Package: bibnets (via r-universe)

May 15, 2026

Title Importing, Constructing, and Exporting Bibliometric Networks

Version 0.4.3

Description Imports, constructs, and exports bibliometric networks from scholarly metadata. Reads 'Scopus', 'Web of Science', 'BibTeX', 'RIS', 'OpenAlex', 'Lens.org', 'Dimensions', and 'Crossref' exports. Goes beyond standard co-networks with attention-weighted networks (lead, last, proximity, circular position weights), position-aware counting (harmonic, arithmetic, geometric, golden-ratio), similarity and dissimilarity normalisations, temporal networks with fixed, sliding, and cumulative windows, disparity-filter backbone extraction, historiograph construction, and local citation scoring. Methods described in López-Pernas, Saqr & Apiola (2023) <doi:10.1007/978-3-031-25336-2_5>.

License MIT + file LICENSE

URL https://github.com/mohsaqr/bibnets

BugReports https://github.com/mohsaqr/bibnets/issues

Depends R (>= 4.1.0)

Imports Matrix, stats, utils

Suggests bibliometrix, biblionetwork, cograph, data.table, igraph, knitr, openalexR, rcrossref, rmarkdown, testthat (>= 3.0.0), tidygraph

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

Date/Publication 2026-05-15 14:06:04 UTC

RemoteUrl https://github.com/mohsaqr/bibnets

RemoteRef HEAD

RemoteSha 2c2b2a50df25eb3f4bcecc2812824e76a588b2c4

Contents

author_network *Build an author network*

Description

Constructs a network between authors using one of four relationship types and any of 13 counting methods, including 9 position-dependent methods that respect author byline order.

Usage

```
author_network(
  data,
  type = "collaboration",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  position_weights = c(1, 0.8, 0.6, 0.4),
  first_last_weight = 2,
  attention = NULL,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

data	A data frame with at least `id` and `authors` (list-column, order preserved). For coupling/co-citation, also needs `references`.
type	Character. Relationship type:
	`"collaboration"` Co-authorship: authors linked when they co-author a publication.
	`"coupling"` Bibliographic coupling aggregated at author level: authors linked when they cite the same references.
	`"co_citation"` Author co-citation: authors linked when they are cited together by the same paper. Requires a `cited_first_authors` list-column.
	`"equivalence"` Profile similarity: cosine similarity of authors' full collaboration/citation profiles.
counting	Character. Counting method. Position-independent methods (`"full"`, `"fractional"`, `"paper"`, `"strength"`) work for all types. Position-dependent methods (`"harmonic"`, `"arithmetic"`, `"geometric"`, `"adaptive_geometric"`, `"golden"`, `"first"`, `"last"`, `"first_last"`, `"position_weighted"`) are available for `type = "collaboration"`.
similarity	Character. Similarity measure: `"none"`, `"association"`, `"cosine"`, `"jaccard"`, `"inclusion"`, `"equivalence"`.

`threshold` Numeric. Minimum edge weight. Default 0.

`min_occur` Integer. Minimum number of papers for an author to be included. Default 1.

`position_weights`
Numeric vector. Custom weights for `counting = "position_weighted"`. Default `c(1, 0.8, 0.6, 0.4)`.

`first_last_weight`
Numeric. Multiplier for `counting = "first_last"`. Default 2.

`attention` Character or NULL. Attention-based weighting independent of `type` and `counting`. One of `"proximity"` (center authors weighted most), `"lead"` (first author dominates, quadratic drop), `"last"` (last author dominates, quadratic rise), `"circular"` (first and last both prominent). Default `NULL` (disabled).

`top_n` Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).

`self_loops` Logical. If `TRUE`, include self-loops (an entity linked to itself). Default `FALSE`.

`deduplicate` Logical. If `TRUE` (default), each `(paper, entity)` pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to `FALSE` to count every raw occurrence.

`format` Character. Output format:

`"edgelist"` Default. A `bibnets_network` data frame with columns `from`, `to`, `weight`, `count`.

`"gephi"` Gephi-ready data frame: `Source`, `Target`, `Weight`, `Count`, `Type`.

`"igraph"` An igraph graph object (requires igraph).

`"cograph"` A cograph_network object (requires cograph).

`"matrix"` A sparse adjacency matrix.

Value

Depends on `format`: a `bibnets_network` data frame (default), a Gephi-ready data frame, an igraph graph, a cograph_network, or a sparse matrix.

Examples

```
data(biblio_data)
author_network(biblio_data, "collaboration")
author_network(biblio_data, "collaboration", counting = "harmonic")
author_network(biblio_data, "collaboration", counting = "geometric",
               similarity = "association")
```

backbone *Extract network backbone using the disparity filter*

Description

Applies the disparity filter to a weighted edge list. For each edge, it computes an alpha (p-value) from both endpoints and keeps the edge if it is statistically significant from at least one endpoint.

Usage

```
backbone(edges, alpha = 0.05)
```

Arguments

`edges`	A data frame with at least columns `from`, `to`, and `weight`. Must be an undirected edge list (each pair appears once).
`alpha`	Numeric. Significance threshold in (0, 1). Default `0.05`.

Details

The null model asks: given that node i has total strength s_i distributed uniformly across k_i edges, what is the probability that a single edge weight is as large as w_{ij}? The answer is

$$\alpha_{ij} = \left(1 - \frac{w_{ij}}{s_i}\right)^{k_i - 1}$$

An edge is retained if $\min(\alpha_{ij}, \alpha_{ji}) < \alpha$. Nodes with only one edge always have $\alpha = 0$ and are always kept.

Value

The filtered edge data frame with an added `alpha` column (the minimum alpha from the two endpoints).

Examples

```
edges <- data.frame(
  from   = c("A", "A", "A", "B", "C"),
  to     = c("B", "C", "D", "C", "D"),
  weight = c(10,   1,   1,   8,   1)
)
backbone(edges, alpha = 0.05)
```

`biblio_data` *Example bibliometric dataset*

Description

A small synthetic dataset of 10 scholarly papers with overlapping authors, references, and keywords. Designed for testing and demonstrating all network construction functions in bibnets.

Usage

```
biblio_data
```

Format

A data frame with 10 rows and 9 columns:

id Unique document identifier (W1–W10).

title Document title.

year Publication year (2018–2022).

journal Source journal (Scientometrics, Journal of Informetrics, JASIST, Quantitative Science Studies).

doi DOI string.

cited_by_count Times cited.

authors List-column of author name strings (6 unique authors).

references List-column of cited reference IDs (10 unique refs, R1–R10). Each paper cites exactly 4 references.

keywords List-column of keyword strings (24 unique keywords). Each paper has 3 keywords.

Examples

```
data(biblio_data)
reference_network(biblio_data)
document_network(biblio_data, "coupling")
author_network(biblio_data, "collaboration")
```

conetwork *Build a co-occurrence network from any field*

Description

With one field, entities are linked when they co-occur in the same document. With by, entities are linked when they share values of the by field across documents.

Usage

```
conetwork(
  data,
  field,
  by = NULL,
  sep = ";",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

data	A data frame with column id and the specified field(s).
field	Character. The entity field — determines what the nodes are.
by	Character or NULL. What links the nodes. If NULL (default), entities are linked by co-occurring in the same document. If specified, entities are linked when they share values from the by field.
sep	Character or NULL. Delimiter for splitting character columns. Default ";". Set to NULL if columns are already list-columns.
counting	Character. Counting method. Default "full".
similarity	Character. Normalization method. Default "none".
threshold	Numeric. Minimum edge weight. Default 0.
min_occur	Integer. Minimum entity frequency. Default 1.
top_n	Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).
self_loops	Logical. If TRUE, include self-loops (an entity linked to itself). Default FALSE.
deduplicate	Logical. If TRUE (default), each (paper, entity) pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to FALSE to count every raw occurrence.

format — Character. Output format:

- "edgelist" Default. A bibnets_network data frame with columns from, to, weight, count.
- "gephi" Gephi-ready data frame: Source, Target, Weight, Count, Type.
- "igraph" An igraph graph object (requires igraph).
- "cograph" A cograph_network object (requires cograph).
- "matrix" A sparse adjacency matrix.

Details

Fields can be list-columns (already split) or character columns with delimiters (auto-split via sep).

Value

Depends on format: a bibnets_network data frame (default), a Gephi-ready data frame, an igraph graph, a cograph_network, or a sparse matrix.

Examples

```
data(biblio_data)

# Co-occurrence: keywords appearing in the same document
conetwork(biblio_data, "keywords")

# Authors linked by shared keywords
conetwork(biblio_data, "authors", by = "keywords")

# Keywords linked by shared authors
conetwork(biblio_data, "keywords", by = "authors")

# Journals linked by shared references (= journal coupling)
conetwork(biblio_data, "journal", by = "references", similarity = "cosine")

# Auto-splits semicolon-delimited string columns
d <- data.frame(id = 1:3, tags = c("ml; dl; nlp", "ml; cv", "dl; cv"))
conetwork(d, "tags")
```

---

country_network *Build a country network*

Description

Constructs a network between countries based on collaboration or coupling.

Usage

```
country_network(
  data,
  type = "collaboration",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  attention = NULL,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

data	A data frame with id and countries (list-column of country names). For coupling, also needs references.
type	Character. "collaboration" (default), "coupling", or "equivalence".
counting	Character. Counting method. Default "full".
similarity	Character. Similarity measure. Default "none".
threshold	Numeric. Minimum edge weight. Default 0.
min_occur	Integer. Minimum papers per country. Default 1.
attention	Character or NULL. Attention-based weighting independent of type and counting. One of "proximity" (center authors weighted most), "lead" (first author dominates, quadratic drop), "last" (last author dominates, quadratic rise), "circular" (first and last both prominent). Default NULL (disabled).
top_n	Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).
self_loops	Logical. If TRUE, include self-loops (an entity linked to itself). Default FALSE.
deduplicate	Logical. If TRUE (default), each (paper, entity) pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to FALSE to count every raw occurrence.
format	Character. Output format: "edgelist" Default. A bibnets_network data frame with columns from, to, weight, count. "gephi" Gephi-ready data frame: Source, Target, Weight, Count, Type. "igraph" An igraph graph object (requires igraph). "cograph" A cograph_network object (requires cograph). "matrix" A sparse adjacency matrix.

Value

Depends on format: a bibnets_network data frame (default), a Gephi-ready data frame, an igraph graph, a cograph_network, or a sparse matrix.

Examples

```
data(open_alex_gold_open_access_learning_analytics)
country_network(open_alex_gold_open_access_learning_analytics, "collaboration")
```

`document_network` *Build a document network*

Description

Constructs a network between documents (papers) in the dataset.

Usage

```
document_network(
  data,
  type = "coupling",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

`data`	A data frame with `id` and `references` (list-column).
`type`	Character. Relationship type:
	`"coupling"` Bibliographic coupling: documents linked when they share cited references.
	`"citation"` Direct citation: directed edges from citing to cited documents (internal citations only).
	`"co_citation"` Co-citation: documents linked when they are cited together by other documents in the dataset.
	`"equivalence"` Profile similarity of reference vectors.
`counting`	Character. Counting method. Default `"full"`. Position-dependent methods are not applicable to document networks.
`similarity`	Character. Similarity measure. Default `"none"`.
`threshold`	Numeric. Minimum edge weight. Default 0.
`min_occur`	Integer. Minimum reference frequency. Default 1.
`top_n`	Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).

self_loops	Logical. If TRUE, include self-loops (an entity linked to itself). Default FALSE.
deduplicate	Logical. If TRUE (default), each (paper, entity) pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to FALSE to count every raw occurrence.
format	Character. Output format:
	"edgelist" Default. A bibnets_network data frame with columns from, to, weight, count.
	"gephi" Gephi-ready data frame: Source, Target, Weight, Count, Type.
	"igraph" An igraph graph object (requires igraph).
	"cograph" A cograph_network object (requires cograph).
	"matrix" A sparse adjacency matrix.

Value

Depends on format. For type = "citation", edges are directed (from = citing, to = cited) with weight and count both 1.

Examples

```
data(biblio_data)
document_network(biblio_data, "coupling")
document_network(biblio_data, "coupling", counting = "strength")
```

---

filter_top *Filter edges to top-n nodes*

Description

Keeps only edges between the most frequent nodes. Node frequency is determined by how many edges each node participates in.

Usage

```
filter_top(edges, n)
```

Arguments

edges	A data frame with at least from, to, weight columns.
n	Integer. Number of top nodes to keep.

Value

A filtered data frame with edges among the top n nodes.

Examples

```
data(biblio_data)
edges <- author_network(biblio_data, "collaboration")

# Keep only edges among the top 3 most connected authors
filter_top(edges, 3)
```

`historiograph` *Build a historiograph (chronological citation network)*

Description

Constructs a Garfield-style historiograph: a directed citation network among the most locally cited documents, laid out chronologically.

Usage

```
historiograph(data, n = 30, min_lcs = 1)
```

Arguments

`data`	A data frame with `id`, `references` (list-column), and `year`. Optionally `title`, `journal`, `doi`, `cited_by_count`.
`n`	Integer. Number of top locally cited documents to include. Default 30.
`min_lcs`	Integer. Minimum local citation score for inclusion. Default 1.

Value

A list with:

`$nodes` Data frame of included documents with `id`, `lcs`, `gcs`, `year`, `title`, `journal`, `doi`.

`$edges` Data frame of directed citation edges with `from` (citing), `to` (cited), `year_from`, `year_to`.

Examples

```
data(biblio_data)
h <- historiograph(biblio_data, n = 5)
h$nodes
h$edges
```

`institution_network` *Build an institution network*

Description

Constructs a network between institutions (affiliations).

Usage

```
institution_network(
  data,
  type = "collaboration",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  attention = NULL,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

`data`	A data frame with `id` and `affiliations` (list-column of institution names). For coupling, also needs `references`.
`type`	Character. `"collaboration"` (default), `"coupling"`, or `"equivalence"`.
`counting`	Character. Counting method. Default `"full"`.
`similarity`	Character. Similarity measure. Default `"none"`.
`threshold`	Numeric. Minimum edge weight. Default 0.
`min_occur`	Integer. Minimum papers per institution. Default 1.
`attention`	Character or NULL. Attention-based weighting independent of `type` and `counting`. One of `"proximity"` (center authors weighted most), `"lead"` (first author dominates, quadratic drop), `"last"` (last author dominates, quadratic rise), `"circular"` (first and last both prominent). Default `NULL` (disabled).
`top_n`	Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).
`self_loops`	Logical. If `TRUE`, include self-loops (an entity linked to itself). Default `FALSE`.
`deduplicate`	Logical. If `TRUE` (default), each `(paper, entity)` pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to `FALSE` to count every raw occurrence.
`format`	Character. Output format:

"edgelist" Default. A bibnets_network data frame with columns from, to, weight, count.

"gephi" Gephi-ready data frame: Source, Target, Weight, Count, Type.

"igraph" An igraph graph object (requires igraph).

"cograph" A cograph_network object (requires cograph).

"matrix" A sparse adjacency matrix.

Value

Depends on format: a bibnets_network data frame (default), a Gephi-ready data frame, an igraph graph, a cograph_network, or a sparse matrix.

Examples

```
data(open_alex_gold_open_access_learning_analytics)
institution_network(open_alex_gold_open_access_learning_analytics, "collaboration")
```

---

keyword_network *Build a keyword co-occurrence network*

Description

Constructs a network where two keywords are linked when they appear together in the same document.

Usage

```
keyword_network(
  data,
  field = "keywords",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  attention = NULL,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

data	A data frame with id and a keyword list-column.
field	Character. Name of the keyword list-column. Default "keywords". Alternatives: "author_keywords", "index_keywords", "keywords_plus".

counting	Character. Counting method. Default "full".
similarity	Character. Similarity measure. Default "none".
threshold	Numeric. Minimum edge weight. Default 0.
min_occur	Integer. Minimum keyword frequency. Default 1.
attention	Character or NULL. Attention-based weighting independent of type and counting. One of "proximity" (center authors weighted most), "lead" (first author dominates, quadratic drop), "last" (last author dominates, quadratic rise), "circular" (first and last both prominent). Default NULL (disabled).
top_n	Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).
self_loops	Logical. If TRUE, include self-loops (an entity linked to itself). Default FALSE.
deduplicate	Logical. If TRUE (default), each (paper, entity) pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to FALSE to count every raw occurrence.
format	Character. Output format: "edgelist" Default. A bibnets_network data frame with columns from, to, weight, count. "gephi" Gephi-ready data frame: Source, Target, Weight, Count, Type. "igraph" An igraph graph object (requires igraph). "cograph" A cograph_network object (requires cograph). "matrix" A sparse adjacency matrix.

Value

Depends on format: a bibnets_network data frame (default), a Gephi-ready data frame, an igraph graph, a cograph_network, or a sparse matrix.

Examples

```
data(biblio_data)
keyword_network(biblio_data)
keyword_network(biblio_data, similarity = "association")
```

---

local_citations *Compute local citation scores*

Description

Counts how many times each document is cited by other documents within the dataset.

Usage

```
local_citations(data)
```

Arguments

`data`	A data frame with `id` and `references` (list-column). Optionally `year`, `title`, `journal`, `doi`, `cited_by_count`.

Value

A data frame with columns:

`id` Document identifier.

`lcs` Local Citation Score: times cited within the dataset.

`gcs` Global Citation Score: `cited_by_count` if available.

Plus any metadata columns present in the input (`title`, `year`, `journal`, `doi`).

Examples

```
data(biblio_data)
local_citations(biblio_data)
```

`normalize` *Normalize a co-occurrence matrix*

Description

Applies a similarity normalization to a square co-occurrence matrix. The diagonal of the input matrix is used as the total occurrence count for each item. Operates entirely in sparse representation.

Usage

```
normalize(A, method = "none")
```

Arguments

`A`	A square symmetric matrix (dense or sparse) representing co-occurrence counts.
`method`	Character. Normalization method:
	`"none"` No normalization. Returns raw co-occurrence counts.
	`"association"` Association strength (probabilistic affinity index). $s_{ij} = c_{ij}/(w_i \cdot w_j)$. Often recommended as the best normalization for co-occurrence data.
	`"cosine"` Salton's cosine. $s_{ij} = c_{ij}/\sqrt{w_i \cdot w_j}$.
	`"jaccard"` Jaccard index. $s_{ij} = c_{ij}/(w_i + w_j - c_{ij})$.
	`"inclusion"` Inclusion index (Simpson coefficient). $s_{ij} = c_{ij}/\min(w_i, w_j)$.
	`"equivalence"` Equivalence index (Salton's cosine squared). $s_{ij} = c_{ij}^2/(w_i \cdot w_j)$.

Value

A normalized sparse matrix of the same dimensions.

Examples

```
# Create a small co-occurrence matrix
A <- matrix(c(10, 3, 1, 3, 8, 2, 1, 2, 5), nrow = 3,
            dimnames = list(c("a", "b", "c"), c("a", "b", "c")))
normalize(A, "association")
normalize(A, "cosine")
normalize(A, "jaccard")
```

---

open_alex_gold_open_access_learning_analytics

OpenAlex Gold Open Access Learning Analytics dataset

Description

A corpus of 1,508 gold open-access scholarly works on learning analytics, retrieved from OpenAlex (CC0 licence). All records have a verified title, publication year, and at least one author. Journal names are present for works published in a named source; preprints and book chapters may have `NA` in `journal`.

Usage

```
open_alex_gold_open_access_learning_analytics
```

Format

A data frame with 1,508 rows and 11 columns:

id OpenAlex work ID (e.g. `"W2769342982"`).

title Work title.

year Publication year (integer).

journal Source name, or `NA` if not available.

doi DOI string without the `https://doi.org/` prefix, or `NA`.

cited_by_count Number of citing works as recorded in OpenAlex.

type Work type (`"article"`, `"review"`, `"preprint"`, `"book-chapter"`, etc.).

authors List-column of author display names (pipe-split from the OpenAlex flat export; one name per authorship slot).

keywords List-column with one element: the primary OpenAlex topic for the work (e.g. `"Online Learning and Analytics"`).

affiliations List-column of institution display names (one entry per authorship–institution pair).

countries List-column of two-letter ISO country codes (one entry per authorship–institution pair).

Source

OpenAlex https://openalex.org, CC0 licence.

Examples

```
data(open_alex_gold_open_access_learning_analytics)
d <- open_alex_gold_open_access_learning_analytics
author_network(d, "collaboration")
country_network(d, "collaboration")
```

`print.bibnets_network` *Print a bibnets network edge list*

Description

Print a bibnets network edge list

Usage

```
## S3 method for class 'bibnets_network'
print(x, n = 10L, ...)
```

Arguments

`x`	A `bibnets_network` data frame.
`n`	Integer. Number of rows to show. Default 10.
`...`	Ignored.

Value

Invisibly returns `x`.

Examples

```
data(biblio_data)
edges <- author_network(biblio_data, "collaboration")
print(edges)
```

`prune` *Prune a weighted edge list*

Description

Reduces a weighted edge list by removing weak or excess edges.

Usage

```
prune(edges, threshold = NULL, top_n = NULL)
```

Arguments

edges	A data frame with at least columns from, to, and weight.
threshold	Numeric. Keep only edges with weight >= threshold.
top_n	Integer. For each node, keep only its top_n strongest edges. An edge is kept if it is in the top top_n for *either* endpoint.

Value

The filtered edge data frame (same columns as input).

Examples

```
edges <- data.frame(
  from   = c("A","A","A","B","B","C"),
  to     = c("B","C","D","C","D","D"),
  weight = c(5,  1,  2,  4,  1,  3)
)

# Keep only edges with weight >= 3
prune(edges, threshold = 3)

# Keep the 2 strongest edges per node
prune(edges, top_n = 2)
```

---

read_biblio *Read bibliometric data*

Description

Universal reader that handles files, folders, format detection, and generic CSV input. Accepts a single file, multiple files, or a directory.

Usage

```
read_biblio(path, format = "auto", id = NULL, actors = NULL, sep = ";", ...)
```

Arguments

path	Character. Path to a file, a vector of file paths, or a directory containing export files.
format	Character. File format: "auto" Default. Auto-detect from file content. "scopus" Scopus CSV. "wos" Web of Science plaintext. "wos_tab" Web of Science tab-delimited. "bibtex" BibTeX .bib file.

	`"ris"` RIS file. `"dimensions"` Dimensions CSV. `"lens"` Lens.org CSV. `"openalex_csv"` Flat OpenAlex CSV export (pipe-delimited fields). `"generic"` Any CSV. Use `id` and `actors` to specify columns.
`id`	Character. Column name for document identifier. Only used when `format = "generic"`. Default `NULL` (uses row numbers).
`actors`	Character vector. Column names to split into list-columns. Only used when `format = "generic"`.
`sep`	Character. Delimiter for splitting actor columns. Default `";"`.
`...`	Additional arguments passed to the format-specific reader.

Value

A data frame.

Examples

```
# Auto-detect format from file content (here: a bundled OpenAlex CSV)
f <- system.file("extdata", "openalex_works.csv", package = "bibnets")
data <- read_biblio(f)
head(data[, c("id", "title", "year", "journal")])

# Read multiple files at once; auto-detects each format
f_scopus <- system.file("extdata", "scopus_sample.csv", package = "bibnets")
f_wos    <- system.file("extdata", "wos_sample.txt",  package = "bibnets")
combined <- read_biblio(c(f_scopus, f_wos))
head(combined[, c("id", "title", "year", "journal")])

# Read every supported export in a directory (here: the bundled extdata)
folder <- system.file("extdata", package = "bibnets")
all_data <- read_biblio(folder)
nrow(all_data)

# Generic CSV: point read_biblio at any CSV and name the list-column fields
tmp <- tempfile(fileext = ".csv")
write.csv(data.frame(
  doc_id  = c("a", "b"),
  Authors = c("Smith J; Jones A", "Davis M"),
  Keywords = c("networks; bibliometrics", "analytics")
), tmp, row.names = FALSE)
generic <- read_biblio(tmp, format = "generic",
                       id = "doc_id",
                       actors = c("Authors", "Keywords"),
                       sep = ";")
head(generic)
```

read_bibtex *Read a BibTeX file*

Description

Parses a .bib file into a standardized bibliometric data frame. Note: standard BibTeX does not contain cited references, so the references column will be empty unless the file includes a non-standard cited-references or note field with reference data.

Usage

```
read_bibtex(file, encoding = "UTF-8")
```

Arguments

file	Path to a .bib file.
encoding	Character. File encoding. Default "UTF-8".

Value

A data frame in the standard bibnets format: id, title, year, journal, doi, cited_by_count, abstract, type, plus list-columns authors, references (typically empty for BibTeX), and keywords.

Examples

```
# Write a minimal BibTeX entry to a temp file, then read it
bib <- '@article{smith2020,
  title  = {Bibliometric networks},
  author = {Smith, J. and Jones, K.},
  journal = {Test Journal},
  year   = {2020},
  doi    = {10.1000/test}
}'
f <- tempfile(fileext = ".bib")
writeLines(bib, f)
data <- read_bibtex(f)
data[, c("id", "title", "year", "journal", "doi")]
unlink(f)
```

read_crossref *Convert Crossref API data to bibnets format*

Description

Takes the output of `rcrossref::cr_works()` (the `$data` tibble/data frame) and converts it to the standardized bibnets format.

Usage

```
read_crossref(data)
```

Arguments

`data`	A data frame from `cr_works(...)$data`.

Value

A data frame in the standard bibnets format: `id`, `title`, `year`, `journal`, `doi`, `cited_by_count`, `abstract`, `type`, plus list-columns `authors`, `references`, and `keywords`.

Examples

```
# Construct a minimal data frame matching the structure of
# rcrossref::cr_works(...)$data. In practice, pass that data frame directly.
raw <- data.frame(
  doi = c("10.1/a", "10.2/b"),
  title = c("First paper", "Second paper"),
  issued = c("2022-01-01", "2021-06-15"),
  container.title = c("Journal A", "Journal B"),
  is.referenced.by.count = c("3", "9"),
  type = c("journal-article", "journal-article"),
  stringsAsFactors = FALSE
)
raw$author <- list(
  data.frame(given = c("Jane", "Anne"),
             family = c("Smith", "Jones"),
             stringsAsFactors = FALSE),
  data.frame(given = "Mark", family = "Davis", stringsAsFactors = FALSE)
)
data <- read_crossref(raw)
head(data[, c("id", "title", "year", "journal")])
```

read_dimensions *Read Dimensions CSV export*

Description

Parses a CSV file exported from Dimensions into a standardized bibliometric data frame.

Usage

```
read_dimensions(file, encoding = "UTF-8")
```

Arguments

file	Path to a Dimensions CSV export file.
encoding	Character. File encoding. Default "UTF-8".

Value

A data frame in the standard bibnets format: id, title, year, journal, doi, cited_by_count, abstract, type, plus list-columns authors, references, and keywords. Dimensions-specific extras: affiliations (list-column), countries (list-column).

Examples

```
f <- system.file("extdata", "dimensions_sample.csv", package = "bibnets")
data <- read_dimensions(f)
head(data[, c("id", "title", "year", "journal")])
```

read_lens *Read Lens.org CSV export*

Description

Parses a CSV file exported from Lens.org into a standardized bibliometric data frame.

Usage

```
read_lens(file, encoding = "UTF-8")
```

Arguments

file	Path to a Lens.org CSV export file.
encoding	Character. File encoding. Default "UTF-8".

Value

A data frame in the standard bibnets format: `id`, `title`, `year`, `journal`, `doi`, `cited_by_count`, `abstract`, `type`, plus list-columns `authors`, `references`, and `keywords`.

Examples

```
f <- system.file("extdata", "lens_sample.csv", package = "bibnets")
data <- read_lens(f)
head(data[, c("id", "title", "year", "journal")])
```

`read_openalex` *Convert OpenAlex data to bibnets format*

Description

Takes the output of `openalexR::oa_fetch()` (a tibble/data frame of works) and converts it to the standardized bibnets format with list-columns.

Usage

```
read_openalex(data)
```

Arguments

`data`	A data frame from `oa_fetch(entity = "works", ...)`. Must contain at least an `id` column. Common columns include `display_name`, `publication_year`, `so`, `doi`, `cited_by_count`, `referenced_works`, `ab`, and `author` (nested).

Value

A data frame in the standard bibnets format: `id`, `title`, `year`, `journal`, `doi`, `cited_by_count`, `abstract`, `type`, plus list-columns `authors`, `references`, and `keywords`.

Examples

```
# Construct a minimal data frame matching the structure returned by
# openalexR::oa_fetch(entity = "works", ...). In practice, pass the
# result of oa_fetch() directly.
raw <- data.frame(
  id = c("W123", "W456"),
  display_name = c("First paper", "Second paper"),
  publication_year = c(2022L, 2021L),
  so = c("Journal A", "Journal B"),
  doi = c("https://doi.org/10.1/a", "https://doi.org/10.2/b"),
  cited_by_count = c(5L, 12L),
  stringsAsFactors = FALSE
)
raw$author <- list(
  data.frame(au_display_name = c("Smith J", "Jones A"),
```

```
            stringsAsFactors = FALSE),
  data.frame(au_display_name = "Davis M", stringsAsFactors = FALSE)
)
raw$referenced_works <- list(c("W100", "W200"), "W123")
data <- read_openalex(raw)
head(data[, c("id", "title", "year", "journal", "doi")])
```

`read_openalex_csv` *Read a flat OpenAlex CSV export*

Description

Reads the flat CSV format downloaded directly from the OpenAlex website (`openalex.org/works` exports). Multi-value fields are pipe-delimited (`|`). This is distinct from the nested tibble produced by `openalexR::oa_fetch()`, which is handled by `read_openalex()`.

Usage

```
read_openalex_csv(file, sep = "|")
```

Arguments

`file`	Path to the CSV file.
`sep`	Character. Delimiter for multi-value fields. Default `"|"`.

Value

A data frame in the standard bibnets format: `id`, `title`, `year`, `journal`, `doi`, `cited_by_count`, `abstract`, `type`, plus list-columns `authors`, `references`, `keywords`, `affiliations`, `countries`. `abstract` and `references` are always `NA` / empty (not available in the flat export).

Examples

```
f <- system.file("extdata", "openalex_works.csv", package = "bibnets")
data <- read_openalex_csv(f)
```

`read_ris` *Read an RIS file*

Description

Parses a `.ris` file into a standardized bibliometric data frame. Like BibTeX, standard RIS does not include cited references.

Usage

```
read_ris(file, encoding = "UTF-8")
```

Arguments

file	Path to a .ris file.
encoding	Character. File encoding. Default "UTF-8".

Value

A data frame in the standard bibnets format: id, title, year, journal, doi, cited_by_count, abstract, type, plus list-columns authors, references (typically empty for RIS), and keywords.

Examples

```
# Write a minimal RIS record to a temp file, then read it
ris <- "TY  - JOUR
AU  - Smith, J.
AU  - Jones, K.
TI  - Bibliometric networks
JO  - Test Journal
PY  - 2020
DO  - 10.1000/test
ER  - "
f <- tempfile(fileext = ".ris")
writeLines(ris, f)
data <- read_ris(f)
data[, c("id", "title", "year", "journal", "doi")]
unlink(f)
```

---

read_scopus *Read Scopus CSV export*

Description

Parses a CSV file exported from Scopus into a standardized bibliometric data frame with list-columns for multi-valued fields.

Usage

```
read_scopus(file, encoding = "UTF-8")
```

Arguments

file	Path to a Scopus CSV export file.
encoding	Character. File encoding. Default "UTF-8".

Value

A data frame in the standard bibnets format: id, title, year, journal, doi, cited_by_count, abstract, type, plus list-columns authors, references, and keywords. Scopus-specific extras: index_keywords (list-column), affiliations (character), language (character).

Examples

```
f <- system.file("extdata", "scopus_sample.csv", package = "bibnets")
data <- read_scopus(f)
head(data[, c("id", "title", "year", "journal")])
```

---

read_wos *Read Web of Science plaintext or tab-delimited export*

Description

Parses a Web of Science export file (plaintext or tab-delimited) into a standardized bibliometric data frame.

Usage

```
read_wos(file, format = "plaintext")
```

Arguments

file	Path to a WoS export file (.txt).
format	Character. "plaintext" (default) for WoS tagged format, or "tab" for tab-delimited export.

Value

A data frame in the standard bibnets format: id, title, year, journal, doi, cited_by_count, abstract, type, plus list-columns authors, references, and keywords. WoS-specific extra: keywords_plus (list-column).

Examples

```
f <- system.file("extdata", "wos_sample.txt", package = "bibnets")
data <- read_wos(f)
head(data[, c("id", "title", "year", "journal")])
```

---

reference_network *Build a reference network*

Description

Constructs a co-citation or equivalence network among cited references. Two references are linked when they are cited together by the same paper.

Usage

```
reference_network(
  data,
  type = "co_citation",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

data	A data frame with id and references (list-column).
type	Character. "co_citation" (default) or "equivalence".
counting	Character. Counting method. Default "full".
similarity	Character. Similarity measure. Default "none".
threshold	Numeric. Minimum edge weight. Default 0.
min_occur	Integer. Minimum times a reference must be cited. Default 1.
top_n	Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).
self_loops	Logical. If TRUE, include self-loops (an entity linked to itself). Default FALSE.
deduplicate	Logical. If TRUE (default), each (paper, entity) pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to FALSE to count every raw occurrence.
format	Character. Output format: "edgelist" Default. A bibnets_network data frame with columns from, to, weight, count. "gephi" Gephi-ready data frame: Source, Target, Weight, Count, Type. "igraph" An igraph graph object (requires igraph). "cograph" A cograph_network object (requires cograph). "matrix" A sparse adjacency matrix.

Value

Depends on format: a bibnets_network data frame (default), a Gephi-ready data frame, an igraph graph, a cograph_network, or a sparse matrix.

Examples

```
data(biblio_data)
reference_network(biblio_data)
reference_network(biblio_data, similarity = "association")
```

scopus_quantum_cloud *Scopus dataset — Green Cloud Computing and Quantization (2020–2025)*

Description

First 500 records from a Scopus bibliometric export on the intersection of green cloud computing and quantization, covering 2020–2025. Includes full references, author keywords, index keywords, and affiliations.

Usage

```
scopus_quantum_cloud
```

Format

A data frame with 499 rows and 12 columns:

id Scopus EID.

title Work title.

year Publication year (integer).

journal Source title.

doi DOI string without the `https://doi.org/` prefix.

cited_by_count Times cited in Scopus.

abstract Abstract text.

type Document type (`"Article"`, `"Review"`, etc.).

authors List-column of author name strings.

references List-column of cited reference strings.

keywords List-column of author keywords.

affiliations List-column of affiliation strings.

Source

Scopus bibliometric export. Dataset archived at doi:10.5281/zenodo.17142636 (CC BY 4.0).

Examples

```
data(scopus_quantum_cloud)
author_network(scopus_quantum_cloud, "collaboration")
keyword_network(scopus_quantum_cloud)
document_network(scopus_quantum_cloud, "coupling", similarity = "cosine")
```

source_network *Build a source (journal) network*

Description

Constructs a network between publication sources (journals, book series).

Usage

```
source_network(
  data,
  type = "coupling",
  counting = "full",
  similarity = "none",
  threshold = 0,
  min_occur = 1L,
  top_n = NULL,
  self_loops = FALSE,
  deduplicate = TRUE,
  format = "edgelist"
)
```

Arguments

data	A data frame with id and journal (character column). For coupling, also needs references. For co-citation, needs a cited_journals list-column.
type	Character. "coupling" (default), "co_citation", or "equivalence".
counting	Character. Counting method. Default "full".
similarity	Character. Similarity measure. Default "none".
threshold	Numeric. Minimum edge weight. Default 0.
min_occur	Integer. Minimum papers per source. Default 1.
top_n	Integer or NULL. Return only the top n edges by weight. Default NULL (all edges).
self_loops	Logical. If TRUE, include self-loops (an entity linked to itself). Default FALSE.
deduplicate	Logical. If TRUE (default), each (paper, entity) pair is counted at most once — duplicate entries in the source data (e.g., the same author listed twice on a paper) are treated as one occurrence. Set to FALSE to count every raw occurrence.
format	Character. Output format: "edgelist" Default. A bibnets_network data frame with columns from, to, weight, count. "gephi" Gephi-ready data frame: Source, Target, Weight, Count, Type. "igraph" An igraph graph object (requires igraph). "cograph" A cograph_network object (requires cograph). "matrix" A sparse adjacency matrix.

Value

Depends on `format`: a `bibnets_network` data frame (default), a Gephi-ready data frame, an igraph graph, a cograph_network, or a sparse matrix.

Examples

```
data(biblio_data)
source_network(biblio_data, "coupling")
```

`split_field` *Parse semicolon-delimited strings into list-column*

Description

Splits semicolon-separated strings (common in Scopus/WoS exports) into character vectors, trimming whitespace.

Usage

```
split_field(x, sep = ";")
```

Arguments

`x`	A character vector of semicolon-delimited strings.
`sep`	Character. Delimiter. Default `";"`.

Value

A list of character vectors.

Examples

```
split_field(c("Alice; Bob; Carol", "Dave; Eve"))
```

summary.bibnets_network *Summarise a bibnets network*

Description

Summarise a bibnets network

Usage

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

Arguments

object	A bibnets_network data frame.
...	Ignored.

Value

Invisibly returns object.

Examples

```
data(biblio_data)
edges <- author_network(biblio_data, "collaboration")
summary(edges)
```

temporal_network *Build time-windowed networks*

Description

Splits data by time windows and builds a separate network for each window using any network function.

Usage

```
temporal_network(
  data,
  network_fun,
  ...,
  window = 3,
  step = NULL,
  strategy = "fixed",
  time_col = "year"
)
```

Arguments

data	A data frame with a numeric time column.
network_fun	Function or character string naming a network function (e.g., author_network, "reference_network", conetwork).
...	Additional arguments passed to network_fun (e.g., type, counting, similarity, threshold, top_n).
window	Integer. Width of each time window in units of the time column (years, months, quarters, etc.). Default 3.
step	Integer or NULL. Step size between windows. Default NULL (equals window for fixed, 1 for sliding).
strategy	Character. Time window strategy: "fixed" Disjoint non-overlapping windows (default). "sliding" Overlapping windows advancing by step units. "cumulative" Each window starts at the earliest value and extends further.
time_col	Character. Name of the column containing the time variable. Default "year". Works with any numeric time unit: years, months, quarters, semesters, weeks, etc. (e.g., "month", "quarter", "time").

Value

A named list of data frames (edge lists). Names are window labels like "2018-2020".

Examples

```
data(biblio_data)

# Fixed 3-year windows
temporal_network(biblio_data, author_network, "collaboration")

# Sliding window
temporal_network(biblio_data, author_network, "collaboration",
                 window = 2, strategy = "sliding")

# Cumulative
temporal_network(biblio_data, reference_network,
                 threshold = 0, strategy = "cumulative", window = 2)

# With string name
temporal_network(biblio_data, "keyword_network", window = 3)
```

`to_cograph` *Prepare network for cograph::splot()*

Description

Converts a bibnets edge list to a `cograph_network` object by calling `cograph::as_cograph()`. Optionally merges node metadata (e.g., from `local_citations()`) into the network's node table so attributes like `lcs` or `year` can be used directly in `splot()` aesthetic parameters (e.g., `node_size = "lcs"`).

Usage

```
to_cograph(edges, nodes = NULL, directed = FALSE)
```

Arguments

`edges`	A data frame with at least `from`, `to`, `weight` columns.
`nodes`	Optional data frame of node attributes with an `id` column (e.g., output of `local_citations()`). All columns are merged into the `cograph_network$nodes` table and become available as aesthetic mappings.
`directed`	Logical. Default `FALSE`.

Details

Note: bibnets edge lists (`from`, `to`, `weight`) are accepted directly by `cograph::splot()` without conversion. This function is only needed when you want to attach node-level metadata.

Value

A `cograph_network` object (S3 list with `$nodes` and `$edges`).

Examples

```
data(biblio_data)

# Without metadata: splot() accepts bibnets edges directly
edges <- author_network(biblio_data, "collaboration")

# With metadata: document network + local citation scores as node size
edges <- document_network(biblio_data, type = "coupling")
nodes <- local_citations(biblio_data)   # keyed by document id
net   <- to_cograph(edges, nodes = nodes)
```

to_gephi *Export to Gephi node and edge tables*

Description

Converts a bibnets edge list (and optional node table) to the CSV format expected by Gephi's Data Laboratory. Column names are remapped to Gephi conventions (`Source`, `Target`, `Weight`, `Id`, `Label`).

Usage

```
to_gephi(edges, nodes = NULL, file = NULL, directed = FALSE)
```

Arguments

`edges`	A data frame with at least `from`, `to`, `weight` columns.
`nodes`	Optional data frame of node attributes. Must contain an `id` column. All other columns are included as Gephi node attributes.
`file`	Optional directory path. If supplied, writes `nodes.csv` and `edges.csv` into that directory. If `NULL` (default), returns a list.
`directed`	Logical. Sets the `Type` column. Default `FALSE`.

Value

If `file = NULL`: a list with `$nodes` and `$edges` data frames. If `file` is a directory path: writes two CSV files invisibly and returns the file paths.

Examples

```
data(biblio_data)
edges <- author_network(biblio_data, "collaboration")
gephi <- to_gephi(edges)
head(gephi$edges)
```

to_graphml *Export to GraphML*

Description

Writes a bibnets edge list (and optional node attributes) to a GraphML file using pure base R — no XML package required.

Usage

```
to_graphml(edges, nodes = NULL, file = NULL, directed = FALSE)
```

Arguments

edges	A data frame with at least from, to, weight columns.
nodes	Optional data frame of node attributes with an id column.
file	File path to write. If NULL (default), returns the GraphML as a character string.
directed	Logical. Default FALSE.

Value

If file = NULL: GraphML as a character string. Otherwise writes the file and returns the path invisibly.

Examples

```
data(biblio_data)
edges <- keyword_network(biblio_data)
xml <- to_graphml(edges)
cat(substr(xml, 1, 300))
```

---

to_igraph *Convert edge data frame to igraph*

Description

Convert edge data frame to igraph

Usage

```
to_igraph(edges, directed = FALSE)
```

Arguments

edges	A data frame with at least from, to, weight columns, as returned by any network function in bibnets.
directed	Logical. Default FALSE.

Value

An igraph graph object.

Examples

```
data(biblio_data)
edges <- author_network(biblio_data, "collaboration")
g <- to_igraph(edges)
```

to_matrix *Convert edge data frame to adjacency matrix*

Description

Convert edge data frame to adjacency matrix

Usage

```
to_matrix(edges, symmetric = TRUE)
```

Arguments

edges	A data frame with from, to, weight columns.
symmetric	Logical. If TRUE (default), produces a symmetric matrix.

Value

A sparse Matrix.

Examples

```
data(biblio_data)
edges <- reference_network(biblio_data, min_occur = 2)
to_matrix(edges)
```

to_tbl_graph *Convert edge data frame to tbl_graph*

Description

Convert edge data frame to tbl_graph

Usage

```
to_tbl_graph(edges, directed = FALSE)
```

Arguments

edges	A data frame with at least from, to, weight columns.
directed	Logical. Default FALSE.

Value

A tbl_graph object (tidygraph).

Examples

```
data(biblio_data)
edges <- keyword_network(biblio_data)
tg <- to_tbl_graph(edges)
```

Index