Package 'BOLDconnectR'

August 15, 2024

Type Package

Title Retrieve, transform and analyze the Barcode of Life Data Systems data in R

Version 0.1.0

Author Sameer M. Padhye, Claudia-Liliana Ballesteros Mejia, Sujeevan Ratnasingham

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

Description 'BOLDconnectR' is a package designed for the for data retrieval, transformation and analysis of the data available on the Barcode of Life Data Systems (BOLD) database.
Both publicly available as well as private user data can be downloaded easily into the R environment based on variety of inputs including processids, sampleids, taxonomy, geography etc. Data transformation functions convert the imported data into objects and/or files commonly used in other R packages and third party tools while data analysis functions offer basic biodiversity, clustering and mapping analyses for the downloaded data.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.2.0)

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

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

R topics documented:

align.seq . . . 2
analyze.alphadiv . . . 4
analyze.betadiv . . . 6
analyze.seq . . . 7
bold.connectr . . . 9
bold.connectr.filters . . . 11
bold.connectr.public . . . 13
bold.fields.info . . . 15
bold.multirecords.set.csv . . . 16
data.summary . . . 16
fetch.bold.bins . . . 17
fetch.bold.datasets . . . 17
fetch.bold.processid . . . 18
fetch.bold.sampleid . . . 18
fetch.data . . . 18
fetch.public.data . . . 19
gen.comm.mat . . . 19
generate.batches . . . 21
get_data_bins . . . 21
get_data_datasets . . . 22
get_data_processid . . . 22
globals . . . 22
id.files . . . 23
post.api.res.fetch . . . 23
reassign.data.type . . . 23
test.data . . . 24
visualize.geo . . . 24

Index **26**

`align.seq` *Transform and align the sequence data retrieved from BOLD*

Description

Transforms and aligns the data retrieved from the `bold.connectr` and `bold.connectr.public` functions for various downstream analyses

Usage

```
align.seq(
  bold.df,
  marker = NULL,
  name.fields = NULL,
  file.path = NULL,
  file.name = NULL,
```

```
  raw.fas = FALSE
)
```

Arguments

bold.df	A data frame obtained from bold.connectr() or bold.connectr.public().
marker	A single or multiple character vector specifying the gene marker for which the output is generated. Default is NULL (all data is used).
name.fields	A single or multiple character vector specifying column headers which should be used to name each sequence in the fasta file. Default is NULL in which case, only the BIN id is used as a name.
file.path	A character value specifying the folder path where the file should be saved.Default value is NULL.
file.name	A character value specifying the name of the exported file.Default value is NULL.
raw.fas	A logical input to specify whether a unaligned(raw) 'fasta' file should be created. Default value is FALSE.

Details

'align.seq' fetches the sequence information obtained using the connectr functions and performs a ClustalOmega multiple sequence alignment on it. This is done using msa::msa() function with method = "ClustalOmega" & default settings. In addition, the function also provides a a) 'ape' 'DNAbin' object , b) a data frame of the sequence data and the respective names and c) a raw (unaligned.fas) 'fasta' file. File path and file name need to be provided for if raw.fas=TRUE. 'marker' name provided must match with the standard marker names available in BOLD. Name for individual sequences in the output can be customized by using the names.field argument. If more than one field is specified, the name will follow the sequence of the fields given in the vector. Please note that a multiple sequence alignment on large sequence data might slow the machine.Also note that the function does not detect any STOP codons and indels in the data.

Value

A list containing:

- 'Biostrings' DNAStringSet object of the multiple sequence alignment
- 'ape': a 'DNAbin' object
- seq.df: a data frame with sequences as one column and its name in the other (unaligned)
- raw.fas = TRUE: a '.fas' file of unaligned sequences

Examples

```
# Download the data
seq<-bold.connectr.public(taxonomy = c("Oreochromis tanganicae","Oreochromis karongae"))

# Align the data (using species", bin_uri & country.ocean as a composite name for each sequence)
seq.align<-align.seq(seq,name.fields = c("species","bin_uri"),marker="COI-5P")

# Dataframe of the sequences (not aligned) with their corresponding names
head(seq.align$seq.df)

#ape DNAbin object
seq.align$ape_obj
```

```
#A DNAStringSet object of the Multiple sequence alignment
seq.align$msa.result
```

`analyze.alphadiv` *Create an alpha diversity profile of the retrieved data*

Description

This function analyzes the output from the `gen.comm.mat` to provide a richness and alpha diversity profile of the downloaded data.

Usage

```
analyze.alphadiv(
  bin.comm,
  plot.curve = FALSE,
  curve.index = NULL,
  curve.xval = NULL,
  preston.res = FALSE,
  pres.plot.y.label = NULL
)
```

Arguments

`bin.comm`	the site X species like output from the `gen.comm.mat` function.
`plot.curve`	A logical value specifying whether an accumulation curve should be plotted.Default value is FALSE.
`curve.index`	A character value specifying which index should be used for the `curve.index` argument. Default value is NULL.
`curve.xval`	A character value specifying whether sample or individuals should be used against the `curve.index`.Default value is NULL.
`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` in `gen.comm.mat()`) which was used to generate the matrix.

Details

`analyze.alphadiv` estimates the richness and calculates the shannon diversity values using the `gen.comm.mat()` output. The estimations are based on BIN counts or presence absence data at the taxonomic level specified by the user in the `gen.comm.mat` function. The function also generates Preston plots and the associated numerical results. The richness profile is created using `BAT::alpha.accum()` while the preston and shannon diversity results are obtained using the `vegan::prestondistr()` and `vegan::diversity()` functions respectively. Preston plots are created using the data from the `prestondistr` results in ggplot2.The cyan bars in the preston plot represent the observed species (or equivalent taxonomic group) while the orange dots represent the

expected number of the same. Please note that some of the results (like Shannon/Preston) would depend on the input data (true abundances vs counts vs incidences). Additionally, Preston result's output gives a number of species by default. If another taxonomic rank is used in the gen.comm.mat() function, the 'species' number in the results would be the number of that taxonomic rank (even though the output would still print 'species').

Value

A list containing containing:

- richness = A richness profile matrix
- output$Shannon_div = Shannon diversity values for the given sites/grids
- output$richness_plot = A ggplot2 visualization of the richness curve
- output$preston.res = a Preston plot numerical data output
- output$preston.plot = a ggplot2 visualization of the preston.plot

Examples

```
# Download data from BOLD (removing species with blanks)
comm.mat.data<-bold.connectr.public(taxonomy = "Poecilia")

comm.mat.data<-comm.mat.data[!comm.mat.data$species=="",]

# Generate the community matrix based on grids
comm.data.grid<-gen.comm.mat(comm.mat.data,taxon.rank="species",grids = TRUE,gridsize = 1000000)

grid.data<-comm.data.grid$comm.matrix

# Diversity results with estimation curve and without preston results
div.res1<-analyze.alphadiv(grid.data,plot.curve=TRUE,curve.index="Jack1ab",curve.xval = "Sampl")

# Richness estimations
head(div.res1$richness)

# Richness plot
div.res1$richness_plot

# Shannon diversity
head(div.res1$Shannon_div)


# Diversity results without estimation curve and with preston results
div.res2<-analyze.alphadiv(grid.data,preston.res = TRUE,pres.plot.y.label = "species")

# Preston results
div.res2$preston.res

# Preston plot
div.res2$preston.plot
```

`analyze.betadiv` *Create a beta diversity profile of the retrieved data*

Description

This function analyzes the output from the `gen.comm.mat` to provide a beta diversity (and its partitions) profile of the downloaded data.

Usage

```
analyze.betadiv(
  bin.comm,
  index,
  pre.abs = FALSE,
  heatmap = FALSE,
  component,
  grids = FALSE,
  grids.df = NULL
)
```

Arguments

`bin.comm`	the site X species like output from the `gen.comm.mat` function.Default value is FALSE.
`index`	A character vector specifying the type of beta diversity index ('jaccard' or 'sorenson' currently available).
`pre.abs`	A logical value specifying whether the input data is presence-absence. Default value is FALSE.
`heatmap`	A logical value specifying whether a heatmap of the beta diversity values should be plotted.Default value is FALSE.
`component`	A character value specifying which beta diversity component should be used for the heatmap. Default value is NULL.
`grids`	A logical value specifying Whether the community matrix is generated using grids. Default value is FALSE.
`grids.df`	If `grids` = TRUE, a sf grid data frame generated along with the community matrix using the `gen.comm.mat()` function.Default value is NULL.

Details

`analyze.betadiv` calculates either a sorenson or jaccard beta dissimilarity using the `gen.comm.mat()` output. It also generates matrices of 'species replacement' and 'richness difference' components of the total beta diversity. The values are calculated using `BAT::beta()` function which partitions the data using the Podani approach. A corresponding 'heatmap' can also be obtained when `heatmap`=TRUE. In case of grid based heatmaps, grids are arranged on the heatmap based on their centroid distances (i.e. nearest grids are placed closest). For site categories, the heatmap labels are arranged alphabetically. Grid based heatmaps can only be generated when `grids` = TRUE and a sf 'grid.df' which is generated from the `gen.comm.mat` function is provided to the function.

Value

A list containing:

- output$total.beta = beta.total
- output$replace = beta.replace (replacement)
- output$richnessd = beta.richnessd (richness difference)
- output$heatmap.viz = heatmap_final

Examples

```
#Download data from BOLD (removing species with blanks)
comm.mat.data<-bold.connectr.public(taxonomy = "Poecilia")

#Generate the community matrix based on grids
comm.data.beta<-gen.comm.mat(comm.mat.data,taxon.rank="species",site.cat = "country.ocean")

beta.data<-comm.data.beta$comm.matrix

#beta diversity without the heatmaps
beta.div.res<-analyze.betadiv(beta.data,index="sorenson")

#Total diversity
beta.div.res$total.beta

#Replacement
beta.div.res$replace

#Richness difference
beta.div.res$richnessd


#beta diversity with the heatmaps
beta.div.res2<-analyze.betadiv(beta.data,index="sorenson",heatmap = TRUE,component = "total")

#Total diversity
beta.div.res2$total.beta

#Replacement
beta.div.res2$replace

#Richness difference
beta.div.res2$richnessd

#Visualize the heatmap
beta.div.res$heatmap.viz
```

---

analyze.seq *Analyze and visualize the output from the* align.seq

---

Description

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

Usage

```
analyze.seq(
  aligned.seq,
  dist.model,
  clus = c("nj", "njs"),
  tree.export = FALSE,
  file.path = NULL,
  file.name = NULL,
  plot = FALSE,
  ...
)
```

Arguments

aligned.seq	A DNAStringset multiple sequence alignment object returned by align.seq function
dist.model	A character string specifying the model to generate the distances
clus	A character vector specifying either ape::nj() (neighbour joining) or ape::njs() (neighbour joining with NAs) clustering algorithm
tree.export	Logical value specifying whether newick tree should be generated and exported. Default value is FALSE,
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.
plot	Logical value specifying if a neighbour joining plot should be generated. Default value is FALSE
...	additional arguments from ape::dist.dna()

Details

analyze.seq analyzes the multiple sequence alignment output of the align.seq function to generate a distance matrix using the models available in the ape::dist.dna(). plot= TRUE will generate a basic visualization of the Neighbor Joining (NJ) tree of the distance matrix using the ggtree package. Both ape::nj() and ape::njs() are available for generating the tree. Additionally, the function provides base frequencies and a option to export the trees in 'newick' format.

Value

A list containing:

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

Examples

```
#Download the data
seq.data<-bold.connectr.public(taxonomy = c("Eulimnadia"),marker = "COI-5P")
seq<-seq.data[!seq.data$species=="",]

# Align the data (using species" and bin_uri  as a composite name for each sequence)
seq.align<-align.seq(seq,name.fields = c("species","bin_uri"),marker="COI-5P")

#Analyze the data
seq.analysis<-analyze.seq(seq.align$msa.result,"K80",clus="njs",plot=TRUE)

#Visualize the plot
seq.analysis$plot

#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
```

`bold.connectr` *Retrieve data from the BOLD database*

Description

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

Usage

```
bold.connectr(
  input.data,
  param,
  param.index,
  api_key,
  taxonomy = NULL,
  geography = NULL,
  latitude = NULL,
  longitude = NULL,
  shapefile = NULL,
  institutes = NULL,
  identified.by = NULL,
  seq.source = NULL,
  marker = NULL,
  collection.period = NULL,
  basecount = NULL,
  altitude = NULL,
  depth = NULL,
```

```
  fields = NULL,
  export = FALSE,
  file.type = NULL,
  file.path = NULL,
  file.name = NULL
)
```

Arguments

input.data	A file path pointing to either a CSV/TSV/txt file with the ids or a data frame where ids are stored
param	A character string specifying one of "processid", "sampleid", "bin_uri" or "dataset_codes"
param.index	A number indicating the column index of the params in the dataset.
api_key	A character string required for authentication and data access.
taxonomy	A single or multiple character vector of taxonomic names at any hierarchical level. Default value is NULL.
geography	A single or multiple character vector of any of country/province/state/region/sector/site names/codes. Default value is NULL.
latitude	A single or a vector of two numbers specifying the latitudinal range. Values should be separated by a comma. Default value is NULL.
longitude	A single or a vector of two numbers specifying the longitudinal range. Values should be separated by a comma. Default value is NULL.
shapefile	A file path pointing to a shapefile or name of the shapefile imported in the R session. Default value is NULL.
institutes	A single or multiple character vector specifying names of institutes. Default value is NULL.
identified.by	A single or multiple character vector specifying names of people responsible for identifying the organism. Default value is NULL.
seq.source	A single or multiple character vector specifying the data portals from where the (sequence) data was mined. Default value is NULL.
marker	A single or multiple character vector specifying of gene names. Default value is NULL.
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.
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.
altitude	A single or a vector of two numbers specifying the altitude range. Values should be separated by a comma. Default value is NULL.
depth	A single or a vector of two numbers specifying the depth range. Values should be separated by a comma. Default value is NULL.
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

This function retrieves both public as well as private user data and can effectively download data in bulk. Currently 'processids', 'sampleids', 'bin_uri' and 'dataset codes' are available as search `param`.There is no cap on the upper limit of the data that can be retrieved though, that depends on the net connection and the machine specs. Data input is either as a data path to a flat file having `.csv/.tsv/.txt` extensions or a R `data.frame` object. Import assumes a header present for the input data. The function provides post download (optional) filters on various fields like `taxonomy`, `geography`, `institutions` etc. with the default being NULL for all. Using the `fields` argument will let the user select any specific columns that need to be in the final data frame instead of the whole data though, processids and sampleids will be present in the data by default. The default NULL value of the argument will result in all columns being downloaded. `api_key` is a UUID v4 hexadecimal string obtained by requesting BOLD. The validity of the key will be for one year and would need to be renewed after. Please note that the `param` and `param.index` must be correctly matched to avoid getting any errors.Also, it could be likely that certain values/fields are not currently available and will be so in the near 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)
#With processids ('processid' param is assumed to be in the first column (param.index=1))
res<-bold.connectr(input.data = test.data, param = 'processid',param.index = 1,api_key = "key")
head(data,10)

#With sampleids (the 'sampleid' param is assumed to be in the second column (param.index=2))
res<-bold.connectr(test.data,'sampleid',2,api_key = "key")
head(data,10)

## Using filters

#Geography
res<-bold.connectr(test.data, param = 'processid',param.index = 1,api_key = "key",geography="India")

#Sequence length
res<-bold.connectr(test.data, 'processid',1,api_key = "key",nuc_basecount=c(500,600))

#Gene marker
res<-bold.connectr(test.data, 'processid',1,api_key = "key",marker="COI-5P")


## End(Not run)
```

`bold.connectr.filters` *Filters for specific parameters to customize the search for private data*

Description

Filters for specific parameters to customize the search for private data

Usage

```
bold.connectr.filters(
  bold.df,
  taxon.name = NULL,
  location.name = NULL,
  latitude = NULL,
  longitude = NULL,
  shapefile = NULL,
  institutes = NULL,
  identified.by = NULL,
  seq.source = NULL,
  marker = NULL,
  collection.period = NULL,
  basecount = NULL,
  altitude = NULL,
  depth = NULL
)
```

Arguments

bold.df	the output from the connectr functions
taxon.name	A single character string or a vector of taxonomic names at any hierarchical level. Default value is NULL
location.name	A single character string or a vector of country/province/state/region/sector/site names/codes. Default value is NULL.
latitude	A number or a vector of two numbers indicating the latitudinal range. Values separated by a comma. Default value is NULL.
longitude	A number or a vector of two numbers indicating the longitudinal range. Values separated by a comma. Default value is NULL.
shapefile	A file path pointing to a shapefile or name of the shapefile imported in the session. Default value is NULL.
institutes	A character string or a vector specifying names of institutes. Default value is NULL.
identified.by	A character string or a vector specifying names of people responsible for identifying the organism. Default value is NULL.
seq.source	A character string or a vector specifying data portals from where the (sequence) data was mined. Default value is NULL.
marker	A character string or a vector specifying of gene names. Default value is NULL.
collection.period	A Date or a vector of two values specifying the collection period range (start, end). Values separated by comma. Default value is NULL.
basecount	A number or a vector of two numbers indicating the base pairs number range. Values separated by a comma. Default value is NULL.
altitude	A number or a vector of two numbers indicating the altitude range. Values separated by a comma. Default value is NULL.

depth — A number or a vector of two numbers indicating the depth range. Values separated by a comma. Default value is NULL.

`bold.connectr.public` *Retrieve publicly available data from the BOLD database*

Description

Retrieves publicly available data based on taxonomy,geography or ids (processid,sampleid, dataset codes & bin_uri) input.

Usage

```
bold.connectr.public(
  taxonomy = NULL,
  geography = NULL,
  bins = NULL,
  ids = NULL,
  datasets = NULL,
  latitude = NULL,
  longitude = NULL,
  shapefile = NULL,
  institutes = NULL,
  identified.by = NULL,
  seq.source = NULL,
  marker = NULL,
  collection.period = NULL,
  basecount = NULL,
  altitude = NULL,
  depth = NULL,
  fields = NULL,
  export = NULL,
  file.type,
  file.path,
  file.name
)
```

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.
ids	A single or multiple character vector specifying either processids or sampleids. Default value is NULL.
datasets	A single or multiple character vector specifying dataset codes. Default value is NULL.

latitude	A single or a vector of two numbers specifying the latitudinal range. Values should be separated by a comma. Default value is NULL.
longitude	A single or a vector of two numbers specifying the longitudinal range. Values should be separated by a comma. Default value is NULL.
shapefile	A file path pointing to a shapefile or name of the shapefile imported in the R session. Default value is NULL.
institutes	A single or multiple character vector specifying names of institutes. Default value is NULL.
identified.by	A single or multiple character vector specifying names of people responsible for identifying the organism. Default value is NULL.
seq.source	A single or multiple character vector specifying the data portals from where the (sequence) data was mined. Default value is NULL.
marker	A single or multiple character vector specifying of gene names. Default value is NULL.
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.
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.
altitude	A single or a vector of two numbers specifying the altitude range. Values should be separated by a comma. Default value is NULL.
depth	A single or a vector of two numbers specifying the depth range. Values should be separated by a comma. Default value is NULL.
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

Function downloads publicly available data on BOLD. Data can be retrieved by providing either one or a combination of `taxonomy`, `geography`, `bins`, `ids` or `datasets` codes. There is no limit on the data that can be downloaded but complex combinations of the search parameters can lead to weburl character length exceeding the predetermined limit (2048 characters). Single parameter searches are exempt from this limit as downloads are carried out in batches of 5 (values of the parameter). Downloaded data can then be filtered on either one or a combination of arguments (such as `institutes`, `identifiers`, `altitude`, `depth` etc.). Using the `fields` argument will let the user select any specific columns that need to be in the final data frame instead of the whole data. The default NULL value will result in all data being acquired.Please note that for every request, it could be likely that certain values/fields are not currently available and will be so in the near future.

Value

A data frame containing all the information related to the query search

Examples

```
# Taxonomy
bold.data<-bold.connectr.public(taxonomy = "Panthera leo")
head(bold.data,10)#'

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

# Taxonomy, Geography and BINs
bold.data.taxo_geo<-bold.connectr.public(taxonomy = "Panthera uncia",geography = "India",bins=c(""))
```

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

Description

#' @description This function provides information on the field (column) names and their respective data type

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.multirecords.set.csv

Helper function bold.multirecords.set.csv

Description

Helper function bold.multirecords.set.csv

Usage

```
bold.multirecords.set.csv(edited.json)
```

Arguments

edited.json	edited json file obtained after the fetch functions for any of the id types

data.summary

Generate a summary of the data retrieved by the connectr functions

Description

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

Usage

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

Arguments

bold.df	the data.frame retrieved from the connectr functions
cols	Logical value indicating whether the names of (all) the columns currently available in the database be printed in the console

Details

'data.summary' provides summaries for each data type available in the fetched dataset. The function uses the `skimr::skim()` function to generate a list of data frames followed by the `skimr::partition()` which separates the summary based on the data type for easy export. The summary includes counts for NULL, unique values along with proportion of complete cases. The 'columns' argument will select any specific field required. The output is printed on the console and can be saved as well. Please note that if the 'fields' argument from the 'bold.fetch.data' has been used to filter certain columns, summaries of only those columns will be available by default.

Value

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

Examples

```
# Download data
bold_data<-bold.connectr.public(taxonomy = "Oreochromis")

# Generate summary for specific fields (cols)
test.data.summary<-data.summary(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
```

---

fetch.bold.bins *fetch processid data using bin ids*

Description

fetch processid data using bin ids

Usage

```
fetch.bold.bins(data.input, param.index, api_key)
```

Arguments

data.input	input data
param.index	the column number in the data which has bin_uri data
api_key	API key required to fetch the data

---

fetch.bold.datasets *fetch processids data using the dataset codes*

Description

fetch processids data using the dataset codes

Usage

```
fetch.bold.datasets(data.input, param.index, api_key)
```

Arguments

data.input	input data
param.index	the column number in the data which has the dataset_codes
api_key	API key required to fetch the data

fetch.bold.processid *bold.fetch.data processid*

Description

bold.fetch.data processid

Usage

```
fetch.bold.processid(data.input, param.index, api_key)
```

Arguments

data.input	input data
param.index	the column number in the data which has bin_uri data
api_key	API key required to fetch the data

fetch.bold.sampleid *bold.fetch.data sampleid*

Description

bold.fetch.data sampleid

Usage

```
fetch.bold.sampleid(data.input, param.index, api_key)
```

Arguments

data.input	input data
param.index	the column number in the data which has bin_uri data
api_key	API key required to fetch the data

fetch.data *Helper Function: Fetching the data based on the temp file*

Description

Helper Function: Fetching the data based on the temp file

Usage

```
fetch.data(result)
```

Arguments

result	the output of any of the id fetch functions

fetch.public.data *Helper function to retrieve public data*

Description

Helper function to retrieve public data

Usage

```
fetch.public.data(query)
```

Arguments

query	the output of any of the id fetch functions

gen.comm.mat *Create a community matrix based on BINs abundances/incidences*

Description

This function generates a community matrix (~site X species) using the data retrieved bold.connectr() or bold.connectr.public() based on BIN abundances/incidences

Usage

```
gen.comm.mat(
  bold.df,
  taxon.rank,
  taxon.name = NULL,
  site.cat = NULL,
  grids = FALSE,
  gridsize = NULL,
  pre.abs = FALSE,
  view.grids = FALSE
)
```

Arguments

bold.df	the bold 'data.frame' generated from bold.connectr() or bold.connectr.public()
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 or multiple character vector specifying the countries for which a community matrix should be created. Default value is NULL.
grids	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.
pre.abs	A logical value specifying whether the generated matrix should be converted into a 'presence-absence' matrix
view.grids	A logical value specifying viewing the grids overlaid on a map with respective cell ids. Default value is FALSE.

Details

The function transforms the bold.connectr() or bold.connectr.public() downloaded data into a site X species like matrix using BINs instead of species. Values in each cell are the counts of a specific BIN from a site.cat site category or a 'grid'. These counts can be generated at any taxonomic hierarchical level for a single or multiple taxa (even for bin_uri though, in that case, the numbers in each cell would be the number of times that respective BIN is found at a particular site.cat or 'grid'). site.cat can be any of the geography fields (Meta data on fields can be checked using the bold.fields.info()). Alternatively, grids = TRUE will generate grids based on the BIN occurrences with the size of the grid being determined by the user (in sq.m.). For grids generation, rows with no latitude and longitude data are removed (even if a corresponding site.cat name is available) while null entries for site.cat are allowed if they have a latitude and longitude value (This is done because grids are drawn based on the bounding boxes which only use latitude and longitude values).grids converts the Coordinate Reference System (CRS) of the data to 'Mollweide' projection by which distance based grid can be correctly specified. A cell id is also given to each grid with the lowest number assigned to the lowest latitudinal point in the dataset. The cellids can be changed as per user requirement using the grids_final sf data frame stored in the output. This can be visualized with the view.grids argument. The plot obtained is a visualization of the grid centroids with their respective names. Please note that a) data points having NA values for bin_uri, latitude or longitude or both are removed, b) if the data has many closely located grids, visualization with view.grids can get difficult. The argument pre.abs will convert the counts to 1 and 0. This dataset can then directly be used as the input data for functions from packages like vegan for biodiversity analyses.

Value

a list containing

- comm.matrix = A site X species like matrix based on BINs
- grids = A sf data frame containing the grid geometry and corresponding cell id
- grid_plot = A grid_plot overlaid on a world map with cell ids

Examples

```
# Using countries as a site.cat
# Download data from BOLD
comm.mat.data<-bold.connectr.public(taxonomy = "Panthera")

# Generate the community matrix based on countries
comm.matrix<-gen.comm.mat(comm.mat.data,taxon.rank="species",site.cat = "country.ocean")

# View the community matrix
head(comm.matrix$comm.matrix)

# Using grids instead of site.cat
```

```
# Generate the community matrix based on grids
comm.data.grid<-gen.comm.mat(comm.mat.data,taxon.rank="species",grids = TRUE,gridsize = 1000000)

# View the community matrix
head(comm.data.grid$comm.matrix)

# View the sf dataframe of the grids
head(comm.data.grid$grids)
```

`generate.batches` *Helper function to generate batches*

Description

Helper function to generate batches

Usage

```
generate.batches(data, param.index, batch.size)
```

Arguments

`data`	input data
`param.index`	the column number in the data which has bin_uri data
`batch.size`	Number specifying the number of batches to be created

`get_data_bins` *Helper function using GET to get the processids from bin ids*

Description

Helper function using GET to get the processids from bin ids

Usage

```
get_data_bins(data_for_bins, api.key)
```

Arguments

`data_for_bins`	input data containing BIN information
`api.key`	API key required to fetch the data

get_data_datasets *Helper function using GET to get the processids from dataset codes*

Description

Helper function using GET to get the processids from dataset codes

Usage

```
get_data_datasets(data_for_datasets, api.key)
```

Arguments

data_for_datasets	input data containing dataset codes information
api.key	API key required to fetch the data

get_data_processid *GET_data helper functions for single records of processids and sampleids*

Description

GET_data helper functions for single records of processids and sampleids

Usage

```
get_data_processid(input_data, param.index, api.key)
```

Arguments

input_data	input data
param.index	the column number in the data which has bin_uri data
api.key	API key required to fetch the data

globals *global variables*

Description

global variables

id.files *Helper Function id.files*

Description

Helper Function id.files

Usage

```
id.files(ids)
```

Arguments

ids	Character string which are converted to temp file/s which then are used by the POST function

post.api.res.fetch *Helper function to fetch the data using POST API*

Description

Helper function to fetch the data using POST API

Usage

```
post.api.res.fetch(base.url, query.params, api.key, temp.file)
```

Arguments

base.url	base url for POST API
query.params	the query parameters for the POST API
api.key	the API key needed to access the data
temp.file	the tempfile on which data is written for the POST API

reassign.data.type *Helper function reassign.data.type*

Description

Helper function reassign.data.type

Usage

```
reassign.data.type(x)
```

Arguments

x	Data frame

test.data *A data frame with 2101 processids and sampleids*

Description

A data frame containing proccessids and sampleids for data retrieval from BOLD Report ...

Usage

```
test.data
```

Format

`test.data`:
A data frame with 2205 rows and 2 columns:

processid Character vector of processids
sampleid Character vector of sampleids corresponding to the processids

visualize.geo *Visualize organism occurrence data on maps*

Description

#' @description This function creates basic maps of organism occurrence data on different scales

Usage

```
visualize.geo(
  bold.df,
  country = NULL,
  bbox = NULL,
  export = FALSE,
  file.path = NULL,
  file.name = NULL,
  file.type = NULL
)
```

Arguments

bold.df	the data.frame retrieved from `bold.connectr()` or `bold.connectr.public()`
country	A single or multiple character vector of country names. Default is NULL
bbox	A numeric vector specifying the min,max values of the latitude and longitude. Default is NULL
export	A logical value asking if the output should be exported locally
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.
file.type	A character value specifying the type of file to be exported. Currently '.csv' and '.tsv' options are available

Details

visualize.geo extracts out the geographic information from the bold.connectr() or bold.connectr.public() output. Data points having NA values for either latitude or longitude or both are removed. Latitude and longitude values are in 'decimal degrees' format. Default view includes data mapped on a world shape file downloaded using the rnaturalearth::ne_countries() at a 110 scale. If the 'country' is specified (single or multiple values), the function will specifically plot the occurrences on the specified country. Alternatively, a bounding box can be defined for a specific region to be visualized. If export = TRUE, an image file will be saved based on the type (jpg, tiff), and the file path. The function also provides a sf data frame of the GIS data which can be used for any other application/s.

Value

geo.df = A simple features (sf) 'data.frame' containing the geographic data map_plot = A visualization of the occurrences

Examples

```
#Download data
geo.data<-bold.connectr.public(taxonomy = "Musca domestica")

geo.viz<-visualize.geo(geo.data,export = FALSE)

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

# Visualization
geo.viz$plot
```

Index

∗ **datasets**
    test.data, 24

align.seq, 2
analyze.alphadiv, 4
analyze.betadiv, 6
analyze.seq, 7
ape::dist.dna(), *8*
ape::nj(), *8*
ape::njs(), *8*

BAT::alpha.accum(), *4*
BAT::beta(), *6*
bold.connectr, 9
bold.connectr(), *3*, *19, 20*, *24, 25*
bold.connectr.filters, 11
bold.connectr.public, 13
bold.connectr.public(), *3*, *19, 20*, *24, 25*
bold.fields.info, 15
bold.fields.info(), *20*
bold.multirecords.set.csv, 16

data.summary, 16

fetch.bold.bins, 17
fetch.bold.datasets, 17
fetch.bold.processid, 18
fetch.bold.sampleid, 18
fetch.data, 18
fetch.public.data, 19

gen.comm.mat, 19
gen.comm.mat(), *4–6*
generate.batches, 21
get_data_bins, 21
get_data_datasets, 22
get_data_processid, 22
ggplot2, *4*
ggtree, *8*
globals, 22

id.files, 23

msa::msa(), *3*

post.api.res.fetch, 23

reassign.data.type, 23
rnaturalearth::ne_countries(), *25*

sf, *6*, *20*, *25*
skimr::partition(), *16*
skimr::skim(), *16*

test.data, 24

vegan, *20*
vegan::diversity(), *4*
vegan::prestondistr(), *4*
visualize.geo, 24