Package ‘BOLDconnectR’

September 1, 2024

Title Retrieve, Transform and Analyze the Barcode of Life Data Systems Data in R

Version 1.0.0

Maintainer Sameer Padhye `<spadhye@uoguelph.ca>`

Description Facilitate retrieval, transformation and analysis of the data from the Barcode of Life Data Systems (BOLD) database. This package allows both public and private user data to be easily downloaded easily into the R environment using a variety of inputs such as: processids, sampleids, taxonomy, geography etc. It provides the functionality to convert data into formats compatible with other R-packages and third-party tools, as well as analysis functions for biodiversity, clustering and mapping.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.2.0)

Imports ape,
BAT,
data.table,
dplyr,
ggplot2,
glue,
httr,
jsonlite,
lifecycle,
methods,
reshape2,
rlang,
rnaturalearth,
sf,
skimr,
tidyr,
utils,
vegan

Suggests Biostrings,
BiocManager,
msa

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Contents

`bold.analyze.align` *Transform and align the sequence data retrieved from BOLD (Internal function)*

Description

Function designed to transform and align the sequence data retrieved from the function `bold.fetch`.

Usage

```
bold.analyze.align(
  bold.df,
  marker = NULL,
  align.method,
  seq.name.fields = NULL,
  ...
)
```

Arguments

`bold.df`	A data frame obtained from `bold.fetch()`.
`marker`	A single or multiple character vector specifying the gene marker for which the output is generated. Default is NULL (all data is used).
`align.method`	Character vector specifying the type of multiple sequence alignment algorithm to be used. Default value is ClustalOmega.
`seq.name.fields`	A single or multiple character vector specifying the column headers to be used to name each sequence in the fasta file. Default is NULL in which case, only the processid is used as a name.
`...`	additional arguments that can be passed to msa::msa() function.

Details

`bold.analyze.align` retrieves the sequence information obtained using `bold.fetch` function and performs a multiple sequence alignment. Type of clustering method can be specified using the `align.method` argument (Default value is ClustalOmega). It utilizes the msa::msa() function with default settings but additional arguments can be passed via the `...` argument. Marker name provided must match with the standard marker names available on the BOLD webpage. Name for individual sequences in the output can be customized by using the `seq.name.fields` argument. If more than one field is specified, the name will follow the sequence of the fields given in the vector. Performing a multiple sequence alignment on large sequence data might slow the system. Additionally, users are responsible for verifying the sequence quality and integrity, as the function does not provide any checks on issues like STOP codons and indels within the data. The output of this function is a modified Barcode Core Data Model (BCDM) dataframe, which includes two additional columns: one for the aligned sequences and another for the names given to the sequences.

Note: This function is currently an internal function with documentation and can be accessed only by using the `:::` operator (`BOLDconnectR:::bold.analyze.align`). Users are required to install and load the `Biostrings` and `msa` packages using `BiocManager` (Please see the examples for the usage) before running this function that are maintained by `Bioconductor`.

Value

An 'output' list containing:

- bold.df.mod = A modified BCDM data frame with two additional columns ('aligned_seq' and 'msa.seq.name').

Examples

```
## Not run:
# Search for ids
seq.data.ids <- bold.public.search(taxonomy = c("Oreochromis tanganicae", "Oreochromis karongae"))

# Fetch the data using the ids
# api_key must be obtained from BOLD support before usage
seq.data<-bold.fetch(param.data = seq.data.ids,
query.param = "processid",
param.index = 1,
api_key=apikey)

# msa is required for this function to run. msa is installed using BiocManager
if (!requireNamespace("BiocManager", quietly=TRUE))
install.packages("BiocManager")
BiocManager::install("msa")
BiocManager::install("Biostrings")
library(msa)
library(Biostrings)

# Align the data (using  bin_uri as the name for each sequence)
seq.align <- BOLDconnectR:::bold.analyze.align(seq.data,
seq.name.fields = c("bin_uri"),
align.method="ClustalOmega")

# Dataframe of the sequences (aligned) with their corresponding names
 head(seq.align)

## End(Not run)
```

---

bold.analyze.diversity *Create an biodiversity profile of the retrieved data*

Description

This function creates a biodiversity profile of the downloaded data using bold.fetch().

Usage

```
bold.analyze.diversity(
  bold.df,
  taxon.rank,
  taxon.name = NULL,
  site.cat = NULL,
  grids.cat = FALSE,
  gridsize = NULL,
  presence.absence = FALSE,
  richness.res = FALSE,
  rich.plot.curve = FALSE,
  rich.curve.estimatr = NULL,
  rich.curve.x.axis = NULL,
  shannon.res = FALSE,
  preston.res = FALSE,
  pres.plot.y.label = NULL,
  beta.res = FALSE,
  beta.index = NULL
)
```

Arguments

`bold.df`	A data frame obtained from bold.fetch().
`taxon.rank`	A single character value specifying the taxonomic hierarchical rank. Needs to be provided by default.
`taxon.name`	A single or multiple character vector specifying the taxonomic names associated with the 'taxon.rank'. Default value is NULL.
`site.cat`	A single character vector specifying the geographic category for which a community matrix should be created. Default value is NULL.
`grids.cat`	A logical value specifying Whether the community matrix should be based on grids as 'sites'. Default value is NULL.
`gridsize`	A numeric value of the size of the grid if the grids=TRUE; Size is in sq.m. Default value is NULL.
`presence.absence`	A logical value specifying whether the generated matrix should be converted into a 'presence-absence' matrix.
`richness.res`	A logical value specifying whether richness results should be calculated. Default value is FALSE.

`rich.plot.curve`	A logical value specifying whether an accumulation curve should be plotted. Default value is FALSE.
`rich.curve.estimatr`	A character value specifying which index should be used. Default value is NULL.
`rich.curve.x.axis`	A character value specifying whether 'samples' or 'individuals' should be used against the `rich.curve.estimatr`. Default value is NULL.
`shannon.res`	A logical value specifying whether the Shannon diversity results should be generated. Default value is FALSE.
`preston.res`	A logical value specifying whether the Preston results should be generated. Default value is FALSE.
`pres.plot.y.label`	A character value specifying the taxonomic category (`taxon.rank`) which was used to generate the matrix.
`beta.res`	A logical value specifying whether beta diversity results should be calculated. Default value is FALSE.
`beta.index`	A character vector specifying the type of beta diversity index ('jaccard' or 'sorensen' available).

Details

`bold.analyze.diversity` estimates the richness, Shannon diversity and beta diversity from the BIN counts or presence-absence data. Internally, the function converts the downloaded BCDM data into a community matrix (site X species) which is also generated as a part of the output. grids.cat converts the Coordinate Reference System (CRS) of the data to a 'Mollweide' projection by which distance-based grid can be correctly specified (Gott III et al. 2007).Each grid is assigned a cell id, with the lowest number given to the lowest latitudinal point in the dataset. The community matrix generated by the function is used to create richness profiles using `BAT::alpha.accum()` and Preston and Shannon diversity analyses using `vegan::prestondistr()` and `vegan::diversity()` respectively. The `BAT::alpha.accum()` currently offers various richness estimators, including Observed diversity (Obs); Singletons (S1); Doubletons (S2); Uniques (Q1); Duplicates (Q2); Jackknife1 abundance (Jack1ab); Jackknife1 incidence (Jack1in); Jackknife2 abundance (Jack2ab); Jackknife2 incidence (Jack2in); Chao1 and Chao2. The results depend on the input data (true abundances vs counts vs incidences) and users should be careful in the subsequent interpretation. Preston plots are generated using the data from the `prestondistr` results in `ggplot2` featuring cyan bars for observed species (or equivalent taxonomic group) and orange dots for expected counts. The `presence.absence` argument converts the counts (or abundances) to 1s and 0s. This dataset can then be directly used as input data for biodiversity analysis functions from packages like vegan. Beta diversity values are calculated using `BAT::beta()` function, which partitions the data using the Podani & Schmera (2011)/Carvalho et al. (2012) approach partitioning the beta diversity into 'species replacement' and 'richness difference' components. These results are stored as distance matrices in the output. *Note on the community matrix*: Each cell in this matrix contains the counts (or abundances) of the specimens whose sequences have an assigned BIN, in a given a site category (`site.cat`) or a grid (`grids.cat`). These counts can be generated at any taxonomic hierarchical level, applicable to one or multiple taxa including 'bin_uri'. The `site.cat` can refer to any geographic field, and metadata on these fields can be checked using the `bold.fields.info()`. If, `grids.cat` = TRUE, grids are generated based on BIN occurrence data (latitude, longitude) with grid size determined by the user in square meters using the `gridsize` argument. Rows lacking latitude and longitude data are removed, while NULL entries for site.cat are allowed if they have a latitude and longitude value. This is because grids are drawn based on the bounding boxes which only use latitude and longitude values *Important Note*: Results, including species counts, adapt based on taxon.rank argument although the output label remains ‘species’ in some instances (`preston.res`).

Value

An ’output’ list containing containing:

- comm.matrix = site X species like matrix required for the biodiversity results
- richness = A richness profile matrix
- Shannon_div = Shannon diversity values for the given sites/grids (from gen.comm.mat)
- richness_plot = A ggplot2 visualization of the richness curve
- preston.res = a Preston plot numerical data output
- preston.plot = a ggplot2 visualization of the preston.plot
- total.beta = beta.total
- replace = beta.replace (replacement)
- richnessd = beta.richnessd (richness difference)

References

Carvalho, J.C., Cardoso, P. & Gomes, P. (2012) Determining the relative roles of species replacement and species richness differences in generating beta-diversity patterns. Global Ecology and Biogeography, 21, 760-771.

Podani, J. & Schmera, D. (2011) A new conceptual and methodological framework for exploring and explaining pattern in presence-absence data. Oikos, 120, 1625-1638.

Richard Gott III, J., Mugnolo, C., & Colley, W. N. (2007). Map projections minimizing distance errors. Cartographica: The International Journal for Geographic Information and Geovisualization, 42(3), 219-234.

Examples

```
## Not run:
# Search for ids
comm.mat.data <- bold.public.search(taxonomy = "Poecilia")

# Fetch the data using the ids
# api_key must be obtained from BOLD support before usage.
BCDMdata <- bold.fetch(param.data = comm.mat.data,
query.param = "processid",
param.index = 1,
api_key = apikey)

# Remove rows which have no species data
BCDMdata <- BCDMdata[!BCDMdata$species== "",]

#1. Analyze richness data
res.rich <- bold.analyze.diversity(BCDMdata,
taxon.rank = "species",
site.cat = 'country.ocean',
richness.res = TRUE)

# Community matrix (BCDM data converted to community matrix)
res.rich$comm.matrix
```

```
# richness results
res.rich$richness

#2. Shannon diversity
res.shannon <- bold.analyze.diversity(BCDMdata,
taxon.rank = "species",
site.cat = 'country.ocean',
shannon.res = TRUE)

# Shannon diversity results
res.shannon

#3. Preston plots and results
pres.res <- bold.analyze.diversity(BCDMdata,
taxon.rank = "species",
site.cat = 'country.ocean',
preston.res = TRUE)

# Preston plot
pres.res$preston.plot

# Preston plot data
pres.res$preston.res

#4. beta diversity
beta.res <- bold.analyze.diversity(BCDMdata,
taxon.rank = "species",
site.cat = 'country.ocean',
beta.res = TRUE,
beta.index = "jaccard")

#Total diversity
beta.res$total.beta

#Replacement
beta.res$replace

#Richness difference
beta.res$richnessd

## End(Not run)
```

`bold.analyze.map` *Visualize BIN occurrence data on maps*

Description

This function creates basic maps of BIN occurrence data at different scales.

Usage

```
bold.analyze.map(bold.df, country = NULL, bbox = NULL)
```

Arguments

`bold.df`	The data.frame retrieved from `bold.fetch()`.
`country`	A single or multiple character vector of country names. Default value is NULL.
`bbox`	A numeric vector specifying the min, max values of the latitude and longitude. Default value is NULL.

Details

`bold.analyze.map` extracts out the geographic information from the `bold.fetch()` output. Data points having NA values for either latitude or longitude or both are removed. Latitude and longitude values are in ‘decimal degrees’ format with a ’WGS84’ Coordinate Reference System (CRS) projection. Default view includes data mapped onto a world shape file using the `rnaturalearth::ne_countries()` at a 110 scale (low resolution). If the country is specified (single or multiple values), the function will specifically plot the occurrences on the specified country. Alternatively, a bounding box (bbox) can be defined for a specific region to be visualized. The function also provides a sf data frame of the GIS data which can be used for any other application/s.

Value

An ’output’ list containing:

- geo.df = A simple features (sf) ‘data.frame’ containing the geographic data.
- plot = A visualization of the occurrences.

Examples

```
## Not run:
# Download the ids
geo.data.ids <- bold.public.search(taxonomy = "Musca domestica")

# Fetch the data using the ids.
# api_key must be obtained from BOLD support before usage.

geo.data <- bold.fetch(param.data = geo.data.ids,
query.param = "processid",
param.index = 1, api_key = apikey)

# All data plotted.
geo.viz <- bold.analyze.map(geo.data)

# Data plotted only in one country
geo.viz <- bold.analyze.map(geo.data,
country = c("Saudi Arabia"))

# The sf dataframe of the downloaded data
geo.viz$geo.df

## End(Not run)
```

`bold.analyze.tree` *Analyze and visualize the multiple sequence alignment*

Description

Calculates genetic distances and performs a Neighbor Joining tree estimation of the multiple sequence alignment output obtained from `bold.analyze.align`.

Usage

```
bold.analyze.tree(
  bold.df,
  dist.model,
  clus.method = c("nj", "njs"),
  dist.matrix = FALSE,
  newick.tree.export = FALSE,
  newick.file.path = NULL,
  newick.file.name = NULL,
  tree.plot = FALSE,
  tree.plot.type,
  ...
)
```

Arguments

`bold.df`	A modified BCDM data frame obtained from `bold.analyze.align()`.
`dist.model`	A character string specifying the model to generate the distances.
`clus.method`	A character vector specifying either `ape::nj()` (neighbour joining) or `ape::njs()` (neighbour joining with NAs) clustering algorithm.
`dist.matrix`	A logical value specifying whether the distance matrix should be saved in the output. Default is FALSE.
`newick.tree.export`	A logical value specifying whether newick tree should be generated and exported. Default value is FALSE.
`newick.file.path`	A character value specifying the folder path where the file should be saved.
`newick.file.name`	A character value specifying the name of the exported file.
`tree.plot`	Logical value specifying if a neighbour joining plot should be generated. Default value is FALSE.
`tree.plot.type`	The layout of the tree. Based on `ape::plot.phylo()` type.
`...`	additional arguments from `ape::dist.dna()`

Details

`bold.analyze.tree` analyzes the multiple sequence alignment output of the `bold.analyze.align` function to generate a distance matrix using the models available in the `ape::dist.dna()`. Setting `dist.matrix`= TRUE will store the underlying distance matrix in the output; however, the default value for the argument is deliberately kept at FALSE to avoid potential memory issues with large data. Additional arguments for calculating distances can passed using the argument .... Setting `tree.plot=` TRUE generates a basic visualization of the Neighbor Joining (NJ) tree using the distance matrix from `ape::dist.dna()` and the `ape::plot.phylo()` function. `tree.plot.type` specifies the type of tree and has the following options ("phylogram", "cladogram", "fan", "unrooted", "radial", "tidy" based on `type` argument of `ape::plot.phylo()`;The first alphabet can be used instead of the whole word). Both `ape::nj()` and `ape::njs()` are available for generating the tree. Additionally, the function provides base frequencies and offers an option to export the trees in a Newick format by specifying the name and path for output file.

Value

An 'output' list containing:

- dist_mat = A distance matrix based on the model selected if dist.matrix=TRUE.
- base_freq = Overall base frequencies of the align.seq result.
- plot = Neighbor Joining clustering visualization (if tree.plot=TRUE).
- data_for_plot = A phylo object used for the plot.
- NJ/NJS tree in a newick format (only if newick.tree.export=TRUE).

Examples

```
## Not run:
library(msa)
library(Biostrings)

# Download the data ids
seq.data.ids <- bold.public.search(taxonomy = c("Eulimnadia"), filt.marker = "COI-5P")

# Fetch the data using the ids.
# api_key must be obtained from BOLD support before usage.

seq.data <- bold.fetch(param.data = seq.data.ids, query.param = "processid",
                       param.index = 1, api_key = apikey)

# Remove rows without species name information
seq <- seq.data[seq.data$species!="", ]

# Align the data
# Users need to install and load packages `msa` and `Biostrings`.
seq.align<-BOLDconnectR:::bold.analyze.align(seq.data,
                                             seq.name.fields = c("species","bin_uri"),
                                             marker="COI-5P",
                                             align.method="ClustalOmega")

#Analyze the data to get a tree
seq.analysis<-bold.analyze.tree(seq.align,
                                dist.model = "K80",
                                clus="nj",
                                tree.plot.type='p',
                                tree.plot=TRUE,
                                dist.matrix = T)

# Output
# A 'phylo' object of the plot
seq.analysis$data_for_plot
```

```
# A distance matrix based on the distance model selected
seq.analysis$dist_matrix
# Base frequencies of the sequences
seq.analysis$base_freq

## End(Not run)
```

`bold.data.summarize` *Generate a summary of the data downloaded from BOLD*

Description

The function is used to obtain a detailed summary of the data obtained by `bold.fetch` function.

Usage

```
bold.data.summarize(bold.df, cols = NULL)
```

Arguments

`bold.df`	the data.frame retrieved from the `bold.fetch` function.
`cols`	A single or multiple character vector specifying the columns for which a data summary is sought. Default value is NULL.

Details

`bold.data.summarize` provides summaries for each data type available in the downloaded dataset. The function uses the `skimr::skim()` function to generate a list of data frames which are then separated by data type using `skimr::partition()` to facilitate streamlined export. The summary includes counts for NULL values, unique values and the proportion of complete cases. The cols argument allows users to select specific fields for summarization; by default, it is set to NULL, meaning all columns are summarized. Summaries are printed to the console and can also be saved. Please note that if the fields argument from bold.fetch has been used to filter for specific columns, and only those will be summarized by default. For specific details on the skim output, refers to the `skimr` package documentation.

Value

A list of data frames. Each data frame is a data summary of a specific data type.

Examples

```
## Not run:
# Download data
bold_data.ids <- bold.public.search(taxonomy = "Oreochromis")

bold.data <- bold.fetch(bold_data.ids,
query.param = "processid",
param.index = 1,
api_key = apikey)

# Generate summary for specific fields (cols)
```

```
test.data.summary <- bold.data.summarize(bold.data,
                              cols = c("country.ocean", "nuc_basecount", "inst", "elev"))

# Character data fields summary
test.data.summary$character

# Numerical data fields summary
test.data.summary$numeric

## End(Not run)
```

bold.export *Export files generated by BOLDconnectR*

Description

The function is used to export some of the output data generated by BOLDconnectR

Usage

```
bold.export(
  bold.df,
  export = c("mod.df", "msa.fas", "fas"),
  fas.seq.name.fields = NULL,
  df.export.file.type = NULL,
  export.file.path = NULL,
  export.file.name = NULL
)
```

Arguments

`bold.df`	The data.frame either retrieved from `bold.fetch()`,`bold.analyze.align` or a user modified BCDM dataset.
`export`	A character input specifying the type of output required. Should be either of "mod.df","msa.fas" or "fas".
`fas.seq.name.fields`	A single or multiple character vector indicating the column headers that should be used to name each sequence for the unaligned FASTA file. Default is NULL; in this case, only the processid is used as the name.
`df.export.file.type`	A character value specifying the type of file to be exported for the modified dataframe. Currently '.csv' and '.tsv' options are available.
`export.file.path`	A character value specifying the folder path where the file should be saved.
`export.file.name`	A character value specifying the name of the exported file.

Details

`bold.export` offers an export option for some of the sequence-based outputs obtained from functions within the BOLDconnectR package. Sequence information downloaded using `bold.fetch()` or the aligned sequences obtained using `bold.analyze.align` can be exported as a FASTA file for any third party tool (via `export=`'fas' or 'msa.fas'). Data fetched by `bold.fetch()` can be directly used to export the unaligned FASTA file, while the modified dataframe from `bold.analyze.align` is required for exporting the multiple sequence alignment. The name for individual sequences in the unaligned FASTA file output can be customized by using the `fas.seq.name.fields` argument. If more than one field is specified, the name will follow the sequence of the fields given in the vector. The multiple sequence aligned FASTA file uses the same name provided by the user in the `bold.analyze.align` function. Additionally, this function allows for the export of user-edited data (in taxonomy, geography etc.) as a csv/tsv file while retaining its BCDM format. This functionality is developed with the future potential of uploading data to BOLD using the package. Edits to the BCDM data can be made using any other R packages so long as it maintains the BCDM format.

Value

It exports a .fas or a csv/tsv file based on the export argument.

Examples

```
## Not run:
library(msa)
library(Biostrings)

 # Download data
data_for_export_ids <- bold.public.search(taxonomy = "Poecilia reticulata",
filt.basecount= c(500,600),
filt.marker = "COI-5P")

# Fetching the data using the ids
data_for_export <- bold.fetch(data_for_export_ids,
query.param = "processid",
param.index = 1,
apikey= apikey)

# Align the data (using processid and bin_uri as fields for sequence names)
# Users need to install and load packages `msa` and `Biostrings`.

seq.align<-BOLDconnectR:::bold.analyze.align(data_for_export,
                                             seq.name.fields = c("processid","bin_uri"))
# Export the fasta file (unaligned)
# Note that input data is the original BCDM data retrieved using bold.fetch
bold.export(data_for_export, export= "fas",
            fas.seq.name.fields = c("species","bin_uri","processid"),
            export.file.path = "file_path",
            export.file.name = "file_name")

# Export the multiple sequence alignment
# Note the input data here is the modified BCDM data after using bold.analyze.align
bold.export(seq.align,export = "msa.fas",
fas.seq.name.fields = ("species","bin_uri","processid"),
export.file.path = "file_path",export.file.name = "file_name")

## End(Not run)
```

bold.fetch — *Retrieve all data from the BOLD database*

Description

Retrieves public and private user data based on different parameters (processid, sampleid, dataset codes & bin_uri) input.

Usage

```
bold.fetch(
  param.data,
  query.param,
  param.index,
  api_key,
  filt.taxonomy = NULL,
  filt.geography = NULL,
  filt.latitude = NULL,
  filt.longitude = NULL,
  filt.shapefile = NULL,
  filt.institutes = NULL,
  filt.identified.by = NULL,
  filt.seq.source = NULL,
  filt.marker = NULL,
  filt.collection.period = NULL,
  filt.basecount = NULL,
  filt.altitude = NULL,
  filt.depth = NULL,
  filt.fields = NULL,
  export = FALSE,
  file.type = NULL,
  file.path = NULL,
  file.name = NULL
)
```

Arguments

`param.data`	A file path pointing to either a csv/tsv/txt file with the ids or a data frame where ids are stored.
`query.param`	The parameter on which the data should be fetched. “processid”, “sampleid”, "bin_uri" or "dataset_codes".
`param.index`	A number indicating the column index (position) of the `query.params` in the dataset.
`api_key`	A character string required for authentication and data access.
`filt.taxonomy`	A single or multiple character vector of taxonomic names at any hierarchical level. Default value is NULL.
`filt.geography`	A single or multiple character vector specifying any of the country/province/state/region/sector/site names/codes. Default value is NULL.

`filt.latitude`	A single or a vector of two numbers specifying the latitudinal range in decimal degrees. Values should be separated by a comma. Default value is NULL.
`filt.longitude`	A single or a vector of two numbers specifying the longitudinal range in decimal degrees. Values should be separated by a comma. Default value is NULL.
`filt.shapefile`	A file path pointing to a shapefile or name of the shapefile (.shp) imported in the R session. Default value is NULL.
`filt.institutes`	A single or multiple character vector specifying names of institutes. Default value is NULL.
`filt.identified.by`	A single or multiple character vector specifying names of people responsible for identifying the organism. Default value is NULL.
`filt.seq.source`	A single or multiple character vector specifying the data portals from where the (sequence) data was mined. Default value is NULL.
`filt.marker`	A single or multiple character vector specifying of gene names. Default value is NULL.
`filt.collection.period`	A single or a vector of two date values specifying the collection period range (start, end). Values should be separated by a comma. Default value is NULL.
`filt.basecount`	A single or a vector of two numbers specifying range of basepairs number. Values should be separated by a comma. Default value is NULL.
`filt.altitude`	A single or a vector of two numbers specifying the altitude range in meters. Values should be separated by a comma. Default value is NULL.
`filt.depth`	A single or a vector of two numbers specifying the depth range. Values should be separated by a comma. Default value is NULL.
`filt.fields`	A single or multiple character vector specifying columns needed in the final dataframe. Default value is NULL.
`export`	A logical value specifying whether the output should be exported locally. Default value is FALSE.
`file.type`	A character value specifying the type of file to be exported. Currently '.csv' and '.tsv' options are available.
`file.path`	A character value specifying the folder path where the file should be saved.
`file.name`	A character value specifying the name of the exported file.

Details

`bold.fetch` retrieves both public as well as private user data, where private data refers to data that the user has permission to access. The data is downloaded in the Barcode Core Data Model (BCDM) format. It supports effective download data in bulk using search parameters `query.params` such as 'processids', 'sampleids', 'bin_uri' and 'dataset codes'. Data input can be either through a path to a flat file with extensions like `.csv/.tsv/.txt` or a R `data.frame` object. The import process assumes that the input data includes a header. Users must specify only one of the `query.params` at a time for retrieval. Multi-parameter searches combining fields like 'processids'+ 'sampleids' + 'bin_uri' are not supported, regardless of the parameters available. The `filt.` or filter parameter arguments provide further data sorting by which a specific user defined data can be obtained. Note that any/all `filt.` argument names must be written explicitly to avoid any errors (Ex. `filt.institutes` = 'CBG' instead of just 'CBG'). Using the `filt.fields` argument allows users to select specific columns for inclusion in the final data frame, though, processids and sampleids, are included by default. If this argument is left as NULL all columns will be downloaded. There is no upper limit to the volume of data that can be retrieved, however, this depends on the user's internet connection and computer specifications. The `api_key` is a UUID v4 hexadecimal string obtained upon request from BOLD at support@boldsystems.org and is valid for one year, requiring renewal thereafter. The names of the columns in the downloaded data correspond to those specified in bold.fields.info. It is important to correctly match the `query.param` and `param.index` to avoid getting any errors. Note that some values or fields might currently be unavailable but may be accessible future.

Value

A data frame containing all the information related to the processids/sampleids and the filters applied (if/any).

Examples

```
## Not run:
data(test.data)

# key' would the 'api_key' provided to the user

#With processids ('processid' param is the first column in the data (param.index=1))
res <- bold.fetch(param.data = test.data,
query.param = 'processid',
param.index = 1,
api_key = "key")


#With sampleids ('sampleid' param is the second column in the data (param.index=2))
res<-bold.fetch(param.data = test.data,
query.param = 'sampleid',
param.index = 2,
api_key = "key")

## Using filters

#Geography
res <- bold.fetch(param.data = test.data,
query.param = 'processid',
param.index = 1,
api_key = "key",
filt.geography = "Churchill")

#Sequence length
res <- bold.fetch(param.data = test.data,
query.param = 'processid',
param.index = 1,
api_key  =  "key",
filt.basecount = c(500,600))

#Gene marker & sequence length
res<-bold.fetch(param.data = test.data,
query.param = 'processid',
param.index = 1,
api_key  =  "key",
filt.marker = "COI-5P",
filt.basecount = c(500, 600))
```

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

`bold.fields.info` *Retrieve metadata of the BOLD data fields*

Description

Provides information on the field (column) names and their respective data type, all of which are compliant with the Barcode Core Data Model (BCDM), the latest data model of the BOLD database.

Usage

```
bold.fields.info(print.output = FALSE)
```

Arguments

`print.output`	Whether the output should be printed in the console. Default is FALSE.

Details

The function downloads the latest field (column) meta data (file type and brief description) which is currently available for download from BOLD.print,output = TRUE will print the information in the console.

Value

A data frame containing information on all fields (columns).

Examples

```
bold.field.data<-bold.fields.info()
head(bold.field.data,10)
```

`bold.public.search` *Search publicly available data on the BOLD database*

Description

Retrieves record ids for publicly available data based on taxonomy, geography or ids (dataset codes & bin_uri) search.

Usage

```
bold.public.search(
  taxonomy = NULL,
  geography = NULL,
  bins = NULL,
  datasets = NULL,
  filt.latitude = NULL,
  filt.longitude = NULL,
  filt.shapefile = NULL,
  filt.institutes = NULL,
  filt.identified.by = NULL,
  filt.seq.source = NULL,
  filt.marker = NULL,
  filt.collection.period = NULL,
  filt.basecount = NULL,
  filt.altitude = NULL,
  filt.depth = NULL
)
```

Arguments

`taxonomy`	A single or multiple character vector specifying the taxonomic names at any hierarchical level. Default value is NULL.
`geography`	A single or multiple character vector specifying any of the country/province/state/region/sector/site names/codes. Default value is NULL.
`bins`	A single or multiple character vector specifying the BIN ids. Default value is NULL.
`datasets`	A single or multiple character vector specifying dataset codes. Default value is NULL.
`filt.latitude`	A single or a vector of two numbers specifying the latitudinal range in decimal degrees. Values should be separated by a comma. Default value is NULL.
`filt.longitude`	A single or a vector of two numbers specifying the longitudinal range in decimal degrees. Values should be separated by a comma. Default value is NULL.
`filt.shapefile`	A file path pointing to a shapefile or name of the shapefile (.shp) imported in the R session. Default value is NULL.
`filt.institutes`	A single or multiple character vector specifying names of institutes. Default value is NULL.
`filt.identified.by`	A single or multiple character vector specifying names of people responsible for identifying the organism. Default value is NULL.
`filt.seq.source`	A single or multiple character vector specifying the data portals from where the (sequence) data was mined. Default value is NULL.
`filt.marker`	A single or multiple character vector specifying of gene names. Default value is NULL.
`filt.collection.period`	A single or a vector of two date values specifying the collection period range (start, end). Values should be separated by a comma. Default value is NULL.

`filt.basecount`	A single or a vector of two numbers specifying range of basepairs number. Values should be separated by a comma. Default value is NULL.
`filt.altitude`	A single or a vector of two numbers specifying the altitude range in meters. Values should be separated by a comma. Default value is NULL.
`filt.depth`	A single or a vector of two numbers specifying the depth range. Values should be separated by a comma. Default value is NULL.

Details

`bold.public.search` searches publicly available data on BOLD, retrieving associated proccessids and sampleids, which can then be accessed using `bold.fetch`. Search parameters can include one or a combination of taxonomy, geography, bin uri or dataset codes. While there is no limit on the amount of ID data that can be downloaded, complex combinations of the search parameters may exceed the predetermined weburl character length (2048 characters). Searches using a single parameter are not subject to this limit. For multiparameter searches (e.g. taxonomy + geography + bins/datasets; see the example: Taxonomy + Geography + BIN id), it's crucial that the parameters are logically combined to ensure accurate and non-empty results. Downloaded IDs can be filtered further on one or a combination of arguments (such as institutes, identifiers, altitude, depth etc.). It is essential to explicitly name the filter arguments (eg. `filt.institutes` = 'CBG' instead of just 'CBG') to avoid any errors.

Value

A data frame containing all the processids and sampleids related to the query search.

Examples

```
## Not run:
# Taxonomy
bold.data <- bold.public.search(taxonomy = "Panthera leo")
head(bold.data,10)

# Taxonomy and Geography
bold.data.taxo.geo <- bold.public.search(taxonomy = "Panthera uncia",
geography = "India")
head(bold.data.taxo.geo,10)

# Taxonomy, Geography and BINs
bold.data.taxo.geo.bin <- bold.public.search("Panthera leo",
geography = "India",
bins=c("BOLD:AAD6819"))

bold.data.taxo.geo.bin

## End(Not run)
```

`test.data` *Canadian spider data by Blagoev et al.(2015)*

Description

The test data comprises 1,336 process and sample IDs from the Salticidae (Arthropoda:Arachnida:Araneae) family, sourced from Canadian spider data published by Blagoev et al. (2015). This publication includes a DNA barcode reference library encompassing 1,018 species of Canadian spiders.

Usage

```
test.data
```

Format

A data frame with 1336 rows and 2 columns:

processid Character vector of processids

sampleid Character vector of sampleids corresponding to the processids

Source

https://onlinelibrary.wiley.com/doi/full/10.1111/1755-0998.12444

References

Blagoev, G. A., Dewaard, J. R., Ratnasingham, S., Dewaard, S. L., Lu, L., Robertson, J., ... & Hebert, P. D. (2016). Untangling taxonomy: a DNA barcode reference library for C anadian spiders. Molecular Ecology Resources, 16(1), 325-341.

Index