Structures and Dynamics on (of) Complex Networks (sdcn). I Ecosystems

Authors

2015-02-23

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:

- Null models of different structural features such as degree heterogeneity and modularity.
- Dynamic models. Holling Type I, II dynamic models should be implemented for mutualistic networks, food webs, competitive networks, and mixed networks.
- 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).
- Analysis of simulation results.
- Fit of empirical data?

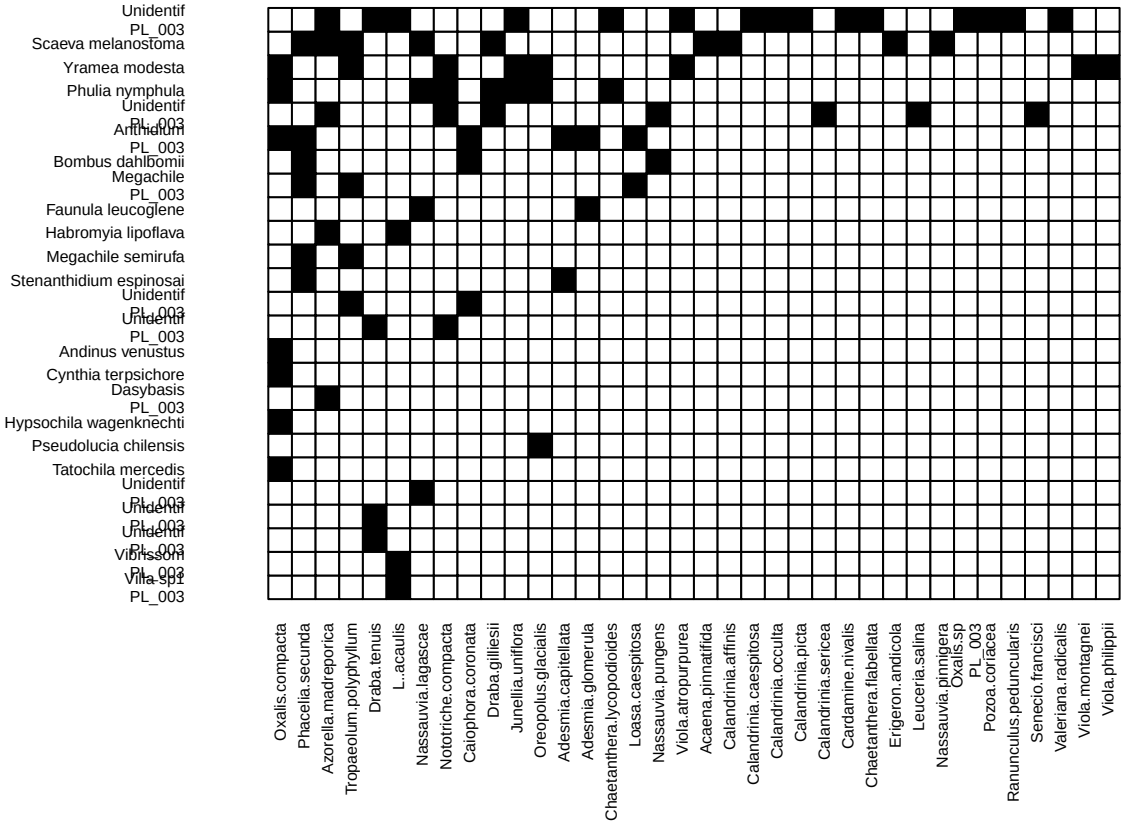
Null models of complex networks (graphs)

The null model of a graph is a randomized version of the original graph. A null model graph matches with the original graph in some of its structural features such as degree distribution or modularity, but is otherwise a completed random graph. Null models of graphs (can) not only serve to identify non-random patterns in real networks, but also highlight(disclose/evaluate) dynamical effects of those patterns.

We impelement a null model function for bipartite graphs. A local rewiring algorithm is applied iteratively to generate a null model graph, which keep the degree distribution of the original graph and is otherwise random as much as possible. [1]

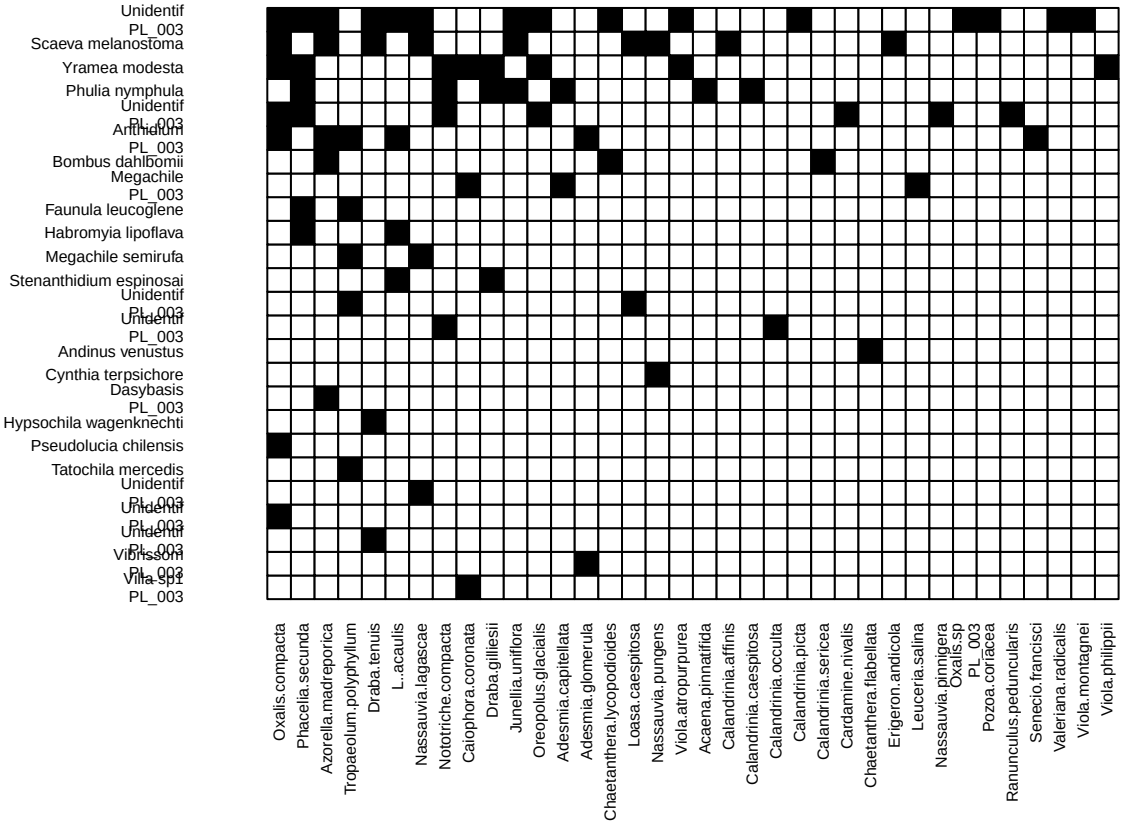
```
require(sdcn)
require(bipartite) # for plot
data(M_PL_003)
# M_PL_003 <- as.matrix(M_PL_003)
bipartite::visweb(M_PL_003)
```

```
## Warning in bipartite::visweb(M_PL_003): Object converted to matrix.
```




```
M_PL_003.rand = swaplinks(M_PL_003)
bipartite::visweb(M_PL_003.rand)
```

```
## Warning in bipartite::visweb(M_PL_003.rand): Object converted to matrix.
```



Degree Heterogeneity

Dynamics of Ecological Communities

All ecological communities consist of (are made up of) large number of species that interact in myriad (many kinds of) ways with each other. Also all the interacting species in a ecological community are effected by environments such as climate change, and haman activities.

How the abundances of species change over time is determined by the complex interactions among species, and the effects of environments on the species. Evolution processes ultimately shape the nature of species interactions and species reaction against environments, that further select survived species.

The dynamics of ecological communities is too much complex that it's necessary to simplify some aspects of them [2]. Models of dynamics of ecological communities were provided such as ODE models that describe dynamics in deterministic environments, SDE models that describe dynamics in stochastic environments, meta-community and meta-population models that highlight the spacial heterogeneity between communities and populations, individual-based models that emphasize the individuals rather than the population of species.

Different models describe different aspects of ecosystems, and can disclose different aspects of resilience (stability) of ecosystems.

We first implement the ODE model of ecological communities.

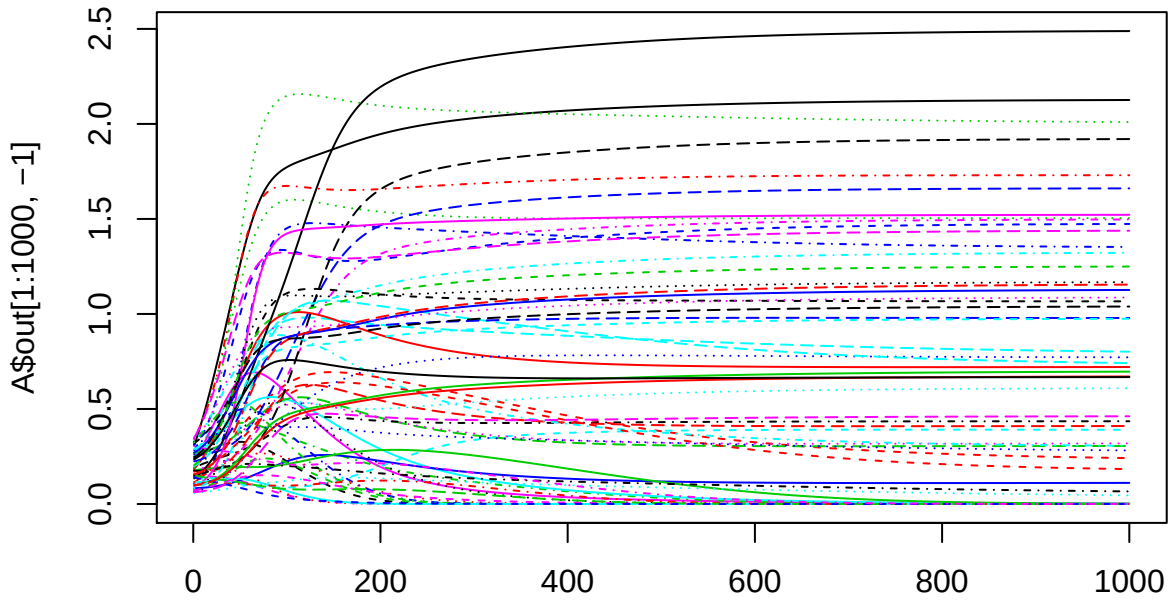
A Example of simulation of dynamics of mutualistic communities

```
library(sdcn)
library(deSolve)
library(rootSolve)
library(plyr)
graph <- as.matrix(M_PL_003)
coeff <- list(alpha.mu = 0.2, alpha.sd = 0.15, beta0.mu = 0.95, beta0.sd = 0.15, beta1.mu = 0.03, beta1
parms <- parms_lv2(graph, coeff)
init <- init_lv2(parms)
```

```
## Warning in init_lv2(parms): Initial state values is less than 0 !!
```

```
# init <- parms$r

extinct_threshold <- .Machine$double.eps * 100  # threshold of species
A = sim_ode_auto(model = model_lv2, parms = parms, init = init, steps = 1000, stepwise = 0.1, extinct_t
matplot(A$out[1:1000, -1], type = 'l', lwd = 1.)
```




```
B = sim_ode_press(model = model_lv2, parms = parms, init = init, steps = 1000, stepwise = 0.1, extinct_
```

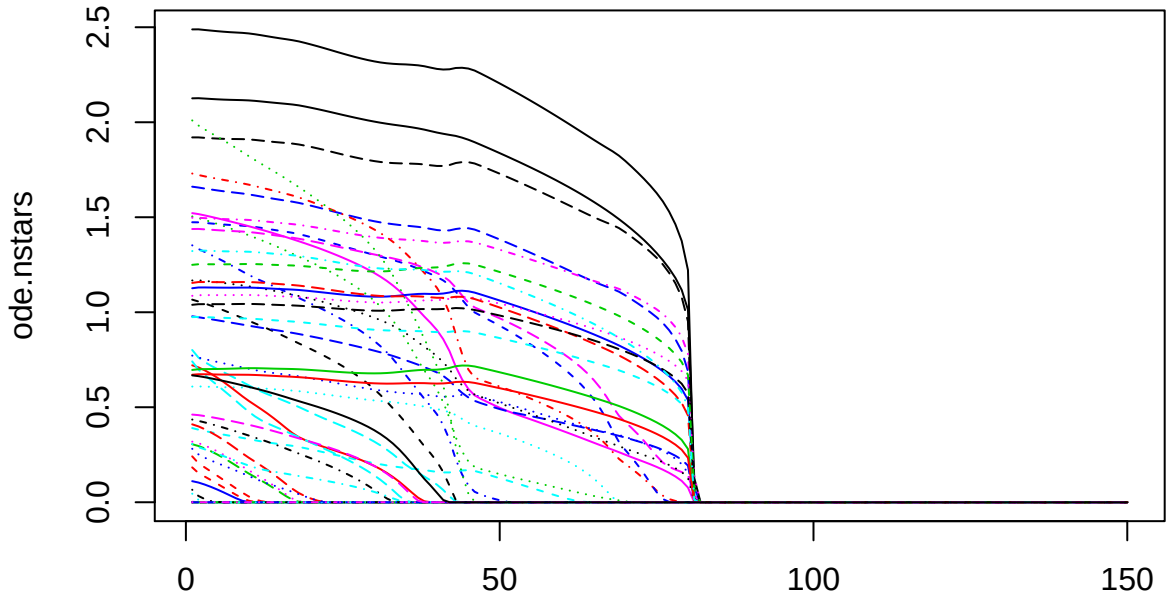
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
```

```
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
```

```
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
## [1] 91
## [1] 92
## [1] 93
## [1] 94
## [1] 95
## [1] 96
## [1] 97
## [1] 98
## [1] 99
## [1] 100
## [1] 101
## [1] 102
## [1] 103
## [1] 104
## [1] 105
## [1] 106
## [1] 107
## [1] 108
## [1] 109
## [1] 110
## [1] 111
## [1] 112
## [1] 113
## [1] 114
## [1] 115
## [1] 116
## [1] 117
## [1] 118
## [1] 119
## [1] 120
```

```
## [1] 121
## [1] 122
## [1] 123
## [1] 124
## [1] 125
## [1] 126
## [1] 127
## [1] 128
## [1] 129
## [1] 130
## [1] 131
## [1] 132
## [1] 133
## [1] 134
## [1] 135
## [1] 136
## [1] 137
## [1] 138
## [1] 139
## [1] 140
## [1] 141
## [1] 142
## [1] 143
## [1] 144
## [1] 145
## [1] 146
## [1] 147
## [1] 148
## [1] 149
## [1] 150
```

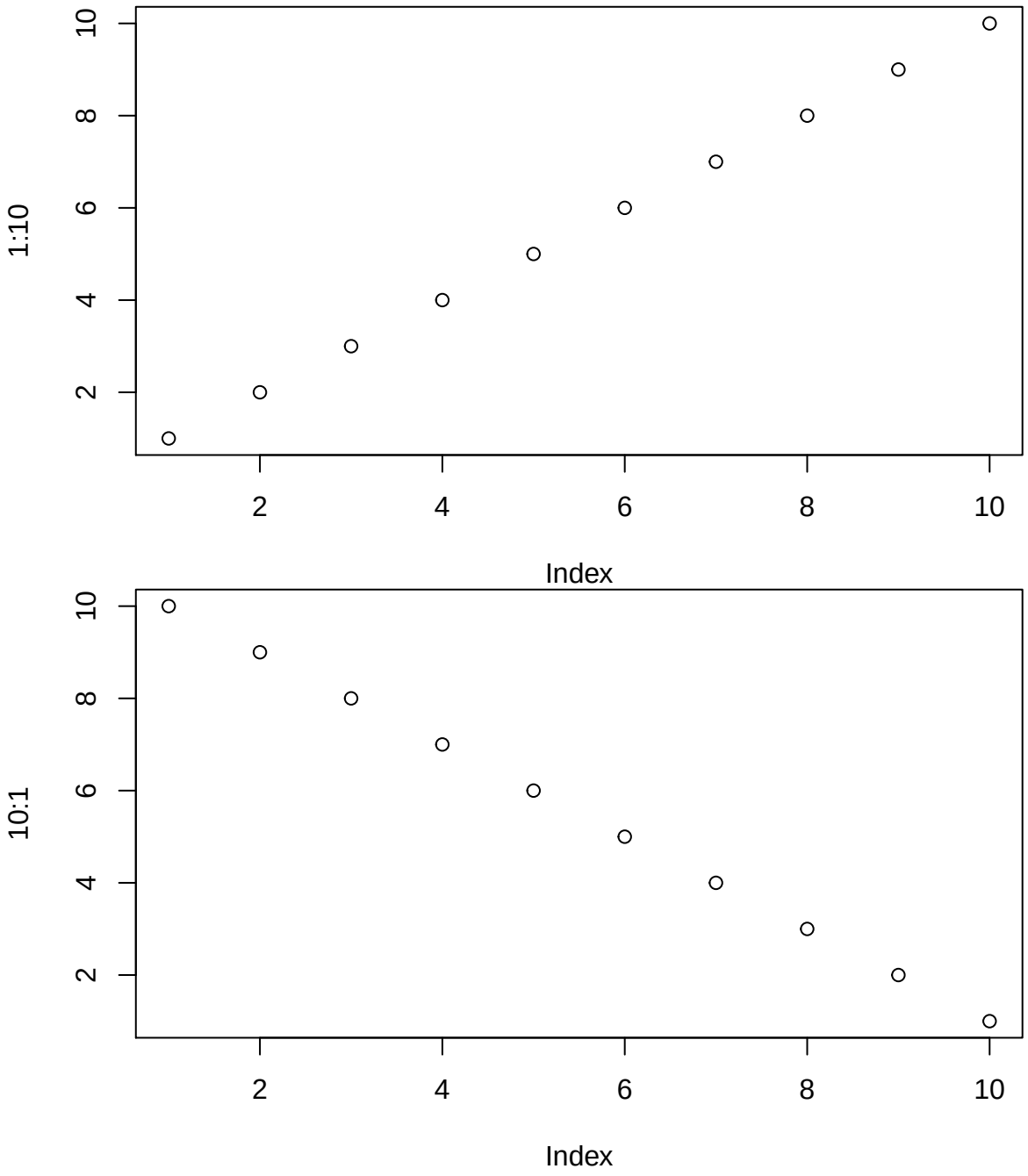
```
ode.nstars = laply(B, function(one) {
  one$nstar
})
matplot(ode.nstars, type = 'l', lwd = 1.)
```



Figures

The figure sizes have been customised so that you can easily put two images side-by-side.

```
plot(1:10)
plot(10:1)
```



You can enable figure captions by `fig_caption: yes` in YAML:

```
output:
  rmarkdown::html_vignette:
    fig_caption: yes
```

Then you can use the chunk option `fig.cap = "Your figure caption."` in **knitr**.

More Examples

You can write math expressions, e.g. $Y = X\beta + \epsilon$, footnotes[1], and tables, e.g. using `knitr::kable()`.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21	6	160	110	3	2	16	0	1	4	4
Mazda RX4 Wag	21	6	160	110	3	2	17	0	1	4	4
Datsun 710	22	4	108	93	3	2	18	1	1	4	1
Hornet 4 Drive	21	6	258	110	3	3	19	1	0	3	1
Hornet Sportabout	18	8	360	175	3	3	17	0	0	3	2
Valiant	18	6	225	105	2	3	20	1	0	3	1
Duster 360	14	8	360	245	3	3	15	0	0	3	4
Merc 240D	24	4	146	62	3	3	20	1	0	4	2
Merc 230	22	4	140	95	3	3	22	1	0	4	2
Merc 280	19	6	167	123	3	3	18	1	0	4	4

Also a quote using `>`:

> "He who gives up [code] safety for [code] speed deserves neither." (via)

1.Squartini, T. & Garlaschelli, D. *et al.* Analytical maximum-likelihood method to detect patterns in real networks. *New Journal of Physics* **13,** 083001 (2011).

2.Godfray, H. C. J. & May, R. M. *et al.* Open questions: are the dynamics of ecological communities predictable? *BMC Biology* **12,** 22 (2014).

[1] A footnote here.