Package 'scalednap'

September 3, 2026

Type Package

Title Community Detection by Scaled Null-Adjusted Persistence

Version 1.0.0

Description Finds the vertex partition of an undirected graph that maximises a persistence-based objective, using the Milano local-search algorithm. Three related quality measures are supported: the persistence probability of a community, its null-adjusted persistence (NAP), and the scaled null-adjusted persistence (Scaled-NAP), which interpolates between NAP and modularity. The methods are described in Avellone et al. (2023) <doi:10.1007/s10288-023-00559-z> and Avellone et al. (2025) <doi:10.1016/j.ins.2025.123032>.

License GPL (>= 2)

Encoding UTF-8

SystemRequirements C++20

URL https://github.com/aavellone/scalednap-r

BugReports https://github.com/aavellone/scalednap-r/issues

Suggests igraph, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Collate 'scalednap-exports.R' 'cluster_milano.R' 'global_persistence.R' 'local_persistence.R'

Config/roxygen2/version 8.0.0

Author Alessandro Avellone [aut, cre], Paolo Bartesaghi [aut], Stefano Benati [aut], Rosanna Grassi [aut]

Maintainer Alessandro Avellone <alessandro.avellone@unimib.it>

Repository CRAN

Date/Publication 2026-09-02 22:10:02 UTC

Contents

scalednap-package 2
cluster_milano 2
global_persistence 5
local_persistence 6

Index **8**

`scalednap-package` *scalednap*

Description

Given a non-oriented graph, calculates the optimal vertex partition using persistence as the objective function.

Details

See manual entries.

Author(s)

Maintainer: Alessandro Avellone <alessandro.avellone@unimib.it>

Authors:

- Alessandro Avellone <alessandro.avellone@unimib.it>
- Paolo Bartesaghi <paolo.bartesaghi@unimi.it>
- Stefano Benati <stefano.benati@unitn.it>
- Rosanna Grassi <rosanna.grassi@unimib.it>

`cluster_milano` *cluster_milano*

Description

Calculates the vertex partition with maximum global null-adjusted persistence. This function is polymorphic: it automatically detects the input type and accepts either a vertex vector (accompanied by an edge list) or directly an `igraph` object.

Usage

```
cluster_milano(x, ...)

## Default S3 method:
cluster_milano(
  x,
  edge_list,
  weights = NULL,
  membership = NULL,
  H0 = 0,
  seed = NULL,
  n_restarts = 1L,
  tol = NULL,
  max_level = 0L,
  num_threads = 1L,
  collect_stats = FALSE,
  ...
)

## S3 method for class 'igraph'
cluster_milano(
  x,
  membership = NULL,
  H0 = 0,
  seed = NULL,
  n_restarts = 1L,
  tol = NULL,
  max_level = 0L,
  num_threads = 1L,
  collect_stats = FALSE,
  ...
)
```

Arguments

x	An integer or character vector representing the graph vertices, OR an object of class igraph.
...	Additional arguments passed to specific methods (e.g., edge_list, weights, tol, max_level, etc.).
edge_list	Integer matrix with two columns representing the graph edge list.
weights	Numeric vector of positive edge weights. If NULL, all weights default to 1.
membership	Integer vector representing the starting partition: x_i = k if i in C_k. If NULL, each vertex starts in its own cluster.
H0	Selects the quality measure. If NULL, the persistence probability is used. If 0 (default), the null-adjusted persistence (NAP). If a value in $(0, 1]$, the scaled null-adjusted persistence (Scaled-NAP, $sNAP_\alpha$), with scaling exponent α equal to H0.

seed	Two forms are accepted. A single non-negative integer is the *master seed*: if NULL or 0 a random master is generated (and reported in the result); if positive, the run is fully reproducible — the seeds of the individual restarts are derived deterministically from the master in C++ (SplitMix64 mixing, identical to the command-line tool). Alternatively, a digit string (e.g. the seed_winning field of a previous result) is an *exact 64-bit seed* used as-is, with no derivation: it reproduces that single restart and requires n_restarts = 1.
n_restarts	Positive integer: number of independent restarts (default 1). The best partition across restarts is returned; winning_restart in the result tells which one won.
tol	Optional numeric tolerance for the stopping criterion. If NULL (default), an adaptive threshold is calculated dynamically in C++.
max_level	Optional integer representing the maximum number of aggregation levels. If 0 (default) or NULL, the algorithm runs until convergence.
num_threads	Number of threads for the move-node phase. 1 (default) runs the deterministic sequential mode; values greater than 1 enable the lock-free parallel mode (Hogwild!), which is NOT deterministic even for a fixed seed.
collect_stats	Logical. If TRUE, the per-cluster scores are computed and returned in clusters_value. Default FALSE.

Value

A list with five elements:

membership The optimal vertex partition.

score The measure value of the optimal partition.

clusters_value Numeric vector with the score of each cluster, or NULL unless collect_stats = TRUE.

iterations Integer vector: number of move-node sweeps performed at each coarsening level.

seed The master seed actually used (numeric). Re-running with seed set to this value and the same n_restarts reproduces the result exactly. NA when the run was launched with an exact 64-bit seed string (no master exists in that case).

winning_restart 1-based index of the restart that produced the returned partition.

seed_winning Character string: the exact 64-bit seed used by the winning restart (returned as a string because R doubles are exact only up to 2^53). Pass it back as seed to reproduce that single winning run directly.

Examples

```
library(scalednap)

# --- EXAMPLE 1: Standard input (vectors and matrices) ---
edg <- c(1, 2, 1, 3, 1, 4, 2, 3, 3, 4, 4, 5, 5, 6, 5, 7, 6, 7)
edge_list <- matrix(edg, ncol = 2, byrow = TRUE)
vertex <- c(1, 2, 3, 4, 5, 6, 7)
cluster_milano(x = vertex, edge_list = edge_list)

# --- EXAMPLE 2: igraph input ---
```

```
if (requireNamespace("igraph", quietly = TRUE)) {
  g <- igraph::make_ring(10)
  cluster_milano(g)
}
```

`global_persistence` *global_persistence*

Description

Given a partition of the graph vertices, calculates the global persistence as the sum of the local persistences of the individual clusters. Persistence can be either null-adjusted or probability-based. This function is polymorphic: it automatically detects the input type and accepts either a vertex vector (accompanied by an edge list) or directly an `igraph` object.

Usage

```
global_persistence(x, ...)

## Default S3 method:
global_persistence(x, edge_list, weights = NULL, membership, H0 = 0, ...)

## S3 method for class 'igraph'
global_persistence(x, membership, H0 = 0, ...)
```

Arguments

`x`	An integer or character vector representing the graph vertices, OR an object of class `igraph`.
`...`	Additional arguments passed to specific methods (e.g., `edge_list`, `weights`, `membership`, `H0`).
`edge_list`	Integer matrix with two columns representing the graph edge list.
`weights`	Numeric vector of edge weights. If `NULL`, all weights default to 1.
`membership`	Integer vector of vertex cluster assignments: `x_i = k` if `i in C_k`.
`H0`	Selects the quality measure. If `NULL`, the persistence probability is used. If `0` (default), the null-adjusted persistence (NAP). If a value in $(0, 1]$, the scaled null-adjusted persistence (Scaled-NAP, $sNAP_\alpha$), with scaling exponent α equal to `H0`.

Value

A list with two elements:

score The global persistence of the partition.

clusters_value The local persistence of each cluster. A value of `NaN` indicates that cluster C_k is empty in the input `membership`.

Examples

```
library(scalednap)

# --- EXAMPLE 1: Standard input (vectors and matrices) ---
edg <- c(1, 2, 1, 3, 1, 4, 2, 3, 3, 4, 4, 5, 5, 6, 5, 7, 6, 7)
edge_list <- matrix(edg, ncol = 2, byrow = TRUE)
vertex <- c(1, 2, 3, 4, 5, 6, 7)
mem <- c(1, 1, 1, 1, 2, 2, 2)
global_persistence(x = vertex, edge_list = edge_list, membership = mem)

# --- EXAMPLE 2: igraph input ---
if (requireNamespace("igraph", quietly = TRUE)) {
  g <- igraph::make_ring(10)
  mem <- c(rep(1, 5), rep(2, 5))
  global_persistence(g, membership = mem)
}
```

---

local_persistence *local_persistence*

Description

Given the incidence vector of a vertex subset, calculates either the persistence probability or the null-adjusted persistence of cluster C. This function is polymorphic: it automatically detects the input type and accepts either a vertex vector (accompanied by an edge list) or directly an `igraph` object.

Usage

```
local_persistence(x, ...)

## Default S3 method:
local_persistence(x, edge_list, weights = NULL, cluster, H0 = 0, ...)

## S3 method for class 'igraph'
local_persistence(x, cluster, H0 = 0, ...)
```

Arguments

`x`	An integer or character vector representing the graph vertices, OR an object of class `igraph`.
`...`	Additional arguments passed to specific methods (e.g., `edge_list`, `weights`, `cluster`, `H0`).
`edge_list`	Integer matrix with two columns representing the graph edge list.
`weights`	Numeric vector of edge weights. If `NULL`, all weights default to 1.
`cluster`	Binary incidence vector of the cluster: `x_i = 1` if `i in C`, 0 otherwise.

H0 — Selects the quality measure. If NULL, the persistence probability is used. If 0 (default), the null-adjusted persistence (NAP). If a value in $(0, 1]$, the scaled null-adjusted persistence (Scaled-NAP, $sNAP_\alpha$), with scaling exponent α equal to H0.

Value

Numeric scalar: the persistence probability when H0 = NULL, the null-adjusted persistence (NAP) when H0 = 0, or the scaled null-adjusted persistence (Scaled-NAP, $sNAP_\alpha$, with α = H0) when H0 is in $(0, 1]$.

Examples

```
library(scalednap)

# --- EXAMPLE 1: Standard input (vectors and matrices) ---
edg <- c(1, 2, 1, 3, 1, 4, 2, 3, 3, 4, 4, 5, 5, 6, 5, 7, 6, 7)
edge_list <- matrix(edg, ncol = 2, byrow = TRUE)
vertex <- c(1, 2, 3, 4, 5, 6, 7)
cluster_bin <- c(1, 1, 1, 1, 0, 0, 0)
local_persistence(x = vertex, edge_list = edge_list, cluster = cluster_bin)

# --- EXAMPLE 2: igraph input ---
if (requireNamespace("igraph", quietly = TRUE)) {
  g <- igraph::make_ring(10)
  cluster_bin <- c(rep(1, 5), rep(0, 5))
  local_persistence(g, cluster = cluster_bin)
}
```

Index

cluster_milano, 2

global_persistence, 5

local_persistence, 6

scalednap *(*scalednap-package*)*, 2
scalednap-package, 2