

Package ‘cooccure’

May 8, 2026

Title Co-Occurrence Network Construction and Manipulation

Version 0.1.1

Description Constructs co-occurrence networks from several types of input data, such as delimited fields, long/bipartite tables, binary matrices, or wide sequences. Returns tidy edge data frames and supports optional scaling, splitting into several networks, thresholding, and subsetting. Provides eight similarity measures, including Jaccard, cosine, and association strength. Supports export to several network and file formats. Network construction and analysis methods follow Saqr, Lopez-Pernas, Conde, and Hernandez-Garcia (2024, <doi:10.1007/978-3-031-54464-4_15>).

License MIT + file LICENSE

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.3

Imports graphics, Matrix, methods, stats, utils

Suggests igraph, cograph, Nestimate, tidygraph, testthat (>= 3.0.0), knitr, rmarkdown, shiny, DT

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 3.5.0)

URL <https://github.com/mohsaqr/cooccure>

BugReports <https://github.com/mohsaqr/cooccure/issues>

NeedsCompilation no

Author Mohammed Saqr [aut, cre, cph],
Sonsoles López-Pernas [aut, cph],
Kamila Misiejuk [aut, cph]

Maintainer Mohammed Saqr <saqr@saqr.me>

Repository CRAN

Date/Publication 2026-04-24 20:20:10 UTC

Contents

actors	2
actor_genres	3
as_cograph	3
as_igraph	4
as_matrix	5
as_netobject	5
as_tidygraph	6
cooccurrence	7
demo	10
launch_app	11
movies	11
plot.cooccurrence	12
print.cooccurrence	13
summary.cooccurrence	13
Index	15

actors	<i>IMDB actor-movie long table (1970-2024)</i>
--------	--

Description

Long-format bipartite table linking actors to movies in [movies](#). Pre-filtered to the 624 actors who appear in at least two movies, so all similarity measures compute instantly. Pass `field = "actor"` and `by = "tconst"` to [cooccurrence](#) to build an actor co-appearance network.

Usage

```
actors
```

Format

A data frame with 1,267 rows and 7 variables:

actor Actor name.

tconst IMDB title identifier linking to [movies](#).

primaryTitle Movie title.

startYear Release year (integer).

decade Release decade as a character string.

genres Comma-separated genre labels for the linked movie.

averageRating IMDB average user rating for the linked movie.

Source

<https://developer.imdb.com/non-commercial-datasets/>

Examples

```
head(actors)
cooccurrence(actors, field = "actor", by = "tconst", similarity = "jaccard")
```

actor_genres	<i>IMDB actor-genre long table (1970-2024)</i>
--------------	--

Description

Long-format table mapping each of the 624 actors in [actors](#) to every genre of every movie they appeared in. Use this to build an actor co-occurrence network grouped by genre: which actors share the same genres? Pass `field = "actor"` and `by = "genre"` to [cooccurrence](#).

Usage

```
actor_genres
```

Format

A data frame with 2,502 rows and 2 variables:

actor Actor name.

genre Genre label (one row per actor-genre combination).

Source

<https://developer.imdb.com/non-commercial-datasets/>

Examples

```
head(actor_genres)
cooccurrence(actor_genres, field = "actor", by = "genre", similarity = "jaccard")
```

as_cograph	<i>Convert to cograph network</i>
------------	-----------------------------------

Description

Creates a `cograph_network` object from a cooccurrence edge list, compatible with `cograph::splot()` and other cograph functions.

Usage

```
as_cograph(x, ...)

## S3 method for class 'cooccurrence'
as_cograph(x, ...)
```

Arguments

x A cooccurrence data frame.
 ... Ignored.

Value

A `cograph_network` object.

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))
if (requireNamespace("cograph", quietly = TRUE)) {
  net <- as_cograph(res)
  net$n_nodes
}
```

as_igraph

Convert to igraph

Description

Creates an undirected, weighted `igraph` graph from a cooccurrence edge list.

Usage

```
as_igraph(x, ...)

## S3 method for class 'cooccurrence'
as_igraph(x, ...)
```

Arguments

x A cooccurrence data frame.
 ... Passed to `igraph::graph_from_data_frame`.

Value

An `igraph` object.

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))
if (requireNamespace("igraph", quietly = TRUE)) {
  g <- as_igraph(res)
  igraph::vcount(g)
}
```

as_matrix	<i>Extract the co-occurrence matrix</i>
-----------	---

Description

Returns the full square co-occurrence matrix (normalized + scaled). Use type = "raw" for the raw count matrix.

Usage

```
as_matrix(x, ...)

## S3 method for class 'cooccurrence'
as_matrix(x, type = c("normalized", "raw"), ...)
```

Arguments

x	A cooccurrence data frame.
...	Ignored.
type	Character. "normalized" (default) or "raw".

Value

A numeric matrix.

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))
as_matrix(res)
as_matrix(res, type = "raw")
```

as_netobject	<i>Convert to Nestimate netobject</i>
--------------	---------------------------------------

Description

Creates a netobject from a cooccurrence edge list, compatible with Nestimate::centrality(), Nestimate::bootstrap_network(), etc.

Usage

```
as_netobject(x, ...)

## S3 method for class 'cooccurrence'
as_netobject(x, ...)
```

Arguments

x A cooccurrence data frame.
 ... Ignored.

Value

A netobject with class c("netobject", "cograph_network").

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))
if (requireNamespace("Nestimate", quietly = TRUE)) {
  net <- as_netobject(res)
  net$n_nodes
}
```

as_tidygraph

Convert to tidygraph

Description

Creates a tbl_graph from a cooccurrence edge list.

Usage

```
as_tidygraph(x, ...)

## S3 method for class 'cooccurrence'
as_tidygraph(x, ...)
```

Arguments

x A cooccurrence data frame.
 ... Ignored.

Value

A tbl_graph object.

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))
if (requireNamespace("tidygraph", quietly = TRUE) &&
    requireNamespace("igraph", quietly = TRUE)) {
  as_tidygraph(res)
}
```

`cooccurrence`*Build a co-occurrence network*

Description

Constructs an undirected co-occurrence network from various input formats and returns a tidy edge data frame. Argument names follow the `citennets` convention.

Usage

```
cooccurrence(  
  data,  
  field = NULL,  
  by = NULL,  
  sep = NULL,  
  weight_by = NULL,  
  split_by = NULL,  
  similarity = c("none", "jaccard", "cosine", "inclusion", "association", "dice",  
    "equivalence", "relative"),  
  counting = c("full", "fractional"),  
  scale = NULL,  
  threshold = 0,  
  min_occur = 1L,  
  top_n = NULL,  
  output = c("default", "gephi", "igraph", "cograph", "matrix"),  
  ...  
)  
  
co(  
  data,  
  field = NULL,  
  by = NULL,  
  sep = NULL,  
  weight_by = NULL,  
  split_by = NULL,  
  similarity = c("none", "jaccard", "cosine", "inclusion", "association", "dice",  
    "equivalence", "relative"),  
  counting = c("full", "fractional"),  
  scale = NULL,  
  threshold = 0,  
  min_occur = 1L,  
  top_n = NULL,  
  output = c("default", "gephi", "igraph", "cograph", "matrix"),  
  ...  
)
```

Arguments

data	<p>Input data. Accepts:</p> <ul style="list-style-type: none"> • A <code>data.frame</code> with a delimited column (<code>field + sep</code>). • A <code>data.frame</code> in long/bipartite format (<code>field + by</code>). • A binary (0/1) <code>data.frame</code> or <code>matrix</code> (auto-detected). • A wide sequence <code>data.frame</code> or <code>matrix</code> (non-binary). • A list of character vectors (each element is a transaction).
field	<p>Character. The entity column — determines what the nodes are. For delimited format, a single column split by <code>sep</code>. For long/bipartite, the item column. For multi-column delimited, a vector of column names pooled per row. Use <code>field = "all"</code> for wide sequence data (e.g. TraMineR / tna format) where every column is a time point and cell values are the items.</p>
by	<p>Character or NULL. Grouping column for long/bipartite format. Each unique value defines one transaction.</p>
sep	<p>Character or NULL. Separator for splitting delimited fields.</p>
weight_by	<p>Character or NULL. Column name containing a numeric association strength for each entity-transaction pair. Only accepted for long format (<code>field + by</code>). When supplied, each entity contributes its weight rather than 1, so $C_{ij} = \sum_d w_{id} \cdot w_{jd}$. Typical use: topic-document probability matrices from LDA or similar models.</p>
split_by	<p>Character or NULL. Column name to split the data by before computing co-occurrence. A separate network is computed per group and the results are combined into a single <code>data.frame</code> with an additional group column. Only works with <code>data.frame</code> inputs.</p>
similarity	<p>Character. Similarity measure:</p> <p>"none" Raw co-occurrence counts.</p> <p>"jaccard" $C_{ij}/(f_i + f_j - C_{ij})$.</p> <p>"cosine" Salton's cosine: $C_{ij}/\sqrt{f_i \cdot f_j}$.</p> <p>"inclusion" Simpson coefficient: $C_{ij}/\min(f_i, f_j)$.</p> <p>"association" Association strength: $C_{ij}/(f_i \cdot f_j)$ (van Eck & Waltman, 2009).</p> <p>"dice" $2C_{ij}/(f_i + f_j)$.</p> <p>"equivalence" Salton's cosine squared: $C_{ij}^2/(f_i \cdot f_j)$.</p> <p>"relative" Row-normalized: each row sums to 1.</p>
counting	<p>Character. Counting method:</p> <p>"full" Each co-occurring pair adds 1 regardless of transaction size. Default.</p> <p>"fractional" Each pair adds $1/(n_i - 1)$ where n_i is the number of items in transaction i. Transactions with many items contribute less per pair (Perianes-Rodriguez et al., 2016).</p>
scale	<p>Character or NULL. Optional scaling applied to weights after similarity normalization:</p> <p>NULL or "none" No scaling.</p> <p>"minmax" Min-max to $[0, 1]$.</p> <p>"log" Natural log: $\log(1 + w)$.</p>

	"log10" Log base 10: $\log_{10}(1 + w)$.
	"binary" Binary: 1 if $w > 0$, else 0.
	"zscore" Z-score standardization.
	"sqrt" Square root.
	"proportion" Divide by sum of all weights.
threshold	Numeric. Minimum edge weight to retain. Applied after similarity and scaling. Default 0.
min_occur	Integer. Minimum entity frequency. Entities appearing in fewer than min_occur transactions are dropped. Default 1.
top_n	Integer or NULL. Keep only the top top_n edges by weight. When split_by is used, applied per group. Default NULL (all edges).
output	Character. Column naming convention for the output: "default" from, to, weight, count. "gephi" Source, Target, Weight, Type (= "Undirected"). Ready for Gephi import. "igraph" Returns an igraph graph object directly. "cograph" Returns a cograph_network object directly. "matrix" Returns the square co-occurrence matrix.
...	Currently unused.

Value

Depends on output:

- "default": A cooccurrence data frame with columns from, to, weight, count (and group when split_by is used).
- "gephi": A data frame with columns Source, Target, Weight, Type, Count. Ready for Gephi CSV import.
- "igraph": An igraph graph object.
- "cograph": A cograph_network object.
- "matrix": A square numeric co-occurrence matrix.

For the data frame outputs, rows are sorted by weight descending and attributes store the full matrix, item frequencies, and parameters.

References

van Eck, N. J., & Waltman, L. (2009). How to normalize co-occurrence data? An analysis of some well-known similarity measures. *Journal of the American Society for Information Science and Technology*, 60(8), 1635–1651.

Examples

```
# Delimited keywords
df <- data.frame(
  id = 1:4,
  keywords = c("network; graph", "graph; matrix; network",
              "matrix; algebra", "network; algebra; graph")
)
cooccurrence(df, field = "keywords", sep = ";")

# Split by a grouping variable
df$year <- c(2020, 2020, 2021, 2021)
cooccurrence(df, field = "keywords", sep = ";", split_by = "year")

# List of transactions with Jaccard similarity
cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")),
             similarity = "jaccard")

# Short alias
co(df, field = "keywords", sep = ";", similarity = "cosine")

# Weighted long format (e.g. LDA topic-document probabilities)
theta <- data.frame(
  doc = c("d1","d1","d1","d2","d2","d3","d3"),
  topic = c("T1","T2","T3","T1","T3","T2","T3"),
  prob = c(0.6, 0.3, 0.1, 0.4, 0.6, 0.5, 0.5)
)
cooccurrence(theta, field = "topic", by = "doc", weight_by = "prob")
```

demo

Demo actor-movie-genre table

Description

A small hand-crafted dataset of 30 well-known actors across 10 classic films with genre labels. Designed for quick exploration in the Shiny app. Use `field = "actor"` with `by = "movie"` or `by = "genre"`.

Usage

demo

Format

A data frame with 34 rows and 3 variables:

movie Movie title.

actor Actor name.

genre Primary genre label.

Examples

```
head(demo)
cooccurrence(demo, field = "actor", by = "movie", similarity = "jaccard")
```

launch_app	<i>Launch the cooccure Shiny explorer</i>
------------	---

Description

Opens an interactive Shiny application for building and exploring co-occurrence networks. Requires the **shiny** and **DT** packages.

Usage

```
launch_app(...)
```

Arguments

... Passed to [runApp](#) (e.g. port, launch.browser).

Value

Called for its side effect (launches the app). No return value.

Examples

```
if (interactive()) {
  launch_app()
}
```

movies	<i>IMDB movie metadata (1970-2024)</i>
--------	--

Description

A sample of 1,000 highly-rated IMDB movies (rating ≥ 7.0 , $\geq 1,000$ votes) released between 1970 and 2024. The genres column is comma-delimited and suitable for use as the field argument to [cooccurrence](#).

Usage

```
movies
```

Format

A data frame with 1,000 rows and 7 variables:

tconst IMDB title identifier (e.g. "tt0068646").

primaryTitle Movie title.

startYear Release year (integer).

genres Comma-separated genre labels (e.g. "Crime,Drama").

decade Release decade as a character string (e.g. "1970s").

averageRating IMDB average user rating.

numVotes Number of IMDB user votes.

Source

<https://developer.imdb.com/non-commercial-datasets/>

Examples

```
head(movies)
cooccurrence(movies, field = "genres", sep = ",", similarity = "jaccard")
```

plot.cooccurrence *Plot a cooccurrence network*

Description

Plots the co-occurrence matrix as a heatmap. If **igraph** is available, plots a network graph instead.

Usage

```
## S3 method for class 'cooccurrence'
plot(x, type = c("heatmap", "network"), ...)
```

Arguments

x A cooccurrence data frame.
type Character. "heatmap" (default) or "network" (requires **igraph**).
... Passed to the plotting function.

Value

Invisibly returns **x**.

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))
plot(res)
```

print.cooccurrence *Print a cooccurrence edge list*

Description

Print a cooccurrence edge list

Usage

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

Arguments

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

Value

Invisibly returns x.

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))  
print(res)
```

summary.cooccurrence *Summarise a cooccurrence network*

Description

Summarise a cooccurrence network

Usage

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

Arguments

object	A cooccurrence data frame.
...	Ignored.

Value

Invisibly returns object.

Examples

```
res <- cooccurrence(list(c("A","B","C"), c("B","C"), c("A","C")))
summary(res)
```

Index

* datasets

- actor_genres, 3
- actors, 2
- demo, 10
- movies, 11

- actor_genres, 3
- actors, 2, 3
- as_cograph, 3
- as_igraph, 4
- as_matrix, 5
- as_netobject, 5
- as_tidygraph, 6

- co (cooccurrence), 7
- cooccurrence, 2, 3, 7, 11

- demo, 10

- launch_app, 11

- movies, 2, 11

- plot.cooccurrence, 12
- print.cooccurrence, 13

- runApp, 11

- summary.cooccurrence, 13