Package ‘CITAN’

March 22, 2022

Version 2022.1.1
Date 2022-03-22
Type Package
License LGPL (>= 3)
Encoding UTF-8
URL https://github.com/gagolews/CITAN
BugReports https://github.com/gagolews/CITAN/issues
Title CITation ANalysis Toolpack
Description Supports quantitative research in scientometrics and bibliometrics. Provides various tools for preprocessing bibliographic data retrieved, e.g., from Elsevier's SciVerse Scopus, computing bibliometric impact of individuals, or modelling phenomena encountered in the social sciences.
This package is deprecated, see ‘agop’ instead.
Depends R (>= 3.2.0), agop, RSQLite
Imports stringi, DBI, grDevices, graphics, stats, utils
RoxygenNote 7.1.2
NeedsCompilation no
Author Marek Gagolewski [aut, cre] (<https://orcid.org/0000-0003-0637-6028>)
Maintainer Marek Gagolewski <marek@gagolewski.com>
Repository CRAN
Date/Publication 2022-03-21 23:20:02 UTC

R topics documented:

CITAN-package .......................................................... 2
as.character.authorinfo .................................................. 3
as.character.docinfo ..................................................... 4
dbExecQuery ............................................................ 5
Description

CITAN is a library of functions useful in — but not limited to — quantitative research in the field of scientometrics.

Details

The package is deprecated, see agop instead.

For the complete list of functions, call library(help="CITAN").

Author(s)

Marek Gagolewski
References


Kosmulski M. (2007). MAXPROD - A new index for assessment of the scientific output of an individual, and a comparison with the h-index, Cybermetrics 11(1).


as.character.authorinfo

Coerce an authorinfo object to character string

Description

Converts an object of class authorinfo to a character string. Such an object is returned by e.g. lbsGetInfoAuthors.

Usage

## S3 method for class 'authorinfo'
as.character(x, ...)
Arguments

x a single object of class authorinfo.

... unused.

Details

An authorinfo object is a list with the following components:

- IdAuthor — numeric; author's identifier in the table Biblio_Authors,
- Name — character; author's name.

Value

A character string

See Also

print.authorinfo, lbsSearchAuthors, lbsGetInfoAuthors

as.character.docinfo Coerce a docinfo object to character string

Description

Converts an object of class docinfo to a character string. Such an object is returned by e.g. lbsGetInfoDocuments.

Usage

## S3 method for class 'docinfo'
as.character(x, ...)

Arguments

x a single object of class docinfo.

... unused.

Details

A docinfo object is a list with the following components:

- IdDocument — numeric; document identifier in the table Biblio_Documents,
- Authors — list of authorinfo objects (see e.g. as.character.authorinfo).
- Title — title of the document,
- BibEntry — bibliographic entry,
- AlternativeId — unique character identifier,
dbExecQuery

- Pages — number of pages,
- Citations — number of citations,
- Year — publication year,
- Type — type of document, see lbsCreate.

Value

A character string

See Also

lbsSearchDocuments, as.character.authorinfo, print.docinfo, lbsGetInfoDocuments

dbExecQuery

Execute a query and free its resources

Description

Executes an SQL query and immediately frees all allocated resources.

Usage

dbExecQuery(conn, statement, rollbackOnError = FALSE)

Arguments

- conn: a DBI connection object.
- statement: a character string with the SQL statement to be executed.
- rollbackOnError: logical; if TRUE, then the function executes rollback on current transaction if an exception occurs.

Details

This function may be used to execute queries like CREATE TABLE, UPDATE, INSERT, etc.

It has its own exception handler, which prints out detailed information on caught errors.

See Also

dbSendQuery, dbClearResult, dbGetQuery
lbsAssess

**Calculate impact of given authors**

**Description**

Given a list of authors’ citation sequences, the function calculates values of many impact functions at a time.

**Usage**

```r
lbsAssess(
  citseq,
  f = list(length, index_h),
  captions = c("length", "index_h"),
  orderByColumn = 2,
  bestRanks = 20,
  verbose = T
)
```

**Arguments**

- `citseq`: list of numeric vectors, e.g. the output of `lbsGetCitations`.
- `f`: a list of \( n \) functions which compute the impact of an author. The functions must calculate their values using numeric vectors passed as their first arguments.
- `captions`: a list of \( n \) descriptive captions for the functions in `f`.
- `orderByColumn`: column to sort the results on. 1 for author names, 2 for the first function in `f`, 3 for the second, and so on.
- `bestRanks`: if not NULL, only a given number of authors with the greatest impact (for each function in `f`) will be included in the output.
- `verbose`: logical; TRUE to inform about the progress of the process.

**Value**

A data frame in which each row corresponds to the assessment results of some citation sequence. The first column stands for the authors’ names (taken from names(citseq)), the second for the valuation of \( f[[1]] \), the third for \( f[[2]] \), and so on. See Examples below.

**See Also**

`lbsConnect`, `lbsGetCitations`
Examples

```r
## Not run:
conn <- lbsConnect("Bibliometrics.db");
## ...
citseq <- lbsGetCitations(conn,
surveyDescription="Scientometrics", documentTypes="Article",
idAuthors=c(39264,39265,39266));
print(citseq);
## $'
# Author name
## 40116 34128 39122 29672 32343 32775 # IdDocument
## 11 4 1 0 0 0 # Citation count
## attr("IdAuthor")
## [1] 39264 # IdAuthor
##
## $'
# Author name
## 38680 38605 40035 40030 40124 39829 39745 29672
## 30 14 8 6 6 5 3 0
## attr("IdAuthor")
## [1] 39265
##
## $'
# Author name
## 29992 29672 29777 32906 33858 33864 34704
## 1 0 0 0 0 0 0
## attr("IdAuthor")
## [1] 39266
library("agop")
print(lbsAssess(citseq,
f=list(length, sum, index.h, index.g, function(x) index.rp(x,1),
function(x) sqrt(prod(index.lp(x,1)))),
captions=c("length", "sum", "index.h", "index.g", "index.w",
"index.lp1", "index.lpInf"));
```

```r
# Name length sum index.h index.g index.w index.lp1 index.lpInf
# 3 Xu Y. 8 72 5 8 7 8.573214 5.477226
# 2 Wang Y. 7 1 1 1 1 1.000000 1.000000
# 1 Liu X. 6 16 2 4 3 4.157609 3.316625
```

dbDisconnect(conn);
## End(Not run)
```

---

### lbsClear

**Clear a Local Bibliometric Storage**

**Description**

Clears a Local Bibliometric Storage by dropping all tables named Bibli.o_* and all views named ViewBiblio_*.

**Usage**

```r
lbsClear(conn, verbose = TRUE)
```
**lbsConnect**

*Connect to a Local Bibliometric Storage*

**Description**

Connects to a Local Bibliometric Storage handled by the SQLite engine (see *RSQLite* package documentation).

**Usage**

```r
lbsConnect(dbfilename)
```

**Arguments**

- `dbfilename` filename of an SQLite database.
Details

Do not forget to close the connection (represented by the connection object returned) with the `lbsDisconnect` function after use.

Please note that the database may be also accessed by using lower-level functions from the `DBI` package called on the returned connection object. The table-view structure of a Local Bibliometric Storage is presented in the man page of the `lbsCreate` function.

Value

An object of type `SQLiteConnection`, used to communicate with the SQLite engine.

See Also

`lbsCreate`, `lbsDisconnect`

Examples

```r
## Not run:
conn <- lbsConnect("Bibliometrics.db")
## ... 
lbsDisconnect(conn)
## End(Not run)
```

---

**lbsCreate**

*Create a Local Bibliometric Storage*

Description

Creates an empty Local Bibliometric Storage.

Usage

`lbsCreate(conn, verbose = TRUE)`

Arguments

- `conn` a connection object, see `lbsConnect`.
- `verbose` logical; TRUE to be more verbose.

Details

The function may be executed only if the database contains no tables named `Biblio_*` and no views named `ViewBiblio_*`.

The following SQL code is executed.
CREATE TABLE Biblio_Categories (  
    -- Source classification codes (e.g. ASJC)  
    IdCategory INTEGER PRIMARY KEY ASC,  
    IdCategoryParent INTEGER NOT NULL,  
    Description VARCHAR(63) NOT NULL,  
    FOREIGN KEY(IdCategoryParent) REFERENCES Biblio_Categories(IdCategory)  
);  

CREATE TABLE Biblio_Sources (  
    IdSource INTEGER PRIMARY KEY AUTOINCREMENT,  
    AlternativeId VARCHAR(31) UNIQUE NOT NULL,  
    Title VARCHAR(255) NOT NULL,  
    IsActive BOOLEAN,  
    IsOpenAccess BOOLEAN,  
    Type CHAR(2) CHECK (Type IN ('bs', 'cp', 'jo')),  
    -- Book Series / Conference Proceedings / Journal  
    -- or NULL in all other cases  
    Impact1 REAL, -- value of an impact factor  
    Impact2 REAL, -- value of an impact factor  
    Impact3 REAL, -- value of an impact factor  
    Impact4 REAL, -- value of an impact factor  
    Impact5 REAL, -- value of an impact factor  
    Impact6 REAL, -- value of an impact factor  
);  

CREATE TABLE Biblio_SourcesCategories (  
    -- links Sources and Categories  
    IdSource INTEGER NOT NULL,  
    IdCategory INTEGER NOT NULL,  
    PRIMARY KEY(IdSource, IdCategory),  
    FOREIGN KEY(IdSource) REFERENCES Biblio_Sources(IdSource),  
    FOREIGN KEY(IdCategory) REFERENCES Biblio_Categories(IdCategory)  
);  

CREATE TABLE Biblio_Documents (  
    IdDocument INTEGER PRIMARY KEY AUTOINCREMENT,  
    IdSource INTEGER,  
    AlternativeId VARCHAR(31) UNIQUE NOT NULL,  
    Title VARCHAR(255) NOT NULL,  
    BibEntry TEXT,  
    -- (e.g. Source Title,Year,Volume,Issue,Article Number,PageStart,PageEnd)  
    Year INTEGER,  
    Pages INTEGER,  
    Citations INTEGER NOT NULL,  
    Type CHAR(2) CHECK (Type IN ('ar', 'ip', 'bk', 'cp', 'ed', 'er', 'le', 'no', 'rp', 're', 'sh')),  
    -- Article-ar / Article in Press-ip / Book-bk /  
    -- Conference Paper-cp / Editorial-ed / Erratum-er /
-- Letter-le / Note-no / Report-rp / Review-re / Short Survey-sh
-- or NULL in all other cases
FOREIGN KEY(IdSource) REFERENCES Biblio_Sources(IdSource),
FOREIGN KEY(IdLanguage) REFERENCES Biblio_Languages(IdLanguage)
);

CREATE TABLE Biblio_Citations (  
IdDocumentParent INTEGER NOT NULL, # cited document
IdDocumentChild INTEGER NOT NULL, # reference
PRIMARY KEY(IdDocumentParent, IdDocumentChild),
FOREIGN KEY(IdDocumentParent) REFERENCES Biblio_Documents(IdDocument),
FOREIGN KEY(IdDocumentChild) REFERENCES Biblio_Documents(IdDocument)
);

CREATE TABLE Biblio_Surveys (  
-- each call to lbsImportDocuments() puts a new record here,
-- they may be grouped into 'Surveys' using 'Description' field
IdSurvey INTEGER PRIMARY KEY AUTOINCREMENT,
Description VARCHAR(63) NOT NULL, -- survey group name
FileName VARCHAR(63),  -- original file name
Timestamp DATETIME  -- date of file import
);

CREATE TABLE Biblio_DocumentsSurveys (  
-- note that the one Document may often be found in many Surveys
IdDocument INTEGER NOT NULL,
IdSurvey INTEGER NOT NULL,
PRIMARY KEY(IdDocument, IdSurvey),
FOREIGN KEY(IdDocument) REFERENCES Biblio_Documents(IdDocument),
FOREIGN KEY(IdSurvey) REFERENCES Biblio_Surveys(IdSurvey),
FOREIGN KEY(IdDocument) REFERENCES Biblio_Documents(IdDocument)
);

CREATE TABLE Biblio_Authors (  
IdAuthor INTEGER PRIMARY KEY AUTOINCREMENT,
Name VARCHAR(63) NOT NULL,
AuthorGroup VARCHAR(31), # used to merge authors with non-unique representations
);

CREATE TABLE Biblio_AuthorsDocuments (  
-- links Authors and Documents
IdAuthor INTEGER NOT NULL,
IdDocument INTEGER NOT NULL,
PRIMARY KEY(IdAuthor, IdDocument),
FOREIGN KEY(IdAuthor) REFERENCES Biblio_Authors(IdAuthor),
FOREIGN KEY(IdDocument) REFERENCES Biblio_Documents(IdDocument)
);

In addition, the following views are created.
CREATE VIEW ViewBiblio_DocumentsSurveys AS
SELECT
    Biblio_DocumentsSurveys.IdDocument AS IdDocument,
    Biblio_DocumentsSurveys.IdSurvey AS IdSurvey,
    Biblio_Surveys.Description AS Description,
    Biblio_Surveys.Filename AS Filename,
    Biblio_Surveys.Timestamp AS Timestamp
FROM Biblio_DocumentsSurveys
JOIN Biblio_Surveys
ON Biblio_DocumentsSurveys.IdSurvey=Biblio_Surveys.IdSurvey;

CREATE VIEW ViewBiblio_DocumentsCategories AS
SELECT
    IdDocument AS IdDocument,
    DocSrcCat.IdCategory AS IdCategory,
    DocSrcCat.Description AS Description,
    DocSrcCat.IdCategoryParent AS IdCategoryParent,
    Biblio_Categories.Description AS DescriptionParent
FROM
    (SELECT
        Biblio_Documents.IdDocument AS IdDocument,
        Biblio_SourcesCategories.IdCategory AS IdCategory,
        Biblio_Categories.Description AS Description,
        Biblio_Categories.IdCategoryParent AS IdCategoryParent
    FROM Biblio_Documents
    JOIN Biblio_SourcesCategories
    ON Biblio_Documents.IdSource=Biblio_SourcesCategories.IdSource
    JOIN Biblio_Categories
    ON Biblio_SourcesCategories.IdCategory=Biblio_Categories.IdCategory)
    AS DocSrcCat
JOIN Biblio_Categories
ON DocSrcCat.IdCategoryParent=Biblio_Categories.IdCategory;

Value

TRUE on success.

See Also

lbsConnect, lbsClear, Scopus_ImportSources, lbsTidy /internal/ /internal/ /internal/

Examples

```r
## Not run:
conn <- lbsConnect("Bibliometrics.db");
## ...
lbsCreate(conn);
Scopus_ImportSources(conn);
## ...
```
lbsDeleteAllAuthorsDocuments

Delete all authors, documents and surveys from a Local Bibliometric Storage

Description

Deletes author, citation, document, and survey information from a Local Bibliometric Storage.

Usage

lbsDeleteAllAuthorsDocuments(conn, verbose = TRUE)

Arguments

- conn: database connection object, see lbsConnect.
- verbose: logical; TRUE to be more verbose.

Details

For safety reasons, an SQL transaction opened at the beginning of the removal process is not committed (closed) automatically. You should do manually (or rollback it), see Examples below.

Value

TRUE on success.

See Also

lbsClear, dbCommit, dbRollback

Examples

## Not run:
conn <- lbsConnect("Bibliometrics.db")
lbsDeleteAllAuthorsDocuments(conn)
dbCommit(conn)
## ...
lbsDisconnect(conn)
## End(Not run)
lbsDescriptiveStats

Performs preliminary analysis of data in a Local Bibliometric Storage by creating some basic descriptive statistics (numeric and graphical). Dataset may be restricted to any given document types or a single survey.

Usage

lbsDescriptiveStats(
  conn,
  documentTypes = NULL,
  surveyDescription = NULL,
  which = (1L:7L),
  main = "",
  ask = (prod(par("mfcol")) < length(which) && dev.interactive()),
  ...
)

Arguments

conn
  connection object, see lbsConnect.
documentTypes
surveyDescription
  single character string or NULL; survey to restrict to, or NULL for no restriction.
which
  numeric vector with elements in 1,...,7, or NULL; plot types to be displayed.
main
  title for each plot.
ask
  logical; if TRUE, the user is asked to press return before each plot.
...
  additional graphical parameters, see plot.default.
cex.caption
  controls size of default captions.

Details

Plot types (accessed with which):

- 1 — "Document types",
- 2 — "Publication years",
- 3 — "Citations per document",
- 4 — "Citations of cited documents per type", 
• 5 — "Number of pages per document type",
• 6 — "Categories of documents" (based on source categories),
• 7 — "Documents per author".

Note that this user interaction scheme is similar in behavior to the `plot.lm` function.

See Also

`plot.default`, `lbsConnect`, `lbsDescriptiveStats`

Examples

```r
## Not run:
conn <- lbsConnect("Bibliometrics.db");
## ...
lbsDescriptiveStats(conn, surveyDescription="Scientometrics",
documentTypes=c("Article", "Note", "Report", "Review", "Short Survey"));
## ...
lbsDisconnect(conn);
## End(Not run)
```
lbsGetCitations  Fetch authors’ citation sequences

Description

Creates ordered citation sequences of authors in a Local Bibliometric Storage.

Usage

```r
lbsGetCitations(
  conn,
  documentTypes = NULL,
  surveyDescription = NULL,
  idAuthors = NULL,
  verbose = TRUE
)
```

Arguments

- `conn`: a connection object as produced by `lbsConnect`.
- `surveyDescription`: single character string or `NULL`; survey to restrict to or `NULL` for no restriction.
- `idAuthors`: numeric vector of authors’ identifiers for which the sequences are to be created or `NULL` for all authors in the database.
- `verbose`: logical; `TRUE` to inform about the progress of the process.

Details

A citation sequence is a numeric vector consisting of citation counts of all the documents mapped to selected authors. However, the function may take into account only the documents from a given Survey (using `surveyDescription` parameter) or of chosen types (`documentTypes`).

Value

A list of non-increasingly ordered numeric vectors is returned. Each element of the list corresponds to a citation sequence of some author. List names attribute are set to authors’ names. Moreover, each vector has a set `IdAuthor` attribute, which uniquely identifies the corresponding record in the table `Biblio_Authors`. Citation counts come together with `IdDocuments` (vector elements are named).

The list of citation sequences may then be used to calculate authors’ impact using `lbsAssess` (see Examples below).

See Also

`lbsConnect`, `lbsAssess`
Examples

```r
## Not run:
conn <- lbsConnect("Bibliometrics.db");
## ...
citseq <- lbsGetCitations(conn, 
surveyDescription="Scientometrics", documentTypes="Article", 
idAuthors=c(39264,39265,39266));
print(citseq);
## $'Liu X.' # Author name
## 40116 34128 39122 29672 32343 32775 # IdDocument
## 11 4 1 0 0 0 # Citation count
## attr("IdAuthor")
## [1] 39264 # IdAuthor
##
## $'Xu Y.'
## 38680 38605 40035 40030 40124 39829 39745 29672
## 30 14 8 6 6 5 3 0
## attr("IdAuthor")
## [1] 39265
##
## $'Wang Y.'
## 29992 29672 29777 32906 33858 33864 34704
## 1 0 0 0 0 0 0
## attr("IdAuthor")
## [1] 39266
##
print(lbsAssess(citseq, 
  f=list(length, sum, index.h, index.g, function(x) index.rp(x,1), 
    function(x) sqrt(prod(index.lp(x,1)))),
  function(x) sqrt(prod(index.lp(x,Inf))))),
  captions=c("length", "sum", "index.h", "index.g", "index.w", "index.lp1", "index.lpInf"));
## Name length sum index.h index.g index.w index.lp1 index.lpInf
## 3 Xu Y. 8 72 5 8 7 8.573214 5.477226
## 2 Wang Y. 7 1 1 1 1 1.000000 1.000000
## 1 Liu X. 6 16 2 4 3 4.157609 3.316625
## ...
## End(Not run)
```

lbsGetInfoAuthors  
Retrieves author information

Description

Retrieves basic information on given authors.

Usage

```r
lbsGetInfoAuthors(conn, idAuthors)
```
lbsGetInfoDocuments

Retrieve document information

Description

Retrieves information on given documents.

Usage

lbsGetInfoDocuments(conn, idDocuments)

Arguments

- conn: a connection object as produced by lbsConnect.
- idDocuments: a numeric or integer vector with document identifiers (see column IdDocument in the table Biblio_Documents).
Value

A list of docinfo objects, that is lists with the following components:

- `IdDocument` — numeric; document identifier in the table Biblio_Documents,
- `Authors` — list of authorinfo objects (see e.g. `as.character.authorinfo`).
- `Title` — title of the document,
- `BibEntry` — bibliographic entry,
- `AlternativeId` — unique character identifier,
- `Pages` — number of pages,
- `Citations` — number of citations,
- `Year` — publication year,
- `Type` — document type, e.g. Article or Conference Paper.

See Also

`print.docinfo, lbsSearchDocuments, lbsGetInfoAuthors, as.character.authorinfo, as.character.docinfo`

Examples

```r
## Not run:
conn <- dbBiblioConnect("Bibliometrics.db");
## ...
id <- lbsSearchDocuments(conn,
idAuthors=lbsSearchAuthors(conn, "Knuth"
lbsGetInfoDocuments(conn, id);
## ...
## End(Not run)
```

lbsImportDocuments

*Import bibliographic data into a Local Bibliometric Storage.*

Description

Imports bibliographic data from a special 11-column data.frame object (see e.g. `Scopus_ReadCSV`) into a Local Bibliometric Storage.

Usage

```r
lbsImportDocuments(
  conn,
  data,
surveyDescription = "Default survey",
surnameFirstnameCommaSeparated = FALSE,  
originalFilename = attr(data, "filename"),
```
excludeRows = NULL,
updateDocumentIfExists = TRUE,
warnSourceTitle = TRUE,
warnExactDuplicates = FALSE,
verbose = TRUE
)

Arguments

c conn a connection object, see lbsConnect.
data 11 column data.frame with bibliometric entries; see above.
surveyDescription description of the survey. Allows for documents grouping.
surnameFirstnameCommaSeparated logical; indicates wher surnames are separated from first names (or initials) by comma or by space (FALSE, default).
originalFilename original filename; attr(data,"filename") used by default.
excludeRows a numeric vector with row numbers of data to be excluded or NULL.
updateDocumentIfExists logical; if TRUE then documents with existing AlternativeId will be updated.
warnSourceTitle logical; if TRUE then warnings are generated if a given SourceTitle is not found in Biblio_Sources.
warnExactDuplicates logical; TRUE to warn if exact duplicates are found (turned off by default).
verbose logical; TRUE to display progress information.

Details

data must consist of the following 11 columns (in order). Otherwise the process will not be executed.

1 Authors character Author(s) name(s), comma-separated, surnames first.
2 Title character Document title.
3 Year numeric Year of publication.
4 SourceTitle character Title of the source containing the document.
5 Volume character Volume.
6 Issue character Issue.
7 PageStart numeric Start page; numeric.
8 PageEnd numeric End page; numeric.
9 Citations numeric Number of citations; numeric.
10 AlternativeId character Alternative document identifier.
11 DocumentType factor Type of the document.

DocumentType is one of “Article”, “Article in Press”, “Book”, “Conference Paper”, “Editorial”,
lbsSearchAuthors

Find authors that satisfy given criteria

Description

Finds authors by name.

Usage

lbsSearchAuthors(conn, names.like = NULL, group = NULL)

Arguments

conn  
connection object, see lbsConnect.

names.like  
character vector of SQL-LIKE patterns to match authors’ names.

group  
character vector of author group identifiers.
Details

names.like is a set of search patterns in an SQL LIKE format, i.e. an underscore _ matches a single character and a percent sign % matches any set of characters. The search is case-insensitive.

Value

Integer vector of authors’ identifiers which match at least one of given SQL-LIKE patterns.

See Also

lbsGetInfoAuthors, lbsSearchDocuments, lbsGetInfoDocuments

Examples

```r
## Not run:
conn <- dbBiblioConnect("Bibliometrics.db");
## ...
id <- lbsSearchAuthors(conn, c("Smith\nlbsGetInfoAuthors(conn, id);
## ...
## End(Not run)
```

## lbsSearchDocuments

Find documents that satisfy given criteria

Description

Searches for documents meeting given criteria (e.g. document titles, documents’ authors identifiers, number of citations, number of pages, publication years or document types).

Usage

```r
lbsSearchDocuments(
  conn,
  titles.like = NULL,
  idAuthors = NULL,
  citations.expr = NULL,
  pages.expr = NULL,
  year.expr = NULL,
  documentTypes = NULL,
  alternativeId = NULL,
  surveyDescription = NULL
)
```
**lbsSearchDocuments**

**Arguments**

- **conn** connection object, see `lbsConnect`.
- **titles.like** character vector of SQL-LIKE patterns to match documents’ titles or NULL.
- **idAuthors** numeric or integer vector with author identifiers (see column `IdAuthor` in the table `Biblio_Authors`) or NULL.
- **citations.expr** expression determining the desired number of citations or NULL, see Examples below.
- **pages.expr** expression determining the desired number of pages or NULL, see Examples below.
- **year.expr** expression determining the desired publication year or NULL, see Examples below.
- **alternativeId** character vector of documents’ AlternativeIds.
- **surveyDescription** single character string or NULL: survey description to restrict to or NULL.

**Details**

titles.like is a set of search patterns in an SQL LIKE format, i.e. an underscore _ matches a single character and a percent sign % matches any set of characters. The search is case-insensitive.

The expressions passed as parameters citations.expr, pages.expr, year.expr must be acceptable by SQL WHERE clause in the form `WHERE field <expression>`, see Examples below.

**Value**

Integer vector of documents’ identifiers matching given criteria.

**See Also**

`lbsGetInfoAuthors, lbsSearchAuthors, lbsGetInfoDocuments`

**Examples**

```r
## Not run:
conn <- dbBiblioConnect("Bibliometrics.db");
## ...
idd <- lbsSearchDocuments(conn, pages.expr=">= 400", year.expr="BETWEEN 1970 AND 1972");
lbsGetInfoDocuments(conn, idd);
## ...
## End(Not run)
```
lbsTidy  

*Clean up a Local Bibliometric Storage*

**Description**

Cleans up a Local Bibliometric Storage by removing all authors with no documents, fixing documents with missing survey information, and executing the `VACUUM` SQL command.

**Usage**

```r
lbsTidy(
  conn,
  newSurveyDescription = "lbsTidy_Merged",
  newSurveyFilename = "lbsTidy_Merged"
)
```

**Arguments**

- `conn`: database connection object, see `lbsConnect`.
- `newSurveyDescription`: character; default survey description for documents with missing survey info.
- `newSurveyFilename`: character; default survey filename for documents with missing survey info.

**Value**

TRUE on success.

**See Also**

`lbsConnect`, `lbsCreate`, `Scopus_ImportSources`, `lbsDeleteAllAuthorsDocuments`, `dbCommit`, `dbRollback`

---

print.authorinfo  

*Print an authorinfo object*

**Description**

Prints out an object of class `authorinfo`. Such an object is returned by e.g. `lbsGetInfoAuthors`.

**Usage**

```r
## S3 method for class 'authorinfo'
print(x, ...)
```
**print.docinfo**

**Arguments**

- `x` an object of class `authorinfo`.
- `...` unused.

**Details**

For more information see man page for `as.character.authorinfo`.

**See Also**

`as.character.authorinfo`, `lbsSearchAuthors`, `lbsGetInfoAuthors`

---

**print.docinfo**  
*Print a docinfo object*

**Description**

Prints out an object of class docinfo. Such an object is returned by e.g. `lbsGetInfoDocuments`.

**Usage**

```r
## S3 method for class 'docinfo'
print(x, ...)
```

**Arguments**

- `x` an object of class docinfo.
- `...` unused.

**Details**

For more information see man page for `as.character.docinfo`.

**See Also**

`as.character.docinfo`, `lbsSearchDocuments`, `lbsGetInfoDocuments`
Scopus ImportSources

<table>
<thead>
<tr>
<th>Scopus ASJC</th>
<th>Scopus ASJC (All Science. Journals Classification) classification</th>
</tr>
</thead>
</table>

**Description**

List of Elsevier’s SciVerse Scopus ASJC (All Science. Journals Classification) source classification codes.

**Usage**

Scopus ASJC

**Format**

An object of class NULL of length 0.

**Details**

Last update: October 2011. The data file is based on the official and publicly available (no permission needed as stated by Elsevier) Scopus list of covered titles.

It consists of 334 ASJC 4-digit integer codes (column ASJC) together with their group identifiers (column ASJC Parent) and descriptions (column Description).

ASJC codes are used to classify Scopus sources (see Scopus SourceList).

**See Also**

Scopus SourceList, Scopus ReadCSV, Scopus ImportSources

Scopus ImportSources

Import SciVerse Scopus coverage information and ASJC codes to a Local Bibliometric Storage

**Description**

Imports SciVerse Scopus covered titles and their ASJC codes to an empty Local Bibliometric Storage (LBS).

**Usage**

Scopus ImportSources(conn, verbose = T)

**Arguments**

- conn: a connection object, see lbsConnect.
- verbose: logical; TRUE to display progress information.
Details

This function should be called prior to importing any document information to the LBS with the function `lbsImportDocuments`.

Note that adding all the sources takes some time.

Only elementary ASJC and SciVerse Scopus source data read from `Scopus_ASJC` and `Scopus_SourceList` will be added to the LBS (`Biblio_Categories`, `Biblio_Sources`, `Biblio_SourcesCategories`).

Value

`TRUE` on success.

See Also

`Scopus_ASJC`, `Scopus_SourceList`, `Scopus_ReadCSV`, `lbsConnect`, `lbsCreate`

Examples

```r
## Not run:
conn <- lbsConnect("Bibliometrics.db");
lbsCreate(conn);
Scopus_ImportSources(conn);
## ...
lbsDisconnect(conn);
## End(Not run)
```

`Scopus_ReadCSV`  
Import bibliography entries from a CSV file.

Description

Reads bibliography entries from a UTF-8 encoded CSV file.

Usage

```r
Scopus_ReadCSV(
  filename,
  stopOnErrors = TRUE,
  dbIdentifier = "Scopus",
  alternativeIdPattern = \^.*\|\d\|.*\$",
  ...
)
```
Arguments

filename the name of the file which the data are to be read from, see read.csv.
stopOnErrors logical; TRUE to stop on all potential parse errors or just warn otherwise.
dbIdentifier character or NA; database identifier, helps detect parse errors, see above.
alternativeIdPattern character; regular expression used to extract AlternativeId, NA to get the id as is,
... further arguments to be passed to read.csv.

Details

The read.csv function is used to read the bibliography. You may therefore freely modify its behavior by passing further arguments (...), see the manual page of read.table for details.

The CSV file should consist at least of the following columns.

1. Authors: Author name(s) (surname first; multiple names are comma-separated, e.g. “Smith John, Nowak G. W.”),
2. Title: Document title,
3. Year: Year of publication,
4. Source.title: Source title, e.g. journal name,
5. Volume: Volume number,
6. Issue: Issue number,
7. Page.start: Start page number,
8. Page.end: End page number,
9. Cited.by: Number of citations received,
10. Link: String containing unique document identifier, by default of the form ...id=UNIQUE_ID&...
    (see alternativeIdPattern parameter),
12. Source: Data source identifier, must be the same as the dbIdentifier parameter value. It is used for parse errors detection.

The CSV file to be read may, for example, be created by SciVerse Scopus (Export format=comma separated file, .csv (e.g. Excel), Output=Complete format or Citations only). Note that the exported CSV file sometimes needs to be corrected by hand (wrong page numbers, single double quotes in character strings instead of two-double quotes etc.). We suggest to make the corrections in a “Notepad”-like application (in plain text). The function tries to indicate line numbers causing potential problems.

Value

A data.frame containing the following 11 columns:

Authors Author name(s), comma-separated, surnames first.
Scopus_SourceList

<table>
<thead>
<tr>
<th>Title</th>
<th>Document title.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>Year of publication.</td>
</tr>
<tr>
<td>AlternativeId</td>
<td>Unique document identifier.</td>
</tr>
<tr>
<td>SourceTitle</td>
<td>Title of the source containing the document.</td>
</tr>
<tr>
<td>Volume</td>
<td>Volume.</td>
</tr>
<tr>
<td>Issue</td>
<td>Issue.</td>
</tr>
<tr>
<td>PageStart</td>
<td>Start page; numeric.</td>
</tr>
<tr>
<td>PageEnd</td>
<td>End page; numeric.</td>
</tr>
<tr>
<td>Citations</td>
<td>Number of citations; numeric.</td>
</tr>
<tr>
<td>DocumentType</td>
<td>Type of the document; see above.</td>
</tr>
</tbody>
</table>

The object returned may be imported into a local bibliometric storage via `lbsImportDocuments`.

See Also

`Scopus_ASJC`, `Scopus_SourceList`, `lbsConnect`, `Scopus_ImportSources`, `read.table`, `lbsImportDocuments`

Examples

```r
## Not run:
conn <- lbsConnect("Bibliometrics.db");
## ...
data <- Scopus_ReadCSV("db_Polish_MATH/Poland_MATH_1987-1993.csv");
lbsImportDocuments(conn, data, "Poland_MATH");
## ...
lbsDisconnect(conn);
## End(Not run)
```

---

**Scopus_SourceList**  
*Scopus covered source list*

**Description**

List of Elsevier’s *SciVerse Scopus* covered titles (journals, conference proceedings, book series, etc.)

**Usage**

Scopus_SourceList

**Format**

An object of class `NULL` of length 0.
### Details

Last update: October 2011. The data file is based on the official and publicly available (no permission needed as stated by Elsevier) Scopus list of covered titles.

This data frame consists of 30794 records. It has the following columns.

<table>
<thead>
<tr>
<th>SourceId</th>
<th>Unique source identifier in SciVerse Scopus (integer).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Title of the source.</td>
</tr>
<tr>
<td>Status</td>
<td>Status of the source, either Active or Inactive.</td>
</tr>
<tr>
<td>OpenAccess</td>
<td>Type of Open Access, see below.</td>
</tr>
<tr>
<td>Type</td>
<td>Type of the source, see below.</td>
</tr>
<tr>
<td>ASJC</td>
<td>A list of semicolon-separated ASJC classification codes, see Scopus_ASJC.</td>
</tr>
</tbody>
</table>

OpenAccess is one of DOAJ, Not OA (not Open Access source), OA but not registered, OA registered.

Type is one of Book Series, Conference Proceedings, Journal, Trade Journal

The data.frame is sorted by Status (Active sources first) and then by SJR_2011 (higher values first).

### See Also

Scopus_ASJC, Scopus_ReadCSV, Scopus_ImportSources
Index

* datasets
  Scopus_ASJC, 26
  Scopus_SourceList, 29
as.character.authorinfo, 3, 4, 5, 18, 19, 25
as.character.docinfo, 4, 19, 25

  CITAN(CITAN-package), 2
  CITAN-package, 2
dbClearResult, 5
dbCommit, 8, 13, 24
dbQuery, 5
dbGetQuery, 5
dbRollback, 8, 13, 24
dbSendQuery, 5

lbsAssess, 6, 16
lbsClear, 7, 12, 13
lbsConnect, 6, 8, 8, 9, 12–16, 18, 20, 21, 23, 24, 26, 27, 29
lbsCreate, 5, 8, 9, 9, 21, 24, 27
lbsDeleteAllAuthorsDocuments, 8, 13, 24
lbsDescriptiveStats, 14
lbsDisconnect, 9, 15
lbsGetCitations, 6, 16
lbsGetInfoAuthors, 3, 4, 17, 19, 22–25
lbsGetInfoDocuments, 4, 5, 18, 18, 22, 23, 25
lbsImportDocuments, 19, 27, 29
lbsSearchAuthors, 4, 18, 21, 23, 25
lbsSearchDocuments, 3, 18, 19, 22, 22, 25
lbsTidy, 12, 24

plot.default, 14, 15
plot.lm, 15
print.authorinfo, 4, 18, 24
print.docinfo, 5, 19, 25

read.csv, 28
read.table, 28, 29