

# Package ‘pmparser’

December 1, 2023

**Title** Create and Maintain a Relational Database of Data from PubMed/MEDLINE

**Version** 1.0.18

**Description** Provides a simple interface for extracting various elements from the publicly available PubMed XML files, incorporating PubMed's regular updates, and combining the data with the NIH Open Citation Collection. See Schoenbachler and Hughey (2021) <[doi:10.7717/peerj.11071](https://doi.org/10.7717/peerj.11071)>.

**URL** <https://pmparser.hugheylab.org>,  
<https://github.com/hugheylab/pmparser>

**License** GPL-2

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Depends** R (>= 3.6)

**Imports** curl (>= 4.3.2), data.table (>= 1.12.2), DBI (>= 1.1.0),  
foreach (>= 1.5.0), glue (>= 1.4.2), iterators (>= 1.0.12),  
jsonlite (>= 1.7.0), R.utils (>= 2.10.1), RCurl (>= 1.98),  
withr (>= 2.3.0), xml2 (>= 1.3.3)

**Suggests** bigrquery (>= 1.3.2), doParallel (>= 1.0.16), RMariaDB (>= 1.0.9), RPostgres (>= 1.2.0), RSQLite (>= 2.2.0), testthat (>= 2.3.2), knitr, rmarkdown

**SystemRequirements** head, unzip, sqlite

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Jake Hughey [aut, cre],  
Josh Schoenbachler [aut],  
Elliot Outland [aut]

**Maintainer** Jake Hughey <[jakejhughey@gmail.com](mailto:jakejhughey@gmail.com)>

**Repository** CRAN

**Date/Publication** 2023-12-01 11:00:02 UTC

## R topics documented:

getCitation . . . . .	2
getPgParams . . . . .	3
modifyPubmedDb . . . . .	4
parseElement . . . . .	5

<b>Index</b>	<b>9</b>
--------------	----------

---

getCitation	<i>Get public-domain citation data</i>
-------------	--

---

### Description

Get the latest version of the NIH Open Citation Collection from figshare [here](#), and optionally write it to the database. This function requires the shell command `unzip`, available by default on most Unix systems. This function should not normally be called directly, as it is called by `modifyPubmedDb()`.

### Usage

```
getCitation(
  localDir,
  filename = "open_citation_collection.zip",
  nrows = Inf,
  tableSuffix = NULL,
  overwrite = FALSE,
  con = NULL,
  checkMd5 = TRUE
)
```

### Arguments

<code>localDir</code>	String indicating path to directory containing the citation file or to which the citation file should be downloaded.
<code>filename</code>	String indicating name of the citation file. This should not normally be changed from the default.
<code>nrows</code>	Number indicating how many rows of the citation file to read. This should not normally be changed from the default.
<code>tableSuffix</code>	String indicating suffix, if any, to append to the table name.
<code>overwrite</code>	Logical indicating whether to overwrite an existing table.
<code>con</code>	Connection to the database, created using <code>DBI::dbConnect()</code> .
<code>checkMd5</code>	Logical indicating whether to download the citation file if the MD5 sums of the local and remote versions do not match. This should not normally be changed from the default.

**Value**

If con is NULL, the function returns a data.table with columns citing\_pmid and cited\_pmid. Beware this is a large table and could swamp the machine's memory. If con is not NULL, the function returns NULL invisibly.

**See Also**

[parsePmidStatus\(\)](#), [modifyPubmedDb\(\)](#)

**Examples**

```
## Not run:  
dCitation = getCitation('.')  
  
## End(Not run)
```

---

getPgParams

*Get Postgres connection parameters*

---

**Description**

This is a helper function to get parameters from a .pgpass file. See [here](#) for details.

**Usage**

```
getPgParams(path = "~/ .pgpass")
```

**Arguments**

path                    Path to .pgpass file.

**Value**

A data.table with one row for each set of parameters.

**See Also**

[modifyPubmedDb\(\)](#)

**Examples**

```
pg = getPgParams(system.file('extdata', 'pgpass', package = 'pmparser'))
```

---

 modifyPubMedDb

*Create or update a PubMed database*


---

### Description

This function downloads PubMed/MEDLINE XML files, parses them, and adds the information to the database, then downloads the NIH Open Citation Collection and adds it to the database. Only the most recent version of each PMID is retained. Parsing of XML files will use a parallel backend if one is registered, such as with `doParallel::registerDoParallel()`.

### Usage

```
modifyPubMedDb(
  localDir,
  dbname,
  dbtype = c("postgres", "mariadb", "mysql", "sqlite"),
  nFiles = Inf,
  retry = TRUE,
  nCitations = Inf,
  mode = c("create", "update"),
  ...
)
```

### Arguments

<code>localDir</code>	Directory in which to download the files from PubMed.
<code>dbname</code>	Name of database.
<code>dbtype</code>	Type of database, either 'postgres', 'mariadb', 'mysql', or 'sqlite'. Make sure to install the corresponding DBI driver package first: RPostgres, RMariaDB (for both 'mariadb' and 'mysql'), or RSQLite. Due to the large size of the database, SQLite is recommended only for small-scale testing.
<code>nFiles</code>	Maximum number of xml files to parse that are not already in the database. This should not normally be changed from the default.
<code>retry</code>	Logical indicating whether to retry parsing steps that fail.
<code>nCitations</code>	Maximum number of rows of the citation file to read. This should not normally be changed from the default.
<code>mode</code>	String indicating whether to create the database using the baseline files or to update the database using the update files.
<code>...</code>	Other arguments passed to <code>DBI::dbConnect()</code> .

### Value

NULL, invisibly. Tab-delimited log files will be created in a logs folder in `localDir`.

**See Also**

[parsePmidStatus\(\)](#), [getCitation\(\)](#), [getPgParams\(\)](#)

**Examples**

```
## Not run:  
modifyPubmedDb('.', 'pmdb', mode = 'create')  
  
## End(Not run)
```

---

parseElement

*Parse elements from a PubMed XML file*

---

**Description**

Elements are parsed according to the MEDLINE®PubMed® XML Element Descriptions and their Attributes [here](#). These functions should not normally be called directly, as they are called by [modifyPubmedDb\(\)](#).

**Usage**

```
parsePmidStatus(rawXml, filename, con = NULL, tableSuffix = NULL)  
  
parseArticleId(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseArticle(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parsePubHistory(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseJournal(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parsePubType(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseMesh(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseKeyword(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseGrant(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseChemical(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseDataBank(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseComment(pmXml, dPmid, con = NULL, tableSuffix = NULL)  
  
parseAbstract(pmXml, dPmid, con = NULL, tableSuffix = NULL)
```

```
parseOther(pmXml, dPmid, con = NULL, tableSuffix = NULL)
```

```
parseAuthor(pmXml, dPmid, con = NULL, tableSuffix = NULL)
```

```
parseInvestigator(pmXml, dPmid, con = NULL, tableSuffix = NULL)
```

## Arguments

rawXml	An xml document obtained by loading a PubMed XML file using <code>xml2::read_xml()</code> .
filename	A string that will be added to a column <code>xml_filename</code> .
con	Connection to the database, created using <code>DBI::dbConnect()</code> .
tableSuffix	String to append to the table names.
pmXml	An xml nodeset derived from <code>rawXml</code> , such as that returned by <code>parsePmidStatus()</code> , where each node corresponds to a PMID.
dPmid	A <code>data.table</code> with one row for each node of <code>pmXml</code> , should have columns <code>pmid</code> , <code>version</code> , and possibly <code>xml_filename</code> .

## Value

`parsePmidStatus()` returns a list of two objects. The first is an xml nodeset in which each node corresponds to a `PubmedArticle` in the `rawXml` object. The second is a `data.table` with columns `pmid`, `version`, `xml_filename`, and `status`, in which each row corresponds to a `PubmedArticle` in the `rawXml` object or a deleted `pmid`. The `status` column is parsed from the `DeleteCitation` and `MedlineCitation` sections.

The following functions return a `data.table` or list of `data.tables` with columns from `dPmid` plus the columns specified.

`parseArticleId()`: a `data.table` with columns `id_type` and `id_value`, parsed from the `ArticleIDList` section. Only `id_types` "doi" and "pmc" are retained.

`parseArticle()`: a `data.table` with columns `title`, `language`, `vernacular_title`, `pub_model`, and `pub_date`, parsed from the `Article` section.

`parsePubHistory()`: a `data.table` with columns `pub_status` and `pub_date`, parsed from the `History` section.

`parseJournal()`: a `data.table` with columns `journal_name`, `journal_iso`, `pub_date`, `pub_year`, `pub_month`, `pub_day`, `medline_date`, `volume`, `issue`, and `cited_medium`, parsed from the `Journal` section.

`parsePubType()`: a `data.table` with columns `type_name` and `type_id`, parsed from the `PublicationTypeList` section.

`parseMesh()`: a list of three `data.tables` parsed mostly from the `MeshHeadingList` section. The first has column `indexing_method` (parsed from the `MedlineCitation` section), the second has columns `descriptor_pos`, `descriptor_name`, `descriptor_ui`, and `descriptor_major_topic`, the third has columns `descriptor_pos`, `qualifier_name`, `qualifier_ui`, and `qualifier_major_topic`.

`parseKeyword()`: a list of two `data.tables` parsed from the `KeywordList` section. The first has column `list_owner`, the second has columns `keyword_name` and `major_topic`.

parseGrant(): a list of two data.tables parsed from the GrantList section. The first has column complete, the second has columns grant\_id, acronym, agency, and country.

parseChemical(): a data.table with columns registry\_number, substance\_name, and substance\_ui, parsed from the ChemicalList section.

parseDataBank(): a data.table with columns data\_bank\_name and accession\_number, parsed from the DataBankList section.

parseComment(): a data.table with columns ref\_type and ref\_pmid, parsed from the CommentsCorrectionsList section.

parseAbstract(): a list of two data.tables parsed from the Abstract section. The first has column copyright. The second has columns text, label, and nlm\_category.

parseAuthor(): a list of data.tables parsed from the AuthorList section. The first is for authors and has columns author\_pos, last\_name, fore\_name, initials, suffix, valid, equal\_contrib, and collective\_name. The second is for affiliations and has columns author\_pos, affiliation\_pos, and affiliation. The third is for author identifiers and has columns author\_pos, source, and identifier. The fourth is for author affiliation identifiers and has columns author\_pos, affiliation\_pos, source, and identifier. The fifth is for the author list itself and has a column complete.

parseInvestigator(): a list of data.tables similar to those returned by parseAuthor(), except parsed from the InvestigatorList section, with column names containing "investigator" instead of "author", and where the first data.table lacks columns for equal\_contrib and collective\_name and the fifth data.table does not exist.

parseOther(): a list of data.tables parsed from the OtherAbstract and OtherID sections. The first has columns text, type, and language. The second has columns source and id\_value.

## See Also

[getCitation\(\)](#), [modifyPubmedDb\(\)](#)

## Examples

```
library('data.table')
library('xml2')

filename = 'pubmed20n1016.xml.gz'
rawXml = read_xml(system.file('extdata', filename, package = 'pmparser'))

pmidStatusList = parsePmidStatus(rawXml, filename)
pmXml = pmidStatusList[[1L]]
dPmidRaw = pmidStatusList[[2L]]
dPmid = dPmidRaw[status != 'Deleted', !'status']

dArticleId = parseArticleId(pmXml, dPmid)
dArticle = parseArticle(pmXml, dPmid)
dJournal = parseJournal(pmXml, dPmid)
dPubType = parsePubType(pmXml, dPmid)
dPubHistory = parsePubHistory(pmXml, dPmid)
meshRes = parseMesh(pmXml, dPmid)
keywordRes = parseKeyword(pmXml, dPmid)
grantRes = parseGrant(pmXml, dPmid)
```

```
dChemical = parseChemical(pmXml, dPmid)
dDataBank = parseDataBank(pmXml, dPmid)
dComment = parseComment(pmXml, dPmid)
abstractRes = parseAbstract(pmXml, dPmid)
authorRes = parseAuthor(pmXml, dPmid)
investigatorRes = parseInvestigator(pmXml, dPmid)
otherRes = parseOther(pmXml, dPmid)
```



# Index

DBI::dbConnect(), 2, 4, 6  
doParallel::registerDoParallel(), 4

getCitation, 2  
getCitation(), 5, 7  
getPgParams, 3  
getPgParams(), 5

modifyPubmedDb, 4  
modifyPubmedDb(), 2, 3, 5, 7

parseAbstract (parseElement), 5  
parseArticle (parseElement), 5  
parseArticleId (parseElement), 5  
parseAuthor (parseElement), 5  
parseChemical (parseElement), 5  
parseComment (parseElement), 5  
parseDataBank (parseElement), 5  
parseElement, 5  
parseGrant (parseElement), 5  
parseInvestigator (parseElement), 5  
parseJournal (parseElement), 5  
parseKeyword (parseElement), 5  
parseMesh (parseElement), 5  
parseOther (parseElement), 5  
parsePmidStatus (parseElement), 5  
parsePmidStatus(), 3, 5  
parsePubHistory (parseElement), 5  
parsePubType (parseElement), 5

xml2::read\_xml(), 6