Package: webseq (via r-universe)

November 18, 2026

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>

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

BugReports https://github.com/stitam/webseq/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports curl, dataverse, dplyr, httr, httr2, jsonlite, magrittr, rentrez, rlang, stringi, stringr, tibble, tidyr, XML, xml2

Suggests knitr, rmarkdown, testthat

Remotes IQSS/dataverse-client-r

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev libx11-dev

Repository https://stitam.r-universe.dev

Date/Publication 2026-05-22 11:34:41 UTC

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

RemoteRef HEAD

RemoteSha 4337b497f8679484b3b1d3b12f4792abc5dd1f7d

Contents

dv_connect . . . 2
dv_download . . . 4
dv_list_files . . . 5
ena_download_reads . . . 5
ena_query . . . 6
ena2ncbi . . . 8
examples . . . 8
extract_accn . . . 9
flag_files . . . 10
flatten . . . 11
mgnify_endpoints . . . 11
mgnify_instance . . . 12
mgnify_list . . . 12
ncbi_download_genome . . . 13
ncbi_get_meta . . . 15
ncbi_get_summary . . . 16
ncbi_get_uid . . . 17
ncbi_link . . . 18
ncbi_link_uid . . . 19
ncbi_meta_assembly . . . 20
ncbi_parse . . . 21
ncbi_parse_assembly_xml . . . 22
ncbi_parse_biosample_txt . . . 23
ncbi_parse_biosample_xml . . . 24
ncbi_post_uid . . . 24
ncbi_recover_id . . . 25
ncbi2ena . . . 26
parse_report . . . 27

Index **29**

`dv_connect` *Connect to a Dataset on a Dataverse server*

Description

Connect to a Dataset on a Dataverse server

Usage

```
dv_connect(dataset, version = ":latest", server, apikey = "")
```

Arguments

`dataset`	character, "persistentId" of the Dataset.
`version`	character, version of the Dataset.
`server`	character, URL of the Dataverse server.
`apikey`	character, your API key for the Dataverse server.
`...`	Additional arguments to pass to ‘dataverse::get_dataset()‘.

Details

The persistentId and the server address can be obtained from the URL of the Dataset. The URL starts with the server address, and the persistentId is the value of the "persistentId" query parameter. For example, in the URL "https://dataverse.no/dataset.xhtml?persistentId=doi:10.18710/NCKZD7", the server address is "https://dataverse.no" and the persistentId is "doi:10.18710/NCKZD7".

If the Dataset is private, you will need to provide your API key to access it. You can obtain an API key by creating an account on the Dataverse server and generating a new API token in your account settings.

Value

A connection object with Dataset metadata.

Note

The function returns an attribute called "conn" which contains the connection information you provided when you called the function. If you are accessing a private Dataset, this attribute will contain your API key!

See Also

[dv_list_files()], [dv_download()]

Examples

```
# connect to a public dataset on dataverse.no
conn <- dv_connect(
  dataset = "doi:10.18710/NCKZD7",
  server = "https://dataverse.no"
)
# connect to a private dataset on dataverse.no
conn <- dv_connect(
 dataset = "private_persistentId_here",
 server = "https://dataverse.no",
 apikey = "your_api_key_here"
)
```

dv_download *Download files from a Dataset on a Dataverse server*

Description

Download files from a Dataset on a Dataverse server

Usage

```
dv_download(files, conn, dirpath = NULL, verbose = getOption("verbose"))
```

Arguments

files	a tibble of files from a ‘dv_connection‘ object, typically obtained from [dv_list_files()] plus optional filtering. Must contain columns called ‘filename‘, ‘id‘ and ‘md5‘.
conn	a ‘dv_connection‘ object from [dv_connect()].
dirpath	character, path to the directory where files will be saved. Defaults to the current working directory.
verbose	logical, should verbose messages be printed to the console?

Value

A character vector of file paths for successfully downloaded files, or ‘NA‘ for files that failed to download. Returned invisibly.

See Also

[dv_connect()], [dv_list_files()]

Examples

```
conn <- dv_connect(
  dataset = "doi:10.18710/NCKZD7",
  server  = "https://dataverse.no"
)

conn |>
  dv_list_files() |>
  dplyr::filter(grepl("\\.fasta$", filename)) |>
  dv_download(conn = conn)
```

dv_list_files *List files in a Dataset on a Dataverse server*

Description

List files in a Dataset on a Dataverse server

Usage

```
dv_list_files(conn)
```

Arguments

conn	a ‘dv_connection‘ object from [dv_connect()].

Value

A tibble with information about the files in the Dataset.

See Also

[dv_connect()], [dv_download()]

Examples

```
conn <- dv_connect(
  dataset = "doi:10.18710/NCKZD7",
  server  = "https://dataverse.no"
)
conn |> dv_list_files()
```

ena_download_reads *Download sequencing reads from the European Nucleotide Archive (ENA)*

Description

Download sequencing reads from the European Nucleotide Archive (ENA)

Usage

```
ena_download_reads(
  accession,
  type = "fastq",
  dirpath = NULL,
  mirror = TRUE,
  verbose = getOption("verbose")
)
```

Arguments

accession	character; a character vector of ENA Accession IDs.
type	character; A character string specifying the type of file to download, either "run", "fastq" or "err". See Details for more information.
dirpath	character; the path to the directory where the file should be downloaded. If NULL, download file to the working directory.
mirror	logical; should the download directory mirror the structure of the FTP directory?
verbose	logical; should verbose messages be printed to console?

Details

If type == "run" the function will download the read files that were originally submitted to ENA. If a run was originally submitted to another INSDC database (NCBI SRA, DDBJ) then a file of this category will not be available. If type == "fastq" the function will download one or more FASTQ files for each run. If type = "err" the function will download files that can be used with NCBI's SRA Toolkit.

References

https://ena-docs.readthedocs.io/en/latest/retrieval/file-download/sra-ftp-structure.html

Examples

```
## Not run:
ena_download_reads("ERR1649607", type = "fastq", verbose = TRUE)

## End(Not run)
```

---

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,
  include_links = 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; ???
include_links	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)
```

ena2ncbi *Convert accessions from ENA to NCBI*

Description

Take a vector of ENA accessions and convert them to NCBI accessions.

Usage

```
ena2ncbi(accessions, type)
```

Arguments

accessions	character; a vector or ENA accessions.
type	character; type of ENA accessions. Supported types: ‘sample‘, ‘study‘.

Value

A tibble with two columns, ‘ena‘ and ‘ncbi‘.

Examples

```
ena2ncbi("ERS3202441", type = "sample")
ena2ncbi(c("ERS3202441", "ERS3202442"), type = "sample")
ena2ncbi("ERP161024", type = "study")
```

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)
```

flatten *Bind rows in a list of list of data frames*

Description

This is a utility function that converts a list of lists, each containing a names list of data frames into a single flat list of data frames.

Usage

```
flatten(x)
```

Arguments

x	list; a list which contains a named list of data frames

Value

a flat list of data frames where lists of data frames with the same name are merged.

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 Assembly Database*

Description

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

Usage

```
ncbi_download_genome(
  query,
  type = "genomic.fna",
  dirpath = NULL,
  mirror = FALSE,
  verbose = getOption("verbose")
)
```

Arguments

query	an object of class 'ncbi_uid', 'ncbi_uid_link', 'ncbi_link', or an integer vector of NCBI Assembly UIDs. See Details for more information.
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.
mirror	logical; should the download directory mirror the structure of the FTP directory?
verbose	logical; should verbose messages be printed to console?

Details

'ncbi_get_uid()' returns an object of class 'ncbi_uid'; 'ncbi_link_uid' returns an object of class 'ncbi_uid_link'; 'ncbi_link' returns and object of class 'ncbi_link'. These objects may be used directly as query input for 'ncbi_download_genome'. It is recommended to use this approach. Alternatively, you can also use a character vector of UIDs as query input. This approach is not recommended because there are no consistency checks, the function will just attempt to interpret the query as NCBI Assembly UIDs.

Examples

```
## Not run:
# Download a single genome
ncbi_get_uid("GCF_003007635.1", db = "assembly") |>
  ncbi_download_genome()

"SAMN08619567" |>
  ncbi_get_uid(db = "biosample") |>
  ncbi_link_uid(to = "assembly") |>
  ncbi_download_genome()

"SAMN08619567" |>
  ncbi_link(from = "biosample", to = "assembly") |>
  ncbi_download_genome()

# Download multiple genomes, mirror FTP directory structure
data(examples)

examples$assembly |>
  ncbi_get_uid(db = "assembly") |>
  ncbi_download_genome()

## End(Not run)
```

ncbi_get_meta *Get sequence metadata from NCBI*

Description

This function retrieves sequence metadata from a given NCBI sequence database.

Usage

```
ncbi_get_meta(
  query,
  db = NULL,
  batch_size = 100,
  use_history = TRUE,
  parse = TRUE,
  mc_cores = NULL,
  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(). Note that not all of these 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?
mc_cores	integer; number of cores to use for parallel processing. Only used if parse = TRUE.
verbose	logical; Should verbose messages be printed to console?

Details

You can give UIDs to ncbi_get_meta() in two ways: 1. You can use functions like ncbi_get_uid() or ncbi_link_uid to get UIDs, and then use the returned ncbi_uid objects directly with ncbi_get_meta. If you follow this approach then you do not have to specify the db argument since the function can extract it from the ncbi_uid object. However, if you do provide it, then it must be identical to the db attribute of the "ncbi_uid" object. 2. Alternatively, you can just provide a vector of UIDs, but then you must specify the db argument as well.

Value

If parse = FALSE the function will return an object of class ncbi_meta, which is a character vector with some extra information about the database. This output can be used directly with ncbi_parse. If parse = TRUE the function will attempt to parse the data using ncbi_parse. If parsing is successful, the function will return a tibble, otherwise it will return the unparsed ncbi_meta object.

Examples

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

## End(Not run)
```

---

ncbi_get_summary *Get object summary from NCBI*

Description

This function retrieves object summary data from a given NCBI database.

Usage

```
ncbi_get_summary(
  query,
  db = NULL,
  batch_size = 100,
  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().
batch_size	integer; the number of items to query at once. If query length is larger than 'batch_size', the query will be split into batches.
verbose	logical; should verbose messages be printed to the console?

Details

If query is an 'ncbi_uid' object, the 'db' argument is optional. If 'db' is not specified, the function will retrieve it from the query object. However, if it is specified, it must be identical to the 'db' attribute of the query.

Value

A list of rentrez summary objects.

Examples

```
## Not run:
assemblies <- c("GCF_000002435.2", "GCF_000299415.1")
uids <- ncbi_get_uid(assemblies, db = "assembly")
ncbi_get_summary(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,
  na_strings = "NA",
  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.
`use_history`	logical; should the function use web history for faster API queries?
`na_strings`	character; a vector of strings which should be interpreted as 'NA'.
`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

```
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")
```

---

ncbi_link *Link ID-s from one NCBI database to another*

Description

Each entry in an NCBI database has its unique 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 ID-s from one database to another.

Usage

```
ncbi_link(
  query,
  from,
  to,
  multiple = "all",
  batch_size = 100,
  verbose = getOption("verbose")
)
```

Arguments

query	character; a vector of IDs
from	character; the database the queried ID-s 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.
multiple	character; handling of rows in x with multiple matches in y. For more information see '?dplyr::left_join()'.
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.
verbose	logical; should verbose messages be printed to the console?

Value

A tibble with two columns. The first column contains IDs in the ‘from‘ database, the second column contains linked IDs in the ‘to‘ database.

Examples

```
## Not run:
ncbi_link("GCF_000002435.2", from = "assembly", to = "biosample")
ncbi_link("SAMN02714232", from = "biosample", to = "assembly")

## 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,
  verbose = getOption("verbose")
)
```

Arguments

`query`	either an object of class ‘ncbi_uid‘ or ‘ncbi_uid_link‘, 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. See Details for more information.
`to`	character; the database in which the function should look for links. `ncbi_dbs()` lists all available options.
`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.
`verbose`	logical; should verbose messages be printed to the console?

Details

The function can take three query classes: It can take ‘ncbi_uid‘ objects, these are returned by ‘ncbi_get_uid()‘. In this case, the ‘from‘ argument will be retrieved from the query object, by default. It can also take ‘ncbi_uid_link‘ objects, which means ‘ncbi_link_uid()‘ can be called several times in a sequence to perform a number of successive conversions. When the query is an ‘ncbi_uid_link‘ object, the function will always convert the UIDs in the last column of the query object, and will retrieve the ‘from‘ argument from the name of the last column. This means links should always be interpreted "left-to-right". Note, when tibbles are joined during subsequent ‘ncbi_link_uid‘ calls they are joined using "many-to-many" relationships; see ‘?dplyr::left_join()‘ for more information. Lastly, the function can also take a vector of integer UIDs.

Value

A tibble with two or more columns. When ‘ncbi_link_uid()‘ is called on a ‘ncbi_uid‘ object or a vector of UIDs, the function returns a tibble with exactly two columns: the first column contains UIDs in the ‘from‘ database, and the second column contains linked UIDs in the ‘to‘ database. However, ‘ncbi_link_uid()‘ can be called multiple times in succession. Each call after the first call will add a new column to the returned tibble. See Details for more information.

Examples

```
# Simple call with integer UIDs
ncbi_link_uid(5197591, "assembly", "biosample")
ncbi_link_uid(c(1226742659, 1883410844), "protein", "nuccore")

# Complex call with ncbi_get_uid() and several ncbi_link_uid() calls
"GCF_000299415.1" |>
  ncbi_get_uid(db = "assembly") |>
  ncbi_link_uid(to = "biosample") |>
  ncbi_link_uid(to = "bioproject") |>
  ncbi_link_uid(to = "pubmed")
```

---

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 = NULL,
  format = "xml",
  mc_cores = NULL,
  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.
`mc_cores`	integer; Number of cores to use for parallel processing.
`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.

If `meta` is an unparsed `ncbi_meta` object returned by `ncbi_get_meta()` then the `db` argument is optional. If `db` is not specified, the function will extract it automatically. However, if it is specified, it must be identical to the `db` attribute of the metadata object. If `meta` is not an `ncbi_meta` object, the `db` argument is required.

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, the function will extract 'db' automatically.
ncbi_parse(meta = meta)

# 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,
  mc_cores = NULL,
  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.
`mc_cores`	integer; number of cores to use for parallel processing. If `NULL` use all available cores.
`verbose`	logical; Should verbose messages be printed to console?

`ncbi_post_uid` *Post a vector of NCBI UIDs*

Description

This function posts a vector of NCBI UIDs to the NCBI webservice. The purpose of this function is to generate an ncbi_uid object with web history which can then be used in downstream functions.

Usage

```
ncbi_post_uid(uid, db, batch_size = 100, verbose = getOption("verbose"))
```

Arguments

uid	numeric; a vector of NCBI UIDs.
db	character; the database the UIDs belong to. 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.
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 vector of UIDs is 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 the UIDs belong to.
- web_history: a tibble of web histories.

Note

In webseq NCBI UIDs are acquired using 'ncbi_get_uid()' or 'ncbi_link_uid()' and are used for acquiring linked UIDs in other NCBI databases or for retrieving the data itself. In some cases data retrieval is faster when using web history. However, you may have UIDs without a web history or your web history may expire. In this case you can use 'ncbi_post_uid()' to generate a new ncbi_uid object with web history.

Examples

```
ncbi_post_uid(4505768, db = "biosample")
```

---

ncbi_recover_id *Recover NCBI IDs from NCBI UIDs*

Description

Many functions that interact with NCBI return UIDs. This function converts the UIDs to NCBI IDs.

Usage

```
ncbi_recover_id(
  query,
  db = NULL,
  batch_size = 100,
  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().
batch_size	integer; the number of items to query at once. If query length is larger than 'batch_size', the query will be split into batches.
verbose	logical; should verbose messages be printed to the console?

Details

If query is an 'ncbi_uid' object, the 'db' argument is optional. If 'db' is not specified, the function will retrieve it from the query object. However, if it is specified, it must be identical to the 'db' attribute of the query.

Value

A vector of matching IDs.

Examples

```
## Not run:
uid <- ncbi_get_uid("GCF_000002435.2", db = "assembly")
ncbi_recover_id(uid)

## End(Not run)
```

---

ncbi2ena *Convert accessions from NCBI to ENA*

Description

Take a vector of NCBI accessions and convert them to ENA accessions.

Usage

```
ncbi2ena(accessions, type)
```

Arguments

accessions	character; a vector or ENA accessions.
type	character; type of NCBI accessions. Supported types: 'biosample', 'bioproject'.

Value

A tibble with two columns, 'ncbi' and 'ena'.

Examples

```
ncbi2ena("SAMEA111452506", type = "biosample")
```

---

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

∗ **datasets**
examples, 8

dv_connect, 2
dv_download, 4
dv_list_files, 5

ena2ncbi, 8
ena_download_reads, 5
ena_query, 6
examples, 8
extract_accn, 9

flag_files, 10
flatten, 11

mgnify_endpoints, 11
mgnify_instance, 12
mgnify_list, 12

ncbi2ena, 26
ncbi_download_genome, 13
ncbi_get_meta, 15
ncbi_get_summary, 16
ncbi_get_uid, 17
ncbi_link, 18
ncbi_link_uid, 19
ncbi_meta_assembly, 20
ncbi_parse, 21
ncbi_parse_assembly_xml, 22
ncbi_parse_biosample_txt, 23
ncbi_parse_biosample_xml, 24
ncbi_post_uid, 24
ncbi_recover_id, 25

parse_report, 27