Package: SportMiner (via r-universe)

May 19, 2026

Type Package

Title Text Mining and Topic Modeling for Sport Science Literature

Version 0.1.0

Description A comprehensive toolkit for mining, analyzing, and visualizing scientific literature in sport science domains. Provides functions for retrieving abstracts from 'Scopus', preprocessing text data, performing advanced topic modeling using Latent Dirichlet Allocation ('LDA'), Structural Topic Models ('STM'), and Correlated Topic Models ('CTM'), and creating publication-ready visualizations including keyword co-occurrence networks and topic trends. For methodological details see Blei et al. (2003) <doi:10.1162/jmlr.2003.3.4-5.993> for 'LDA', Roberts et al. (2014) <doi:10.1111/ajps.12103> for 'STM', and Blei and Lafferty (2007) <doi:10.1214/07-AOAS114> for 'CTM'.

License MIT + file LICENSE

URL https://github.com/praveenmaths89/SportMiner

BugReports https://github.com/praveenmaths89/SportMiner/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports rscopus, dplyr, tidytext, topicmodels, stm, ggplot2, rlang, SnowballC, scales, textmineR, Matrix, tidyr, ggraph, igraph, widyr, magrittr, slam

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

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libglpk-dev make libgsl0-dev libicu-dev libxml2-dev libssl-dev

Repository https://praveenmaths89.r-universe.dev

Date/Publication 2026-01-19 14:07:42 UTC

RemoteUrl https://github.com/praveenmaths89/sportminer

RemoteRef HEAD

RemoteSha 7f70ab2e7a7e8d88bb86f6482270acc810121e33

Contents

`sm_compare_models` *Compare Multiple Topic Models*

Description

Trains and compares three topic modeling approaches: LDA (Latent Dirichlet Allocation), STM (Structural Topic Model), and CTM (Correlated Topic Model). Calculates semantic coherence and exclusivity metrics for each model and suggests the optimal model.

Usage

```
sm_compare_models(
  dtm,
  k = 10,
  metadata = NULL,
  prevalence = NULL,
  seed = 1729,
  lda_method = "gibbs",
  verbose = TRUE
)
```

Arguments

dtm	A DocumentTermMatrix object.
k	Number of topics to extract. Default is 10.
metadata	Optional data frame with document-level covariates for STM. Must have the same number of rows as dtm. Default is NULL.
prevalence	Optional formula for STM prevalence specification. Default is NULL.
seed	Random seed for reproducibility. Default is 1729.
lda_method	Method for LDA. Options: "gibbs" or "vem". Default is "gibbs".
verbose	Logical indicating whether to print progress messages. Default is TRUE.

Value

A list containing:

models	List of fitted models (lda, stm, ctm)
metrics	Data frame comparing coherence and exclusivity
recommendation	Character string naming the optimal model

Examples

```
## Not run:
# Requires document-term matrix from sm_create_dtm()
dtm <- sm_create_dtm(processed_data)
comparison <- sm_compare_models(dtm, k = 10)
print(comparison$metrics)
print(comparison$recommendation)

## End(Not run)
```

---

sm_create_dtm *Create Document-Term Matrix*

Description

Converts preprocessed word counts into a document-term matrix suitable for topic modeling. Filters rare terms and empty documents.

Usage

```
sm_create_dtm(word_counts, min_term_freq = 3, max_term_freq = 0.5)
```

Arguments

word_counts	A data.frame with columns doc_id, stem, and n, typically produced by sm_preprocess_text().
min_term_freq	Minimum number of documents a term must appear in to be retained. Default is 3.
max_term_freq	Maximum proportion of documents a term can appear in. Useful for removing ubiquitous terms. Default is 0.5 (50 percent).

Value

A DocumentTermMatrix object from the tm package.

Examples

```
## Not run:
processed <- sm_preprocess_text(papers)
dtm <- sm_create_dtm(processed)

## End(Not run)
```

`sm_get_indexed_keywords` *Get Indexed Keywords from Scopus*

Description

Retrieves indexed keywords for a single paper using its DOI or EID. This function makes an additional API call per paper, so use judiciously.

Usage

```
sm_get_indexed_keywords(doi = NA, eid = NA, verbose = FALSE)
```

Arguments

`doi`	Character string containing the paper's DOI.
`eid`	Character string containing the paper's EID (Scopus identifier).
`verbose`	Logical indicating whether to print error messages. Default is FALSE.

Value

Character string of indexed keywords separated by " | ", or NA if not available.

Examples

```
## Not run:
# Requires Scopus API key
keywords <- sm_get_indexed_keywords(
  doi = "10.1016/j.jsams.2020.01.001"
)

## End(Not run)
```

`sm_keyword_network` *Create Keyword Co-occurrence Network*

Description

Generates and visualizes a keyword co-occurrence network from author-provided keywords. Shows which keywords frequently appear together in the same papers.

Usage

```
sm_keyword_network(
  data,
  keyword_col = "author_keywords",
  separator = "; ",
  min_cooccurrence = 2,
  top_n = 30,
  layout = "fr"
)
```

Arguments

`data`	Data frame containing papers with keyword information.
`keyword_col`	Name of the column containing keywords. Default is "author_keywords".
`separator`	Character string separating keywords within a cell. Default is "; " (Scopus format).
`min_cooccurrence`	Minimum number of times keywords must co-occur to be included in the network. Default is 2.
`top_n`	Number of top keywords (by frequency) to include. If NULL, includes all keywords meeting min_cooccurrence. Default is 30.
`layout`	Network layout algorithm. Options include "fr" (Fruchterman-Reingold), "kk" (Kamada-Kawai), "circle". Default is "fr".

Value

A ggraph/ggplot object displaying the keyword network.

Examples

```
## Not run:
# Requires API data from sm_search_scopus()
papers <- sm_search_scopus(query, max_count = 100)
network_plot <- sm_keyword_network(papers, top_n = 25)
print(network_plot)

## End(Not run)
```

sm_plot_topic_frequency *Plot Topic Frequency Distribution*

Description

Creates a bar chart showing how many documents are assigned to each topic.

Usage

```
sm_plot_topic_frequency(model, dtm, threshold = 0.3)
```

Arguments

model	A fitted topic model (LDA, STM, or CTM).
dtm	The document-term matrix used to train the model.
threshold	Minimum gamma probability for topic assignment. Default is 0.3.

Value

A ggplot object.

Examples

```
## Not run:
# Requires trained model from sm_train_lda()
lda_model <- sm_train_lda(dtm, k = 10)
sm_plot_topic_frequency(lda_model, dtm)

## End(Not run)
```

sm_plot_topic_terms *Plot Topic Term Probabilities*

Description

Creates a bar chart showing the top terms for each topic, based on their beta (topic-word) probabilities.

Usage

```
sm_plot_topic_terms(model, n_terms = 10, topics = NULL)
```

Arguments

model	A fitted topic model (LDA, STM, or CTM).
n_terms	Number of top terms to display per topic. Default is 10.
topics	Vector of topic numbers to display. If NULL, shows all topics. Default is NULL.

Value

A ggplot object.

Examples

```
## Not run:
# Requires trained model from sm_train_lda()
lda_model <- sm_train_lda(dtm, k = 10)
sm_plot_topic_terms(lda_model, n_terms = 15)

## End(Not run)
```

---

sm_plot_topic_trends *Plot Topic Trends Over Time*

Description

Creates a stacked percentage bar chart showing how topic proportions change over publication years.

Usage

```
sm_plot_topic_trends(
  model,
  dtm,
  metadata,
  doc_id_col = "doc_id",
  year_filter = NULL
)
```

Arguments

model	A fitted topic model (LDA, STM, or CTM).
dtm	The document-term matrix used to train the model.
metadata	Data frame with a 'year' column and document identifiers.
doc_id_col	Name of the document ID column in metadata. Default is "doc_id".
year_filter	Optional vector of years to include. Default is NULL (includes all years).

Value

A ggplot object.

Examples

```
## Not run:
# Requires trained model and metadata
papers$doc_id <- paste0("doc_", seq_len(nrow(papers)))
lda_model <- sm_train_lda(dtm, k = 10)
sm_plot_topic_trends(lda_model, dtm, metadata = papers)

## End(Not run)
```

`sm_preprocess_text` *Preprocess Text for Topic Modeling*

Description

Tokenizes, cleans, and stems text data in preparation for topic modeling. Removes stopwords, numbers, and performs stemming using the Porter algorithm.

Usage

```
sm_preprocess_text(
  data,
  text_col = "abstract",
  id_col = NULL,
  min_word_length = 3,
  custom_stopwords = NULL
)
```

Arguments

`data`	A data.frame containing text data.
`text_col`	Name of the column containing text to preprocess. Default is "abstract".
`id_col`	Name of the column containing document IDs. If NULL, a doc_id column will be created. Default is NULL.
`min_word_length`	Minimum word length to retain. Default is 3.
`custom_stopwords`	Additional stopwords to remove beyond the standard English stopwords. Default is NULL.

Value

A data.frame with columns: doc_id, stem, and n (word count).

Examples

```
## Not run:
# Requires API data from sm_search_scopus()
papers <- sm_search_scopus(query, max_count = 50)
processed <- sm_preprocess_text(papers)

## End(Not run)
```

`sm_search_scopus` *Search Scopus Database*

Description

Retrieves abstracts and metadata from the Scopus database based on a structured query. Handles pagination automatically and provides progress feedback.

Usage

```
sm_search_scopus(
  query,
  max_count = 200,
  batch_size = 100,
  view = "COMPLETE",
  verbose = TRUE
)
```

Arguments

`query`	Character string containing the Scopus search query. Should follow Scopus query syntax (e.g., 'TITLE-ABS-KEY("machine learning")').
`max_count`	Maximum number of papers to retrieve. Use Inf to retrieve all available papers. Default is 200.
`batch_size`	Number of records to retrieve per API call. Maximum is 100. Default is 100.
`view`	Level of detail in the response. Options are "STANDARD" or "COMPLETE". Default is "COMPLETE".
`verbose`	Logical indicating whether to print progress messages. Default is TRUE.

Value

A data.frame containing the retrieved papers with columns including title, abstract, author_keywords, year, DOI, and EID.

Examples

```
## Not run:
# Requires Scopus API key
sm_set_api_key()
query <- 'TITLE-ABS-KEY("sport science" AND "machine learning")'
papers <- sm_search_scopus(query, max_count = 50)

## End(Not run)
```

`sm_select_optimal_k` *Select Optimal Number of Topics*

Description

Tests multiple values of k (number of topics) and calculates topic coherence for each. Returns the optimal k based on maximum coherence score, along with a comparison plot.

Usage

```
sm_select_optimal_k(
  dtm,
  k_range = seq(2, 20, by = 2),
  method = "gibbs",
  seed = 1729,
  iter = 500,
  burnin = 100,
  plot = TRUE
)
```

Arguments

`dtm`	A DocumentTermMatrix object.
`k_range`	Vector of k values to test. Default is seq(2, 20, by = 2).
`method`	Topic modeling method. Options: "gibbs" or "vem". Default is "gibbs".
`seed`	Random seed for reproducibility. Default is 1729.
`iter`	Number of Gibbs iterations (if method = "gibbs"). Default is 500.
`burnin`	Number of burn-in iterations (if method = "gibbs"). Default is 100.
`plot`	Logical indicating whether to display the coherence plot. Default is TRUE.

Value

A list containing:

`optimal_k`	The k value with the highest coherence score
`results`	Data frame with k and coherence for each tested value
`plot`	A ggplot object showing coherence vs k

Examples

```
## Not run:
# Requires document-term matrix from sm_create_dtm()
dtm <- sm_create_dtm(processed_data)
k_selection <- sm_select_optimal_k(dtm, k_range = c(5, 10, 15, 20))
print(k_selection$optimal_k)

## End(Not run)
```

`sm_set_api_key` *Set Scopus API Key*

Description

Configures the Scopus API key for the current R session. The key can be provided directly or set via the SCOPUS_API_KEY environment variable.

Usage

```
sm_set_api_key(api_key = NULL)
```

Arguments

`api_key`	Character string containing your Scopus API key. If NULL, the function will attempt to read from the SCOPUS_API_KEY environment variable.

Value

Invisible NULL. Called for side effects.

Examples

```
## Not run:
# Requires Scopus API key
sm_set_api_key("your_api_key_here")

## End(Not run)
```

`sm_train_lda` *Train LDA Topic Model*

Description

Fits a Latent Dirichlet Allocation (LDA) model to a document-term matrix.

Usage

```
sm_train_lda(
  dtm,
  k = NULL,
  method = "gibbs",
  seed = 1729,
  iter = 500,
  burnin = 100,
  alpha = NULL,
  beta = 0.1
)
```

Arguments

`dtm`	A DocumentTermMatrix object.
`k`	Number of topics. If NULL, will attempt to use sm_select_optimal_k first. Default is NULL.
`method`	Method for fitting. Options: "gibbs" or "vem". Default is "gibbs".
`seed`	Random seed for reproducibility. Default is 1729.
`iter`	Number of Gibbs iterations (if method = "gibbs"). Default is 500.
`burnin`	Number of burn-in iterations (if method = "gibbs"). Default is 100.
`alpha`	Hyperparameter for document-topic distributions. Default is 50/k (following Griffiths & Steyvers 2004).
`beta`	Hyperparameter for topic-word distributions. Default is 0.1.

Value

An LDA_Gibbs or LDA_VEM object from the topicmodels package.

Examples

```
## Not run:
# Requires document-term matrix from sm_create_dtm()
dtm <- sm_create_dtm(processed_data)
lda_model <- sm_train_lda(dtm, k = 10)

## End(Not run)
```

theme_sportminer *SportMiner Custom ggplot2 Theme*

Description

A clean, professional, and colorblind-friendly ggplot2 theme designed for academic publications and presentations in sport science.

Usage

```
theme_sportminer(base_size = 11, base_family = "", grid = TRUE)
```

Arguments

base_size	Base font size in points. Default is 11.
base_family	Base font family. Default is "".
grid	Logical indicating whether to display grid lines. Default is TRUE.

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_sportminer()
```

Index