Package ‘sdcn’

March 27, 2015

Type Package

Title Structures and Dynamics on (of) Complex Networks (sdcn)

Version 0.0.0.9000

Date 2015-02-10

Author

Maintainer Wenfeng Feng <fengwenfeng@gmail.com>

Description The package intends to implement general simulation of dynamics on (of) networks which have different structural features. The current goal is to simuate ecological interactions among species in ecological networks, as the first instance of complex networks.
Modules should include:
1) Dynamic models. Holling Type I, II dynamic models should be implemented for mutualistic networks, food webs, competitive networks, and mixed networks.
2) Environmental Perturbations. Two types of perturbations: continuously pressed env. and repeated pulsed env. (stochastics). The perturbations can effect not only on (all or part of) species(nodes) but also on (all or part of) interactions(links).
3) Null models of different structural features such as degree heterogeneity and modularity.
4) Analysis of simulation results.
5) Fit of empirical data?

Imports deSolve (>= 1.10-8),
simecol (>= 0.8-4),
rootSolve (>= 1.6.5),
plyr (>= 1.8.1),
igraph (>= 0.7.1)

License What license is it under?

Suggests knitr,
testthat

VignetteBuilder knitr

LazyData true

R topics documented:

`fragility` *compute fragility of mutualistic communities in gradual pressed context*

Description

compute fragility of mutualistic communities in gradual pressed context

Usage

```
fragility(sim.out)
```

Arguments

`sim.out`	output of simulation under gradual pressed conditions (`sim_ode_press`)

Value

resistance measured by the length of community trajectory

fragility measured by the variance of community trajectory

graph_connected — *Generate a connected graph using package [igraph]*

Description

Generate a connected graph using package [igraph]

Usage

```
graph_connected(s, k, gtype, maxtried = 100, expower = 2.5, ...)
```

Arguments

`s,`	size of network. if graph type is bipartite, s[1], s[2] represent size of two groups; else s is size of network
`k,`	average degree for the network. 1 < k < s for unipartite network, 1 < k < s[1]*s[2]/(s[1]+s[2]) for bipartite network.
`gtype,`	Graph type generated: 'bipartite', 'sf', 'er', 'regular'.
`maxtried,`	the maximum number of tried times. If have tried [maxtried] times, the function will return no matter whether the connected graph is generated.
`expower`	exponent coefficient of Scale-Free network
`...`	the parms conform to the implementation functions of [igraph]

Details

.

Value

the connected graph

inc_to_adj — *transfer an incidence matrix to an adjacency matrix*

Description

transfer an incidence matrix to an adjacency matrix

Usage

```
inc_to_adj(inc)
```

Arguments

`inc,`	an incidence matrix

Value

adj, an adiacency matrix

`init_lv2_cm` *initial values of state variables, i.e., abundances of species*

Description

Assign initial values according to two criteria: 1. using the equilibrium values of LV1 model as initial values. 2. If any of the initial values is less than 0, using the intrinsic growth rates as initial values.

Usage

```
init_lv2_cm(parms)
```

Arguments

`parms,`	the parameters assigned to LV2 model

`model_lv2_ca` *Lotka-Volterra (LV) Equations of Holling type II for a community mixed by Competition and Antagonistic interactions*

Description

Lotka-Volterra (LV) Equations of Holling type II for a community mixed by Competition and Antagonistic interactions

Usage

```
model_lv2_ca(time, init, parms, ...)
```

Arguments

`time,`	time step of simulation
`init,`	the initial state of the LV system, a vector
`parms,`	parameters passed to LV model, a list of: **r** a vector of the intrinsic growth rates of species **C** a matrix of intra-species and inter-species competitions **A** a matrix of antagonistic interactions among species **h** the saturate coefficient, handling time of species feed **g** the conversion efficiency of antagonistic interactions

Value

the derivation

`model_lv2_cm` *Lotka-Volterra (LV) Equations of Holling type II for a community mixed by Competition and Mutualism interactions*

Description

Lotka-Volterra (LV) Equations of Holling type II for a community mixed by Competition and Mutualism interactions

Usage

```
model_lv2_cm(time, init, parms, ...)
```

Arguments

`time,`	time step of simulation
`init,`	the initial state of the LV system, a vector
`parms,`	parameters passed to LV model, a list of: **r** a vector of the intrinsic growth rates of species **C** a matrix of intra-species and inter-species competitions **M** a matrix of mutualism interactions among species **h** the saturate coefficient, handling time of species feed

Value

the derivation

`M_PL_001` *web-of-life data sets from Bascompte et atl.*

Description

web-of-life data sets from Bascompte et atl.

Usage

```
M_PL_001
```

Format

data frames with rows and cols represent two different species groups

Source

http://www.web-of-life.es/map.php

parms_antago_interactions
generate the antagonistic interaction matrix according to the antagonistic network and coefficients

Description

generate the antagonistic interaction matrix according to the antagonistic network and coefficients

Usage

```
parms_antago_interactions(graph, gamma.mu, gamma.sd)
```

Arguments

graph	the antagonistic interaction topology of communities, which is a bi-directional network
gamma.mu	
gamma.sd	coefficients that determin a uniform distribution of antagonistic interaction strengths

parms_competitive_interactions
generate the competitive interaction matrix according to the competitive network and coefficients

Description

generate the competitive interaction matrix according to the competitive network and coefficients

Usage

```
parms_competitive_interactions(graph, beta0.mu, beta0.sd, beta1.mu, beta1.sd)
```

Arguments

graph	the competitive interaction topology of communities, which is the adjacency matrix of a network
beta0.mu	
beta0.sd	coefficients that determin a uniform distribution of intra-species interaction strengths
beta1.mu	
beta1.sd	coefficients that determin a uniform distribution of inter-species interaction strengths

---

`parms_lv2_ca` *parmaters for antagonism LV2 model according to the network and the coefficients*

Description

parmaters for antagonism LV2 model according to the network and the coefficients

Usage

```
parms_lv2_ca(antago_graph, competitive_graph, coeff)
```

Arguments

`antago_graph` the antagonistic interaction topology of communities, which is the adjacency matrix of a network

`competitive_graph` the competitive interaction topology of communities

`coeff,` a list of coefficients:

- **alpha.mu, alpha.sd** coefficients of the intrinsic growth rates of species
- **beta0.mu, beta0.sd** the intra-species competition coefficients which determin a uniform distribution in [beta.mu - beta.sd, beta.mu + beta.sd]
- **beta1.mu, beta1.sd** the inter-species competition coefficients
- **gamma.mu, gamma.sd** the inter-species antagonism coefficients
- **g.mu, g.sd** conversion efficiency of antagonistic interactions
- **h.mu, h.sd** coefficients of the handling time of species

Value

a list of parameters for ode model:

- **r** a vector of the intrinsic growth rates of species
- **C** a matrix of intra-species and inter-species competitions
- **A** a matrix of antagonistic interactions among species
- **h** the saturate coefficient, handling time of species feed
- **g** the conversion efficiency of antagonistic interactions

`parms_lv2_cm` *parmaters for mutualism LV2 model according to the network and the coefficients*

Description

parmaters for mutualism LV2 model according to the network and the coefficients

Usage

```
parms_lv2_cm(mutual_graph, competitive_graph, coeff)
```

Arguments

`mutual_graph` the mutualistic interaction topology of communities, which is the adjacency matrix of a network

`competitive_graph` the competitive interaction topology of communities

`coeff,` a list of coefficients:

- **alpha.mu, alpha.sd** coefficients of the intrinsic growth rates of species
- **beta0.mu, beta0.sd** the intra-species competition coefficients which determin a uniform distribution in [beta.mu - beta.sd, beta.mu + beta.sd]
- **beta1.mu, beta1.sd** the inter-species competition coefficients
- **gamma.mu, gamma.sd** the inter-species mutualism coefficients
- **delta** trade-off coefficients of mutualistic interaction strengths
- **h.mu, h.sd** coefficients of the handling time of species

Value

a list of parameters for ode model:

- **r** a vector of the intrinsic growth rates of species
- **C** a competitive interaction matrix
- **M** a mutualistic interaction matrix among species
- **h** the saturate coefficient, handling time of species feed

`parms_mutual_interactions` *generate the mutualistic interaction matrix according to the mutualistic network and coefficients*

Description

generate the mutualistic interaction matrix according to the mutualistic network and coefficients

Usage

```
parms_mutual_interactions(graph, gamma.mu, gamma.sd, delta)
```

Arguments

graph	the mutualistic interaction topology of communities, which is the adjacency matrix of a network
gamma.mu	
gamma.sd	coefficients that determin a uniform distribution of mutualistic interaction strengths
delta	coefficient that determin the trade-off between the interaction strength and width(node degree) of species

---

perturb_growthrate *perturbations that effect on species by increasing/decreasing the intrinsic growth rates of all species*

Description

perturbations that effect on species by increasing/decreasing the intrinsic growth rates of all species

Usage

```
perturb_growthrate(parms, nstar, r.delta.mu = 0.01, r.delta.sd = 0.01)
```

Arguments

parms	parameters assigned to the ODE model
nstar	state values at equilibrium
r.delta	deviation of intrinsic growth rates at each iterating step

---

perturb_growthrate_part *perturbations that effect on species by increasing/decreasing the intrinsic growth rates of a part of species*

Description

perturbations that effect on species by increasing/decreasing the intrinsic growth rates of a part of species

Usage

```
perturb_growthrate_part(parms, nstar, r.delta.mu = 0.01, r.delta.sd = 0.01,
  perturbed_species)
```

Arguments

parms	parameters assigned to the ODE model
nstar	state values at equilibrium
perturbed_species	the index of perturbed species
r.delta	deviation of intrinsic growth rates at each iterating step

---

`perturb_mutualistic_strength`
perturbations that effect on mutualistic interactions by increasing/decreasing strengths of them

---

Description

perturbations that effect on mutualistic interactions by increasing/decreasing strengths of them

Usage

```
perturb_mutualistic_strength(parms, nstar, gamma.delta.mu = 0.01,
  gamma.delta.sd = 0.01)
```

Arguments

`parms`	parameters assigned to the ODE model
`nstar`	state values at equilibrium
`gamma.delta`	deviation of mutualistic interaction strengths at each iterating step

---

`perturb_primary_extinct`
perturbations that remove one species

Description

perturbations that remove one species

Usage

```
perturb_primary_extinct(parms, nstar, extinct_species)
```

Arguments

`parms`	parameters assigned to the ODE model
`nstar`	state values at equilibrium
`extinct_species`	the removed species

resistance — *compute resistance of mutualistic communities in gradual pressed context*

Description

compute resistance of mutualistic communities in gradual pressed context

Usage

```
resistance(sim.out)
```

Arguments

`sim.out`	output of simulation under gradual pressed conditions (`sim_ode_press`)

Value

resistance measured by the length of community trajectory

runif2 — *another form of uniform distribution between [mean - sd, mean + sd]*

Description

another form of uniform distribution between [mean - sd, mean + sd]

Usage

```
runif2(n, mean, sd)
```

sdcn — *sdcn: Structures and Dynamics on (of) Complex Networks.*

Description

The sdcn package provides three categories of functions:

1. Structures
2. Dynamics
3. Analysis

Structures functions

`swaplinks`

Dynamics functions

`model_lv2`, `parms_lv2`

`sim_ode_auto`	*Simulate ODE dynamics of autonomous systems.The dynamic starts at initialized state variables, and ends in equilibrium (or error where some values of state variables approach infinity?)*

Description

Simulate ODE dynamics of autonomous systems.The dynamic starts at initialized state variables, and ends in equilibrium (or error where some values of state variables approach infinity?)

Usage

```
sim_ode_auto(model, parms, init, steps = 1000, stepwise = 1,
  extinct_threshold)
```

Arguments

`model`	model of ODE dynamics
`parms`	parameters assigned to the model
`init`	initial values of the model according to the parameters
`steps`	steps of simulation
`stepwise`	step length
`extinct_threshold`	abundance threshold, species with abundance less than that is considered to be exintct

Value

a list of:

out output of one ODE simulation, including the trajectory of values of state variables

nstar the values of state variables in equilibrium

Phi the Jacobian matrix in equilibrium

model model of ODE dynamics

parms parameters assigned to the model

extinct number of extinct species

survived number of survived species

`sim_ode_press` *Simulate ODE dynamics of non-autonomous systems.A example is ecosystems under "press" perturbations. The dynamic is iteration of successive ODE dynamics of automous sytems (`sim_ode_auto`), while at each iterating step, the parameters and/or state values of systems are changed to reflect "press" perturbations.*

Description

Simulate ODE dynamics of non-autonomous systems.A example is ecosystems under "press" perturbations. The dynamic is iteration of successive ODE dynamics of automous sytems (`sim_ode_auto`), while at each iterating step, the parameters and/or state values of systems are changed to reflect "press" perturbations.

Usage

```
sim_ode_press(model, parms, init, steps = 1000, stepwise = 1,
  extinct_threshold, perturb, iter_steps = 500, isout = TRUE, ...)
```

Arguments

`model`	model of ODE dynamics
`parms`	parameters assigned to the model
`init`	initial values of the model according to the parameters
`steps`	steps of simulation
`stepwise`	step length
`extinct_threshold`	abundance threshold, species with abundance less than that is considered to be exintct
`perturb`	a function that change the parameters and state values after each iteration step
`iter_steps`	possiblely maximum iteration steps
`isout`	if output the transiting trajectory of each ODE iterate step
`...`	any arguments which are transfered to perturbation function

Value

a list of lists :

out output of one ODE simulation, including the trajectory of values of state variables

nstar the values of state variables in equilibrium

Phi the Jacobian matrix in equilibrium

parms parameters assigned to the model

extinct.species a vector of species that extincted swaplinks — *Swapping links Algorithm for null model of bipartite networks, that generates random network (ensembles) which keep the node degree distribution of a real network.*

Description

Swapping links Algorithm for null model of bipartite networks, that generates random network (ensembles) which keep the node degree distribution of a real network.

Usage

```
swaplinks(bigraph, ntry = 5000)
```

Arguments

`bigraph,`	incidence matrix of a bipartite network, rows and cols represent two groups of nodes/species
`ntry,`	the maximum possible times of swapping links to try

Value

an incidence matrix of bipartite network whose links being randomly swapped.

Examples

```
## Not run:
require(bipartite) # for plot
data(M_PL_003)
# M_PL_003 <- as.matrix(M_PL_003)
bipartite::visweb(M_PL_003)
M_PL_003.rand = swaplinks(M_PL_003)
bipartite::visweb(M_PL_003.rand)

## End(Not run)
```

Index