Package: webseq (via r-universe)

March 3, 2025

Type Package

Title Access data from biological sequence databases like NCBI, ENA, MGnify

Description This package interacts with online biological sequence databases. It provides functions to search for sequences, convert identifiers and download sequences and associated metadata.

Version 0.1

Date 2022-01-05

Author Tamas Stirling

Maintainer Tamas Stirling <stirling.tamas@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports curl, dplyr, httr, jsonlite, plyr, rentrez, stringi, stringr, tibble, tidyr, XML, xml2

Suggests knitr, rmarkdown, testthat

RoxygenNote 7.2.3

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev

Repository https://r-multiverse-staging.r-universe.dev

RemoteUrl https://github.com/stitam/webseq

RemoteRef b710f93da5b8dd8a30fa170cdf3da8da943fbae1

RemoteSha b710f93da5b8dd8a30fa170cdf3da8da943fbae1

Contents

`ena_query` *Retrieve sequences from ENA*

Description

Retrieve sequences from ENA

Usage

```
ena_query(
  accessions,
  mode = "fasta",
  expanded = FALSE,
  annotation_only = FALSE,
  line_limit = 0,
  download = FALSE,
  destfile_by = "all",
  gzip = FALSE,
  set = FALSE,
  range = NULL,
  complement = FALSE,
  batch_size = 0,
  verbose = getOption("verbose")
)
```

Arguments

accessions	character; Accessions to query.
mode	character; Can be either "embl", "fasta", or "xml".
expanded	logical; Get expanded records for CON sequences.
annotation_only	logical; Only retrieve annotation, no sequence.
line_limit	integer; Limit the number of text lines returned.
download	logical; Download the result as a file.
destfile_by	character; Number of files to download. "batch": one file for each batch, "all": one file altogether.
gzip	logical; Download the result as a gzip file.
set	logical; ???
range	character; ???
complement	logical; ???
batch_size	integer; Number of accessions to query in a single request. Using this value, accessions will be broken down into one or more batches. If set to 0, all accessions will be queried in a single request.
verbose	logical; Should verbose messages be printed to the console?

Examples

```
## Not run:
ena_query("LC136852")
ena_query(c("LC136852", "LC136853"))

## End(Not run)
```

---

examples *Examples*

Description

This data set contains a list of IDs which can be used to access data from various data sources. These IDs are used across the package in function documentations, tests, vignettes.

Usage

```
examples
```

Format

A list with 6 elements:

assembly NCBI Assembly IDs

bioproject NCBI BioProject IDs

biosample NCBI BioSample IDs

gene NCBI Gene IDs

protein NCBI Protein IDs

sra NCBI SRA IDs

`extract_accn` *Extract Accession Numbers from a parsed assembly report*

Description

All assembly reports contain GenBank and/or RefSeq identifiers that uniquely identify a contig. This function can be used to extract both GenBank and RefSeq accessions a parsed assembly report.

Usage

```
extract_accn(report)
```

Arguments

`report`	list; a parsed assembly report. use `parse_report()` to parse assembly reports.

Value

a data frame

Note

This is the fifth step within the pipeline for downloading GenBank files.

See Also

`get_genomeid`, `get_report_url()`, `download_report()`, `parse_report()`, `download_gb()`

Examples

```
## Not run:
phages <- get_genomeid("Autographiviridae", db = "assembly")
report_url <- get_report_url(phages$ids[1])
download_report(report_url)
filename <- dir(paste0(tempdir(), "/assembly_reports"))
filepath <- paste0(tempdir(), "/assembly_reports/", filename)
rpt <- parse_report(filepath)
extract_accn(rpt)

## End(Not run)
```

---

flag_files *Flag files to keep for analysis*

Description

Some functions may download files that only differ in their source (e.g. GCA from GenBank assemblies or GCF for RefSeq assemblies) or their version number (v1, v2, etc.). This function helps remove redundant files by flagging which files should be kept for further analysis.

Usage

```
flag_files(filenames)
```

Arguments

filenames	character; a character vector of filenames. Currently the function only supports GCA/GCF identifiers. Look at the examples for more details.

Details

The function first prioritises GCF over GCA and then the highest version number.

Value

The function returns a data frame where each file is listed in the first column and the recommendation to keep the file for further analysis is listed in the last column.

Examples

```
# keep GCF
filenames <- c("GCA_003012895.2_ASM301289v2_genomic.fna",
               "GCF_003012895.2_ASM301289v2_genomic.fna")
flag_files(filenames)

# keep GCF even when version number is lower
filenames <- c("GCA_003012895.2_ASM301289v2_genomic.fna",
               "GCF_003012895.1_ASM301289v1_genomic.fna")
```

```
flag_files(filenames)

filenames <- c("GCA_003012895.1_ASM301289v1_genomic.fna",
               "GCA_003012895.2_ASM301289v2_genomic.fna")
flag_files(filenames)
```

`mgnify_endpoints` *MGnify endpoints*

Description

This functions queries MGnify for all available endpoints

Usage

```
mgnify_endpoints(verbose = getOption("verbose"))
```

Arguments

`verbose`	logical; should verbose messages be printed to console?

Value

a tibble of API-s and their respective endpoints

Note

The function prints contents of the following url: https://www.ebi.ac.uk/metagenomics/api/v1/

Examples

```
## Not run:
mgnify_endpoints(verbose = TRUE)

## End(Not run)
```

mgnify_instance *Search for a specific entry in MGnify*

Description

This function can be used for searching MGnify using an identifier.

Usage

```
mgnify_instance(query, from)
```

Arguments

query	character; the indentifier
from	character; the api which contains this identifier. See `mgnify_endpoints()` for a list of possible apis.

Value

a list

Examples

```
## Not run:
# look up an assembly
mgnify_instance("ERZ477576", from = "assemblies")

## End(Not run)
```

mgnify_list *Retrieve a list of instances from MGnify*

Description

This function retrieves a list of identifiers to look up with other functions.

Usage

```
mgnify_list(
  query,
  from,
  from_id,
  page = NULL,
  sleep = 0.2,
  verbose = getOption("verbose")
)
```

Arguments

query	character; what to look for.
from	character; API. See "mgnify_endpoints()".
from_id	character; more precise filtering for the API.
page	numeric; the API's response is paginated this tells the API which page to return. If NULL, the function will return all pages.
sleep	character; number of seconds to sleep before requesting the next page.
verbose	logical; should verbose messages be printed to console?

Examples

```
## Not run:
# Query samples collected from biogas plants
mgnify_list(query = "samples",
            from = "biomes",
            from_id = "root:Engineered:Biogas plant",
            page = 1)

## End(Not run)
```

---

ncbi_download_genome *Download Genomes from NCBI*

Description

This function directly downloads genome data through the NCBI FTP server.

Usage

```
ncbi_download_genome(
  accession,
  type = "genomic.gbff",
  dirpath = NULL,
  verbose = getOption("verbose")
)
```

Arguments

accession	character; a character vector of assembly accessions.
type	character; the file extension to download. Valid options are "assembly_report", "assembly_stats", "cds", "feature_count", "feature_table", "genomic.fna", "genomic.gbff", "genomic.gff", "genomic.gtf", "protein.faa", "protein.gpff", "translated_cds".
dirpath	character; the path to the directory where the file should be downloaded. If NULL, download file to the working directory.
verbose	logical; should verbose messages be printed to console?

Examples

```
## Not run:
# Download genbank file for GCF_003007635.1.
# The function will access files within this directory:
# ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/003/007/635/
ncbi_download_genome("GCF_003007635.1", type = "genomic.gbff", verbose = TRUE)

# Download multiple files
accessions <- c("GCF_000248195.1", "GCF_000695855.3")
ncbi_download_genome(accessions, type = "genomic.gbff", verbose = TRUE)

## End(Not run)
```

---

ncbi_get_meta *Get sequence metadata from NCBI*

Description

This function retrieves metadata from a given NCBI sequence database.

Usage

```
ncbi_get_meta(
  query,
  db = NULL,
  batch_size = 100,
  use_history = TRUE,
  parse = TRUE,
  verbose = getOption("verbose")
)
```

Arguments

query	either an object of class ncbi_uid or an integer vector of UIDs. See Details for more information.
db	character; the database to search in. For options see ncbi_dbs(). Not all databases are supported.
batch_size	integer; the number of search terms to query at once. If the number of search terms is larger than batch_size, the search terms are split into batches and queried separately.
use_history	logical; should the function use web history for faster API queries?
parse	logical; Should the function attempt to parse the output into a tibble? If unsuccessful, the function will return the unparsed output.
verbose	logical; Should verbose messages be printed to console?

Details

Some functions in webseq, e.g. `ncbi_get_uid()` or `ncbi_link_uid()` return objects of class `"ncbi_uid"`. These objects may be used directly as query input for `ncbi_get_meta()`. This approach is recommended because the internal structure of these objects make `ncbi_get_meta()` queries more robust. Alternatively, you can also use a character vector of UIDs as query input.

If query is a `"ncbi_uid"` object, the `db` argument is optional. If `db` is not specified, the function will use the `db` attribute of the `"ncbi_uid"` object as `db` argument. However, if it is specified, it must be identical to the `db` attribute of the `"ncbi_uid"` object. If query is a character vector, the `db` argument is required.

Value

The function returns a list with two elements:

- `meta`: if `parse = TRUE` then either a tibble with the metadata or if parsing is unsuccessful, the unparsed metadata. If `parse = FALSE` the unparsed metadata.
- `history`: a tibble of web histories.

Examples

```
## Not run:
data(examples)
uids <- ncbi_get_uid(examples$biosample, db = "biosample")
meta <- ncbi_get_meta(uids)

## End(Not run)
```

---

ncbi_get_uid *Get UID-s from NCBI databases*

Description

This function replicates the NCBI website's search utility. It searches one or more search terms in the chosen database and returns internal NCBI UID-s for the hits. These can be used e.g. to link NCBI entries with entries in other NCBI databases or to retrieve the data itself.

Usage

```
ncbi_get_uid(
  term,
  db,
  batch_size = 100,
  use_history = TRUE,
  verbose = getOption("verbose")
)
```

Arguments

term	character; one or more search terms.
db	character; the database to search in. For options see ncbi_dbs().
batch_size	integer; the number of search terms to query at once. If the number of search terms is larger than batch_size, the search terms are split into batches and queried separately. Not used when using web history.
use_history	logical; should the function use web history for faster API queries?
verbose	logical; should verbose messages be printed to the console?

Details

The default value for batch_size should work in most cases. However, if the search terms are very long, the function may fail with an error message. In this case, try reducing the batch_size value.

Value

An object of class "ncbi_uid" which is a list with three elements:

- uid: a vector of UIDs.
- db: the database used for the query.
- web_history: a tibble of web histories.

Examples

```
## Not run:
ncbi_get_uid("GCA_003012895.2", db = "assembly")
ncbi_get_uid("Autographiviridae OR Podoviridae", db = "biosample")
ncbi_get_uid(c("WP_093980916.1", "WP_181249115.1"), db = "protein")

## End(Not run)
```

---

ncbi_link_uid *Link UID-s from one NCBI database to another*

Description

Each entry in an NCBI database has its unique internal id. Entries in different databases may be linked. For example, entries in the NCBI Assembly database may be linked with entries in the NCBI BioSample database. This function attempts to link uids from one database to another.

Usage

```
ncbi_link_uid(
  query,
  from = NULL,
  to,
  batch_size = 100,
  use_history = FALSE,
  verbose = getOption("verbose")
)
```

Arguments

query	either an object of class ncbi_uid or an integer vector of UIDs. See Details for more information.
from	character; the database the queried UIDs come from. ncbi_dbs() lists all available options.
to	character; the database in which the function should look for links. ncbi_dbs() lists all available options. See Details for more information.
batch_size	integer; the number of search terms to query at once. If the number of search terms is larger than batch_size, the search terms are split into batches and queried separately. Not used when using web history.
use_history	logical; should the function use web history for faster API queries?
verbose	logical; should verbose messages be printed to the console? ncbi_dbs() lists all available options.

Details

Some functions in webseq, e.g. ncbi_get_uid() or ncbi_link_uid() return objects of class "ncbi_uid". These objects may be used directly as query input for ncbi_link_uid(). This approach is recommended because the internal structure of these objects make ncbi_link_uid() queries more robust. Alternatively, you can also use a character vector of UIDs as query input.

If query is a "ncbi_uid" object, the from argument is optional. If from is not specified, the function will use the db attribute of the "ncbi_uid" object as from argument. However, if it is specified, it must be identical to the db attribute of the "ncbi_uid" object. If query is a character vector, the from argument is required.

Value

An object of class "ncbi_uid" which is a list with three elements:

- uid: a vector of UIDs.
- db: the database used for the query.
- web_history: a tibble of web histories.

Note

ncbi_link_uid() can work with or without web histories, but the behaviour of the function with web histories is unreliable. The option is there but it is recommended NOT to use web histories with this function.

Examples

```
## Not run:
ncbi_link_uid("4253631", "assembly", "biosample")
ncbi_link_uid(c("1226742659", "1883410844"), "protein", "nuccore")

## End(Not run)
```

---

ncbi_meta_assembly *Retrieve NCBI Assembly metadata*

Description

Retrieve NCBI Assembly metadata

Usage

```
ncbi_meta_assembly(assembly_uid)
```

Arguments

assembly_uid	numeric

Examples

```
## Not run:
ncbi_meta_assembly(419738)

## End(Not run)
```

---

ncbi_parse *Parse NCBI sequence metadata*

Description

This function can be used to parse various retrieved non-sequence data sets from NCBI into a tibble. These data sets usually accompany the biological sequences and contain additional information e.g. identifiers, information about the sample, the sequencing platform, etc.

Usage

```
ncbi_parse(meta, db, format = "xml", verbose = getOption("verbose"))
```

Arguments

meta	character; either an unparsed metadata object returned by ncbi_get_meta() or the path to a file that was downloaded from NCBI.
db	character; the NCBI database from which the data was retrieved.
format	character; the format of the data set. Currently only "xml" is supported.
verbose	logical; Should verbose messages be printed to console?

Details

This function is integrated into ncbi_get_meta() and is called automatically if parse = TRUE (default). However, it can also be used separately e.g. when you want to examine the unparsed metadata object before parsing, or when you already downloaded the metadata manually and you just want to parse it into a tabular format.

Value

a tibble.

Examples

```
## Not run:
data(examples)
#'
# NCBI Assembly, download XML file from NCBI and parse

# Manually download the XML file
# https://www.ncbi.nlm.nih.gov/assembly/GCF_000299415.1
# upper right corner -> send to -> file -> format = xml -> create file
# Parse XML
ncbi_parse(meta = "assembly_summary.xml", db = "assembly", format = "xml")

# NCBI BioSample, fully programmatic access, separate retrieval and parsing

# Get metadata but do not parse
meta <- ncbi_get_meta(examples$biosample, db = "biosample", parse = FALSE)
# Parse metadata separately
ncbi_parse(meta = meta, db = "biosample", format = "xml")

# NCBI BioSample, download XML file from NCBI and parse

# Manually download the XML file
# https://www.ncbi.nlm.nih.gov/biosample/?term=SAMN02714232
# upper right corner -> send to -> file -> format = full (xml) -> create file
# Parse XML
ncbi_parse(meta = "biosample_result.xml", db = "biosample", format = "xml")#'

## End(Not run)
```

ncbi_parse_assembly_xml

Parse NCBI assembly metadata

Description

This function can be used to parse an xml file from the NCBI assembly database into a tibble.

Usage

```
ncbi_parse_assembly_xml(file, verbose = getOption("verbose"))
```

Arguments

file	character; path to an xml file.
verbose	logical; Should verbose messages be printed to console?

Value

a tibble.

Examples

```
## Not run:
# search for Acinetobacter baumannii within the NCBI Assembly database
# https://www.ncbi.nlm.nih.gov/assembly/?term=acinetobacter%20baumannii
# upper right corner -> send to -> file -> format = xml -> create file
# parse the downloaded file
ncbi_parse_assembly_xml("assembly_summary.xml")

## End(Not run)
```

ncbi_parse_biosample_txt

Parse NCBI BioSample metadata

Description

This function parses a txt file from the NCBI BioSample database.

Usage

```
ncbi_parse_biosample_txt(
  file,
  resolve_na = TRUE,
  verbose = getOption("verbose")
)
```

Arguments

file	character; path to a txt file.
resolve_na	logical; replace strings that match NA terms with NA.
verbose	logical; should verbose output be printed to console?

Value

a tibble.

Examples

```
## Not run:
# search for Acinetobacter baumannii within the NCBI BioSample database
# https://www.ncbi.nlm.nih.gov/biosample/?term=acinetobacter+baumannii
# upper right corner -> send to -> file -> format = full (text) -> create file
# parse the downloaded file
ncbi_parse_biosample_txt("biosample_summary.txt")

## End(Not run)
```

---

ncbi_parse_biosample_xml

Parse NCBI BioSample metadata

Description

BioSample metadata from NCBI can be retrieved in multiple file formats. This function parses metadata retrieved in XML format.

Usage

```
ncbi_parse_biosample_xml(biosample_xml, verbose = getOption("verbose"))
```

Arguments

biosample_xml	character; unparsed XML metadata either returned by ncbi_get_meta() or the path to a file that was downloaded from NCBI.
verbose	logical; Should verbose messages be printed to console?

parse_gb_header *Parse headers from GenBank files*

Description

Parse headers from GenBank files

Usage

```
parse_gb_header(
  file,
  dir = getwd(),
  outfile = "./cache/annotation_headers.rda",
  errorfile = "./cache/annotation_headers_parse_error.rda",
  batch_size = 10,
  verbose = getOption("verbose")
)
```

Arguments

file	character;
dir	character;
outfile	character;
errorfile	character;
batch_size	integer;
verbose	logical;

parse_report *Parse assembly reports*

Description

This function can be used to parse a downloaded assembly report.

Usage

```
parse_report(file)
```

Arguments

file	character; the file path to the assembly report.

Value

The function returns an object of classes `arpt` and `list`. The unique class is required for compatibility with subsequent functions in the pipeline. Otherwise data from the returned object can be extracted through general list operations.

Note

This is the fourth step within the pipeline for downloading GenBank files.

See Also

`get_genomeid`, `get_report_url()`, `download_report()`, `extract_accn()`, `download_gb()`

Examples

```
## Not run:
phages <- get_genomeid("Autographiviridae", db = "assembly")
report_url <- get_report_url(phages$ids[1])
download_report(report_url)
filename <- dir(paste0(tempdir(), "/assembly_reports"))
filepath <- paste0(tempdir(), "/assembly_reports/", filename)
parse_report(filepath)

## End(Not run)
```

Index