Package ‘synthesisr’

June 3, 2020

Type Package

Title Import, Assemble, and Deduplicate Bibliographic Datasets

Version 0.3.0

Description A critical first step in systematic literature reviews and mining of academic texts is to identify relevant texts from a range of sources, particularly databases such as ‘Web of Science’ or ‘Scopus’. These databases often export in different formats or with different metadata tags. ‘synthesisr’ expands on the tools outlined by Westgate (2019) <doi:10.1002/jrsm.1374> to import bibliographic data from a range of formats (such as ‘bibtex’, ‘ris’, or ‘ciw’) in a standard way, and allows merging and deduplication of the resulting dataset.

Depends R (>= 3.5.0)

Imports stringdist

Suggests knitr, rmarkdown, testthat

Date 2020-05-18

License GPL-3

LazyData true

RoxygenNote 7.1.0

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Martin Westgate [aut, cre] (<https://orcid.org/0000-0003-0854-2034>), Eliza Grames [aut] (<https://orcid.org/0000-0003-1743-6815>)

Maintainer Martin Westgate <martinjwestgate@gmail.com>

Repository CRAN

Date/Publication 2020-06-03 16:30:03 UTC
**R topics documented:**

- `add_line_breaks` ............................................. 2
- `bibliography-class` ........................................ 3
- `clean` .......................................................... 4
- `code_lookup` .................................................. 5
- `deduplicate` ................................................... 5
- `detect_` .......................................................... 7
- `extract_unique_references` .................................... 8
- `find_duplicates` ............................................... 9
- `format_citation` ............................................. 11
- `fuzz_` ........................................................... 12
- `merge_columns` ................................................ 13
- `override_duplicates` ......................................... 14
- `parse_` .......................................................... 15
- `read_refs` ....................................................... 16
- `review_duplicates` ........................................... 17
- `string_` .......................................................... 18
- `synthesisr` ..................................................... 19
- `write_bib` ....................................................... 20

**Index**

| add_line_breaks | Add line breaks to one or more strings |

**Description**

This function takes a vector of strings and adds line breaks every n characters. Primarily built to be called internally by `format_citation`, this function has been made available as it can be useful in other contexts.

**Usage**

```
add_line_breaks(x, n = 50, max_n = 80, html = FALSE, max_time = 60)
```

**Arguments**

- `x` Either a string or a vector; if the vector is not of class character if will be coerced to one using `as.character`.
- `n` Numeric: The desired number of characters that should separate consecutive line breaks.
- `max_n` Numeric: The maximum number of characters that may separate consecutive line breaks.
- `html` logical: Should the line breaks be specified in html?
- `max_time` Numeric: What is the maximum amount of time (in seconds) allowed to adjust groups until character thresholds are reached?
Details

Line breaks are only added between words, so the value of n is actually a threshold value rather than being matched exactly. max_n is matched exactly if a limit is set and max_time is not reached finding new break points between words.

Value

Returns the input vector unaltered except for the addition of line breaks.

Examples

```r
add_line_breaks(c("On the Origin of Species"), n = 10)
```

Description

This is a small number of standard methods for interacting with class 'bibliography'. More may be added later.

Usage

```r
## S3 method for class 'bibliography'
summary(object, ...)

## S3 method for class 'bibliography'
print(x, n, ...)

## S3 method for class 'bibliography'
x[n]

## S3 method for class 'bibliography'
c(...)

## S3 method for class 'bibliography'
as.data.frame(x, ...)

as.bibliography(x, ...)
```

Arguments

- `object` An object of class 'bibliography'
- `...` Any further information
- `x` An object of class 'bibliography'
- `n` Number of items to select/print
Details

Methods for class bibliography

---

**clean_**

*Clean a data.frame or vector*

---

**Description**

Cleans column and author names

**Usage**

```
clean_df(data)
clean_authors(x)
clean_colnames(x)
```

**Arguments**

- `data` A data.frame with bibliographic information.
- `x` A vector of strings

**Value**

Returns the input, but cleaner.

**Examples**

```r
df <- data.frame(
  X..title. = c(
    "EviAtlas: a tool for visualising evidence synthesis databases",
    "revtools: An R package to support article screening for evidence synthesis",
    "An automated approach to identifying search terms for systematic reviews",
    "Reproducible, flexible and high-throughput data extraction from primary literature"),
  YEAR = c("2019", "2019", "2019", "2019"),
  authors = c(
    "Haddaway et al",
    "Westgate",
    "EM Grames AND AN Stillman & MW Tingley and CS Elphick",
    "Pick et al"
  )
)
clean_df(df)
```

# or use sub-functions
colnames(df) <- clean_colnames(df)
# colnames(df) <- clean_colnames(colnames(df)) # also works
df$author <- clean_authors(df$author)
```
code_lookup

Bibliographic code lookup for search results assembly

Description

A data frame that can be used to look up common codes for different bibliographic fields across databases and merge them to a common format.

Usage

code_lookup

Format

A data frame with 226 obs of 12 variables

code  code used in search results
order  the order in which to rank fields in assembled results
category_description  type of bibliographic data
entry_description  description of field
field  bibliographic field that codes correspond to
ris_generic  logical: If the code is used in generic ris files
ris_wos  logical: If the code is used in Web of Science ris files
ris_pubmed  logical: If the code is used in PubMed ris files
ris_scopus  logical: If the code is used in Scopus ris files
ris_asp  logical: If the code is used in Academic Search Premier ris files
ris_ovid  logical: If the code is used in Ovid ris files
ris_synthesisr  logical: If the code used in synthesisr imports & exports

deduplicate

Remove duplicates from a bibliographic data set

Description

Removes duplicates using sensible defaults

Usage

deduplicate(data, match_by, method, type = "merge", ...)


Arguments

data A data.frame containing bibliographic information.
match_by Name of the column in data where duplicates should be sought.
method The duplicate detection function to use; see see link{string_} or link{fuzz_} for examples. Passed to find_duplicates.
type How should entries be selected? Default is "merge" which selected the entries with the largest number of characters in each column. Alternatively "select" returns the row with the highest total number of characters.
... Arguments passed to find_duplicates.

Details

This is a wrapper function to find_duplicates and extract_unique_references, which tries to choose some sensible defaults. Use with care.

Value

A data.frame containing data identified as unique.

See Also

find_duplicates and extract_unique_references for underlying functions.

Examples

my_df <- data.frame(
  title = c("EviAtlas: a tool for visualising evidence synthesis databases",
    "revtools: An R package to support article screening for evidence synthesis",
    "An automated approach to identifying search terms for systematic reviews",
    "Reproducible, flexible and high-throughput data extraction from primary literature",
    "eviatlas: tool for visualizing evidence synthesis databases.",
    "REVTOOLS a package to support article-screening for evidence synthesis"
  ),
  year = c("2019", "2019", "2019", "2019", NA, NA),
  authors = c("Haddaway et al", "Westgate",
    "Grames et al", "Pick et al", NA, NA),
  stringsAsFactors = FALSE
)

# run deduplication
dups <- find_duplicates(
  my_df$title,
  method = "string_os",
  rm_punctuation = TRUE,
  to_lower = TRUE
)

extract_unique_references(my_df, matches = dups)
# or, in one line:
deduplicate(my_df, "title",
    method = "string_os", 
    rm_punctuation = TRUE,
    to_lower = TRUE)

---

**detect_**

**Detect file formatting information**

**Description**

Bibliographic data can be stored in a number of different file types, meaning that detecting consistent attributes of those files is necessary if they are to be parsed accurately. These functions attempt to identify some of those key file attributes. Specifically, `detect_parser` determines which `parse_` function to use; `detect_delimiter` and `detect_lookup` identify different attributes of RIS files; and `detect_year` attempts to fill gaps in publication years from other information stored in a `data.frame`.

**Usage**

- `detect_parser(x)`
- `detect_delimiter(x)`
- `detect_lookup(tags)`
- `detect_year(df)`

**Arguments**

- **x** A character vector containing bibliographic data
- **tags** A character vector containing RIS tags.
- **df** a `data.frame` containing bibliographic data

**Value**

detect_parser and detect_delimiter return a length-1 character; detect_year returns a character vector listing estimated publication years; and detect_lookup returns a `data.frame`.

**Examples**

```r
revtools <- c("","PMID- 31355546","VI - 10","IP - 4","DP - 2019 Dec","TI - revtools: An R package to support article
```
### extract_unique_references

Remove duplicates from a bibliographic data set

#### Description

Given a list of duplicate entries and a data set, this function extracts only unique references.

#### Usage

```r
extract_unique_references(data, matches, type = "merge")
```

#### Arguments

- **data**: A `data.frame` containing bibliographic information.
- **matches**: A vector showing which entries in `data` are duplicates.
- **type**: How should entries be selected to retain? Default is "merge" which selects the entries with the largest number of characters in each column. Alternatively "select" which returns the row with the highest total number of characters.

#### Value

Returns a `data.frame` of unique references.
find_duplicates

Detect duplicate values

Description

Identifies duplicate bibliographic entries using different duplicate detection methods.

Usage

find_duplicates(
  data,
  method = "exact",
  group_by,
find_duplicates

find_duplicates(data, method, group_by, threshold = 5, to_lower = FALSE, rm_punctuation = FALSE)

Arguments

- **data**: A character vector containing duplicate bibliographic entries.
- **method**: A string indicating how matching should be calculated. Either "exact" for exact matching (the default), or the name of a function for calculating string distance.
- **group_by**: An optional vector, data.frame or list containing data to use as 'grouping' variables; that is, categories within which duplicates should be sought. Defaults to NULL, in which case all entries are compared against all others. Ignored if method = "exact".
- **threshold**: Numeric: the cutoff threshold for deciding if two strings are duplicates. Sensible values depend on the method chosen. Defaults to 5 if method = "string_osa" and must be specified in all other instances except method = "exact" (where no threshold is required).
- **to_lower**: Logical: Should all entries be converted to lower case before calculating string distance? Defaults to FALSE.
- **rm_punctuation**: Logical: Should punctuation should be removed before calculating string distance? Defaults to FALSE.

Value

Returns a vector of duplicate matches, with attributes listing methods used.

See Also

- string_ or fuzz_ for suitable functions to pass to methods;
- extract_unique_references and deduplicate for higher-level functions.

Examples

```r
my_df <- data.frame(
  title = c("EviAtlas: a tool for visualising evidence synthesis databases",
            "revtools: An R package to support article screening for evidence synthesis",
            "An automated approach to identifying search terms for systematic reviews",
            "Reproducible, flexible and high-throughput data extraction from primary literature",
            "eviatlas: tool for visualizing evidence synthesis databases.",
            "REVTOOLS a package to support article-screening for evidence synthesis"),
  year = c("2019", "2019", "2019", "2019", NA, NA),
  authors = c("Haddaway et al", "Westgate",
              "Grames et al", "Pick et al", NA, NA),
stringsAsFactors = FALSE
)
```
# run deduplication

dups <- find_duplicates(
  my_df$title,
  method = "string_osa",
  rm_punctuation = TRUE,
  to_lower = TRUE
)

extract_unique_references(my_df, matches = dups)

# or, in one line:
deduplicate(my_df, "title",
  method = "string_osa",
  rm_punctuation = TRUE,
  to_lower = TRUE)

---

**format_citation**

*Format a citation*

**Description**

This function takes an object of class data.frame, list, or bibliography and returns a formatted citation.

**Usage**

```r
format_citation(
  data,
  details = TRUE,
  abstract = FALSE,
  add_html = FALSE,
  line_breaks = FALSE,
  ...
)
```

**Arguments**

- **data**
  - An object of class data.frame, list, or bibliography.

- **details**
  - Logical: Should identifying information such as author names & journal titles be displayed? Defaults to TRUE.

- **abstract**
  - Logical: Should the abstract be shown (if available)? Defaults to FALSE.

- **add_html**
  - Logical: Should the journal title be italicized using html codes? Defaults to FALSE.

- **line_breaks**
  - Either logical, stating whether line breaks should be added, or numeric stating how many characters should separate consecutive line breaks. Defaults to FALSE.

- **...**
  - any other arguments.
Value

Returns a string of length equal to length(data) that contains formatted citations.

Examples

```r
gender <- c("@article{haddaway2018,
title={ROSES RepOrting standards for Systematic Evidence Syntheses: pro forma, flow-diagram and descriptive summary of the plan and conduct of environmental systematic reviews and systematic maps},
author={Haddaway, Neal R and Macura, Biljana and Whaley, Paul and Pullin, Andrew S},
journal={Environmental Evidence},
volume={7},
number={1},
pages={7},
year={2018},
publisher={Springer}
})

tmp <- tempfile()
writeLines(gender, tmp)
citation <- read_ref(tmp)
format_citation(citation)
```

Description

These functions duplicate the approach of the ‘fuzzywuzzy’ Python library for calculating string similarity.

Usage

```r
fuzzdist(
  a,
  b,
  method = c("fuzz_m_ratio", "fuzz_partial_ratio", "fuzz_token_sort_ratio",
             "fuzz_token_set_ratio")
)
fuzz_m_ratio(a, b)
fuzz_partial_ratio(a, b)
fuzz_token_sort_ratio(a, b)
fuzz_token_set_ratio(a, b)
```
**Arguments**

- **a**: A character vector of items to match to b.
- **b**: A character vector of items to match to a.
- **method**: The method to use for fuzzy matching.

**Value**

Returns a score of same length as b, giving the proportional dissimilarity between a and b.

**Note**

- **fuzz_m_ratio**: is a measure of the number of letters that match between two strings. It is calculated as one minus two times the number of matched characters, divided by the number of characters in both strings.
- **fuzz_partial_ratio**: calculates the extent to which one string is a subset of the other. If one string is a perfect subset, then this will be zero.
- **fuzz_token_sort_ratio**: sorts the words in both strings into alphabetical order, and checks their similarity using fuzz_m_ratio.
- **fuzz_token_set_ratio**: is similar to fuzz_token_sort_ratio, but compares both sorted strings to each other, and to a third group made of words common to both strings. It then returns the maximum value of fuzz_m_ratio from these comparisons.
- **fuzzdist**: is a wrapper function, for compatibility with stringdist.

**Examples**

```r
fuzzdist("On the Origin of Species", 
"Of the Original Specs", 
method = "fuzz_m_ratio")
```

---

**merge_columns**

**Bind two or more data frames with different columns**

**Description**

Takes two or more data.frames with different column names or different column orders and binds them to a single data.frame.

**Usage**

```r
merge_columns(x, y)
```

**Arguments**

- **x**: Either a data.frame or a list of data.frames.
- **y**: A data.frame, optional if x is a list.
override_duplicates

Description

Manually override duplicates

Override duplicates

Value

Returns a single data.frame with all the input data frames merged.

Examples

def_1 <- data.frame(
    title = c("EviAtlas: a tool for visualising evidence synthesis databases",
              "revtools: An R package to support article screening for evidence synthesis" ),
    year = c("2019", "2019")
)
def_2 <- data.frame(
    title = c("An automated approach to identifying search terms for systematic reviews",
              "Reproducible, flexible and high-throughput data extraction from primary literature" ),
    authors = c("Grames et al", "Pick et al")
)
merge_columns(df_1, df_2)

override_duplicates

Description

Re-assign group numbers to text that was classified as duplicated but is unique.

Usage

override_duplicates(matches, overrides)

Arguments

matches Numeric: a vector of group numbers for texts that indicates duplicates and unique values returned by the find_duplicates function.

overrides Numeric: a vector of group numbers that are not true duplicates.

Value

The input matches vector with unique group numbers for members of groups that the user overrides.
Parse bibliographic text in a variety of formats

Description

Text in standard formats - such as imported via `readLines` - can be parsed using a variety of standard formats. Use `detect_parser` to determine which is the most appropriate parser for your situation.

Usage

```r
parse_pubmed(x)
parse_ris(x, tag_naming = "best_guess")
parse_bibtex(x)
parse_csv(x)
parse_tsv(x)
```

Arguments

- `x` A character vector containing bibliographic information in ris format.
- `tag_naming` What format are ris tags in? Defaults to "best_guess" See `read.refs` for a list of accepted arguments.

Value

Returns an object of class `bibliography` (ris, bib, or pubmed formats) or `data.frame` (csv or tsv).

Examples

```r
eviallas <- c(
  "TY - JOUR",
  "AU - Haddaway, Neal R.",
  "AU - Feierman, Andrew",
  "AU - Grainger, Matthew J.",
  "AU - Gray, Charles T.",
  "AU - Tanriver-Ayder, Ezgi",
  "AU - Dhaubanjar, Sanita",
  "AU - Westgate, Martin J.",
  "PY - 2019",
  "DA - 2019/06/04",
  "TI - EviAtlas: a tool for visualising evidence synthesis databases",
  "JO - Environmental Evidence",
  "SP - 22",
  "VL - 8",
)```
read.refs  

Import bibliographic search results

Description

Imports common bibliographic reference formats (i.e. .bib, .ris, or .txt).

Usage

```
read.refs(
  filename,
  tag_naming = "best_guess",
  return_df = TRUE,
  verbose = FALSE
)
```

```
read_ref(
  filename,
  tag_naming = "best_guess",
  return_df = TRUE,
  verbose = FALSE
)
```

Arguments

filename  
A path to a filename or vector of filenames containing search results to import.

tag_naming  
Either a length-1 character stating how should ris tags be replaced (see details for a list of options), or an object inheriting from class data.frame containing user-defined replacement tags.

return_df  
If TRUE (default), returns a data.frame; if FALSE, returns a list.

verbose  
If TRUE, prints status updates (defaults to FALSE).
Details

The default for argument tag_naming is "best_guess", which estimates what database has been used for ris tag replacement, then fills any gaps with generic tags. Any tags missing from the database (i.e. code_lookup) are passed unchanged. Other options are to use tags from Web of Science ("wos"), Scopus ("scopus"), Ovid ("ovid") or Academic Search Premier ("asp"). If a data.frame is given, then it must contain two columns: "code" listing the original tags in the source document, and "field" listing the replacement column/tag names. The data.frame may optionally include a third column named "order", which specifies the order of columns in the resulting data.frame; otherwise this will be taken as the row order. Finally, passing "none" to replace_tags suppresses tag replacement.

Value

Returns a data.frame or list of assembled search results.

Functions

• read_ref: Import a single file

Examples

litsearchr <- c(
  "@article{grames2019,
    title={An automated approach to identifying search terms for systematic reviews using keyword co-occurrence networks},
    author={Grames, Eliza M and Stillman, Andrew N and Tingley, Morgan W and Elphick, Chris S},
    journal={Methods in Ecology and Evolution},
    volume={10},
    number={10},
    pages={1645--1654},
    year={2019},
    publisher={Wiley Online Library}
  }
"
)

tmp <- tempfile()

writeLines(litsearchr, tmp)

df <- read_refs(tmp, return_df = TRUE, verbose = TRUE)

review_duplicates Manually review potential duplicates

Description

Allows users to manually review articles classified as duplicates.
Usage
review_duplicates(text, matches)

Arguments
- **text**: A character vector of the text that was used to identify potential duplicates.
- **matches**: Numeric: a vector of group numbers for texts that indicates duplicates and unique values returned by the `find_duplicates` function.

Value
A data.frame of potential duplicates grouped together.

string_ Calculate similarity between two strings

Description
These functions each access a specific "methods" argument provided by stringdist, and are provided for convenient calling by `find_duplicates`. They do not include any new functionality beyond that given by stringdist, which you should use for your own analyses.

Usage
- `string_os(a, b)`
- `string_lv(a, b)`
- `string_dl(a, b)`
- `string_hamming(a, b)`
- `string_lcs(a, b)`
- `string_qgram(a, b)`
- `string_cosine(a, b)`
- `string_jaccard(a, b)`
- `string_jw(a, b)`
- `string_soundex(a, b)`

Arguments
- **a**: A character vector of items to match to b.
- **b**: A character vector of items to match to a.
Value

Returns a score of same length as b, giving the dissimilarity between a and b.

Description

Systematic review searches include multiple databases that export results in a variety of formats with overlap in coverage between databases. To streamline the process of importing, assembling, and deduplicating results, synthesisr recognizes bibliographic files exported from databases commonly used for systematic reviews and merges results into a standardized format.

Import & Export

The key task performed by synthesisr is flexible import and presentation of bibliographic data. This is typically achieved by read.refs, which can import multiple files at once and link them together into a single data.frame. Conversely, export is via write.refs. Users that require more detailed control can use the following functions:

- detect_ Detect file attributes
- parse_ Parse a vector containing bibliographic data
- clean_ Cleaning functions for author and column names
- code_lookup A dataset of potential ris tags

Data formatting

- bibliography-class Methods for class 'bibliography'
- merge_columns rbind two data.frames with different numbers of columns
- format_citation Return a clean citation from a bibliography or data.frame
- add_line_breaks Set a maximum character width for strings

Deduplication

When importing from multiple databases, it is likely that there will be duplicates in the resulting dataset. The easiest way to deal with this problem in synthesisr is using the deduplicate command; but this can be risky, particularly if there are no DOIs in the dataset. To get finer control of the deduplication process, consider using the sub-functions:

- find_duplicates Locate potentially duplicated references
- extract_unique_references Return a data.frame with only ‘unique’ references
- review_duplicates Manually review potential duplicates
- override_duplicates Manually override identified duplicates
- fuzz_ Fuzzy string matching c/o ‘fuzzywuzzy’
- string_ Fuzzy string matching c/o stringdist
**write_bib**  
*Export data to a bibliographic format*

**Description**

This function exports data.frames containing bibliographic information to either a .ris or .bib file.

**Usage**

```r
write_bib(x)
write_ris(x, tag_naming = "synthesisr")
write.refs(x, format = "ris", tag_naming = "synthesisr", file = FALSE)
```

**Arguments**

- **x**: Either a data.frame containing bibliographic information or an object of class bibliography.
- **tag_naming**: what naming convention should be used to write RIS files? See details for options.
- **format**: What format should the data be exported as? Options are ris or bib.
- **file**: Either logical indicating whether a file should be written (defaulting to FALSE), or a character giving the name of the file to be written.

**Value**

Returns a character vector containing bibliographic information in the specified format if file is FALSE, or saves output to a file if TRUE.

**Functions**

- `write_bib`: Format a bib file for export
- `write_ris`: Format a ris file for export

**Examples**

```r
eviatlas <- c(
  "TY - JOUR",
  "AU - Haddaway, Neal R.",
  "AU - Feierman, Andrew",
  "AU - Grainger, Matthew J.",
  "AU - Gray, Charles T.",
  "AU - Tanriver-Ayder, Ezgi",
  "AU - Dhaubanjar, Sanita",
  "AU - Westgate, Martin J.",
  "PY - 2019",
)
```
"DA - 2019/06/04",
"TI - EviAtlas: a tool for visualising evidence synthesis databases",
"JO - Environmental Evidence",
"SP - 22",
"VL - 8",
"IS - 1",
"SN - 2047-2382",
"UR - https://doi.org/10.1186/s13750-019-0167-1",
"DO - 10.1186/s13750-019-0167-1",
"ID - Haddaway2019",
"ER - ")

detect_parser(eviatlas) # = "parse_ris"
df <- as.data.frame(parse_ris(eviatlas))
ris_out <- writeRefs(df, format = "ris", file = FALSE)
Index

*Topic datasets
  code_lookup, 5
  [.bibliography (bibliography-class), 3
  add_line_breaks, 2, 19
  as.bibliography (bibliography-class), 3
  as.data.frame.bibliography
    (bibliography-class), 3
  bibliography-class, 3
  c.bibliography (bibliography-class), 3
  c.bibliography, (bibliography-class), 3
  clean_., 4, 19
  clean_authors (clean_), 4
  clean_colnames (clean_), 4
  clean_df (clean_), 4
  code_lookup, 5, 19
  deduplicate, 5, 9, 10, 19
  detect_., 7, 19
  detect_delimiter (detect_), 7
  detect_lookup (detect_), 7
  detect_parser, 15
  detect_parser (detect_), 7
  detect_year (detect_), 7
  extract_unique_references, 6, 8, 10, 19
  find_duplicates, 6, 9, 14, 18, 19
  format_citation, 11, 19
  fuzz_, 10, 12, 19
  fuzz_m_ratio (fuzz_), 12
  fuzz_partial_ratio (fuzz_), 12
  fuzz_token_set_ratio (fuzz_), 12
  fuzz_token_sort_ratio (fuzz_), 12
  fuzzdist (fuzz_), 12
  merge_columns, 13, 19
  override_duplicates, 14, 19
  parse_, 7, 15, 19
  parse_bibtex (parse_), 15
  parse_csv (parse_), 15
  parse_pubmed (parse_), 15
  parse_ris (parse_), 15
  parse_tsv (parse_), 15
  print.bibliography
    (bibliography-class), 3
  print.bibliography,
    (bibliography-class), 3
  read_ref (readRefs), 16
  read_refs, 15, 16, 19
  readLines, 15
  review_duplicates, 17, 19
  string_, 10, 18, 19
  string_cosine (string_), 18
  string_dl (string_), 18
  string_hamming (string_), 18
  string_jaccard (string_), 18
  string_jw (string_), 18
  string_lcs (string_), 18
  string_lv (string_), 18
  string_osaa (string_), 18
  string_qgram (string_), 18
  string_soundex (string_), 18
  summary.bibliography
    (bibliography-class), 3
  summary.bibliography,
    (bibliography-class), 3
  synthesisr, 19
  write_bib, 20
  write_refs, 19
  write.refs (write_bib), 20
  write_ris (write_bib), 20