Package 'EcoNetGen'

August 27, 2023

Version 0.2.4

Title Simulate and Sample from Ecological Interaction Networks

Description Randomly generate a wide range of interaction networks with specified size, average degree, modularity, and topological structure. Sample nodes and links from within simulated networks randomly, by degree, by module, or by abundance. Simulations and sampling routines are implemented in 'FORTRAN', providing efficient generation times even for large networks. Basic visualization methods also included. Algorithms implemented here are described in de Aguiar et al. (2017) <arXiv:1708.01242>.

License GPL-3

URL `https://github.com/cboettig/EcoNetGen`

BugReports `https://github.com/cboettig/EcoNetGen/issues`

Encoding UTF-8

ByteCompile true

RoxygenNote 6.1.1

Suggests spelling, testthat, covr, ggraph

Imports igraph, ggplot2

Language en-US

NeedsCompilation yes

Author Marcus de Aguiar [aut, cph] (<`https://orcid.org/0000-0003-1379-7568`>), Erica Newman [aut] (<`https://orcid.org/0000-0001-6433-8594`>), Mathias Pires [aut] (<`https://orcid.org/0000-0003-2500-4748`>), NIMBioS [fnd], Carl Boettiger [aut, cre] (<`https://orcid.org/0000-0003-4580-091X`>)

Maintainer Carl Boettiger <`cboettig@gmail.com`>

Repository CRAN

Date/Publication 2023-08-27 05:10:02 UTC

R topics documented:

adj_plot *Plot network adjacency matrix*

Description

Plot network adjacency matrix

Usage

```
adj_plot(graph)
```

Arguments

graph	an igraph object

Examples

```
set.seed(12345)
graph <- netgen()
adj_plot(graph)
```

netgen *netgen*

Description

Randomly generate a wide range of interaction networks

Usage

```
netgen(net_size = 50, ave_module_size = 10, min_module_size = 6,
  min_submod_size = 1, net_type = c("mixed", "random", "scalefree",
  "nested", "bi-partite nested", "bi-partite random",
  "tri-trophic bipartite nested-random",
  "tri-trophic bipartite nested-bipartite nested", "bn", "br", "tt-bn-r",
  "tt-bn-bn"), ave_degree = 5, rewire_prob_global = 0.2,
  rewire_prob_local = 0, mixing_probs = c(0.2, 0.2, 0.2, 0.2, 0.2, 0,
  0), verbose = FALSE)
```

Arguments

`net_size`	network size (number of nodes)
`ave_module_size`	average module size
`min_module_size`	cutoff for the minimum modules size
`min_submod_size`	cutoff for submodules, used only for bipartite and tripartite networks
`net_type`	network type, see details
`ave_degree`	average degree of connection
`rewire_prob_global`	probability any given edge should be rewired
`rewire_prob_local`	probability that edges within a module should be rewire locally (within the module)
`mixing_probs`	module probabilities for first 7 types, used for constructing mixed networks
`verbose`	logical, default TRUE. Should a message report summary statistics?

Details

network type is one of

- mixed
- random
- scalefree
- nested
- bi-partite nested (or short-hand "bn")
- bi-partite random (or short-hand "br")
- tri-trophic bipartite nested-random. (Can use short-hand "ttbnr")
- tri-trophic bipartite nested-bipartite nested (Can use short-hand "ttbnbn")

Valid Parameter Ranges

Please note that not all combinations of parameters will create valid networks. If an invalid combination is requested, `netgen()` will error with an informative message. A list of these constraints is provided below for reference.

1. `net_size` >= `ave_module_size`. If ‘net_size = ave_module_size“ the program generates a network with a single module.
2. `ave_module_size` > `min_module_size`
3. `ave_degree` >= `1`. Preferably larger than 4, to ensure single component modules.
4. `rewire_prob_global` = `0` produces completely uncoupled modules. To ensure a single component network use `rewire_prob_global` > `0` and sufficiently large.
5. `rewire_prob_local` = `0` produces idealized modules. Use `rewire_prob_local` > `0` to add stochasticity to the modules.

6. For tripartite networks `min_module_size` > `min_submod_size`. This also implies `min_module_size` `>= 2`.
7. For scalefree networks (or mixed networks involving scalefree modules) `ave_degree` < `min_module_size`
8. For mixed networks `mixing_probs` need to sum to `1`. If the sum is larger than one, only the first types, corresponding to `sum <=1`, will be sampled.

Value

an `igraph` object

Examples

```
library(EcoNetGen)

set.seed(12345)
net <- netgen()
adj_plot(net)
```

`netgen_v1` *netgen_v1*

Description

netgen function

Usage

```
netgen_v1(n_modav = c(50, 10), cutoffs = c(3, 0), net_type = 1,
  net_degree = 10, net_rewire = c(0.3, 0), mod_probs = c(0.2, 0.2,
  0.2, 0.2, 0.2, 0, 0), verbose = FALSE)
```

Arguments

`n_modav`	network size and average module size (integer vector, length 2)
`cutoffs`	module and submodule minimum sizes (integer vector, length 2). (submodules are used only for bipartite and tripartite networks)
`net_type`	integer indicating type, see details
`net_degree`	average degree of connection
`net_rewire`	global and local network rewiring probabilities
`mod_probs`	module probabilities for types 1 to 51, used for constructing mixed networks, net_type = 0
`verbose`	logical, default TRUE. Should a message report summary statistics?

Details

network type

- 0 = mixed
- 1 = random
- 2 = scalefree
- 3 = nested
- 41 = bi-partite nested
- 42 = bi-partite random
- 51 = tri-trophic bipartite nested-random "ttbnr"
- 52 = tri-trophic bipartite nested-bipartite nested "ttbnbn"

Value

an `igraph` object

`netsampler` *Network Sampling Routine*

Description

Network Sampling Routine

Usage

```
netsampler(network_in, key_nodes_sampler = c("random", "lognormal",
  "Fisher log series", "exponential", "degree", "module"),
  neighbors_sampler = c("random", "exponential"), n_key_nodes = 10,
  n_neighbors = 0.5, hidden_modules = NULL, module_sizes = NULL,
  cluster_fn = igraph::cluster_edge_betweenness)
```

Arguments

`network_in`	input network (as igraph object)
`key_nodes_sampler`	sampling criteria for key nodes. See details.
`neighbors_sampler`	sampling criteria for neighbors. see details.
`n_key_nodes`	number of key nodes to sample.
`n_neighbors`	number of first neighbors or fraction of first neighbors. See details.
`hidden_modules`	list of the modules to exclude (max 10 modules; only the first numb_hidden are used)
`module_sizes`	integer vector giving the size of each module. see details.
`cluster_fn`	a clustering function, from `igraph::cluster_*`. Default is `igraph::cluster_edge_betweeness`. Only used to compute module sizes if not provided.

Details

Algorithm first samples n_key_nodes according the the requested `key_nodes_sampler` criterion. For each key node, the requested number or fraction of neighbors is then sampled according to the `neighbors_sampler` criterion. Optionally, a list of modules can be designated as "hidden" and will be excluded from sampling.

if `n_neighbors is greater than 1, assumes this is the number to sample. If n_neighborsis between 0 and 1, assumes this is the fration of neighbors to sample. (To sample 1 neighbor, use an explicit integer,` 1L (or `as.`integer(1)‘) to sample 100

Provide `module_sizes` list to improve performance. If not provided, this will will be calculated based on `igraph::cluster_edge_betweeness`. Be sure to provide a `module_sizes` vector whenever calling `netsampler` repeatedly on the same network to avoid unnecessary performance hit from recalculating modules every time. See examples.

Value

the original input network (as an igraph network object), with the attribute `label` added to the edges and vertices indicating if that edge or vertex was `sampled` or `unsampled`.

Examples

```
set.seed(12345)
net <- netgen()
sample <- netsampler(net)

## Precompute `module_sizes` for replicate sampling of the same network:
 library(igraph)
 modules <- cluster_edge_betweenness(as.undirected(net))
 module_sizes <- vapply(igraph::groups(modules), length, integer(1))
 sample <- netsampler(net, module_sizes = module_sizes)
```

Index