

Package ‘networksis’

February 14, 2012

Version 1.4

Date January 26, 2010

Title Simulate bipartite graphs with fixed marginals through sequential importance sampling

Author Ryan Admiraal <ryan@stat.washington.edu> and Mark S. Handcock
<handcock@stat.ucla.edu>

Maintainer Ryan Admiraal <ryan@stat.washington.edu>

Description Tools to simulate bipartite networks/graphs with the degrees of the nodes fixed and specified. Part of the “statnet” suite of packages for network analysis.

Depends ergm, network

Suggests snow

License GPL-3 + file LICENSE

URL <http://statnet.org>

Repository CRAN

Date/Publication 2010-06-16 04:17:06

R topics documented:

networksis-package	2
finch	3
network.form	5
simulate.sisnetwork	6
simulate_sis	8

Index	11
--------------	-----------

networksis-package *User-defined terms used in Exponential Family Random Graph Models*

Description

The `networksis` package is a collection of functions to simulate bipartite graphs with fixed marginals. For a list of functions, type: `help(package='networksis')`.

This package is compatible with the `statnet` suite of packages, a collection of functions to plot, fit, diagnose, and simulate from random graph models.

For a complete list of the functions, use `library(help="networksis")` or read the rest of the manual.

When publishing results obtained using this package, the original authors are to be cited as:

Admiraal, Ryan and Mark S. Handcock (2008). *networksis: a package to simulate bipartite Publications graphs with fixed marginals through sequential importance sampling*, Journal of Statistical Software.

All programs derived from this package must cite it.

Author(s)

Ryan Admiraal <ryan@stat.washington.edu>,
Mark S. Handcock <handcock@stat.washington.edu>

Maintainer: Ryan Admiraal <ryan@stat.washington.edu>

References

Admiraal, Ryan and Mark S. Handcock (2008). *networksis: a package to simulate bipartite Publications graphs with fixed marginals through sequential importance sampling*, Journal of Statistical Software.

See Also

`statnet`, `network`, `ergm`

Examples

```
data(finch)
as.sociomatrix(finch)
sim <- simulate(finch, nsim=100)
sim
sim
sim
```

finch

Co-location of Darwin's Finches as a bipartite graph

Description

Data on co-location of finches noted during Charles Darwin's visit to the Galapagos Islands.

Usage

```
data(finch)
```

Details

Charles Darwin compiled this data for thirteen finch species on a visit to the Galapagos Islands. For each finch type, he recorded on which of seventeen islands that finch could be found. Sanderson (2000) argues that, in examining island biogeography, it is important to condition on the number of islands and species in order to sample from the appropriate null space, so graphs sampled from the null distribution of the observed graph should have the same marginals. Chen, Diaconis, Holmes, and Liu (2005) report the number of graphs matching the marginal constraints of Darwin's finch data to be 67,149,106,137,567,626.

Licenses and Citation

If the source of the data set does not specified otherwise, this data set is protected by the Creative Commons License <http://creativecommons.org/licenses/by-nc-nd/2.5/>. When publishing results obtained using this data set the original authors should be cited. In addition this package should be cited as noted at <http://statnetproject.org/attribution>.

References

Chen Y, Diaconis P, Holmes SP, Liu J (2005) Sequential Monte Carlo Methods for Statistical Analysis of Tables, Journal of the American Statistical Association, 100, 109, V20.

Sanderson, J.G. (2000) Testing Ecological Patterns, American Scientist, 88, p.332-339.

Besag, J. (2000) Markov Chain Monte Carlo for Statistical Inference, Working Paper #9, Center for Statistics and the Social Sciences, University of Washington <http://www.csss.washington.edu/Papers>.

See Also

network, sna

Examples

```

## Darwin's finches example ##
data(finch)
as.sociomatrix(finch)
plot(finch,vertex.col=c(rep(2,13),rep(3,17)),vertex.cex=2.5,displaylabels=FALSE)
## Consider the graph statistic  $\bar{s}^2$  (Roberts and Stone, 1990) ##
prob.vec<-rep(0,500)
s.bar.squared.vec<-rep(0,500)
for(i in 1:500)
{
  sim<-simulate(finch, nsim=1)
  ## Extract new bipartite graph ##
  new.graph<-as.matrix.network(sim)
  ## Calculate custom graph statistic ##
  s.bar.squared<-(sum((new.graph%*%t(new.graph))^2)-
sum(diag((new.graph%*%t(new.graph))^2)))/(13*12)
  s.bar.squared.vec[i]<-s.bar.squared
  ## Graph probability ##
  prob.vec[i]<-exp(sim %n% "log.prob")
}
## Plot the null distribution of  $\bar{s}^2$  ##
nbreaks<-75
w<-(1/prob.vec)/(sum(1/prob.vec))
intervals<-cut(s.bar.squared.vec, nbreaks, include.lowest=TRUE)
weights<-sapply(split(w, f=intervals), sum)
x<-seq(min(s.bar.squared.vec), max(s.bar.squared.vec), length.out=nbreaks)
plot(x,weights, type="h", xlab="", ylab="Probability Density", lwd=2)
abline(v=53.115)
## Consider graph statistics "coincidences(x)", which measure ##
## the number of finches sharing x islands ##
sim<-simulate_sis(finch~coincidences(0:17), nsim=1000)
observed.stats<-summary(finch~coincidences(0:17))
sampled.stats<-sim %n% "samplestatistics"
## Calculate 95% confidence intervals for coincidences(x) ##
library(Hmisc)
p<-exp(sim %n% "log.prob")
p<-p/sum(p)
maxs<-apply(sampled.stats,2,wtd.quantile,weights=p,probs=0.975,normwt=TRUE)
mins<-apply(sampled.stats,2,wtd.quantile,
weights=p,probs=0.025,normwt=TRUE)
means<-apply(sampled.stats,2,wtd.mean,weights=p)
## Plot distributions for coincidences(x) ##
plot(0:17, means, type="b", ylim=c(0,24.5), lwd=3, lty=3,
xlab="Number of Islands", ylab="Pairs of Finches")

for(i in 1:18)
{
  points(rep(i-1,2), c(maxs[i],mins[i]), type="l", lwd=2)
}
points(0:17, observed.stats, type="b", pch=4, lwd=3)

## Calculate p-value for coincidences(0) ##

```

```
r0<-(p%%sweep(sampled.stats,2,observed.stats,"<"))[1,]  
r1<-(p%%sweep(sampled.stats,2,observed.stats,">"))[1,]  
round(apply(cbind(r0,r1),1,min),digits=8)[1]
```

network.form

Generate a bipartite graph with specified marginals

Description

network.form generates a random bipartite graph with marginals given by user-specified row sums and column sums. These row sums and column sums must be consistent with each other (i.e. the sum of the row sums must be the same as the sum of the column sums).

Usage

```
network.form(row.sums, col.sums, nsim=1)
```

Arguments

row.sums	A vector of row sums. These row sums must be natural numbers.
col.sums	A vector of column sums. These column sums must be natural numbers.
nsim	Number of networks to be drawn from the set of all networks with the given marginals.

Details

A bipartite graph with specified row sums and column sums is randomly generated and returned.

Value

network.form bipartite [network](#) object of the randomly generated bipartite graph.

Examples

```
row.sums <- c(3, 2, 0, 1)  
col.sums <- c(2, 2, 1, 1)  
  
bipartite.graph <- network.form(row.sums, col.sums)
```

simulate.sisnetwork *Draw a Bipartite Network Using Sequential Importance Sampling*

Description

[simulate](#) is used to draw from exponential family random network models in their natural parameterizations. See [ergm](#) for more information on these models. The method `simulate.sisnetwork` draws it from graphs with the same marginals as the passed network through sequential importance sampling. That is, the degrees of the nodes are fixed and specified.

Usage

```
## S3 method for class 'sisnetwork'
simulate(object, nsim=1, ...,
         save.networks=TRUE,
         control=ergm::control.simulate(),
         verbose=FALSE)
```

Arguments

<code>object</code>	a <code>sisnetwork</code> object. This should be a list with components <code>row</code> and <code>col</code> to specify the row and column degrees. These are the degrees of the type 1 and type 2 nodes, respectively.
<code>nsim</code>	Number of networks to be randomly drawn from the set of all networks.
<code>control</code>	A list of control parameters for algorithm tuning. Constructed using control.simulate .
<code>save.networks</code>	If this is <code>TRUE</code> , the sampled networks are returned. Otherwise only the last network is returned.
<code>verbose</code>	If this is <code>TRUE</code> , we will print out more information as we run the program, including (currently) some goodness of fit statistics.
<code>...</code>	further arguments passed to or used by methods.

Details

A sample of networks is randomly drawn from the space of networks with the same degrees for each node. More information can be found by looking at the documentation of [ergm](#).

Value

[simulate](#) returns an object of class `network.series` that is a list consisting of the following elements:

<code>log.prob</code>	The vector of the logarithm of the probability of being sampled.
<code>log.graphspace.size</code>	The logarithm of the mean estimate of the number of graphs in the graph space.

```
log.graphspace.SE
    The logarithm of the standard error of the mean estimate of the number of graphs
    in the graph space.
log.graphspace.size.lne
    The logarithm of the lognormal-based estimate of the number of graphs in the
    graph space.
log.graphspace.SE.lne
    The logarithm of the standard error of the lognormal-based estimate of the num-
    ber of graphs in the graph space.
```

In the case of a single network sampled, only the network is returned (as a network object), and the additional information is returned as attributes of the network.

See Also

ergm, network

Examples

```
bipartite.graph<-scan()
1 1 0 0
0 0 1 1
1 1 1 0

bipartite.graph<-matrix(bipartite.graph, nrow=3, byrow=TRUE)
example.net<-network(bipartite.graph)

## Specify which set each node belongs to ##
example.net %v% "set" <- c(rep(1,3),rep(2,4))

## Simulate 1000 graphs with the same ##
## marginals as 'example.net' ##
sim<-simulate(example.net, nsim=1000)

## Estimated graph space size and SE ##
exp(sim %n% "log.graphspace.size")
exp(sim %n% "log.graphspace.SE")

## Darwin's finches example ##
data(finch)
sim<-simulate(finch, nsim=1000)

## Calculate importance weights from the graph probabilities ##
importance.weights<-1/exp(sim %n% "log.prob")
hist(importance.weights,breaks=75, xlab="Inverse Graph Probability",main="")

prob.vec<-rep(0,500)
s.bar.squared.vec<-rep(0,500)
for(i in 1:500)
{
  sim<-simulate(finch, nsim=1)
```

```

## Extract new bipartite graph ##
new.graph<-as.matrix.network(sim)

## Calculate custom graph statistic ##
s.bar.squared<-(sum((new.graph%*%t(new.graph))^2)-
sum(diag((new.graph%*%t(new.graph))^2)))/(13*12)
s.bar.squared.vec[i]<-s.bar.squared

## Graph probability ##
prob.vec[i]<-exp(sim %n% "log.prob")
}

```

simulate_sis	<i>Compute statistics from Bipartite Networks via Sequential Importance Sampling</i>
--------------	--

Description

The function `simulate_sis` draws bipartite graphs from the space of graphs with the same marginals as the passed network using sequential importance sampling. That is, the degrees of the nodes are fixed and specified. The formula specifies a set of network statistics that are evaluated on the sampled networks. See [ergm-terms](#) for more information on the available statistics.

Usage

```
simulate_sis(formula, nsim=1, ...,
             control=ergm::control.simulate.formula(),
             verbose=FALSE)
```

Arguments

formula	formula; an R formula object, of the form $y \sim \langle \text{model terms} \rangle$, where y is a bipartite network object or a matrix that can be coerced to a bipartite network object. For the details on the possible $\langle \text{model terms} \rangle$, see ergm-terms . To create a network object in R, use the <code>network()</code> function, then add nodal attributes to it using the <code>%v%</code> operator if necessary.
nsim	Number of networks to be randomly drawn from the set of all networks.
control	A list of control parameters for algorithm tuning. Constructed using control.simulate.formula .
verbose	If this is TRUE, we will print out more information as we run the program, including (currently) some goodness of fit statistics.
...	further arguments passed to or used by methods.

Details

A sample of networks is randomly drawn from the space of networks with the same degrees for each node.

More information can be found by looking at the documentation of [ergm](#).

Value

Returns a single bipartite `network` object. In addition the object has network attributes (using `%n%`):

`samplestatistics`

The $n \times p$ matrix of the graph statistics produced by the sampled graphs, where n is the number of simulated graphs and p is the number of different graph statistics under consideration.

`log.prob` The vector of the logarithm of the probability of being sampled.

`log.graphspace.size`

The logarithm of the mean estimate of the number of graphs in the graph space.

`log.graphspace.SE`

The logarithm of the standard error of the mean estimate of the number of graphs in the graph space.

`log.graphspace.size.lne`

The logarithm of the lognormal-based estimate of the number of graphs in the graph space.

`log.graphspace.SE.lne`

The logarithm of the standard error of the lognormal-based estimate of the number of graphs in the graph space.

See Also

`ergm`, `network`

Examples

```
data(finch)

## Simulate graphs and record the networks statistics ##
## specified by 'coincidences(x)' ##
sim<-simulate_sis(finch~coincidences(0:17), nsim=10000)

observed.stats<-summary(finch~coincidences(0:17))
sampled.stats<-sim %n% "samplestatistics"

## Calculate 95% confidence intervals for the network statistics ##
library(Hmisc)
p<-exp(sim %n% "log.prob")
p<-p/sum(p)
maxs<-apply(sampled.stats,2,wtd.quantile,
weights=p,probs=0.975,normwt=TRUE)
mins<-apply(sampled.stats,2,wtd.quantile,
weights=p,probs=0.025,normwt=TRUE)
means<-apply(sampled.stats,2,wtd.mean,weights=p)

## Plot the means and CIs for the null distributions of the ##
## statistics. Also plot the observed statistics ##
plot(0:17, means, type="b", ylim=c(0,24.5), lwd=3, lty=3,
xlab="Number of Islands", ylab="Pairs of Finches")
for(i in 1:18)
```

```
{
  points(rep(i-1,2), c(maxs[i],mins[i]), type="l", lwd=2)
}
points(0:17, observed.stats, type="b", pch=4, lwd=3)

## Calculate the p-value for 'coincidences(0)' ##
r0<-(p%%sweep(sampled.stats,2,observed.stats,"<"))[1,]
r1<-(p%%sweep(sampled.stats,2,observed.stats,">"))[1,]
round(apply(cbind(r0,r1),1,min),digits=8)[1]
```

Index

*Topic **datasets**

finch, 3

*Topic **models**

network.form, 5

networksis-package, 2

simulate.sisnetwork, 6

simulate_sis, 8

*Topic **package**

networksis-package, 2

control.simulate, 6

control.simulate.formula, 8

ergm, 6, 8

ergm-terms, 8

finch, 3

formula, 8

network, 5, 8, 9

network.form, 5

networksis, 2

networksis (networksis-package), 2

networksis-package, 2

simulate, 6

simulate.sisnetwork, 6

simulate_sis, 8