Package 'easyPubMed'

May 8, 2026

Type Package

Title Search and Retrieve Scientific Publication Records from PubMed

Version 3.1.6

Date 2025-08-25

Maintainer Damiano Fantini <damiano.fantini@gmail.com>

Description Query NCBI Entrez and retrieve PubMed records in XML or text format. Process PubMed records by extracting and aggregating data from selected fields. A large number of records can be easily downloaded via this simple-to-use interface to the NCBI PubMed API.

URL https://www.data-pulse.com/dev_site/easypubmed/

Depends R(>= 3.5)

Imports methods, utils, rlang

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyData true

Encoding UTF-8

License GPL-3

RoxygenNote 7.3.2

NeedsCompilation no

Author Damiano Fantini [aut, cre]

Repository CRAN

Date/Publication 2025-08-25 18:40:02 UTC

Contents

`custom_grep` *Retrieve Text Between XML Tags*

Description

Extract text form a string containing XML or HTML tags. Text included between tags of interest will be returned. If multiple tagged substrings are found, they will be returned as different elements of a list or character vector.

Usage

```
custom_grep(xml_data, tag, format = "list")
```

Arguments

xml_data	String (of class character and length 1): corresponds to the PubMed record or any string including XML/HTML tags.
tag	String (of class character and length 1): the tag of interest (does NOT include < > chars).
format	c("list", "char"): specifies the format for the output.

Details

The 'custom_grep()' function is now obsolete. This is a helper function that will be replaced by 'easyPubMed:::EPM_custom_grep()', an internal function that won't be exported. The 'custom_grep()' function will be retired in 2026.

Value

List or vector where each element corresponds to an in-tag substring.

Author(s)

Damiano Fantini <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
try({
  ## extract substrings based on regular expressions
  string_01 <- paste0(
    "The itsy bitsy <strong>spider</strong> ",
    "Went up the water spout. Down came the rain ",
    "And <strong>washed the spider out</strong>.")
  print(string_01)
  custom_grep(xml_data = string_01, tag = "strong", format = "char")
  custom_grep(xml_data = string_01, tag = "strong", format = "list")
}, silent = TRUE)
```

---

easyPubMed-class *Class easyPubMed.*

Description

Class easyPubMed defines objects that represent PubMed Query jobs and the corresponding results. Briefly, these objects are initialized using information that will guide the communication with the NCBI Entrez server. Also, easyPubMed objects are used to store raw and processed data retrieved from Pubmed.

Usage

```
## S4 method for signature 'easyPubMed'
initialize(.Object, query_string, job_info)
```

Arguments

.Object	The easyPubMed object being built.
query_string	String (character vector of length 1) corresponding to the user-provided text of the query to be submitted to PubMed.
job_info	List, this should be the output of 'EPM_job_split()'.

Slots

query String (character vector of length 1) corresponding to the PubMed request submitted by the user.

meta List including meta information about the PubMed Query job.

uilist List including all unique identifiers corresponding to the Pubmed records returned by the query. Can be empty.

raw List including the raw data (in 'xml' or 'medline' format) retrieved from the NCBI eFetch server. Can be empty.

data Data.frame including processed data based on the xml raw data retrieved from PubMed.

misc List including additional information.

Author(s)

Damiano Fantini <damiano.fantini@gmail.com>

---

epm_fetch *Fetch Raw Records from Pubmed.*

Description

Fetch raw PubMed records from PubMed. Records can be downloaded in text or xml format and stored into a local object or written to local files.

Usage

```
epm_fetch(
  x,
  format = "xml",
  api_key = NULL,
  write_to_file = FALSE,
  outfile_path = NULL,
  outfile_prefix = NULL,
  store_contents = TRUE,
```

```
  encoding = "UTF-8",
  verbose = TRUE
)
```

Arguments

x	An 'easyPubMed' object.
format	String, the desired format for the raw records. This argument must take one of the following values: 'c("uilist", "medline", "xml")' and defaults to '"xml"'.
api_key	String, corresponding to the NCBI API token (if available). NCBI token strings can be requested from NCBI. Record download will be faster if a valid NCBI token is used. This argument can be 'NULL'.
write_to_file	Logical of length 1. Shall raw records be written to a file on the local machine. It defaults to 'FALSE'.
outfile_path	Path to the folder on the local machine where files will be saved (if 'write_to_file' is 'TRUE'). It must point to an already existing directory. If 'NULL', the working directory will be used.
outfile_prefix	String, prefix that will be added to the name of each file written to the local machine. This argument is parsed only when 'write_to_file' is 'TRUE'. If 'NULL', an arbitrary prefix will be added (easypubmed_job_YYYYMMDDHHMM).
store_contents	Logical of length 1. Shall raw records be stored in the 'easyPubMed' object. It defaults to 'TRUE'. It may convenient to switch this to 'FALSE' when downloading large number of records. If 'store_contents' is 'FALSE', 'write_to_file' must be 'TRUE'.
encoding	String, the encoding of the records retrieved from PubMed. Typically, this is 'UTF-8'.
verbose	Logical, shall details about the progress of the operation be printed to console.

Value

an easyPubMed object.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  x <- epm_query(query_string = 'Damiano Fantini[AU] AND "2018"[PDAT]')
  x <- epm_fetch(x = x, format = 'uilist')
```

```
  x
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

---

epm_import_xml *Import PubMed Records from Local Files.*

Description

Read one or more text files including XML-decorated raw PubMed records and rebuild an 'easyPubMed' object. The function expects all files to be generated from the same query using 'easyPubMed>3.0' and the 'epm_fetch()' function setting 'write_to_file' to 'TRUE'. This function can import a fraction or all of the files resulting from a single query. Files resulting from non-compatible fetch jobs will be dropped.

Usage

```
epm_import_xml(x)
```

Arguments

x	Character vector, the paths to text files including XML-decorated raw PubMed records saved using 'easyPubMed>3.0'.

Value

an 'easyPubMed' object including raw XML PubMed records.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  x <- epm_query(query_string = 'Damiano Fantini[AU] AND "2018"[PDAT]')
  x <- epm_fetch(x = x, format = 'xml', write_to_file = TRUE,
                 outfile_prefix = 'test', store_contents = FALSE)
  y <- epm_import_xml('test_batch_01.txt')
  tryCatch({unlink('test_batch_01.txt')}, error = function(e) { NULL })
```

```
  print(paste0('       Raw Record Num (fetched): ',
               getEPMMeta(x)$raw_record_num))
  print(paste0('Raw Record Num (read & rebuilt): ',
               getEPMMeta(y)$raw_record_num))
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

---

epm_parse *Extract Information from a Raw PubMed Record.*

Description

Read a raw PubMed record, identify XML tags, extract information and cast it into a structured data.frame. The expected input is an XML-tag-decorated string corresponding to a single PubMed record. Information about article title, authors, affiliations, journal name and abbreviation, publication date, references, and keywords are returned.

Usage

```
epm_parse(
  x,
  max_authors = 10,
  autofill_address = TRUE,
  compact_output = TRUE,
  include_abstract = TRUE,
  max_references = 150,
  ref_id_type = "doi",
  verbose = TRUE
)
```

Arguments

x	An 'easyPubMed' object. The object must include raw records (n>0) downloaded in the 'xml' format.
max_authors	Numeric, maximum number of authors to retrieve. If this is set to -1, only the last author is extracted. If this is set to 1, only the first author is returned. If this is set to 2, the first and the last authors are extracted. If this is set to any other positive number (i), up to the leading (n-1) authors are retrieved together with the last author. If this is set to a number larger than the number of authors in a record, all authors are returned. Note that at least 1 author has to be retrieved, therefore a value of 0 is not accepted (coerced to -1).
autofill_address	Logical, shall author affiliations be propagated within each record to fill missing values.

compact_output	Logical, shall record data be returned in a compact format where each row is a single record and author names are collapsed together. If 'FALSE', each row corresponds to a single author of the publication and the record-specific data are recycled for all included authors (legacy approach).
include_abstract	Logical, shall abstract text be included in the output data.frame. If 'FALSE', the abstract text column is populated with a missing value.
max_references	Numeric, maximum number of references to return (for each PubMed record).
ref_id_type	String, must be one of the following values: 'c('pmid', 'doi')'. Type of identifier used to describe citation references.
verbose	Logical, shall details about the progress of the operation be printed to console.

Value

an easyPubMed object including a data.frame ('data' slot) that stores information extracted from its raw XML PubMed records.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.7)
try({
  x <- epm_query(query_string = 'Damiano Fantini[AU] AND "2018"[PDAT]')
  x <- epm_fetch(x = x, format = 'xml')
  x <- epm_parse(x, include_abstract = FALSE, max_authors = 1)
  get_epm_data(x)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

---

epm_parse_record *Extract Information from a Raw PubMed Record.*

Description

Read a raw PubMed record, identify XML tags, extract information and cast it into a structured 'data.frame'. The expected input is an XML-tag-decorated string corresponding to a single PubMed record. Information about article title, authors, affiliations, journal name and abbreviation, publication date, references, and keywords are returned.

Usage

```
epm_parse_record(
  pubmedArticle,
  max_authors = 15,
  autofill_address = TRUE,
  compact_output = TRUE,
  include_abstract = TRUE,
  max_references = 1000,
  ref_id_type = "pmid"
)
```

Arguments

pubmedArticle	String, this is an XML-tag-decorated raw PubMed record.
max_authors	Numeric, maximum number of authors to retrieve. If this is set to -1, only the last author is extracted. If this is set to 1, only the first author is returned. If this is set to 2, the first and the last authors are extracted. If this is set to any other positive number (i), up to the leading (n-1) authors are retrieved together with the last author. If this is set to a number larger than the number of authors in a record, all authors are returned. Note that at least 1 author has to be retrieved, therefore a value of 0 is not accepted (coerced to -1).
autofill_address	Logical, shall author affiliations be propagated within each record to fill missing values.
compact_output	Logical, shall record data be returned in a compact format where each row is a single record and author names are collapsed together. If 'FALSE', each row corresponds to a single author of the publication and the record-specific data are recycled for all included authors.
include_abstract	Logical, shall abstract text be included in the output data.frame. If 'FALSE', the abstract text column is populated with a missing value.
max_references	Numeric, maximum number of references to return (for each PubMed record).
ref_id_type	String, must be one of the following values: 'c('pmid', 'doi')'. Type of identifier used to describe citation references.

Value

a data.frame including information extracted from a raw XML PubMed record.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
data(epm_samples)
x <- epm_samples$bladder_cancer_2018$demo_data_03$raw[[1]]
epm_parse_record(x)
```

---

epm_query *Search for PubMed Records.*

Description

Query PubMed (Entrez) via the PubMed API eSearch utility. Calling this function results in submitting a query to the NCBI EUtils server and then capturing and parsing the response. The number of records expected to be returned by the query is determined. If this number is bigger than n=10,000, the record retrieval job is automatically split in a list of smaller manageable sub-queries. This function returns an "easyPubMed" object, which includes all information required to retrieve PubMed records using the `epm_fetch()` function.

Usage

```
epm_query(query_string, api_key = NULL, verbose = TRUE)
```

Arguments

`query_string`	String (character vector of length 1), corresponding to the query string.
`api_key`	String (character vector of length 1), corresponding to the NCBI API key. Can be 'NULL'.
`verbose`	logical, shall progress information be printed to console. Defaults to 'TRUE'.

Details

This function will use "query_string" for querying PubMed. The Query Term can include one or multiple words, as well as the standard PubMed operators (AND, OR, NOT) and tags (i.e., [AU], [PDAT], [Affiliation], and so on).

Value

An easyPubMed object which includes no PubMed records.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  qry <- 'Damiano Fantini[AU] AND "2018"[PDAT]'
  epm_query(query_string = qry, verbose = FALSE)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

---

epm_query_by_fulltitle

Query PubMed by Full-length Title.

Description

Execute a PubMed query using a full-length publication title as query string. Tokenization and stopword removal is automatically performed. The goal is to mimic a Pubmed citation matching search. Because of this approach, it is possible that a query by full-length title may return more than one record.

Usage

```
epm_query_by_fulltitle(
  fulltitle,
  field = "[Title]",
  api_key = NULL,
  verbose = TRUE
)
```

Arguments

`fulltitle`	String (character vector of length 1) that corresponds to the full-length publication title used for querying PubMed (titles should be used as is, without adding trailing filter tags).
`field`	String (character vector of length 1). This indicates the PubMed record field where the full-length string (fulltitle) should be searched in. By default, this points to the 'Title' field. However, the field can be changed (always use fields supported by PubMed) as required by the user (for example, to attempt an exact-match query using a specific sentence included in the abstract of a record).

api_key	String (character vector of length 1), corresponding to the NCBI API key. Can be 'NULL'.
verbose	Logical, shall details about the progress of the operation be printed to console.

Value

an easyPubMed object.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  q <- 'Analysis of Mutational Signatures Using the mutSignatures R Library.'
  epm_query_by_fulltitle(q)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

---

epm_query_by_pmid *Query PubMed by PMIDs.*

Description

Query PubMed using a list of PubMed record identifiers (PMIDs) as input. The list of identifiers is automatically split into a series of manageable-sized chunks (max n=50 PMIDs per chunk).

Usage

```
epm_query_by_pmid(pmids, api_key = NULL, verbose = TRUE)
```

Arguments

pmids	Vector (character or numeric), list of Pubmed record identifiers (PMIDs). Values will be coerced to character.
api_key	String (character vector of length 1), corresponding to the NCBI API key. Can be 'NULL'.
verbose	Logical, shall details about the progress of the operation be printed to console.

Value

an easyPubMed object.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  my_pmids <- c(34097668, 34097669, 34097670)
  epm_query_by_pmid(my_pmids)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

---

epm_samples *Preprocessed PubMed Records and Data*

Description

This dataset includes a collection of sample data obtained from PubMed records and saved in different formats. This dataset is used to demonstrate specific functionalities of the 'easyPubMed' R library. Each element in the 'epm_samples' list corresponds to a different input or intermediate object.

Usage

```
data("epm_samples")
```

Format

The dataset is formatted as a list including 4 elements:

* 'bladder_cancer_2018': List of 4

* 'bladder_cancer_40y': List of 1

* 'fx': List of 5

Examples

```
## Display some contents
data("epm_samples")
# Display Query String used for collecting the data
print(epm_samples$bladder_cancer_2018$demo_data_01)
```

`epm_stopwords` *PubMed Query Stopwords*

Description

Collection of 133 Stopwords that can be removed from query strings to improve the accuracy of exact-match PubMed queries.

Usage

```
data("epm_stopwords")
```

Format

A character vector including all PubMed stopwords tat are typically filtered out from queries.

Details

Number of stopwords included, n=133.

Examples

```
## Display some contents
data("epm_stopwords")
head(epm_stopwords)
```

`fetchEPMData` *Method fetchEPMData.*

Description

Retrieve PubMed records for an ‘easyPubMed‘ object.

Usage

```
fetchEPMData(x, params)

## S4 method for signature 'easyPubMed,list'
fetchEPMData(x, params)
```

Arguments

x	an easyPubMed-class object.
params	list including parameters to tune the record retrieval job. For more info, see '?easyPunMed:::EPM_validate_fetch_params'.

---

fetch_pubmed_data *Retrieve PubMed Data in XML or TXT Format*

Description

Retrieve PubMed records from Entrez following a search performed via the get_pubmed_ids() function. Data are downloaded in the XML or TXT format and are retrieved in batches of up to 5000 records.

Usage

```
fetch_pubmed_data(
  pubmed_id_list,
  retstart = 0,
  retmax = 500,
  format = "xml",
  encoding = "UTF8",
  api_key = NULL,
  verbose = TRUE
)
```

Arguments

pubmed_id_list	An easyPubMed object.
retstart	Integer (>=0): this argument is ignored.
retmax	Integer (>=1): this argument is ignored.
format	String: element specifying the output format. The following values are allowed: c("xml", "medline", "uilist").
encoding	String, the encoding of the records retrieved from Pubmed. This argument is ignored and set to 'UTF-8'.
api_key	String, corresponding to the NCBI API token (if available). NCBI token strings can be requested from NCBI. Record download will be faster if a valid NCBI token is used. This argument can be NULL.
verbose	Logical, shall details about the progress of the operation be printed to console.

Details

The 'fetch_pubmed_data()' function is now obsolete. You should use the 'epm_fetch()' function instead. Please, have a look at the manual or the vignette. The 'fetch_pubmed_data()' function will be retired in 2026.

Value

Character vector of length >= 1. If format is set to "xml" (default), a single String including all PubMed records (decorated with XML tags) is returned. If a different format is selected, a vector of strings is returned, where each element corresponds to a line of the output document.

Author(s)

Damiano Fantini <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/ https://www.ncbi.nlm.nih.gov/books/NBK25499/table/chapter4.T._valid_values_of__retmode_and/

Examples

```
## Example 01: retrieve PubMed record Unique Identifiers (uilist)
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  q <- 'Damiano Fantini[AU] AND "2018"[PDAT]'
  x <- get_pubmed_ids(pubmed_query_string = q)
  y <- fetch_pubmed_data(x, format = "uilist")
  y
}, silent = TRUE)
setTimeLimit(elapsed = Inf)

## Not run:
## Example 02: retrieve data in XML format
q <- 'Damiano Fantini[AU] AND "2018"[PDAT]'
x <- epm_query(query_string = q)
y <- fetch_pubmed_data(x, format = "xml")
y

## End(Not run)
```

`getEPMData` *Method getEPMData.*

Description

Retrieve processed data from an ‘easyPubMed‘ object.

Usage

```
getEPMData(x)

## S4 method for signature 'easyPubMed'
getEPMData(x)
```

Arguments

x	an object of class ‘easyPubMed‘.

---

getEPMJobList *Method getEPMJobList.*

Description

Retrieve the list of record retrieval sub-jobs from an ‘easyPubMed‘ object. Record retrieval sub-jobs are stored in a ‘data.frame‘ and each row corresponds to an independent non-overlapping PubMed query. This ‘data.frame‘ guides the record retrieval process. The ‘data.frame‘ is obtained from the ’misc’ slot of an ‘easyPubMed‘ object.

Usage

```
getEPMJobList(x)

## S4 method for signature 'easyPubMed'
getEPMJobList(x)
```

Arguments

x	an object of class ‘easyPubMed‘.

---

getEPMMeta *Method getEPMMeta.*

Description

Retrieve meta data from an ‘easyPubMed‘ object.

Usage

```
getEPMMeta(x)

## S4 method for signature 'easyPubMed'
getEPMMeta(x)
```

Arguments

x	an object of class ‘easyPubMed‘.

getEPMMisc *Method getEPMMisc.*

Description

Retrieve miscellaneous information stored in an ‘easyPubMed‘ object.

Usage

```
getEPMMisc(x)

## S4 method for signature 'easyPubMed'
getEPMMisc(x)
```

Arguments

x	an object of class ‘easyPubMed‘.

getEPMQuery *Method getEPMQuery.*

Description

Retrieve the user-provided query string from an ‘easyPubMed‘ object.

Usage

```
getEPMQuery(x)

## S4 method for signature 'easyPubMed'
getEPMQuery(x)
```

Arguments

x	an object of class ‘easyPubMed‘.

getEPMRaw *Method getEPMRaw.*

Description

Retrieve the raw PubMed record data stored in an ‘easyPubMed‘ object.

Usage

```
getEPMRaw(x)

## S4 method for signature 'easyPubMed'
getEPMRaw(x)
```

Arguments

x	an object of class ‘easyPubMed‘.

getEPMUilist *Method getEPMUilist.*

Description

Retrieve the list of unique record identifiers (PMIDs) from an ‘easyPubMed‘ object.

Usage

```
getEPMUilist(x)

## S4 method for signature 'easyPubMed'
getEPMUilist(x)
```

Arguments

x	an object of class ‘easyPubMed‘.

get_epm_data *Get Processed Data from an easyPubMed Object.*

Description

Obtain Processed Data that were extracted from a list of PubMed records. This is a wrapper function that calls the 'getEPMData()' method. This function returns contents from the 'data' slot.

Usage

```
get_epm_data(x)
```

Arguments

x	An 'easyPubMed' object.

Value

a 'data.frame' including processed data from an 'easyPubMed' object.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  x <- epm_query(query_string = 'Damiano Fantini[AU] AND "2018"[PDAT]')
  x <- epm_fetch(x)
  x <- epm_parse(x, max_references = 5, max_authors = 5)
  get_epm_data(x)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

get_epm_meta *Get Meta Data from an easyPubMed Object.*

Description

Request Meta Data from an 'easyPubMed' object. This is a wrapper function that calls the 'getEPMMeta()' method. This function returns contents from the 'meta' slot.

Usage

```
get_epm_meta(x)
```

Arguments

x	An 'easyPubMed' object.

Value

a list including meta data from an 'easyPubMed' object.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  x <- epm_query(query_string = 'Damiano Fantini[AU] AND "2018"[PDAT]')
  get_epm_meta(x)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

get_epm_raw *Get Raw Data from an easyPubMed Object.*

Description

Request Raw Data from an 'easyPubMed' object. This is a wrapper function that calls the 'getEPM-Raw()' method. This function returns contents from the 'raw' slot.

Usage

```
get_epm_raw(x)
```

Arguments

x	An 'easyPubMed' object.

Value

a list including raw data from an 'easyPubMed' object.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  x <- epm_query(query_string = 'Damiano Fantini[AU] AND "2018"[PDAT]')
  x <- epm_fetch(x)
  get_epm_raw(x)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

get_epm_uilist *Get PubMed Record Identifiers from an easyPubMed Object.*

Description

Request the list of unique PubMed Record Identifiers that are contained in an 'easyPubMed' object. This function is a wrapper function calling the 'getEPMUilist()' method. This function returns contents from the 'uilist' slot.

Usage

```
get_epm_uilist(x)
```

Arguments

x	An 'easyPubMed' object.

Value

a character vector including a list of unique record identifiers from an 'easyPubMed' object.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  x <- epm_query(query_string = 'Damiano Fantini[AU] AND "2018"[PDAT]')
  x <- epm_fetch(x)
  get_epm_uilist(x)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

`get_pubmed_ids` *Simple PubMed Record Search*

Description

Query PubMed (Entrez) in a simple way via the PubMed API eSearch function. Calling this function results in posting the query results on the PubMed History Server. This allows later access to the resulting data via the fetch_pubmed_data() function, or other easyPubMed functions. NOTE: this function has become obsolete. You should use the epm_query() function instead. Please, have a look at the manual or the vignette. The `get_pubmed_ids()` function will be retired in 2026.

Usage

```
get_pubmed_ids(pubmed_query_string, api_key = NULL)
```

Arguments

`pubmed_query_string`	String (character vector of length 1), corresponding to the query string used for querying PubMed.
`api_key`	String (character vector of length 1), corresponding to the NCBI API key. Can be NULL.

Details

This function will use the String provided as argument for querying PubMed via the eSearch function of the PubMed API. The Query Term can include one or multiple words, as well as the standard PubMed operators (AND, OR, NOT) and tags (i.e., [AU], [PDAT], [Affiliation], and so on). ESearch will post the UIDs resulting from the search operation onto the History server so that they can be used directly in a subsequent fetchPubmedData() call.

Value

An easyPubMed object which includes no PubMed records.

Author(s)

Damiano Fantini, <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  qry <- 'Damiano Fantini[AU] AND "2018"[PDAT]'
  get_pubmed_ids(pubmed_query_string = qry)
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

`parseEPMData` *Method parseEPMData.*

Description

Extract, parse and format information from raw PubMed records stored in an 'easyPubMed' object.

Usage

```
parseEPMData(x, params)

## S4 method for signature 'easyPubMed,list'
parseEPMData(x, params)
```

Arguments

`x`	an easyPubMed-class object
`params`	list including parameters to tune the record data parsing job. For more info, see '?easyPunMed:::EPM_validate_parse_params'.

`print,easyPubMed-method` *Print method of the easyPubMed Class.*

Description

Print method of the easyPubMed Class.

Usage

```
## S4 method for signature 'easyPubMed'
print(x)
```

Arguments

x	the ‘easyPubMed‘ object being shown.

---

setEPMData *Method setEPMData.*

Description

Attach (or replace) processed data to an ‘easyPubMed‘ object.

Usage

```
setEPMData(x, y)

## S4 method for signature 'easyPubMed,data.frame'
setEPMData(x, y)
```

Arguments

x	an object of class ‘easyPubMed‘.
y	‘data.frame‘ including processed data.

---

setEPMJobList *Method setEPMJobList.*

Description

Attach (or replace) the list of record retrieval sub-jobs to an ‘easyPubMed‘ object. Record retrieval sub-jobs are stored in a data.frame and each row corresponds to an independent non-overlapping PubMed query. This ‘data.frame‘ guides the record retrieval process. The ‘data.frame‘ is written into the ‘misc‘ slot of an ‘easyPubMed‘ object.

Usage

```
setEPMJobList(x, y)

## S4 method for signature 'easyPubMed,data.frame'
setEPMJobList(x, y)
```

Arguments

x	an object of class ‘easyPubMed‘.
y	‘data.frame‘ including the list of PubMed record retrieaval sub-jobs.

setEPMMeta *Method setEPMMeta.*

Description

Attach (or replace) meta data to an ‘easyPubMed‘ object.

Usage

```
setEPMMeta(x, y)

## S4 method for signature 'easyPubMed,list'
setEPMMeta(x, y)
```

Arguments

x	an object of class ‘easyPubMed‘.
y	list including meta data information.

setEPMMisc *Method setEPMMisc.*

Description

Attach (or replace) miscellaneous information to an ‘easyPubMed‘ object.

Usage

```
setEPMMisc(x, y)

## S4 method for signature 'easyPubMed,list'
setEPMMisc(x, y)
```

Arguments

x	an object of class ‘easyPubMed‘.
y	list including miscellaneous data and information.

setEPMQuery *Method setEPMQuery.*

Description

Attach (or replace) a user-provided query string to an ‘easyPubMed‘ object.

Usage

```
setEPMQuery(x, y)

## S4 method for signature 'easyPubMed,character'
setEPMQuery(x, y)
```

Arguments

x	an object of class ‘easyPubMed‘.
y	string (character vector of length 1) corresponding to a PubMed query string.

setEPMRaw *Method setEPMRaw.*

Description

Attach (or replace) raw PubMed record data to an ‘easyPubMed‘ object.

Usage

```
setEPMRaw(x, y)

## S4 method for signature 'easyPubMed,list'
setEPMRaw(x, y)
```

Arguments

x	an object of class ‘easyPubMed‘.
y	list of PubMed records (raw data).

setEPMUilist *Method setEPMUilist.*

Description

Attach (or replace) the list of unique record identifiers (PMIDs) to an 'easyPubMed' object.

Usage

```
setEPMUilist(x, y)

## S4 method for signature 'easyPubMed,list'
setEPMUilist(x, y)
```

Arguments

x	an object of class 'easyPubMed'.
y	list of unique PubMed record identifiers (PMIDs).

show,easyPubMed-method *Show method of the easyPubMed Class.*

Description

Show method of the easyPubMed Class.

Usage

```
## S4 method for signature 'easyPubMed'
show(object)
```

Arguments

object	the 'easyPubMed' object being shown.

table_articles_byAuth *Extract Publication and Affiliation Data from PubMed Records*

Description

Extract Publication Info from PubMed records and cast data into a data.frame where each row corresponds to a different author. It is possible to limit data extraction to first authors or last authors only, or get information about all authors of each PubMed record.

Usage

```
table_articles_byAuth(
  pubmed_data,
  included_authors = "all",
  max_chars = 500,
  autofill = TRUE,
  dest_file = NULL,
  getKeywords = TRUE,
  encoding = "UTF8"
)
```

Arguments

pubmed_data	PubMed Data in XML format: typically, an XML file resulting from a batch_pubmed_download() call or an XML object, result of a fetch_pubmed_data() call.
included_authors	Character: c("first", "last", "all"). Only includes information from the first, the last or all authors of a PubMed record.
max_chars	This argument is ignored. In this version of the function, the whole Abstract Text is returned.
autofill	Logical. If TRUE, missing affiliations are imputed according to the available values (from the same article).
dest_file	String (character of length 1). Name of the file that will be written for storing the output. If NULL, no file will be saved.
getKeywords	This argument is ignored. In this version of the function MeSH terms and codes (i.e., keywords) are parsed by default.
encoding	The encoding of an input/output connection can be specified by name (for example, "ASCII", or "UTF-8", in the same way as it would be given to the function base::iconv(). See iconv() help page for how to find out more about encodings that can be used on your platform. Here, we recommend using "UTF-8".

Details

The ‘table_articles_byAuth()‘ function is now obsolete. You should use the ‘epm_parse()‘ function instead. Please, have a look at the manual or the vignette. The ‘table_articles_byAuth()‘ function will be retired in 2026.

Value

Data frame including the following fields: ‘c("pmid", "doi", "title", "abstract", "year", "month", "day", "jabbrv", "journal", "keywords", "mesh", "lastname", "firstname", "address", "email")‘.

Author(s)

Damiano Fantini <damiano.fantini@gmail.com>

References

https://www.data-pulse.com/dev_site/easypubmed/

Examples

```
# Note: a time limit can be set in order to kill the operation when/if
# the NCBI/Entrez server becomes unresponsive.
setTimeLimit(elapsed = 4.9)
try({
  q0 <- 'Damiano Fantini[AU] AND "2018"[PDAT]'
  q1 <- easyPubMed::get_pubmed_ids(pubmed_query_string = q0)
  q2 <- fetch_pubmed_data(pubmed_id_list = q1)
  df <- table_articles_byAuth(q2, included_authors = 'first')
  df[, c('pmid', 'lastname', 'jabbrv', 'year', 'month', 'day')]
}, silent = TRUE)
setTimeLimit(elapsed = Inf)
```

Index