Package 'BOLDNODE'

October 14, 2026

Title Search and Explore BOLD Data Packages Efficiently

Version 1.0.1

Description Provides efficient tools for exploring and transforming the Barcode of Life Data Systems (BOLD) data packages on local machines. It enables fast local querying of the data packages without relying on live API calls. Results can be easily converted into formats compatible with widely used R packages and third party tools.

License CC BY 4.0

Encoding UTF-8

Imports data.table, DBI, dbplyr, dplyr, duckdb (>= 1.5.5), progressr, rlang, sf, tidyr

Suggests ape, Biostrings, BiocManager, knitr, rmarkdown, DT, ggplot2, ggrepel, maps, muscle, pak, phangorn, tibble, vegan

Config/roxygen2/version 8.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Sameer Padhye [aut, cre],
Spencer K. Monckton [aut],
Dirk Steinke [ctb],
Jireh Agda [ctb],
Teresita M. Porter [ctb]

Maintainer Sameer Padhye <spadhye@uoguelph.ca>

Repository CRAN

Date/Publication 2026-10-14 18:32:08 UTC

Contents

bcdm_field_names 2
bcdm_field_values 3
bcdm_to_dnastringset 4
bcdm_to_dwc 5

bcdm_to_fasta 6
bcdm_to_occmatrix 7
bcdm_to_sf 9
bold_parquet_search 10
bold_search_collect 12
get_bin_consensus 14
get_bin_reps 16
get_concise_summary 20

Index **22**

`bcdm_field_names` *Retrieve BCDM field names and descriptions*

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).

Usage

```
bcdm_field_names(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) metadata (file type and brief description) for the Barcode Core Data Model (BCDM) from https://github.com/boldsystems-central/BCDM/blob/main/field_definitions.tsv; output = TRUE will print the information in the console. *Important Note*: Two field names 'country/ocean' and 'province/state' have been modified to 'country.ocean' and 'province.state' to match with BOLDconnectR output and for operational ease.

Value

A data frame containing definitions for all fields (columns).

Examples

```
bold.field.data <- bcdm_field_names()

head(bold.field.data, 10)
```

bcdm_field_values *Extract unique values from BCDM fields in the BOLD data package*

Description

Extracts distinct values of specified field(s) from the BOLD parquet data.

Usage

```
bcdm_field_values(
  input.data,
  specific.cols,
  save.data = FALSE,
  output.file = NULL
)
```

Arguments

input.data	Path to the input parquet file or the bold_parquet_search result.
specific.cols	Name of the column to extract unique values from.
save.data	Logical value indicating whether to save the results to disk as a .rds file (default: FALSE).
output.file	Path (without extension) for saving results as .rds file (required if save.data = TRUE).

Details

This function extracts unique values from one or more specified columns from the BOLD parquet data. It handles both the parquet file and tbl_sql objects from bold_parquet_search as input. The results can be saved to disk as an .rds file for later use.

Value

A list containing unique values from the specified column. If save.data = T, a .rds file is exported locally.

Examples

```
# Import the parquet file (This is a test parquet file composed of
# records of Cerambycidae beetles from Canada)
parquet_file <- system.file(
"extdata",
"test_data.parquet",
package = "BOLDNODE"
)
```

```
# Search the BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  marker = "COI-5P"
)

# Get the field values#'
vocab.data <- bcdm_field_values(bold_search,
  specific.cols = c("inst", "identified_by")
)
```

`bcdm_to_dnastringset` *Convert the BOLD parquet search into a DNAStringSet object*

Description

Converts the sequence data from the search result into a DNAStringSet object for downstream multiple sequence alignment with customized headers.

Usage

```
bcdm_to_dnastringset(bold.search.res, marker = NULL, cols_for_seq_names)
```

Arguments

`bold.search.res`	A `tbl_sql` object containing BOLD search results.
`marker`	Character vector specifying the genetic marker.
`cols_for_seq_names`	Character vector of field names to include in the header.

Details

This function transforms the search results from `bold_parquet_search` into a `DNAStringSet` object suitable for downstream data analyses. The `cols_for_seq_names` argument lets users create custom headers using the BCDM column names.

Value

A `DNAStringSet` object.

Examples

```
## Not run:

# Search the BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Coleoptera",
  geography = "Canada",
  basecount = c(500, 660)
)

# Get the DNAStringset object (library Biostrings needs to be imported beforehand)

bold.dnastringset <- bcdm_to_dnastringset(bold_search,
  marker = "COI-5P",
  cols_for_seq_names = c("processid", "family")
)

## End(Not run)
```

`bcdm_to_dwc` *Convert the BOLD parquet search in to a Darwin Core (DwC) format data model*

Description

Converts `bold_parquet_search` results from BCDM format to Darwin Core Standard format.

Usage

```
bcdm_to_dwc(bold.search.res)
```

Arguments

`bold.search.res`
: A `tbl_sql` object containing BOLD search results.

Details

This function maps BCDM (Barcode Core Data Model) fields to their Darwin Core equivalents (https://gbif.github.io/dwc-dp/qrg/)

Important Note: All fields should be available in the `bold_parquet_search` `tbl_sql` object otherwise, the function will throw an error.

Value

A data frame with columns mapped to Darwin Core equivalent fields.

Examples

```
# Import the parquet file (This is a test parquet file composed of
# records of Cerambycidae beetles from Canada)
parquet_file <- system.file(
"extdata",
"test_data.parquet",
package = "BOLDNODE"
)

# Search the BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Lepturinae",
  marker = "COI-5P",
  basecount = c(550, 660)
)

# Get the DwC object#'
bold.dwc <- bcdm_to_dwc(bold_search)
```

---

bcdm_to_fasta *Export the BOLD parquet search to FASTA format*

Description

Exports nucleotide sequences from BOLD search results to a FASTA file with customizable headers.

Usage

```
bcdm_to_fasta(bold.search.res, output.file, fas.header, chunk.size = 1e+06)
```

Arguments

bold.search.res	A tbl_sql object containing BOLD search results.
output.file	Path to the output FASTA file.
fas.header	Character vector of field names to include in the FASTA header.
chunk.size	Number of records to process in each chunk (default: 1000000).

Details

This function transforms the search results from bold_parquet_search into a FASTA file. A data chunking option is available to manage large sizes to avoid memory issues. The fas.header argument lets users create custom headers using the BCDM field names. Metadata on fields can be checked using the bcdm_field_names function.

Value

Writes a FASTA file to disk with custom headers as specified by the user.

Examples

```
## Not run:

# Search the BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Coleoptera",
  geography = "Canada",
  marker = "COI-5P",
  basecount = c(500, 660)
)

# Get a fasta file
bcdm_to_fasta(
  bold_search,
  output.file = "trial2.fas",
  fas.header = c("bin_uri", "processid")
)

## End(Not run)
```

---

bcdm_to_occmatrix *Convert the BOLD parquet search into an occurrence matrix*

Description

Extracts occurrence data (specimen counts by taxon and location) from BOLD search results.

Usage

```
bcdm_to_occmatrix(
  bold.search.res,
  kingdom = "Animalia",
  taxon.rank,
  taxon.name = NULL,
  site.cat = NULL,
  pre.abs = FALSE
)
```

Arguments

`bold.search.res`
: A `tbl_sql` object containing BOLD search results.

kingdom	Character value specifying the kingdom (default: Animalia).
taxon.rank	Taxonomic rank to aggregate by (kingdom, phylum, class, order, family, genus, species or bin_uri).
taxon.name	Optional vector of specific taxon names to include (e.g., for taxon.rank = 'class', name can be 'Insecta').
site.cat	Optional categorical variable to group occurrence data by (e.g., region; when site.cat = NULL, coord used as default).
pre.abs	Logical indicating whether to convert counts to presence/absence (1/0) data (default: FALSE).

Details

This function transforms the search results from bold_parquet_search into the occurrence data matrices commonly used in biodiversity and ecological analyses by packages like vegan and betapart. Occurrences differ based on the kingdom. For Animalia, only records with BINs are included. For other kingdoms, all records with a sequence are counted (i.e., records without sequences are removed before calculations). Records can be aggregated at different taxonomic ranks (from kingdom to BINs) for a single or multiple taxa, with optional filtering by specific taxon names. site.cat can be any of the geography fields. The function can convert count data to presence/absence (1/0) format. *Important Note*: The bcdm_to_occmatrix function requires taxonomy (including bin_uri) and geography fields to be available in the bold_parquet_search result. If the specific.cols argument is used in the bold_parquet_search function to retrieve certain columns that do not have taxonomy and geography columns, the function will throw an error.

Value

A data frame with occurrence data (taxon names as columns, site categories or coordinates as rows).

Examples

```
# Import the parquet file (This is a test parquet file composed of
# records of Cerambycidae beetles from Canada)
parquet_file <- system.file(
"extdata",
"test_data.parquet",
package = "BOLDNODE"
)

# Search the BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  marker = "COI-5P",
)

# Get the occurrence matrix#'
occurrence_data <- bcdm_to_occmatrix(
  bold_search,
  taxon.rank = "genus",
  site.cat = "region"
```

```
)
```

---

bcdm_to_sf *Convert the BOLD parquet search into a simple features dataframe*

Description

Converts BOLD search results with coordinate data to a simple features dataframe with point geometry (sf object).

Usage

```
bcdm_to_sf(bold.search.res, chunk.size = 1e+05)
```

Arguments

bold.search.res	A tbl_sql object containing BOLD search results.
chunk.size	Number of records to process in each chunk (default: 100000).

Details

This function transforms the search results from bold_parquet_search into an sf object. A data chunking option is available to manage large sizes to avoid memory issues. The function creates point geometries in the WGS84 coordinate system (EPSG:4326). Records that don't have coordinate data are removed during processing.

Value

An sf object with point geometry in the WGS84 coordinate reference system (EPSG:4326).

Examples

```
## Not run:

# Search the BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Odonata",
  geography = "Malaysia"
)

# Get the occurrence matrix#'
sf_data <- bcdm_to_sf(bold_search, chunk.size = 100000)

## End(Not run)
```

`bold_parquet_search` *Search records within a BOLD parquet data package*

Description

Query records in BOLD parquet data packages using taxonomic, geographic, and other search criteria.

Usage

```
bold_parquet_search(
  input.parquet,
  ids = NULL,
  bins = NULL,
  scope.taxonomy = NULL,
  taxonomy = NULL,
  scope.geography = "any",
  geography = NULL,
  institutes = NULL,
  identified.by = NULL,
  seq.source = NULL,
  marker = NULL,
  basecount = NULL,
  biogeo.cat = NULL,
  dataset.projects = NULL,
  bounding.box = NULL,
  ambi.base.cutoff = NULL,
  specific.cols = NULL
)
```

Arguments

`input.parquet`	Path to the input parquet file.
`ids`	Vector of process IDs or sample IDs used to filter records.
`bins`	Vector of BIN numbers (i.e., URIs) used to filter records.
`scope.taxonomy`	Character value specifying the kingdom. Values include "all", "Animalia", "Plantae", "Protista", "Fungi", "Bacteria" (default: NULL).
`taxonomy`	Vector of taxonomic names to filter by. Values include "kingdom", "phylum", "class", "order", "family", "subfamily","genus", "species" (default: NULL).
`scope.geography`	A character string specifying the geographic hierarchy level for a search. Values include "any", "country.ocean", "province.state", "region", "sector", "site" (default: "any").
`geography`	Vector of geographic locations to filter by based on the `scope.geography`.
`institutes`	Vector of institute codes used to filter records.

identified.by	Vector of identifiers used to filter records.
seq.source	Vector of sequence run sites used to filter records.
marker	Vector of marker codes used to filter records.
basecount	Nucleotide base count filter - either a single value or a vector of two values for a range.
biogeo.cat	Character vector of biogeographic or ecological categories for filter by, such as a biome, realm, or ecoregion.
dataset.projects	Vector of dataset/project codes used to filter records.
bounding.box	Numeric vector of length 4: c(min_lon, max_lon, min_lat, max_lat).
ambi.base.cutoff	Character value for filtering data based proportion of ambiguous bases (IUPAC codes). Valid values are "<1%", "1-5%", and ">5%" (default: NULL).
specific.cols	Optional character vector of specific columns to return.

Details

This function loads the BOLD public data package parquet files (https://boldsystems.org/data/data-packages/) via DuckDB and applies filters based on the provided parameters. It supports filtering by ids (sampleid, processid), taxonomy (using a combination of scope.taxonomy and taxonomy), geography (using a combination of scope.geography and geography), biogeography (from biome to ecoregion level), BINs, institutes, identifiers, sequence sources, genetic markers, nucleotide base counts, dataset or projects, spatial bounding boxes, and ambiguous base percent cutoffs. The taxonomy and geography filters use a two-level scoping system that allows users to define `scope.taxonomy` and `scope.geography` to control how records are filtered. This is particularly useful when the same name exists at multiple geographic or taxonomic levels. For example, the name Azerbaijan may refer to a country, but a region with the same name also exists in Iran. If `scope.geography` is set to "any", records from both geographic levels will be returned. However, setting `scope.geography` to "country.ocean" restricts the search to country records only, returning records associated with the country of Azerbaijan. A similar situation can occur with taxonomic names where identical names are assigned to different groups. For example, the genus **Iris** occurs in both plants and animals. Using `scope.taxonomy` allows users to specify the desired taxonomic context and avoid ambiguity between groups. Users can also specify particular columns to return using the `specific.cols` parameter (column names can be checked using the `bcdm_field_names` function). The `tbl_sql` object can then be used by any `bcdm_to_*` function for data transformations or `bold_search_collect` to load the query results in memory.

Value

A `tbl_sql` object containing the filtered data. The total number of records matching the search criteria is printed to the console.

Examples

```
# Import the parquet file (This is a test parquet file composed of
# records of Cerambycidae beetles from Canada)
```

```
parquet_file <- system.file(
"extdata",
"test_data.parquet",
package = "BOLDNODE"
)

# Search the BOLD data package

# Taxonomy
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Clytus"
)

# Geography
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  geography = "Ontario"
)

# Combination of many search criteria
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Trachysida",
  geography = "British Columbia",
  marker = "COI-5P",
  basecount = c(500, 660)
)
```

`bold_search_collect` *Collect and export parquet search results*

Description

Collects, outputs and exports the results of a `bold_parquet_search` query, processing large datasets in user-defined chunks to improve memory efficiency.

Usage

```
bold_search_collect(
  bold.search.res,
  chunk.size = 1e+06,
  sys.sleep = 0,
  export = FALSE,
  export.type = c("tsv", "parquet"),
  output.path = NULL
)
```

Arguments

bold.search.res	A tbl_sql object obtained from bold_parquet_search.
chunk.size	Maximum number of rows to process in each chunk (default: 1e6).
sys.sleep	Time to sleep between chunks in seconds (default: 0).
export	Logical value that allows user to export the output locally (default: FALSE).
export.type	Character string specifying the data type of the exported file (tsv or parquet). Required when export=TRUE.
output.path	Character string specifying the local path for data export along with the file name and extension. Required when export=TRUE.

Details

This function collects the results of a bold_parquet_search query into the current R session. To facilitate the handling of large datasets, records can be processed in user-defined chunks, with optional pauses between chunks to help manage memory usage and system resources. The function also supports exporting results in TSV or parquet format. When export = FALSE (default), the collected data are returned only within the R session. When export = TRUE, a complete file path, including a file name and extension, must be provided via output.path. *Important Note*: Some queries (for example, all records from the order Diptera) may produce very large result sets that exceed the available RAM on lower-specification systems (e.g., 8 GB RAM), regardless of the chunking and system sleep settings.

Value

A data frame containing all collected results. If export = TRUE, the results are also exported locally as either a TSV or Parquet file.

Examples

```
# Import the parquet file (This is a test parquet file composed of
# records of Cerambycidae beetles from Canada)
parquet_file <- system.file(
"extdata",
"test_data.parquet",
package = "BOLDNODE"
)

# Search the BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  #    geography = "Ontario",
  marker = "COI-5P",
  basecount = c(500, 660)
)

# Collect the data  (no export)
bold_search_collect(
```

```
  bold_search,
  chunk.size = 50000,
  export = FALSE
)
```

`get_bin_consensus` *Compute consensus BIN taxonomy*

Description

Computes and returns consensus taxonomic identifications for each BIN in search results or a BCDM data frame.

Usage

```
get_bin_consensus(
  bold.search.res,
  ranks = c("kingdom", "phylum", "class", "order", "family", "subfamily", "tribe",
    "genus", "species", "subspecies"),
  threshold = 1,
  min.ids = 1,
  enforce.scientific = TRUE,
  groups = "bin_uri",
  discord.format = c("text", "list")
)
```

Arguments

`bold.search.res`
: A `tbl_sql` object obtained from `bold_parquet_search` or a data frame or data table in BCDM format.

`ranks`
: A character vector of ranks to consider for consensus identifications. Defaults to the standard BOLD ranks.

`threshold`
: Numeric value(s) between 0 and 1 indicating the minimum proportion of records in a BIN that must must share the same taxonomic assignment to establish a consensus. Supply as a single value, a vector with length equal to the number of taxonomic ranks considered, or a named list with names corresponding to specific ranks. If supplied as a named list, an optional "default" value can be set for any ranks that are not explicitly specified (e.g., `threshold = list(species = 0.95, default = 0.75)`). The default is 1.0 requiring complete agreement among records at all ranks.

`min.ids`
: Numeric value(s) indicating the minimum number of identifications needed to establish a consensus (names with fewer identifications are still included when calculating proportions). Supply as a single value, a vector with length equal to the number of taxonomic ranks in consideration, or a named list with names corresponding to specific ranks. If supplied as a named list, an optional "default"

value can be set for any ranks that are not explicitly specified (e.g., `min.ids = list(family = 1, default = 2)`). The default is 2, requiring a minimum of two identifications at any rank.

`enforce.scientific`
: A logical value indicating whether non-scientific, provisional names should be ignored when determining consensus. Default value is TRUE, meaning non-scientific names are ignored.

`groups`
: Grouping variable. Default value is "bin_uri".

`discord.format`
: String indicating the desired output format for the `discordant_ids` column. Can be one of "text" or "list". If "text" (the default), the output is a string column with comma-separated values in the format "Taxon (proportion)". If "list", the output is a list column with names indicating competing identifications and values indicating proportions of discordant identifications for each taxon.

Details

Consensus is defined as any name that exceeds the specified `threshold`, expressed as a proportion of records with a concordant identification (i.e., same name, same rank). The function steps backwards (i.e., from subspecies to kingdom) through the eligible `ranks` to determine the lowest available concordant identification that meets the criteria specified by `threshold`, `min.ids`, and `enforce.scientific`. Different thresholds can be supplied for each rank, if desired (either as a vector of equal length to ranks or as a named list). The function can also be applied to any other grouping variable by modifying `groups`.

The provided `bold.search.res` input can be a search result object from `bold_parquet_search` or a BCDM data frame. Alternatively, it can be any data frame or data table minimally containing `bin_uri` (or other grouping variable) and taxonomic identifications for all available records.

Important Note: This function performs operations on the input data and may be slow when applied to very large datasets or run on systems with limited #' resources. Before using this function, check the size of the `bold.search.res`object using `get_concise_summary` and proceed with caution.

Value

A table of consensus identifications for each BIN (or other grouping variable), with the following columns: `bin_uri`, `member_count`, `concordant_rank`, `concordant_id`, `discordant_rank`, `discordant_ids`.

Examples

```
## Not run:

# Search BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Coleoptera",
  geography = "Canada"
)

# Compute strict consensus identifications for BINs in searched data
```

```
strict_consensus <- get_bin_consensus(
  bold.search.res = bold_search,
  threshold = 1.0
)

# Compute identifications concordant among at least 75% of BIN members,
# as long as at least three records carry the majority identifications
# in each BIN.
bin_ids <- get_bin_consensus(
  bold.search.res = bold_search,
  threshold = 0.75,
  min.ids = 3
)

# Include non-scientific names (i.e., interim taxonomy or placeholder names)
# in consideration of BIN consensus.
bin_consensus <- get_bin_consensus(
  bold.search.res = bold_search,
  threshold = 0.9,
  enforce.scientific = FALSE
)

## End(Not run)
```

`get_bin_reps` *Select representative records by BIN*

Description

Obtain one or more representative record(s) from each BIN in search results or a BCDM data frame. BIN representatives can also be selected for each unique BIN-taxon combination. The chosen selection criteria are applied in sequence to select representatives, thus criteria should be given in order of priority. If multiple records are tied after all criteria have been applied, the tie is broken at random by default. Alternatively, it is possible to obtain reproducible results for the same input values by setting a random seed.

Usage

```
get_bin_reps(
  bold.search.res,
  Nreps = 1,
  by.taxon = FALSE,
  enforce.scientific = FALSE,
  non.redundant.taxa = FALSE,
  criteria = list(vouchered = TRUE,
                  seq_length = c("COI_auto", "longest", "shortest", 658),
                  id_method = c("Morphology", "Morphology and sequence based",
                                "Image based", "Image and sequence based",
```

```
                                  "Tree based", "BIN based", "BOLD ID Engine",
                                  "Other sequence based approach", "Other"),
                    inst = "Centre for Biodiversity Genomics",
                    coll_date = c("latest", "oldest"),
                    seq_date = c("latest", "oldest")),
  seed = NULL
)
```

Arguments

`bold.search.res`	A `tbl_sql` object obtained from `bold_parquet_search` or a data frame or data table in BCDM format.
`Nreps`	Integer indicating the maximum number of representatives to select for each BIN (or BIN-taxon combination).
`by.taxon`	Logical value indicating whether to select representatives for each unique combination of BIN and taxonomic identification. If `TRUE`, the additional parameters `non_redundant_taxa` and `enforce_scientific` are also applied. (See 'Sampling representatives by taxon' for more details.)
`enforce.scientific`	Logical value indicating whether to ignore non-scientific, provisional names for the purposes of sampling representatives by taxon. Ignored if `by.taxon` is `FALSE`. (See 'Sampling representatives by taxon' for more details.)
`non.redundant.taxa`	Logical value, when sampling by taxon (by.taxon = TRUE), representatives are selected from the lowest available taxonomic rank within each lineage. For example, the identifications "Apidae", "Bombus", and "Bombus terrestris" are considered to belong to a single taxonomic lineage, and records assigned to "Bombus terrestris" will be selected first. Ignored if `by.taxon` is `FALSE`. (See 'Sampling representatives by taxon' for more details.)
`criteria`	Named list of selection criteria to apply when sampling representatives, given in priority order. See 'Criteria' for more information and default values.
`seed`	Optional positive integer to use as a random seed for reproducible tie-breaking. If `NULL` (the default), ties are broken randomly and selected records may differ between runs.

Value

A data frame of selected representatives.

Input

The provided `bold.search.res` input can be a search result object from `bold_parquet_search` or a BCDM data frame. Alternatively, it can be any data frame or data table minimally containing `bin_uri`, unique record identifiers (e.g. `processid` or `sampleid`), taxonomic identifications for all available records, and any fields relevant to the provided selection `criteria` (see below).

Important Note: This function is not optimized for very large `tbl_sql` search results, particularly those with many unique BINs. On the other hand, input provided as a data frame or data table

can be processed much more efficiently. Therefore, if you intend to keep the full search results in addition to BIN representatives, it is recommended that you first collect the results into a data frame using bold_search_collect.

Criteria

Selection criteria must be listed in order of priority. Each one acts upon data from a particular BCDM field, which must be present in the input data, as indicated below. Available criteria include the following:

vouchered If TRUE, prioritize records with known voucher repositories over those mined from databases like GenBank. Setting this to FALSE will prioritize records *without* vouchers. To exclude this criterion, simply omit it. (Required field: inst.)

seq_length Can be used either to specify target barcode sequence length as an integer, to preferentially select longer or shorter sequences ("longest", "shortest"), or to select sequences that match the modal* barcode length for each BIN ("COI_auto"). If a target length is provided, sequences closest to that target are prioritized. *Modal barcode length ("COI_auto" option) is determined after first rounding sequence lengths to the nearest full codon; in the event of multiple modes, the one closest to 658bp is chosen. (Required field: nuc_basecount.)

id_method A character vector listing preferred identification methods in order of priority. The function definition lists all available values per the BCDM specification. (Required field: identification_method.)

inst A character vector listing preferred voucher specimen repositories in order of priority. Values not present in the input data are ignored. (Required field: inst.)

coll_date Prioritize recently collected specimens ("latest") or those collected longest ago ("oldest"). Records without dates are selected last in either case. (Required field: collection_date_start.)

seq_date Prioritize recently uploaded sequences ("latest") or those with the earliest upload date ("oldest"). (Required field: sequence_upload_date.)

Default selection criteria are as follows:

```
criteria = list(vouchered = TRUE,
                seq_length = "COI_auto",
                id_method = c("Morphology", "Morphology and sequence based",
                              "Image based", "Image and sequence based",
                              "Tree based", "BIN based", "BOLD ID Engine",
                              "Other sequence based approach", "Other"),
                inst = "Centre for Biodiversity Genomics",
                coll_date = "latest",
                seq_date = "latest")
```

Sampling representatives by taxon

When by.taxon = TRUE, representatives are selected for each unique combination of bin_uri and identification. For example: Sampling single representatives from a BIN containing records identified as "Agrilinae", "Agrilus", and "Agrilus VVG_sp.42" will yield three records—one for each name. Two additional parameters can be used to tune this behaviour: enforce.scientific and non.redundant.taxa.

When enforce.scientific = TRUE, interim / provisional names are ignored when considering unique identifications. For example: "Agrilus VVG_sp.42" and "Agrilus" will both be treated as "Agrilus". For the BIN from the previous example, this will yield two representatives: one each for "Agrilus" and "Agrilinae". Note that records with such names may still be selected as representatives if they are prioritized according to the provided criteria, or if they are the only available representatives in a BIN.

When non.redundant.taxa = TRUE, the identifications in a BIN are compared in terms of their full taxonomic classification to determine the most specific name available for each distinct taxonomic lineage, and any records thus identified will be selected first. For example: "Agrilinae", "Agrilus", and "Agrilus VVG_sp.42" will all be treated as a single lineage when selecting representatives by taxon. This taxonomic de-duplication behaviour is also affected by the previous parameter: when enforce.scientific and non.redundant.taxa are both TRUE, the names "Agrilinae", "Agrilus", "Agrilus VVG_sp.42", and "Agrilus crataegi" all collapse to the single taxon "Agrilus crataegi". When non.redundant.taxa is TRUE, and enforce.scientific is FALSE, the same four names collapse to "Agrilus VVG_sp.42" and "Agrilus crataegi". Note that this option prioritizes records with the lowest available taxonomic information, and in some cases can thus override the selection criteria, e.g., if otherwise preferred records have unresolved identifications. Note also that the function will always return at least Nreps records per BIN where possible; if there are fewer than Nreps records bearing the lowest non-redundant identification(s) in a BIN, additional records will be selected as back-fill beginning with the next-most specific identification, working upwards.

Reproducibility

It is possible to consistently obtain the same representatives using the same input parameters by supplying a random seed. However, the same data provided either as a tbl_sql object or a data frame may yield different representative records for a given random seed due to differences in sorting and tie-breaking behaviours across backends (i.e. DuckDB vs. data.table).

Examples

```
## Not run:

# Search BOLD data package
bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Araneae",
  geography = "Canada"
)

# Select three representatives per BIN from the searched data,
# prioritizing those with a morphological ID and with vouchers
# deposited at the CBG (e.g. in case vouchers need to be examined)
bin_reps <- get_bin_reps(
  bold.search.res = bold_search,
  Nreps = 3,
  criteria = list(
    inst = "Centre for Biodiversity Genomics",
    id_method = c(
      "Morphology",
      "Morphology and sequence based"
```

```
    )
  )
)

# Select one representative for each combination of BIN and taxonomic
# lineage (scientific names only), with preference for 658-bp barcodes
# (e.g. for building a sequence tree of all known taxa)
bin_tax_reps <- get_bin_reps(
  bold.search.res = bold_search,
  Nreps = 1,
  by.taxon = TRUE,
  enforce.scientific = TRUE,
  non.redundant.taxa = TRUE,
  criteria = list(seq_length = 658)
)

## End(Not run)
```

`get_concise_summary` *Generate a concise summary of the search results*

Description

Creates a summary statistics table from the `bold_parquet_search` `tb_sql` object.

Usage

```
get_concise_summary(bold.search.res)
```

Arguments

`bold.search.res`
: A `tbl_sql` object containing `bold_parquet_search` results.

Details

The function provides a concise summary of the search obtained by `bold_parquet_search` that includes: total records, unique BINs, unique institutes, unique markers and amplicon size range.

Value

A data frame with the summary statistics.

Examples

```
# Search the BOLD data package

# Import the parquet file (This is a test parquet file composed of
# records of Cerambycidae beetles from Canada)
parquet_file <- system.file(
"extdata",
"test_data.parquet",
package = "BOLDNODE"
)

bold_search <- bold_parquet_search(
  input.parquet = parquet_file,
  taxonomy = "Lamiinae",
  marker = "COI-5P"
)
# Get the concise summary
bold_summary <- get_concise_summary(bold_search)
```

Index

bcdm_field_names, 2
bcdm_field_values, 3
bcdm_to_dnastringset, 4
bcdm_to_dwc, 5
bcdm_to_fasta, 6
bcdm_to_occmatrix, 7
bcdm_to_sf, 9
bold_parquet_search, 10, *14, 15*, *17*
bold_search_collect, 12, *18*

get_bin_consensus, 14
get_bin_reps, 16
get_concise_summary, *15*, 20