Package: netsubsamp (via r-universe)

May 22, 2026

Type Package

Title Multivariate Inference of Network Moments by Subsampling

Version 1.0.0

Description Implements node subsampling methods for multivariate inference on network moments (rescaled motif counts), including: uniform node subsampling to approximate the joint distribution of multiple network moments (Algorithm 1); externally sparsified moments for density-matched comparisons (Algorithm 2); and a two-sample test for unmatchable networks with unequal edge densities via a split-and-sparsify subsampling procedure (Algorithm 3). Built-in support for V-shape (2-star), triangle, and 3-star motifs, with a user-extensible interface for arbitrary additional motifs. Parallel execution is supported via 'doParallel' and 'foreach'. Based on Qi, Hua, Li and Zhou (2024) <doi:10.48550/arXiv.2409.01599>.

License GPL-3

Encoding UTF-8

Imports foreach, doParallel, parallel, stats

Suggests Matrix, randnet, testthat (>= 3.0.0)

RoxygenNote 7.3.1

Config/testthat/edition 3

NeedsCompilation no

Author Mingyu Qi [aut], Chen-Wei Hua [aut], Tianxi Li [aut, cre], Wen Zhou [aut]

Maintainer Tianxi Li <tianxili@umn.edu>

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

Date/Publication 2026-04-21 19:42:33 UTC

RemoteUrl https://github.com/cran/netsubsamp

RemoteRef HEAD

RemoteSha 020fd15eb731ad082ac78b3e1414d71f1ef8808c

Contents

`netsubsamp-package` *netsubsamp: Multivariate Inference of Network Moments by Subsampling*

Description

netsubsamp implements the three subsampling algorithms from Qi, Hua, Li and Zhou (2024) for multivariate inference on network moments (rescaled counts of graph motifs such as V-shapes, triangles, and stars) under a sparse graphon model.

The three algorithms

Algorithm 1 – `subsample_moments`: Uniform node subsampling. Draws N_{sub} random node-induced subgraphs of size b from a single network G and computes the vector of network moments for each subsample. The resulting empirical distribution consistently approximates the joint distribution of multiple network moments for networks of size b drawn from the same graphon. This provides the foundation for any downstream inference task: confidence regions, goodness-of-fit tests, or visualization of the joint moment distribution.

Algorithm 2 – `sparsify_moments`: Externally sparsified moments. Given two independently sampled subgraphs G_1 and G_2 and a target density $\rho^\dagger$, uses G_1 to estimate the required edge-removal probability and then randomly removes edges from G_2 to produce a graph at the target density. Returns the density-normalised moment vector $\bar{\Psi}_{\rho^\dagger}(G_1, G_2)$. This building block handles the practical complication that the true graphon density is unknown.

Algorithm 3 – `two_sample_test`: Two-sample test for unmatchable networks with unequal densities. Tests $H_0 : w = w'$ for two unmatchable networks G (size n) and G' (size b) that may have different edge densities. The nodes of G' are split randomly into two halves; Algorithm 2 is applied to compute an observed statistic. Node subsampling from G (again via Algorithm 2) generates a reference distribution. A final multivariate test—either a Mahalanobis distance test or the Cauchy combination test of Liu and Xie (2020)—yields a p-value. This is the first subsampling-based procedure that is valid for unmatchable networks with unequal densities.

Motif support

Three motifs are built in: `"v_shape"` (2-star, 2 edges), `"triangle"` (3 edges), and `"star3"` (3-star, 3 edges). All three functions accept a `motifs` argument that can include user-supplied specifications: a list with element `fn`, a `function(A)` returning a *named* numeric vector of one or more moment values, and element `n_edges`, an integer vector of the corresponding edge counts used for density normalisation.

Parallel execution

The N_{sub} subsampling loop in `subsample_moments` and `two_sample_test` can be parallelised by setting `parallel = TRUE` (uses **doParallel** and **foreach**). Parallel mode requires the **netsubsamp** package to be installed on all worker nodes.

Author(s)

Maintainer: Tianxi Li <tianxili@umn.edu>

Authors:

- Mingyu Qi
- Chen-Wei Hua
- Wen Zhou

References

Qi M, Hua C-W, Li T, Zhou W (2024). Multivariate Inference of Network Moments by Subsampling. *Biometrika*, **111**(1), 1–18. arXiv:2409.01599.

Liu Y, Xie J (2020). Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures. *Journal of the American Statistical Association*, **115**, 393–402.

Examples

```
library(randnet)

## ---- Algorithm 1: approximate the joint moment distribution ----------
set.seed(1)
G   <- BlockModel.Gen(lambda = 15, n = 200, K = 3)$A
res <- subsample_moments(G, b = 60, N_sub = 500)

# Estimated density
res$rho

# First few rows of the N_sub x 3 moment matrix
head(res$moments)


## ---- Algorithm 2: externally sparsified moments ----------------------
set.seed(2)
G1  <- BlockModel.Gen(lambda = 15, n = 100, K = 3)$A
G2  <- BlockModel.Gen(lambda = 15, n = 100, K = 3)$A
sparsify_moments(G1, G2, rho_target = 0.08)


## ---- Algorithm 3: two-sample test under unequal densities -----------
set.seed(3)
G       <- BlockModel.Gen(lambda = 20, n = 400, K = 3)$A
G_prime <- BlockModel.Gen(lambda = 15, n = 100, K = 3)$A
```

```
# Choose rho_target as 0.7 * min(density(G), density(G_prime))
rho_G       <- sum(G)       / (400 * 399)
rho_G_prime <- sum(G_prime) / (100 * 99)
rho_target  <- 0.7 * min(rho_G, rho_G_prime)

res <- two_sample_test(G, G_prime,
                       rho_target = rho_target,
                       N_sub      = 500,
                       test       = "mahalanobis")
res$p_value
```

`sparsify_moments` *Externally sparsified network moments (Algorithm 2)*

Description

Uses one adjacency matrix (`G1`) to estimate the sparsification probability and a second independent adjacency matrix (`G2`) to compute density-normalised network moments after random edge removal to a common target density. The resulting statistic $\bar{\Psi}_{\rho^\dagger}(G_1, G_2)$ is the building block for the two-sample test in `two_sample_test`.

Usage

```
sparsify_moments(
  G1,
  G2,
  rho_target,
  motifs = list("v_shape", "triangle", "star3")
)
```

Arguments

`G1`	A square symmetric binary adjacency matrix (dense or sparse `Matrix`) used to estimate the sparsification probability.
`G2`	A square symmetric binary adjacency matrix from which moments are computed after edge removal.
`rho_target`	Numeric in $(0, 1)$; the target edge density $\rho^\dagger$.
`motifs`	A list of motif specifications; see `subsample_moments` for the format. Defaults to `list("v_shape", "triangle", "star3")`.

Value

A named numeric vector of density-normalised network moments, or `NA` values if the graph after edge removal has zero density.

Algorithm

1. Estimate $\hat{p} = \min\bigl(1,\, \rho^{\dagger}/\hat{\rho}_{G_1}\bigr)$.
2. Randomly remove edges from G_2: each edge is retained independently with probability $\hat{p}$, yielding $\tilde{G}_2$.
3. Return $\hat{\rho}_{\tilde{G}_2}^{-|E(R)|} \cdot U_R(\tilde{G}_2)$ for each motif R.

References

Qi, Hua, Li and Zhou (2024). Multivariate Inference of Network Moments by Subsampling. *Biometrika*, **111**(1), 1–18. doi:10.1093/biomet/asad056.

Examples

```
library(randnet)
set.seed(2)
G1 <- BlockModel.Gen(lambda = 15, n = 100, K = 3)$A
G2 <- BlockModel.Gen(lambda = 15, n = 100, K = 3)$A
sparsify_moments(G1, G2, rho_target = 0.08)
```

---

subsample_moments *Uniform node subsampling for multivariate network moments (Algorithm 1)*

Description

Repeatedly draws random node-induced subgraphs of size `b` from network `G` and computes network moments for each subsample. The resulting empirical distribution approximates the joint distribution of network moments for networks of size `b` drawn from the same underlying graphon model, providing the foundation for downstream inference.

Usage

```
subsample_moments(
  G,
  b,
  N_sub,
  motifs = list("v_shape", "triangle", "star3"),
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

G	A square symmetric binary adjacency matrix (dense `matrix` or sparse `Matrix`) of size $n \times n$.
b	Integer; the subsampling size. Must satisfy $3 \leq b < n$.
N_sub	Integer; number of subsamples to draw.
motifs	A list of motif specifications. Each element is either: • a character string: `"v_shape"`, `"triangle"`, or `"star3"` (built-in defaults); or • a named list with elements `fn` – a `function(A)` returning a *named* numeric vector – and `n_edges` – an integer vector of edge counts matching the output length of `fn`. Defaults to `list("v_shape", "triangle", "star3")`.
parallel	Logical; if `TRUE` the subsampling loop is run in parallel via **doParallel** and **foreach**. Default `FALSE`. Requires the **netsubsamp** package to be *installed* on all worker nodes when `TRUE`.
n_cores	Integer or `NULL`; number of cores to use when `parallel = TRUE`. If `NULL`, uses `parallel::detectCores() - 1` (minimum 1).

Value

A list with:

`rho` Estimated edge density of `G`.

`moments` An $N_{\text{sub}} \times m$ numeric matrix of subsampled network moments. Rows for which the subsampled graph has zero density contain `NA`.

`motif_names` Character vector of moment names, in column order of `moments`.

`b` The subsampling size used.

References

Qi, Hua, Li and Zhou (2024). Multivariate Inference of Network Moments by Subsampling. *Biometrika*, **111**(1), 1–18. doi:10.1093/biomet/asad056.

Examples

```
library(randnet)
set.seed(1)
G   <- BlockModel.Gen(lambda = 15, n = 200, K = 3)$A
res <- subsample_moments(G, b = 60, N_sub = 500)
head(res$moments)
res$rho
```

`two_sample_test` *Two-sample test for unmatchable networks with unequal densities (Algorithm 3)*

Description

Tests whether two unmatchable networks — differing in node set, size, and possibly edge density — arise from the same underlying graphon model. The procedure avoids the density-mismatch problem via a *split-and-sparsify* strategy: each network is randomly split into two halves, one used to estimate the sparsification probability and the other to compute density-normalised moments at a shared target density. Node subsampling from the larger network then generates a reference distribution against which the observed statistic is compared.

Usage

```
two_sample_test(
  G,
  G_prime,
  rho_target,
  N_sub = 2000L,
  motifs = list("v_shape", "triangle", "star3"),
  test = c("mahalanobis", "cauchy"),
  parallel = FALSE,
  n_cores = NULL
)
```

Arguments

`G`	A square symmetric binary adjacency matrix (dense or sparse `Matrix`) of size $n \times n$. Typically the larger or reference network.
`G_prime`	A square symmetric binary adjacency matrix of size $b \times b$. The network to compare against `G`.
`rho_target`	Numeric in $(0, 1)$; the common target edge density $\rho^\dagger$. See the *Choice of* `rho_target` section.
`N_sub`	Integer; number of subsampling replicates. Default 2000.
`motifs`	A list of motif specifications; see `subsample_moments` for the format. Defaults to `list("v_shape", "triangle", "star3")`.
`test`	Character; the multivariate test to apply. Either `"mahalanobis"` (default) or `"cauchy"`.
`parallel`	Logical; whether to parallelise the subsampling loop. Default `FALSE`. Requires the **netsubsamp** package to be *installed* on all worker nodes when `TRUE`.
`n_cores`	Integer or `NULL`; number of cores when `parallel = TRUE`.

Value

A list with:

`p_value` The p-value of the test.

`test_statistic` Observed test statistic: Mahalanobis distance D_0 when `test = "mahalanobis"`, or Cauchy combination statistic T when `test = "cauchy"`.

`observed_moments` Named numeric vector; the observed $\bar{\Psi}$ statistic.

`null_moments` $N_{\text{sub}} \times m$ matrix of null $\bar{\Psi}$ statistics (including any `NA` rows from degenerate subsamples).

`null_statistic` Numeric vector of per-replicate null Mahalanobis distances (or marginal p-values for `"cauchy"`).

`marginal_p_values` (Only for `test = "cauchy"`.) Named numeric vector of per-motif marginal p-values.

`method` The test method used.

`motif_names` Character vector of moment names.

Algorithm

1. Randomly partition the nodes of `G_prime` into two equal halves G'_1 and G'_2; compute the observed statistic $\bar{\Psi}_{\rho^\dagger}(G'_1, G'_2)$ via Algorithm 2.
2. For $i = 1, \ldots, N_{\text{sub}}$: independently draw two sets of $\lfloor b/2 \rfloor$ nodes from G and compute $\bar{\Psi}_{\rho^\dagger}(G^*_{i1}, G^*_{i2})$ via Algorithm 2.
3. Compare the observed statistic against the empirical null distribution using the selected `test`.

Choice of `rho_target`

A recommended default is $\rho^\dagger = \kappa \cdot \min(\hat{\rho}_G, \hat{\rho}_{G'})$, with $\kappa \in (0.5,\ 0.9)$. Values close to 0 discard too many edges and inflate variance; values too close to $\min(\hat{\rho}_G, \hat{\rho}_{G'})$ leave little room for edge removal and can break the density-matching property. A value of $\kappa = 0.7$ (corresponding to `rho_target = 0.7 * min(rho_hat_G, rho_hat_G_prime)`) works well across a range of settings.

Tests

Mahalanobis (default) Computes a multivariate distance that fully exploits the joint distribution of all moments. The p-value is the proportion of null distances exceeding the observed distance. This test is more powerful when the signal lies in the *joint* rather than the *marginal* distribution of the moments.

Cauchy combination Combines marginal p-values via the Cauchy combination test of Liu & Xie (2020). Controls type I error under arbitrary dependence but ignores cross-moment correlation.

References

Qi, Hua, Li and Zhou (2024). Multivariate Inference of Network Moments by Subsampling. *Biometrika*, **111**(1), 1–18. doi:10.1093/biomet/asad056.

Liu, Y. and Xie, J. (2020). Cauchy combination test: a powerful test with analytic p-value calculation under arbitrary dependency structures. *Journal of the American Statistical Association*, **115**, 393–402. doi:10.1080/01621459.2018.1554485.

Examples

```
library(randnet)
set.seed(3)
G       <- BlockModel.Gen(lambda = 20, n = 400, K = 3)$A
G_prime <- BlockModel.Gen(lambda = 15, n = 100, K = 3)$A

# Choose rho_target as 0.7 * min(density(G), density(G_prime))
rho_G       <- sum(G)       / (400 * 399)
rho_G_prime <- sum(G_prime) / (100 * 99)
rho_target  <- 0.7 * min(rho_G, rho_G_prime)

res <- two_sample_test(G, G_prime,
                       rho_target = rho_target,
                       N_sub      = 500,
                       test       = "mahalanobis")
res$p_value
```

Index