga chrysoparia
birdlife_habitat(22721692)
# Passer domesticus
birdlife_habitat(103818789)

## End(Not run)
```

`birdlife_threats` *Get bird threat information from BirdLife/IUCN*

Description

Get bird threat information from BirdLife/IUCN

Usage

```
birdlife_threats(id)
```

Arguments

`id`	A single IUCN species ID

Value

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

Author(s)

David J. Harris <harry491@gmail.com>

See Also

Other birdlife: `birdlife_habitat()`

Examples

```
## Not run:
# Setophaga chrysoparia
birdlife_threats(22721692)
# Aburria aburri
birdlife_threats(22678440)

## End(Not run)
```

leda *Access LEDA trait data*

Description

Access LEDA trait data

Usage

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

Arguments

trait	(character) Trait to get. See Details.
...	Curl options passed on to crul::verb-GET

Details

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

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

Examples

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

# Seed number
leda("seed_number")

# Releasing height
leda(trait = "releasing_height")

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

all <- c("age_first_flowering", "branching", "buds_seasonality",
  "buds_vertical_dist", "canopy_height",
  "dispersal_type", "leaf_distribution", "ldmc_geo", "leaf_mass",
  "leaf_size", "morphology_disperal", "growth_form", "life_span",
  "releasing_height", "seed_longevity", "seed_mass",
  "seed_number", "seed_shape", "shoot_growth_form",
  "snp", "ssd", "tv", "clonal_growth_organs")
out <- list()
for (i in seq_along(all)) {
```

```
  cat(all[i], sep="\n")
  out[[i]] <- leda(all[i])
}
sapply(out, NROW)

## End(Not run)
```

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

Description

Retrieve gene sequences from NCBI by accession number.

Usage

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

Arguments

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

Details

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

Value

data.frame of the form:

- taxon - taxonomic name (may include some junk, but hard to parse off)
- taxonomy - organism lineage
- gene_desc - gene description
- organelle - if mitochondrial or chloroplast
- gi_no - GI number
- acc_no - accession number
- keyword - if official DNA barcode
- specimen_voucher - museum/lab accession number of vouchered material
- lat_lon - longitude/latitude of specimen collection event
- country - country/location of specimen collection event
- paper_title - title of study

- journal - journal study published in (if published)
- first_author - first author of study
- uploaded_date - date sequence was uploaded to GenBank
- length - sequence length
- sequence - sequence character string

Author(s)

Scott Chamberlain, Rupert Collins

See Also

ncbi_searcher(), ncbi_byname()]

Examples

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

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

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

data.frame

Author(s)

Scott Chamberlain

See Also

ncbi_searcher(), ncbi_byid()

Examples

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

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

## End(Not run)
```

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

Description

Search for gene sequences available for taxa from NCBI.

Usage

```
ncbi_searcher(
  taxa = NULL,
  id = NULL,
  seqrange = "1:3000",
  getrelated = FALSE,
  fuzzy = FALSE,
  limit = 500,
  entrez_query = NULL,
  hypothetical = FALSE,
  verbose = TRUE,
  sleep = 0L
)
```

Arguments

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

verbose	(logical) If TRUE (default), informative messages printed.
sleep	(integer) number of seconds to sleep before each HTTP request. use if running to 429 Too Many Requests errors from NCBI. default: 0 (no sleep)

Value

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

Authentication

NCBI rate limits requests. If you set an API key you have a higher rate limit. Set your API key like Sys.setenv(ENTREZ_KEY="yourkey") or you can use ?rentrez::set_entrez_key. set verbose curl output (crul::set_verbose()) to make sure your api key is being sent in the requests

Author(s)

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

See Also

ncbi_byid, ncbi_byname

Examples

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

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

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

```
out2df[grep("12S", out2df$gene_desc, ignore.case=TRUE), ]

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

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

## End(Not run)
```

`plantatt` *PLANTATT plant traits dataset*

Description

PLANTATT plant traits dataset

`taxa_search` *Search for traits by taxa names*

Description

Search for traits by taxa names

Usage

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

Arguments

`x`	(character) Taxonomic name(s) to search for
`db`	(character) only 'ncbi' for now - other options maybe in the future
`...`	Curl options passed on to `GET`

Value

A `data.frame`

Author(s)

Scott Chamberlain

Examples

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

## End(Not run)
```

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

Description

Search for traits from EOL's Traitbank

Usage

```
traitbank(query, key = NULL, ...)
```

Arguments

`query`	(character) a query to the EOL Cypher service that holds Traitbank data. required. no default query given. see examples
`key`	(character) EOL Cypher query API key. required, either passed in or as an environment variable
`...`	Curl options passed on to `verb-GET`

Details

`traitbank` is an interface to the EOL Cypher query. Note that the previous interface EOL had for Traits has been completely replaced - thus, this function is completely different. You no longer query by EOL page id, but using the query language for a database called Neo4J. See the docs for help. Later we plan to make a more user friendly interface to get Traitbank data that doesn't require knowing the Neo4J query syntax

Value

a list

Authentication

You'll need an EOL cypher key to use this function. Steps:

1. Sign in to (register if necessary) your EOL account https://eol.org/users/sign_in.
2. Send an email to the EOL administrator (hammockj AT si.edu) with your username and request that they make you a "power user".
3. Get your key by visiting https://eol.org/services/authenticate

4. Store your key in your .Renviron file or similar under the name "EOL_CYPHER_KEY" Hint: To do this, you can create or edit this file in your home directory, or use the shortcut `usethis::edit_r_environ()` and add a line like `EOL_CYPHER_KEY="your_key_here"`
5. (not recommended alternative): you can pass your key to the key parameter, but we do not recommend doing that as you risk accidentally committing your key to the public web.

References

https://github.com/EOL/eol_website/blob/master/doc/api.md https://github.com/EOL/eol_website/blob/master/doc/query-examples.md

Examples

```
## Not run:
# traitbank_query function
traitbank(query = "MATCH (n:Trait) RETURN n LIMIT 1;")

# traitbank function
res <- traitbank(query = "MATCH (n:Trait) RETURN n LIMIT 2;")
res

## End(Not run)
```

---

traits-defunct *Defunct functions in traits*

Description

These functions have been removed.

Details

- `eol_invasive_`: This function has moved to a new package. See `originr::eol`
- `fe_native`: This function has moved to a new package. See `originr::flora_europaea`
- `g_invasive`: This function has moved to a new package. See `originr::gisd`
- `is_native`: This function has moved to a new package. See `originr::is_native`
- `tr_usda`: the API behind this function is down for good
- `coral_locations`: API down for good, as far as I can tell
- `coral_methodologies`: API down for good, as far as I can tell
- `coral_resources`: API down for good, as far as I can tell
- `coral_species`: API down for good, as far as I can tell
- `coral_taxa`: API down for good, as far as I can tell
- `coral_traits`: API down for good, as far as I can tell

`traits_cache` *Caching*

Description

Manage cached `traits` package files with **hoardr**

Details

The dafault cache directory is `paste0(rappdirs::user_cache_dir(), "/R/traits")`, but you can set your own path using `cache_path_set()`

`cache_delete` only accepts 1 file name, while `cache_delete_all` doesn't accept any names, but deletes all files. For deleting many specific files, use `cache_delete` in a `lapply()` type call

Useful user functions

- `traits_cache$cache_path_get()` get cache path
- `traits_cache$cache_path_set()` set cache path
- `traits_cache$list()` returns a character vector of full path file names
- `traits_cache$files()` returns file objects with metadata
- `traits_cache$details()` returns files with details
- `traits_cache$delete()` delete specific files
- `traits_cache$delete_all()` delete all files, returns nothing

Examples

```
## Not run:
traits_cache

# list files in cache
traits_cache$list()

# delete certain database files
# traits_cache$delete("file path")
# traits_cache$list()

# delete all files in cache
# traits_cache$delete_all()
# traits_cache$list()

# set a different cache path from the default

## End(Not run)
```

`tr_ernest` *Amniote life history dataset*

Description

Amniote life history dataset

Usage

```
tr_ernest(read = TRUE, ...)
```

Arguments

`read`	(logical) read in csv files. Default: `TRUE`
`...`	Curl options passed on to `crul::HttpClient()`

Details

When using this data, cite the paper:

Myhrvold, N. P., Baldridge, E., Chan, B., Sivam, D., Freeman, D. L. and Ernest, S. K. M. (2015), An amniote life-history database to perform comparative analyses with birds, mammals, and reptiles. Ecology, 96: 3109. https://doi.org/10.1890/15-0846R.1

As well as the Dryad data package:

L. Freeman, Daniel; P. Myhrvold, Nathan; Chan, Benjamin; Sivam, Dhileep; Ernest, S. K. Morgan; Baldridge, Elita (2016): Full Archive. figshare. https://doi.org/10.6084/m9.figshare.3563457.v1

Value

paths to the files (character) if `read=FALSE` or a list of data.frame's if `read=TRUE`

References

https://doi.org/10.1890/15-0846R.1 https://doi.org/10.6084/m9.figshare.3563457.v1

Examples

```
## Not run:
res <- tr_ernest()
res$data
res$references
res$sparse
res$range_count

## End(Not run)
```

tr_zanne *Zanne et al. plant dataset*

Description

Zanne et al. plant dataset

Usage

```
tr_zanne(read = TRUE, ...)
```

Arguments

read	(logical) read in csv files. Default: TRUE
...	Curl options passed on to crul::HttpClient()

Details

This data is a dataset stored on Dryad (doi: 10.5061/dryad.63q27). When using this data, cite the paper:

Zanne AE, Tank DC, Cornwell WK, Eastman JM, Smith SA, FitzJohn RG, McGlinn DJ, O'Meara BC, Moles AT, Reich PB, Royer DL, Soltis DE, Stevens PF, Westoby M, Wright IJ, Aarssen L, Bertin RI, Calaminus A, Govaerts R, Hemmings F, Leishman MR, Oleksyn J, Soltis PS, Swenson NG, Warman L, Beaulieu JM, Ordonez A (2014) Three keys to the radiation of angiosperms into freezing environments. Nature 506(7486): 89-92. http://dx.doi.org/10.1038/nature12872

As well as the Dryad data package:

Zanne AE, Tank DC, Cornwell WK, Eastman JM, Smith SA, FitzJohn RG, McGlinn DJ, O'Meara BC, Moles AT, Reich PB, Royer DL, Soltis DE, Stevens PF, Westoby M, Wright IJ, Aarssen L, Bertin RI, Calaminus A, Govaerts R, Hemmings F, Leishman MR, Oleksyn J, Soltis PS, Swenson NG, Warman L, Beaulieu JM, Ordonez A (2013) Data from: Three keys to the radiation of angiosperms into freezing environments. Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.63q27.2

Value

paths to the files (character) if read=FALSE or a list of data.frame's if read=TRUE

References

http://datadryad.org/resource/doi:10.5061/dryad.63q27

Examples

```
## Not run:
res <- tr_zanne()
res$tax_lookup
res$woodiness
res$freezing
```

```
res$leaf_phenology

## End(Not run)
```

Index