

# Package ‘akc’

January 6, 2023

**Title** Automatic Knowledge Classification

**Version** 0.9.9

**Description** A tidy framework for automatic knowledge classification and visualization. Currently, the core functionality of the framework is mainly supported by modularity-based clustering (community detection) in keyword co-occurrence network, and focuses on co-word analysis of bibliometric research. However, the designed functions in 'akc' are general, and could be extended to solve other tasks in text mining as well.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 4.0.0)

**Imports** igraph, dplyr, ggplot2, stringr, ggraph (>= 1.0.2), tidygraph (>= 1.1.2), ggforce, textstem, tibble, tidytext, rlang, magrittr, data.table (>= 1.13.0), ggwordcloud (>= 0.5.0), tidyfst

**URL** <https://github.com/hope-data-science/akc>

**RoxygenNote** 7.2.1

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Tian-Yuan Huang [aut, cre] (<<https://orcid.org/0000-0002-4151-3764>>)

**Maintainer** Tian-Yuan Huang <[huang.tian-yuan@qq.com](mailto:huang.tian-yuan@qq.com)>

**Repository** CRAN

**Date/Publication** 2023-01-06 08:00:02 UTC

## R topics documented:

|                            |   |
|----------------------------|---|
| bibli_data_table . . . . . | 2 |
| doc_group . . . . .        | 3 |

|                           |           |
|---------------------------|-----------|
| keyword_clean . . . . .   | 4         |
| keyword_cloud . . . . .   | 5         |
| keyword_extract . . . . . | 6         |
| keyword_group . . . . .   | 8         |
| keyword_merge . . . . .   | 9         |
| keyword_network . . . . . | 11        |
| keyword_table . . . . .   | 12        |
| keyword_vis . . . . .     | 13        |
| make_dict . . . . .       | 14        |
| <b>Index</b>              | <b>16</b> |

---

|                  |  |
|------------------|--|
| bibli_data_table | <i>A selected dataset of bibliometric data on the topic of "Library science"</i> |
|------------------|--|

---

### Description

A selected sample of bibliometric data about topics on "Library science".

Period: 2019

Database: Web of Science Core Collection

### Usage

bibli\_data\_table

### Format

A data frame with 1448 rows and 4 variables:

**id** Unique article identifier for each article

**title** Title of the article

**keyword** Keyword list of the article

**abstract** Abstract of the article

---

`doc_group`*Construct network of documents based on keyword co-occurrence*

---

## Description

Create a `tbl_graph`(a class provided by **tidygraph**) from the tidy table with document ID and keyword. Each entry(row) should contain only one document and keyword in the tidy format. This function would group the documents.

## Usage

```
doc_group(  
  dt,  
  id = "id",  
  keyword = "keyword",  
  com_detect_fun = group_fast_greedy  
)
```

## Arguments

|                             |   |
|-----------------------------|---|
| <code>dt</code>             | A data.frame containing at least two columns with document ID and keyword.  |
| <code>id</code>             | Quoted characters specifying the column name of document ID. Default uses "id".   |
| <code>keyword</code>        | Quoted characters specifying the column name of keyword. Default uses "keyword".  |
| <code>com_detect_fun</code> | Community detection function, provided by <b>tidygraph</b> (wrappers around clustering functions provided by <b>igraph</b> ), see <a href="#">group_graph</a> to find other optional algorithms. Default uses <a href="#">group_fast_greedy</a> . |

## Details

As we could classify keywords using document ID, we could also classify documents with keywords. In the output network, the nodes are documents and the edges mean the two documents share same keywords with each other.

## Value

A `tbl_graph`, representing the document relation network based on keyword co-occurrence.

## Examples

```
library(akc)  
bibli_data_table %>%  
  keyword_clean(id = "id", keyword = "keyword") %>%  
  doc_group(id = "id", keyword = "keyword") -> grouped_doc  
  
grouped_doc
```

---

|               |   |
|---------------|---|
| keyword_clean | <i>Automatic keyword cleaning and transfer to tidy format</i> |
|---------------|---|

---

### Description

Carry out several keyword cleaning processes automatically and return a tidy table with document ID and keywords.

### Usage

```
keyword_clean(
  df,
  id = "id",
  keyword = "keyword",
  sep = ";",
  rmParentheses = TRUE,
  rmNumber = TRUE,
  lemmatize = FALSE,
  lemmatize_dict = NULL
)
```

### Arguments

|                |   |
|----------------|---|
| df             | A data.frame containing at least two columns with document ID and keyword strings with separators.  |
| id             | Quoted characters specifying the column name of document ID. Default uses "id".   |
| keyword        | Quoted characters specifying the column name of keywords. Default uses "keyword".   |
| sep            | Separator(s) of keywords. Default uses ";".   |
| rmParentheses  | Remove the contents in the parentheses (including the parentheses) or not. Default uses TRUE.   |
| rmNumber       | Remove the pure number sequence or no. Default uses TRUE.   |
| lemmatize      | Lemmatize the keywords or not. Lemmatization is supported by 'lemmatize_strings' function in 'textstem' package. Default uses FALSE.  |
| lemmatize_dict | A dictionary of base terms and lemmas to use for replacement. Only used when the <b>lemmatize</b> parameter is TRUE. The first column should be the full word form in lower case while the second column is the corresponding replacement lemma. Default uses NULL, this would apply the default dictionary used in <a href="#">lemmatize_strings</a> function. |

**Details**

The entire cleaning processes include: 1.Split the text with separators; 2.Remove the contents in the parentheses (including the parentheses); 3.Remove white spaces from start and end of string and reduces repeated white spaces inside a string; 4.Remove all the null character string and pure number sequences; 5.Convert all letters to lower case; 6.Lemmatization. Some of the procedures could be suppressed or activated with parameter adjustments. Default setting did not use lemmatization, it is suggested to use [keyword\\_merge](#) to merge the keywords afterward.

**Value**

A tbl with two columns, namely document ID and cleaned keywords.

**See Also**

[keyword\\_merge](#)

**Examples**

```
library(akc)

bibli_data_table

bibli_data_table %>%
  keyword_clean(id = "id",keyword = "keyword")
```

---

|               |   |
|---------------|---|
| keyword_cloud | <i>Draw word cloud for grouped keywords</i> |
|---------------|---|

---

**Description**

This function should be used to plot the object exported by [keyword\\_group](#). It could draw a robust word cloud of keywords.

**Usage**

```
keyword_cloud(tibble_graph, group_no = NULL, top = 50, max_size = 20)
```

**Arguments**

|              |  |
|--------------|--|
| tibble_graph | A tibble_graph output by <a href="#">keyword_group</a> .   |
| group_no     | If one wants to visualize a specific group, gives the group number. Default uses NULL, which returns all the groups. |
| top          | How many top keywords (by frequency) should be plot? Default uses 50.  |
| max_size     | Size of largest keyword. Default uses 20.  |

## Details

In the output graph, the size of keywords is proportional to the keyword frequency, keywords in different colours belong to different group. For advanced usage of word cloud, use **ggwordcloud** directly with the grouped keywords yielded by [keyword\\_group](#).

## See Also

[keyword\\_group](#), [geom\\_text\\_wordcloud\\_area](#)

## Examples

```
library(dplyr)
library(akc)

bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") -> grouped_keyword

grouped_keyword %>%
  keyword_ccloud()

grouped_keyword %>%
  keyword_ccloud(group_no = 1)
```

---

|                 |                                       |
|-----------------|---------------------------------------|
| keyword_extract | <i>Extract keywords from raw text</i> |
|-----------------|---------------------------------------|

---

## Description

When we have raw text like abstract or article but not keywords, we might prefer extracting keywords first. The least prerequisite data to be provided are a data.frame with document id and raw text, and a user defined dictionary should be provided. One could use [make\\_dict](#) function to construct his(her) own dictionary with a character vector containing the vocabularies. If the dictionary is not provided, the function would return all the ngram tokens without filtering (not recommended).

## Usage

```
keyword_extract(
  dt,
  id = "id",
  text,
  dict = NULL,
  stopword = NULL,
  n_max = 4,
  n_min = 1
)
```

**Arguments**

|          |  |
|----------|--|
| dt       | A data.frame containing at least two columns with document ID and text strings for extraction.   |
| id       | Quoted characters specifying the column name of document ID. Default uses "id".  |
| text     | Quoted characters specifying the column name of raw text for extraction.   |
| dict     | A data.table with two columns, namely "id" and "keyword"(set as key). This should be exported by <code>make_dict</code> function. The default uses NULL, which means the output keywords are not filtered by the dictionary (usually not recommended). |
| stopword | A vector containing the stop words to be used. Default uses NULL.  |
| n_max    | The number of words in the n-gram. This must be an integer greater than or equal to 1. Default uses 4.   |
| n_min    | This must be an integer greater than or equal to 1, and less than or equal to n_max. Default uses 1.   |

**Details**

In the procedure of keyword extraction from **akc**, first the raw text would be split into independent clause (namely split by punctuations of [ , ; ! ? . ] ). Then the ngrams of the clauses would be extracted. Finally, the phrases represented by ngrams should be in the dictionary created by the user (using `make_dict`). The user could also specify the *n* of ngrams.

This function could take some time if the sample size is large, it is suggested to use `system.time` to do some test first. Nonetheless, it has been optimized by `data.table` codes already and has good performance for big data.

**Value**

A data.frame(tibble) with two columns, namely document ID and extracted keyword.

**See Also**

[make\\_dict](#)

**Examples**

```
library(akc)
library(dplyr)

bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  pull(keyword) %>%
  make_dict -> my_dict

tidytext::stop_words %>%
  pull(word) %>%
  unique() -> my_stopword
```

```
bibli_data_table %>%
  keyword_extract(id = "id", text = "abstract",
    dict = my_dict, stopwords = my_stopword)
```

---

 keyword\_group

---

*Construct network from a tidy table and divide them into groups*


---

### Description

Create a `tbl_graph` (a class provided by **tidygraph**) from the tidy table with document ID and keyword. Each entry (row) should contain only one keyword in the tidy format. This function would automatically compute the frequency and classification group number of nodes representing keywords.

### Usage

```
keyword_group(
  dt,
  id = "id",
  keyword = "keyword",
  top = 200,
  min_freq = 1,
  com_detect_fun = group_fast_greedy
)
```

### Arguments

|                             |   |
|-----------------------------|---|
| <code>dt</code>             | A data.frame containing at least two columns with document ID and keyword.  |
| <code>id</code>             | Quoted characters specifying the column name of document ID. Default uses "id".   |
| <code>keyword</code>        | Quoted characters specifying the column name of keyword. Default uses "keyword".  |
| <code>top</code>            | The number of keywords selected with the largest frequency. If there is a tie, more than <i>top</i> entries would be selected.  |
| <code>min_freq</code>       | Minimum occurrence of selected keywords. Default uses 1.  |
| <code>com_detect_fun</code> | Community detection function, provided by <b>tidygraph</b> (wrappers around clustering functions provided by <b>igraph</b> ), see <a href="#">group_graph</a> to find other optional algorithms. Default uses <a href="#">group_fast_greedy</a> . |

### Details

This function receives a tidy table with document ID and keyword. Only top keywords with largest frequency would be selected and the minimum occurrence of keywords could be specified. For suggestions of community detection algorithm, see the references provided below.



**Value**

A `tbl_graph`, representing the keyword co-occurrence network with frequency and group number of the keywords.

**References**

de Sousa, Fabiano Berardo, and Liang Zhao. "Evaluating and comparing the igraph community detection algorithms." 2014 Brazilian Conference on Intelligent Systems. IEEE, 2014.

Yang, Z., Algesheimer, R., & Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, 6, 30750.

**See Also**

[tbl\\_graph](#), [group\\_graph](#)

**Examples**

```
library(akc)

bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword")

# use 'louvain' algorithm for community detection

bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword",
    com_detect_fun = group_louvain)

# get more alternatives by searching '?tidygraph::group_graph'
```

---

keyword\_merge

*Merge keywords that supposed to have same meanings*

---

**Description**

Merge keywords that have common stem or lemma, and return the majority form of the word. This function receives a tidy table (`data.frame`) with document ID and keyword waiting to be merged.

**Usage**

```
keyword_merge(
  dt,
  id = "id",
  keyword = "keyword",
  reduce_form = "lemma",
```

```

    lemmatize_dict = NULL,
    stem_lang = "porter"
  )

```

### Arguments

|                |   |
|----------------|---|
| dt             | A data.frame containing at least two columns with document ID and keyword.  |
| id             | Quoted characters specifying the column name of document ID. Default uses "id".   |
| keyword        | Quoted characters specifying the column name of keyword. Default uses "keyword".  |
| reduce_form    | Merge keywords with the same stem("stem") or lemma("lemma"). See details. Default uses "lemma". Another advanced option is "partof". If a non-unigram (A) is part (subset) of another non-unigram (B), then the longer one(B) would be replaced by the shorter one(A).  |
| lemmatize_dict | A dictionary of base terms and lemmas to use for replacement. Only used when the <b>lemmatize</b> parameter is TRUE. The first column should be the full word form in lower case while the second column is the corresponding replacement lemma. Default uses NULL, this would apply the default dictionary used in <a href="#">lemmatize_strings</a> function. Applicable when <b>reduce_form</b> takes "lemma". |
| stem_lang      | The name of a recognized language. The list of supported languages could be found at <a href="#">getStemLanguages</a> . Applicable when <b>reduce_form</b> takes "stem".  |

### Details

While `keyword_clean` has provided a robust way to lemmatize the keywords, the returned token might not be the most common way to use. This function first gets the stem or lemma of every keyword using [stem\\_strings](#) or [lemmatize\\_strings](#) from `textstem` package, then find the most frequent form (if more than 1, randomly select one) for each stem or lemma. Last, every keyword would be replaced by the most frequent keyword which share the same stem or lemma with it.

When the 'reduce\_form' is set to "partof", then for non-unigrams in the same document, if one non-unigram is the subset of another, then they would be merged into the shorter one, which is considered to be more general (e.g. "time series" and "time series analysis" would be merged into "time series" if they co-occur in the same document). This could reduce the redundant information. This is only applied to multi-word phrases, because using it for one word would oversimplify the token and cause information loss (therefore, "time series" and "time" would not be merged into "time"). This is an advanced option that should be used with caution (A trade-off between information generalization and detailed information retention).

### Value

A tibble, namely a tidy table with document ID and merged keyword.

### See Also

[stem\\_strings](#), [lemmatize\\_strings](#)

**Examples**

```
library(akc)

bibli_data_table %>%
  keyword_clean(lemmatize = FALSE) %>%
  keyword_merge(reduce_form = "stem")

bibli_data_table %>%
  keyword_clean(lemmatize = FALSE) %>%
  keyword_merge(reduce_form = "lemma")
```

---

|                 |   |
|-----------------|---|
| keyword_network | <i>Flexible visualization of network (alternative to 'keyword_vis')</i> |
|-----------------|---|

---

**Description**

Providing flexible visualization of [keyword\\_vis](#). The group size would be showed, and user could extract specific group to visualize.

**Usage**

```
keyword_network(
  tibble_graph,
  group_no = NULL,
  facet = TRUE,
  max_nodes = 10,
  alpha = 0.7
)
```

**Arguments**

|              |  |
|--------------|--|
| tibble_graph | A tbl_graph output by <a href="#">keyword_group</a> .  |
| group_no     | If one wants to visualize a specific group, gives the group number. Default uses NULL, which returns all the groups. |
| facet        | Whether the figure should use facet or not.  |
| max_nodes    | The maximum number of nodes displayed in each group.   |
| alpha        | The transparency of label. Must lie between 0 and 1. Default uses 0.7.   |

**Details**

If the group\_no is not specified, when facet == TRUE, the function returns a faceted figure with limited number of nodes (adjusted by max\_nodes parameter). The "N=" shows the total size of the group.

When `facet == FALSE`, all the nodes would be displayed in one network. Colors are used to specify the groups, the size of nodes is proportional to the keyword frequency, while the alpha of edges is proportional to the co-occurrence relationship between keywords.

If the `group_no` is specified, returns the network visualization of the group. If you want to display all the nodes, set `max_nodes` to `Inf`.

### Value

An object yielded by `ggraph`

### See Also

`ggraph`, `keyword_vis`

### Examples

```
library(akc)

bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") %>%
  keyword_network()

# use color with `scale_fill_`
bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") %>%
  keyword_network() + ggplot2::scale_fill_viridis_d()

# without facet
bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") %>%
  keyword_network(facet = FALSE)

# get Group 5
bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") %>%
  keyword_network(group_no = 5)
```

---

keyword\_table

*Display the table with different groups of keywords*

---

### Description

Display the result of network-based keyword clustering, with frequency information attached.

**Usage**

```
keyword_table(tibble_graph, top = 10)
```

**Arguments**

`tibble_graph` A `tbl_graph` output by [keyword\\_group](#).

`top` How many keywords should be displayed in the table for each group. Default uses 10. If there is a tie, more than *top* keywords would be selected. To show all the keywords, use *Inf*.

**Value**

A tibble with two columns, namely `group` and `keywords` with frequency attached. Different keywords are separated by semicolon(';').

**See Also**

[keyword\\_group](#)

**Examples**

```
library(akc)

bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") %>%
  keyword_table()
```

---

keyword\_vis

*Visualization of grouped keyword co-occurrence network*

---

**Description**

Visualization of network-based keyword clustering, with frequency and co-occurrence information attached.

**Usage**

```
keyword_vis(tibble_graph, facet = TRUE, max_nodes = 10, alpha = 0.7)
```

**Arguments**

`tibble_graph` A `tbl_graph` output by [keyword\\_group](#).

`facet` Whether the figure should use facet or not.

`max_nodes` The maximum number of nodes displayed in each group.

`alpha` The transparency of label. Must lie between 0 and 1. Default uses 0.7.

**Details**

When `facet == TRUE`, the function returns a faceted figure with limited number of nodes (adjusted by `max_nodes` parameter). When `facet == FALSE`, all the nodes would be displayed in one network. Colors are used to specify the groups, the size of nodes is proportional to the keyword frequency, while the alpha of edges is proportional to the co-occurrence relationship between keywords.

**Value**

An object yielded by [ggraph](#)

**See Also**

[ggraph](#)

**Examples**

```
library(akc)

bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") %>%
  keyword_vis()

# without facet
bibli_data_table %>%
  keyword_clean(id = "id", keyword = "keyword") %>%
  keyword_group(id = "id", keyword = "keyword") %>%
  keyword_vis(facet = FALSE)
```

---

make\_dict

*Making one's own dictionary*

---

**Description**

Constructing a dictionary using a string vector with user defined vocabulary.

**Usage**

```
make_dict(dict_vocabulary_vector)
```

**Arguments**

`dict_vocabulary_vector`

A character vector containing the user defined professional vocabulary.

**Details**

Build a user defined vocabulary for keyword extraction ([keyword\\_extract](#)).

**Value**

A data.table with document id and keyword,using keyword as the key.

**See Also**

[keyword\\_extract](#)

**Examples**

```
library(akc)
library(dplyr)

bibli_data_table %>%
  keyword_clean() %>%
  pull(keyword) %>%
  make_dict() -> dict
```

# Index

## \* datasets

**bibli\_data\_table**, 2

bibli\_data\_table, 2

doc\_group, 3

geom\_text\_wordcloud\_area, 6

getStemLanguages, 10

ggraph, 12, 14

group\_fast\_greedy, 3, 8

group\_graph, 3, 8, 9

keyword\_clean, 4

keyword\_cloud, 5

keyword\_extract, 6, 15

keyword\_group, 5, 6, 8, 11, 13

keyword\_merge, 5, 9

keyword\_network, 11

keyword\_table, 12

keyword\_vis, 11, 12, 13

lemmatize\_strings, 4, 10

make\_dict, 6, 7, 14

stem\_strings, 10

tbl\_graph, 9