# Package 'citelens'

November 14, 2026

**Type** Package

**Title** Tools for Citation Libraries and Text Analytics on Abstracts

**Version** 0.1.0

**Description** Converts 'RefWorks' and 'RIS' exports, and saved 'Google Scholar' results pages, to 'BibTeX', and reads 'BibTeX' files into data frames and 'Excel' workbooks. Looks up missing abstracts in 'OpenAlex'. Runs text analytics on abstracts: topic modeling (latent Dirichlet allocation), cosine document similarity, and trends over time. The computationally heavy steps are implemented in 'C++' via 'Rcpp'.

**URL** https://github.com/cownr10r/citelens

**BugReports** https://github.com/cownr10r/citelens/issues

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** Rcpp, writexl, ggplot2, xml2, jsonlite, curl

**LinkingTo** Rcpp

**Suggests** readxl, testthat (>= 3.0.0), withr

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.1.1

**NeedsCompilation** yes

**Author** Mario Martinez [aut, cre, cph]

**Maintainer** Mario Martinez <mario.a.martinez@live.com>

**Repository** CRAN

**Date/Publication** 2026-11-14 09:10:26 UTC

## Contents

bib_to_df . . . . . 2
citelens_stopwords . . . . . 3
convert_refworks . . . . . 4

doc_similarity . . . . . 4
fetch_abstracts . . . . . 6
find_duplicates . . . . . 7
plot_similarity . . . . . 8
plot_topic_prevalence . . . . . 9
plot_topic_terms . . . . . 9
plot_topic_trends . . . . . 10
plot_word_trends . . . . . 10
refworks_to_bib . . . . . 11
refworks_to_xlsx . . . . . 11
scholar_html_to_bib . . . . . 12
scholar_parse_html . . . . . 13
similarity_to_df . . . . . 14
similar_to . . . . . 15
topic_documents . . . . . 16
topic_k_search . . . . . 16
topic_model . . . . . 17
topic_summary . . . . . 18
topic_terms . . . . . 19
topic_trends . . . . . 19
word_trends . . . . . 20

**Index** **22**

---

`bib_to_df` *Read a BibTeX (.bib) file into a data frame*

---

## Description

Parses a BibTeX file with compiled C++ code and returns one row per entry. The entry type is in ‘bibtype‘, the citation key in ‘bibkey‘, and every BibTeX field found in the file (author, title, year, ...) becomes a column. Fields an entry doesn't have are ‘NA‘. All columns are character vectors.

## Usage

```
bib_to_df(input, clean = TRUE, xlsx = NULL)
```

## Arguments

| | |
|---|---|
| `input` | Path to the ‘.bib‘ (or ‘.txt‘) file. |
| `clean` | If ‘TRUE‘ (the default), tidy each value: remove grouping braces (e.g. a title wrapped in double braces becomes plain text), unescape backslash-escaped ampersands, percent signs, dollar signs, hashes and underscores, and collapse line breaks and repeated spaces. Other LaTeX commands, such as accents, are left as they are. Use ‘FALSE‘ to get the raw values. |
| `xlsx` | Optional path for an Excel file (e.g. ‘"~/Desktop/refs.xlsx"‘). If given, the data frame is also written there as a single sheet with a header row, and the data frame is returned invisibly. A missing ‘.xlsx‘ extension is added. An existing file at that path is overwritten. |

## Details

‘@string‘ macros and the standard month abbreviations (‘jan‘, ‘feb‘, ...) are expanded, ‘#‘ concatenation is handled, and ‘@comment‘ / ‘@preamble‘ blocks are ignored.

## Value

A data frame with one row per BibTeX entry (invisibly when ‘xlsx‘ is given).

## Examples

```
f <- system.file("extdata", "sample.bib", package = "citelens")
bib_to_df(f)

# Also save an Excel copy
out <- tempfile(fileext = ".xlsx")
bib_to_df(f, xlsx = out)
```

---

# citelens_stopwords *Default stop words used by ‘topic_model()‘*

---

## Description

Common English function words plus words that appear in almost every abstract ("study", "results", "findings", ...). Pass your own additions to the ‘stopwords‘ argument of [topic_model()].

## Usage

```
citelens_stopwords()
```

## Value

A character vector.

## Examples

```
head(citelens_stopwords())
```

## convert_refworks *Convert a RefWorks export (Tagged Format or RIS) to a .bib file*

### Description

Reads citations exported from RefWorks in either RefWorks Tagged Format or RIS format (detected automatically) and converts them to BibTeX entries using compiled C++ code.

### Usage

```
convert_refworks(input, output = NULL)
```

### Arguments

| | |
|---|---|
| input | Path to the exported citation file (‘.txt‘ or ‘.ris‘). |
| output | Path for the .bib file. If ‘NULL‘ (the default), nothing is written and the entries are returned. |

### Value

A character vector with one BibTeX entry per record (invisibly when ‘output‘ is given).

### Examples

```
f <- system.file("extdata", "sample_refworks.txt", package = "citelens")
cat(convert_refworks(f)[1])
```

## doc_similarity *Document similarity (cosine similarity of TF-IDF vectors)*

### Description

Measures how alike the abstracts in a data frame are. Each abstract is turned into a vector of word weights (TF-IDF: a word counts for more when it is frequent in that abstract and rare across the collection), and two abstracts are compared by the cosine of the angle between their vectors: 0 means no words in common, 1 means the same word profile. Text is cleaned the same way as in [topic_model()].

## Usage

```
doc_similarity(
  data,
  text = "abstract",
  id = "bibkey",
  min_doc_freq = 2,
  max_doc_prop = 1,
  min_word_length = 3,
  stopwords = NULL,
  singularize = TRUE
)
```

## Arguments

| | |
|---|---|
| data | A data frame with one row per reference. |
| text | Name of the column holding the text to analyse. |
| id | Name of a column that identifies each reference (default ‘"bibkey"‘). If it does not exist, row numbers are used. |
| min_doc_freq | Drop words that occur in fewer than this many abstracts. The default of 2 ignores words that cannot link two abstracts. |
| max_doc_prop | Drop words that occur in more than this share of the abstracts (1 keeps everything; try 0.5 to remove words common to nearly every abstract). |
| min_word_length | Drop words shorter than this many letters. |
| stopwords | Extra words to drop, in addition to [citelens_stopwords()]. |
| singularize | Merge simple plurals (default ‘TRUE‘). |

## Details

Similarity reflects shared vocabulary, not meaning, so two abstracts that use different words for the same idea can look less alike than they are. The result stores a full abstract-by-abstract matrix, which is fine for thousands of abstracts but grows with the square of their number.

## Value

An object of class ‘citelens_similarity‘. Use [similar_to()], [find_duplicates()] and [plot_similarity()] to explore it; the matrix itself is in ‘$sim‘.

## Examples

```
f <- system.file("extdata", "sample_abstracts.bib", package = "citelens")
refs <- bib_to_df(f)
s <- doc_similarity(refs)
similar_to(s, "syn01", n = 3)
```

`fetch_abstracts` *Look up abstracts for a table of references*

## Description

Fills in an ‘abstract‘ column by looking each reference up in [OpenAlex](https://openalex.org), a free, open index of scholarly works with an official API. Use it on the results of [scholar_parse_html()], whose results list has titles but no abstracts, or on any data frame with titles (and optionally DOIs).

## Usage

```
fetch_abstracts(
  data,
  title = "title",
  doi = "doi",
  year = "year",
  text = "abstract",
  api_key = Sys.getenv("OPENALEX_API_KEY"),
  min_match = 0.85,
  max_year_diff = 1,
  overwrite = FALSE,
  delay = 0.1,
  fetch = .get_json
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame, for example from [scholar_parse_html()] or [bib_to_df()]. |
| `title`, `doi`, `year` | Names of the columns holding the title, DOI and year. Missing columns are skipped. |
| `text` | Name of the abstract column to create or fill (default ‘"abstract"‘). |
| `api_key` | Your OpenAlex API key (see above). Defaults to the ‘OPENALEX_API_KEY‘ environment variable. |
| `min_match` | Minimum title similarity (0 to 1) needed to accept a title search result. |
| `max_year_diff` | Largest difference in publication year (your ‘year‘ vs OpenAlex’s) accepted for a title match. The default of 1 allows for online-first dates; raise it (or use ‘Inf‘) for references whose year differs because of re-editions, then check them with ‘closest_year‘. |
| `overwrite` | If ‘FALSE‘ (default), references that already have an abstract keep it. |
| `delay` | Seconds to wait between requests. |
| `fetch` | Function used to download JSON; for testing. |

## Details

A reference is looked up by DOI when it has one, otherwise by title. A title search is only accepted when the best match is very similar to your title (see ‘min_match‘) and the publication years agree to within a year; anything less certain is left as ‘NA‘ instead of risking the wrong abstract. Many, but not all, works in OpenAlex have an abstract, and coverage is best for recent articles, so expect some ‘NA‘s. Check a few matches by eye using ‘abstract_match‘ and ‘abstract_source‘.

OpenAlex now requires an **API key**, which is free: create an account at <https://openalex.org>, then copy the key from <https://openalex.org/settings/api>. A free key allows 100,000 credits a day; a title search costs 10 credits and a DOI lookup 1, so a few thousand references fit in a day. Without a key you get only about 100 credits a day (roughly ten lookups). Pass the key as ‘api_key‘, or save it once with ‘Sys.setenv(OPENALEX_API_KEY = "your-key")‘ (or in your ‘~/.Renviron‘ file) and it is picked up automatically.

Each title (or DOI) is sent to OpenAlex over the internet, along with your key. Nothing is sent to Google. If OpenAlex says the allowance is used up, the function stops and returns what it has; run it again on that result and it looks up only the references still missing an abstract.

## Value

‘data‘ with the abstract column filled in, plus: ‘abstract_source‘ (the OpenAlex id), ‘abstract_match‘ (title similarity, 1 for a DOI match), ‘abstract_status‘ (why a reference has no abstract: ‘"found"‘, ‘"no abstract on file"‘, ‘"title not matched"‘, ‘"year differs"‘, ‘"no results"‘ or ‘"lookup failed"‘), and ‘closest_title‘ / ‘closest_score‘ / ‘closest_year‘ (the nearest OpenAlex title, for references that were not matched, so you can see whether ‘min_match‘ or ‘max_year_diff‘ would help).

## Examples

```
## Not run:
res <- scholar_parse_html("~/Downloads/scholar_page1.html")
res <- fetch_abstracts(res, api_key = "your-openalex-key")
topic_model(res, k = 4, id = "title")

## End(Not run)
```

---

# find_duplicates *Find likely duplicate or near-duplicate references*

---

## Description

Lists every pair of abstracts whose cosine similarity is at or above ‘cutoff‘, most similar first. Useful for cleaning a library where the same article was entered twice.

## Usage

```
find_duplicates(s, cutoff = 0.9, data = NULL, title = "title")
```

### Arguments

| | |
|---|---|
| s | A result from [doc_similarity()]. |
| cutoff | Minimum similarity to report (0 to 1). |
| data | Optional: the data frame passed to [doc_similarity()]. If given, the ‘title‘ of each reference is added. |
| title | Name of the title column in ‘data‘. |

### Value

A data frame with ‘id_1‘, ‘id_2‘ and ‘similarity‘ (and ‘title_1‘, ‘title_2‘); zero rows if no pair reaches the cutoff.

---

## plot_similarity *Heatmap of document similarity*

---

### Description

Heatmap of document similarity

### Usage

```
plot_similarity(
  s,
  order = TRUE,
  labels = length(s$doc_ids) <= 40,
  values = length(s$doc_ids) <= 30,
  digits = 2
)
```

### Arguments

| | |
|---|---|
| s | A result from [doc_similarity()]. |
| order | If ‘TRUE‘ (default), reorder abstracts so that similar ones sit together (hierarchical clustering). |
| labels | Show reference ids on the axes? Defaults to ‘TRUE‘ for up to 40 abstracts. |
| values | Print the similarity number inside each box? Defaults to ‘TRUE‘ for up to 30 abstracts; with more, the numbers get too small to read. |
| digits | Decimal places for the numbers (default 2). |

### Value

A ggplot object.

## plot_topic_prevalence *Bar chart of how common each topic is*

### Description

Bar chart of how common each topic is

### Usage

```
plot_topic_prevalence(model, n_terms = 4)
```

### Arguments

| | |
|---|---|
| model | A model from [topic_model()]. |
| n_terms | Number of top terms shown in each topic's label. |

### Value

A ggplot object.

## plot_topic_terms *Bar charts of each topic's top terms*

### Description

Bar charts of each topic's top terms

### Usage

```
plot_topic_terms(model, n = 8)
```

### Arguments

| | |
|---|---|
| model | A model from [topic_model()]. |
| n | Number of terms to show per topic. |

### Value

A ggplot object (add layers or save it with [ggplot2::ggsave()]).

## plot_topic_trends *Line chart of topic trends*

### Description

Line chart of topic trends

### Usage

```
plot_topic_trends(trends)
```

### Arguments

| | |
|---|---|
| trends | The result of [topic_trends()]. |

### Value

A ggplot object.

## plot_word_trends *Line chart of word trends*

### Description

Line chart of word trends

### Usage

```
plot_word_trends(trends)
```

### Arguments

| | |
|---|---|
| trends | The result of [word_trends()]. |

### Value

A ggplot object.

## refworks_to_bib *Convert RefWorks citations (Tagged Format or RIS) to BibTeX entries*

### Description

The format is detected automatically: lines starting with 'TY - ' mean RIS, otherwise RefWorks Tagged Format ('RT', 'A1', 'T1', ...) is assumed.

### Usage

```
refworks_to_bib(lines)
```

### Arguments

| | |
|---|---|
| lines | Character vector of lines from the export (e.g. from readLines()). |

### Value

Character vector with one BibTeX entry per record.

## refworks_to_xlsx *Convert a RefWorks export straight to an Excel file*

### Description

One step from a RefWorks export (Tagged Format or RIS, detected automatically) to an Excel spreadsheet with one row per reference. Internally it runs [convert_refworks()] and then [bib_to_df()], so the same cleaning of odd characters applies. Optionally it also saves the intermediate BibTeX file.

### Usage

```
refworks_to_xlsx(input, xlsx, bib = NULL, clean = TRUE)
```

### Arguments

| | |
|---|---|
| input | Path to the RefWorks export ('.txt' or '.ris'). |
| xlsx | Path for the Excel file, e.g. '"~/Desktop/references.xlsx"'. A missing '.xlsx' extension is added and an existing file is overwritten. |
| bib | Optional path for a BibTeX file to save as well (e.g. '"~/Desktop/references.bib"'). Leave as 'NULL' to skip it. |
| clean | Passed to [bib_to_df()]: tidy braces, LaTeX escapes and extra whitespace in the values (default 'TRUE'). |

## Value

The data frame of references, invisibly.

## Examples

```
ris <- system.file("extdata", "sample_refworks.ris", package = "citelens")
refworks_to_xlsx(ris, tempfile(fileext = ".xlsx"))
```

---

`scholar_html_to_bib` *Convert a saved Google Scholar results page to BibTeX*

---

## Description

Parses a saved Scholar results page with [scholar_parse_html()] and writes BibTeX entries. Results with a venue become '@article'; others become '@misc'. Entries whose author list or venue was cut short by Scholar get a 'note' field saying so, so you can find and complete them.

## Usage

```
scholar_html_to_bib(
  input,
  output = NULL,
  drop_duplicates = TRUE,
  abstracts = FALSE,
  api_key = Sys.getenv("OPENALEX_API_KEY")
)
```

## Arguments

| | |
|---|---|
| `input` | One or more paths to saved '.html' files, **a folder** (every '.html' / '.htm' file in it is read), or a string of HTML text. Results from several pages are combined. |
| `output` | Path for the '.bib' file. If 'NULL' (default), nothing is written and the entries are returned. |
| `drop_duplicates` | Remove results that appear more than once (same title, case ignored) when you combine several pages. |
| `abstracts` | If 'TRUE', look up each abstract in OpenAlex with [fetch_abstracts()] and include it in the entries. Needs internet access. |
| `api_key` | Your OpenAlex API key, passed to [fetch_abstracts()]. Defaults to the 'OPENALEX_API_KEY' environment variable. |

## Details

The entries only contain what the results list shows (no volume, pages, DOI or abstract). For complete records use Scholar's Cite > BibTeX.

## Value

A character vector with one BibTeX entry per result (invisibly when ‘output‘ is given).

## Examples

```
page <- '<div class="gs_r"><div class="gs_ri">
  <h3 class="gs_rt"><a href="https://example.org/a">Retention and advising</a></h3>
  <div class="gs_a">A Smith, B Jones - Journal of Examples, 2019 - Example Press</div>
  </div></div>'
cat(scholar_html_to_bib(page))
```

---

# scholar_parse_html *Parse a saved Google Scholar results page*

---

## Description

Reads a Google Scholar search-results page that you saved from your browser and pulls out what the results list shows for each hit: title, authors, venue, year, publisher, link, snippet, citation count and number of versions. It parses the page's HTML/XML with the xml2 package; it does **not** contact Google Scholar, which blocks automated requests and whose terms do not allow them.

## Usage

```
scholar_parse_html(input, recursive = FALSE)
```

## Arguments

| | |
|---|---|
| `input` | One or more paths to saved ‘.html‘ files, **a folder** (every ‘.html‘ / ‘.htm‘ file in it is read), or a string of HTML text. Results from several pages are combined. |
| `recursive` | If ‘input‘ is a folder, also read files in its subfolders. |

## Details

To save a page in Chrome or Safari: open the results, then File > Save Page As (choose "Webpage, HTML Only" or "Page Source"). Save each results page you want as its own file.

The results list shortens long author lists and venue names with "...", and does not show volume, pages or DOI. Entries affected are flagged in ‘authors_truncated‘ and ‘venue_truncated‘. For complete records, use Scholar's **Cite > BibTeX** (or **My library > Export**) instead.

## Value

A data frame with one row per result: ‘source_file‘ (the file it came from), ‘title‘, ‘authors‘ (as shown, with initials), ‘author_list‘ (a list column of names), ‘authors_truncated‘, ‘venue‘, ‘venue_truncated‘, ‘year‘, ‘publisher‘, ‘url‘, ‘snippet‘, ‘cited_by‘, ‘versions‘ and ‘cid‘ (Scholar's internal id for the result).

## Examples

```
page <- '<div class="gs_r" data-cid="x1"><div class="gs_ri">
 <h3 class="gs_rt"><a href="https://example.org/a">Retention and <b>advising</b></a></h3>
  <div class="gs_a">A Smith, B Jones - Journal of Examples, 2019 - Example Press</div>
  <div class="gs_fl"><a>Cite</a> <a>Cited by 12</a></div></div></div>'
scholar_parse_html(page)
```

---

`similarity_to_df` *Similarity results as a data frame (optionally saved to Excel)*

---

## Description

Turns the result of [doc_similarity()] into a data frame you can view, filter and save. Two layouts are available:

## Usage

```
similarity_to_df(
  s,
  type = c("matrix", "pairs"),
  data = NULL,
  title = "title",
  digits = 3,
  cutoff = 0,
  xlsx = NULL
)
```

## Arguments

| | |
|---|---|
| `s` | A result from [doc_similarity()]. |
| `type` | ‘"matrix"‘ (default) or ‘"pairs"‘. |
| `data` | Optional: the data frame passed to [doc_similarity()]. If given, the ‘title‘ column is added (‘title‘ in the matrix; ‘title_1‘ and ‘title_2‘ in the pairs). |
| `title` | Name of the title column in ‘data‘. |
| `digits` | Decimal places to round the similarities to (default 3). Use ‘NULL‘ for no rounding. |
| `cutoff` | For ‘type = "pairs"‘ (and the pairs sheet in Excel): keep only pairs at or above this similarity (0 to 1). Default 0 keeps every pair. |
| `xlsx` | Optional path for an Excel file (e.g. ‘"~/Desktop/similarity.xlsx"‘). A missing ‘.xlsx‘ ending is added. When given, the data frame is returned invisibly. |

### Details

* ‘type = "matrix"‘: one row and one column per reference, with the cosine similarity in each cell (the table behind the heatmap). * ‘type = "pairs"‘: one row per pair of references, most similar first, which is easier to sort, filter and scan.

If you give ‘xlsx‘, an Excel workbook is written with **both** layouts, on sheets named ‘similarity_matrix‘ and ‘pairs‘, whichever ‘type‘ you asked for.

### Value

A data frame. For ‘"matrix"‘: ‘id‘ (plus ‘title‘) and one column per reference. For ‘"pairs"‘: ‘id_1‘, ‘id_2‘, ‘similarity‘ (plus ‘title_1‘, ‘title_2‘).

### Examples

```
f <- system.file("extdata", "sample_abstracts.bib", package = "citelens")
refs <- bib_to_df(f)
s <- doc_similarity(refs)
head(similarity_to_df(s, "pairs", data = refs), 3)
out <- tempfile(fileext = ".xlsx")
similarity_to_df(s, data = refs, xlsx = out)
```

---

## similar_to *Abstracts most similar to a given one*

---

### Description

Abstracts most similar to a given one

### Usage

```
similar_to(s, which, n = 5, data = NULL, title = "title")
```

### Arguments

| | |
|---|---|
| s | A result from [doc_similarity()]. |
| which | The reference to look up: its id (for example the ‘bibkey‘) or its position among the abstracts used. |
| n | Number of similar abstracts to return. |
| data | Optional: the data frame passed to [doc_similarity()]. If given, the ‘title‘ column (see ‘title‘) is added to the result. |
| title | Name of the title column in ‘data‘. |

### Value

A data frame with ‘rank‘, ‘id‘, ‘similarity‘ (and ‘title‘).

## topic_documents *Topic assignments for each abstract*

### Description

Gives, for every abstract used in the model, its dominant topic, how strongly it belongs to that topic, and its mix of all topics.

### Usage

```
topic_documents(model, data = NULL)
```

### Arguments

| | |
|---|---|
| model | A model from [topic_model()]. |
| data | Optional: the same data frame you passed to [topic_model()]. If given, the topic columns are added to it (with 'NA' for rows that had no usable abstract), so you can keep working with your reference table. |

### Value

A data frame with 'dominant_topic', 'dominant_prob' and one 'topic_<number>' column per topic (an 'id' and 'row' column are included when 'data' is not given).

## topic_k_search *Compare different numbers of topics*

### Description

Fits a model for each value of 'k' and reports the average topic coherence (UMass; closer to zero means a topic's top words more often appear together). Look for the 'k' where coherence is best or stops improving, then read the topics to see whether they make sense. This is a guide, not a verdict.

### Usage

```
topic_k_search(data, k = 2:8, ...)
```

### Arguments

| | |
|---|---|
| data | A data frame, as for [topic_model()]. |
| k | Vector of topic counts to try. |
| ... | Other arguments passed to [topic_model()] (for example 'seed', 'n_iter', 'stopwords'). |

### Value

A data frame with ‘k‘, ‘mean_coherence‘ and ‘loglik‘; it has a ‘plot()‘ method.

### Examples

```
f <- system.file("extdata", "sample_abstracts.bib", package = "citelens")
topic_k_search(bib_to_df(f), k = 2:4, n_iter = 200, seed = 1)
```

---

## topic_model *Topic modeling (LDA) of abstracts*

---

### Description

Fits a Latent Dirichlet Allocation topic model to the abstracts in a data frame, such as the one returned by [bib_to_df()]. Each abstract is treated as a mixture of ‘k‘ topics and each topic as a distribution over words. Fitting uses a collapsed Gibbs sampler written in C++.

### Usage

```
topic_model(
  data,
  k = 5,
  text = "abstract",
  id = "bibkey",
  n_iter = 1000,
  alpha = 5/k,
  beta = 0.01,
  min_doc_freq = 2,
  max_doc_prop = 1,
  min_word_length = 3,
  stopwords = NULL,
  singularize = TRUE,
  seed = NULL
)
```

### Arguments

| | |
|---|---|
| data | A data frame with one row per reference. |
| k | Number of topics (at least 2). |
| text | Name of the column holding the text to analyse. |
| id | Name of a column that identifies each reference (default ‘"bibkey"‘). If it does not exist, row numbers are used. |
| n_iter | Number of Gibbs sampling sweeps. |
| alpha | Prior on how many topics each abstract mixes. Smaller values mean each abstract concentrates on fewer topics. Default ‘5 / k‘. |

| | |
|---|---|
| beta | Prior on how many words each topic uses. Smaller values mean more focused topics. |
| min_doc_freq | Drop words that occur in fewer than this many abstracts. |
| max_doc_prop | Drop words that occur in more than this share of the abstracts (1 keeps everything; try 0.5 to remove words common to nearly every abstract). |
| min_word_length | Drop words shorter than this many letters. |
| stopwords | Extra words to drop, in addition to [citelens_stopwords()]. |
| singularize | Merge simple plurals (default 'TRUE'). |
| seed | Optional number passed to [set.seed()] so the fit can be reproduced. |

## Details

Text is lower-cased, punctuation and numbers are removed, stop words (see [citelens_stopwords()]) and very short words are dropped, and simple plurals are merged (students and student count as one word). Words that occur in fewer than 'min_doc_freq' abstracts are ignored.

Topic models need a reasonably large collection (dozens of abstracts or more) to give stable themes. With a handful of abstracts the output is only exploratory. Different 'seed' values can give somewhat different topics; use [topic_k_search()] to help choose 'k'.

## Value

An object of class 'citelens_topics'. Use [topic_terms()], [topic_documents()], [topic_summary()], [plot_topic_terms()] and [plot_topic_prevalence()] to explore it. Topics are numbered from the most to the least prevalent.

## Examples

```
f <- system.file("extdata", "sample_abstracts.bib", package = "citelens")
refs <- bib_to_df(f)
filler <- c("note", "time", "follow", "shape", "touch", "studied", "evidence",
            "work", "attention", "discussion")   # words from the synthetic sample
m <- topic_model(refs, k = 3, stopwords = filler, seed = 1)
m
topic_terms(m, n = 5)
```

---

# topic_summary *One-row-per-topic summary*

---

## Description

One-row-per-topic summary

## Usage

```
topic_summary(model, n_terms = 5)
```

### Arguments

| | |
|---|---|
| model | A model from [topic_model()]. |
| n_terms | Number of top terms to list for each topic. |

### Value

A data frame with ‘topic‘, ‘prevalence‘ (average share of an abstract), ‘n_docs‘ (abstracts whose dominant topic it is), ‘coherence‘ (UMass; closer to zero is better) and ‘top_terms‘.

---

## topic_terms *Top terms of each topic*

---

### Description

Top terms of each topic

### Usage

```
topic_terms(model, n = 10)
```

### Arguments

| | |
|---|---|
| model | A model from [topic_model()]. |
| n | Number of terms to return per topic. |

### Value

A data frame with columns ‘topic‘, ‘rank‘, ‘term‘ and ‘beta‘ (the probability of the term within the topic).

---

## topic_trends *Topic trends over time*

---

### Description

Average share of each topic among the abstracts published in each year (or group of years). Shares in a period add up to 1.

### Usage

```
topic_trends(model, data, year = "year", bin = 1, n_terms = 3)
```

### Arguments

| | |
|---|---|
| model | A model from [topic_model()]. |
| data | The same data frame that was passed to [topic_model()]. |
| year | Name of the column holding the publication year. |
| bin | Width of each period in years. |
| n_terms | Number of top words shown in each topic's label. |

### Value

A data frame with 'period', 'topic', 'label', 'share' (average share of an abstract) and 'n_period' (abstracts in the period).

---

## word_trends *Word trends over time*

---

### Description

For each year (or group of years), reports the share of abstracts that mention a word. Words are cleaned the same way as in [topic_model()], so "student" also counts "students". Use it to see whether a concept is rising or fading across the literature.

### Usage

```
word_trends(
  data,
  words = NULL,
  n = 8,
  year = "year",
  text = "abstract",
  bin = 1,
  stopwords = NULL,
  min_word_length = 3,
  singularize = TRUE
)
```

### Arguments

| | |
|---|---|
| data | A data frame with one row per reference. |
| words | Words to track, such as 'c("equity", "belonging")'. If 'NULL', the 'n' words found in the most abstracts overall are used. |
| n | Number of words to track when 'words' is 'NULL'. |
| year | Name of the column holding the publication year. Text such as "2019" or "Spring 2019" is fine; the first 4-digit number is used. |
| text | Name of the column holding the text to analyse. |

| | |
|---|---|
| bin | Width of each period in years (1 = one row per year). |
| stopwords | Extra words to drop, in addition to [citelens_stopwords()]. |
| min_word_length | Drop words shorter than this many letters. |
| singularize | Merge simple plurals (default ‘TRUE‘). |

## Details

Years with only a few abstracts give jumpy shares, so check ‘n_period‘ (the number of abstracts in that year) before reading much into a change. Use ‘bin‘ to group years, for example ‘bin = 5‘.

## Value

A data frame with ‘period‘ (first year of the period), ‘term‘, ‘n_docs‘ (abstracts mentioning it), ‘n_period‘ (abstracts in the period) and ‘share‘ (‘n_docs / n_period‘). Periods with no abstracts are omitted.

## Examples

```
f <- system.file("extdata", "sample_abstracts.bib", package = "citelens")
refs <- bib_to_df(f)
word_trends(refs, words = c("retention", "tuition"), bin = 5)
```

# Index

bib_to_df, 2

citelens_stopwords, 3
convert_refworks, 4

doc_similarity, 4

fetch_abstracts, 6
find_duplicates, 7

plot_similarity, 8
plot_topic_prevalence, 9
plot_topic_terms, 9
plot_topic_trends, 10
plot_word_trends, 10

refworks_to_bib, 11
refworks_to_xlsx, 11

scholar_html_to_bib, 12
scholar_parse_html, 13
similar_to, 15
similarity_to_df, 14

topic_documents, 16
topic_k_search, 16
topic_model, 17
topic_summary, 18
topic_terms, 19
topic_trends, 19

word_trends, 20