Package ‘ergm.ego’

May 26, 2022

Version 1.0.1

Date 2022-05-26

Title Fit, Simulate and Diagnose Exponential-Family Random Graph Models to Egocentrically Sampled Network Data

Depends R (>= 2.10),
ergm (>= 4.2.0),
egor,
network (>= 1.17.1)

LinkingTo ergm

Imports statnet.common (>= 4.5.0),
RColorBrewer (>= 1.1.2),
purrr (>= 0.3.2),
tibble (>= 2.1.1),
dplyr,
survey,
stats,
methods

Suggests testthat (>= 2.1.1),
covr (>= 3.2.1)

Description Utilities for managing egocentrically sampled network data and a wrapper around the 'ergm' package to facilitate ERGM inference and simulation from such data. See Krivitsky and Morris (2017) <doi:10.1214/16-AOAS1010>.

License GPL-3 + file LICENSE

URL `https://statnet.org`

BugReports `https://github.com/statnet/ergm.ego/issues`

RoxygenNote 7.2.0

Roxygen list(markdown = TRUE)

Encoding UTF-8

LazyData true

Config/testthat/parallel true

Config/testthat/edition 3

R topics documented:

`*.svystat` *A scalar multiplication method for* `svystat`

Description

Multiply the values of survey statistics by a specified vector elementwise, adjusting the variance.

Usage

```
## S3 method for class 'svystat'
x * y
```

Arguments

`x`	an object of class `[svystat][survey::svymean]`.
`y`	a numeric vector equal in length to `x`; shorter vectors will be recycled.

Value

a `[svystat][survey::svymean]` object with the updated statistics and variance-covariance matrix.

Examples

```
library(survey)
data(api)
# From example(svymean):
dclus1<-svydesign(id=~dnum, weights=~pw, data=apiclus1, fpc=~fpc)

(m1 <- svymean(~api99, dclus1))
(v1 <- vcov(m1))

# Scale the suvery stat object by a factor of two:
(m2 <- m1 * 2)
(v2 <- vcov(m2))
```

`as.egor.egodata` *Convert (deprecated)* `egodata` *Objects to* `egor` *Objects*

Description

Convert (deprecated) `egodata` Objects to `egor` Objects

Usage

```
## S3 method for class 'egodata'
as.egor(x, ...)

as_egor.egodata(x, ...)
```

Arguments

`x`	a `egodata` object
`...`	additional arguments, currently unused.

Value

An `egor` object.

Author(s)

Pavel N. Krivitsky

as.egor.network — *Construct an Egocentric View of a network Object*

Description

Given a network object, construct an egor object representing a census of all the actors in the network. Used mainly for testing.

Usage

```
## S3 method for class 'network'
as.egor(x, special.cols = c("na"), ...)
```

Arguments

x	A network object.
special.cols	Vertex attributes that should not be copied to the egos and alters tables. Defaults to attributes special to the network objects.
...	Additional arguments, currently unused.

Value

An egor object.

Author(s)

Pavel N. Krivitsky

See Also

template_network, which performs the inverse operation (though drops the ties).

Examples

```
# See example(ergm.ego) and example(template_network).
```

control.ergm.ego *Control parameters for ergm.ego.*

Description

Constructs and checks the list of control parameters for estimation by ergm.ego.

Usage

```
control.ergm.ego(
  ppopsize = c("auto", "samp", "pop"),
  ppopsize.mul = 1,
  ppop.wt = c("round", "sample"),
  stats.wt = c("data", "ppop"),
  stats.est = c("survey", "asymptotic", "bootstrap", "jackknife", "naive"),
  boot.R = 10000,
  ignore.max.alters = TRUE,
  ergm = control.ergm(),
  ...
)
```

Arguments

ppopsize, ppopsize.mul

Parameters to determine the size $|N'|$ of the pseudopopulation network. ppopsize can be

"auto" If the popsize ($|N|$) argument is specified and is different from 1, as if "pop"; otherwise, as "samp".

"samp" set $|N'|$ based on the sample size: $|N'| = |S| \times$ popsize.mul

"pop" set $|N'|$ based on the population size: $|N'| = |N| \times$ popsize.mul

a number set $|N'|$ directly (popsize.mul ignored)

a network object use the specified network as the pseudo-population network directly; use at your own risk

a data frame use the specified data frame as the pseudo-population; use at your own risk

The default is to use the same pseudopopulation size as the sample size, but, particularly if there are sampling weights in the data, it should be bigger.

Note that depending on ppop.wt, this may only be an approximate target specification, with the actual constructed pseudopopulation network being slightly bigger or smaller.

ppop.wt

Because each ego must be represented in the pseuodopopulation network an integral number of times, if the sample is weighted (or the target $|N'|$ calculated from ppopsize and ppopsize.mul is not a multiple of the sample size), it may not be possible, for a finite $|N'|$ to represent each ego exactly according to its relative weight, and ppop.wt controls how the fractional egos are allocated:

"round" (default) Rather than treating `ppopsize` as a hard setting, calculate $|N'|w_i/w.$ for each ego i and round it to the nearest integer. Then, the $|N'|$ actually used will be the sum of these rounded freqencies.

"sample" Resample in proportion to w_i.

`stats.wt` Weight assigned to each ego's contribution to the ERGM's sufficient statistic:

"data" (default) Use weights $|N'|w_i/w.$ for each ego i as in the data.

"ppop" Use weights ultimately used in the pseudopopulation network.

`stats.est, boot.R`

Method to be used to estimate the ERGM's sufficient statistics and their variance:

"survey" Variance estimator returned by `survey::svymean()`, appropriate to the design of the dataset.

"asymptotic" Delta method, as derived by Krivitsky and Morris (2017), assuming the ego weights are sampled alongside the egos.

(default) Delta method, as derived by Krivitsky and Morris (2017), assuming the ego weights are sampled alongside the egos.

"bootstrap" Nonparametric bootstrap with bias correction, resampling egos, using `R` replications.

"jackknife" Jackknife with bias correction.

"naive" "Naive" estimator, assuming that weights are fixed.

`ignore.max.alters`

if `TRUE`, ignores any constraints on the number of nominations. Used to be `FALSE`, now `TRUE` in light of the findings of Krivitsky et. al (2020).

`ergm` Control parameters for the `ergm()` call to fit the model, constructed by `control.ergm()`.

`...` Not used at this time.

Value

A list with arguments as components.

Author(s)

Pavel N. Krivitsky

References

Pavel N. Krivitsky and Martina Morris (2017). "Inference for social network models from egocentrically sampled data, with application to understanding persistent racial disparities in HIV prevalence in the US." *Annals of Applied Statistics*, 11(1): 427–455. doi:10.1214/16AOAS1010

Pavel N. Krivitsky, Martina Morris, and Michał Bojanowski (2019). "Inference for Exponential-Family Random Graph Models from Egocentrically-Sampled Data with Alter–Alter Relations." NIASRA Working Paper 08-19. `https://www.uow.edu.au/niasra/publications/`

Pavel N. Krivitsky, Michał Bojanowski, and Martina Morris (2020). "Impact of survey design on estimation of exponential-family random graph models from egocentrically-sampled data." *Social Networks*, to appear. doi:10.1016/j.socnet.2020.10.001

Pavel N. Krivitsky, Mark S. Handcock, and Martina Morris (2011). "Adjusting for Network Size and Composition Effects in Exponential-Family Random Graph Models." *Statistical Methodology*, 8(4): 319–339. doi:10.1016/j.stamet.2011.01.005

See Also

`control.ergm()`

`control.simulate.ergm.ego` *Control parameters for* `simulate.ergm.ego`*.*

Description

Constructs and checks the list of control parameters for simulation by `simulate.ergm.ego`.

Usage

```
control.simulate.ergm.ego(
  ppop.wt = c("round", "sample"),
  SAN = control.san(),
  simulate = control.simulate(),
  ...
)
```

Arguments

`ppop.wt` Because each ego must be represented in the pseuodopopulation network an integral number of times, if the sample is weighted (or the target $|N'|$ calculated from `ppopsize` and `ppopsize.mul` is not a multiple of the sample size), it may not be possible, for a finite $|N'|$ to represent each ego exactly according to its relative weight, and `ppop.wt` controls how the fractional egos are allocated:

"round" (default) Rather than treating `ppopsize` as a hard setting, calculate $|N'|w_i/w.$ for each ego i and round it to the nearest integer. Then, the $|N'|$ actually used will be the sum of these rounded freqencies.

"sample" Resample in proportion to w_i.

`SAN` A list of control parameters for `san` constructed by `control.ergm`, called to construct a pseudopopulation network consistent with the data.

`simulate` A list of control parameters for `simulate.formula` constructed by `control.simulate`, called to simulate from the model fit.

`...` Not used at this time.

Value

A list with arguments as components.

Author(s)

Pavel N. Krivitsky

See Also

control.simulate, control.san

`degreedist.egor` *Plotting the degree distribution of an egocentric dataset*

Description

A `degreedist()` method for `egodata` objects: plot a histogram of the degree distribution of actors in the egocentric dataset, optionally broken down by group and/or compared with a Bernoulli graph.

Usage

```
## S3 method for class 'egor'
degreedist(
  object,
  freq = FALSE,
  prob = !freq,
  by = NULL,
  brgmod = FALSE,
  main = NULL,
  plot = brgmod,
  weight = TRUE,
  ...
)
```

Arguments

`object` A `egor` object.

`freq, prob` Whether to plot the raw frequencies or the conditional proportions of the degree values. Defaults to the latter.

`by` A character vector giving the name of a vertex attribute; if given, plots the frequences broken down by that attribute.

`brgmod` Plot the range of predicted frequencies/probabilities according to a Bernoulli graph having the same expected density as the observed.

`main` Main title of the plot.

`plot` Whether to plot the histogram; defaults to the same value as`brgmod`, i.e., `FALSE`.

`weight` Whether sampling weights should be incorporated into the calculation (`TRUE`, the default) or ignored (`FALSE`).

`...` Additional arguments to `simulate.ergm.ego()`.

Value

Returns either a vector of degree frequencies/proportions if `by=NULL` or a matrix with a row for each category if not. If `plot==TRUE` returns invisibly.

See Also

`degreedist`, `summary`

Examples

```
data(faux.mesa.high)
fmh.ego <- as.egor(faux.mesa.high)

degreedist(fmh.ego,by="Grade",brgmod=TRUE)
# Compare:
degreedist(faux.mesa.high)
```

`ergm.ego` *Inference for Exponential-Family Random Graph Models based on Egocentrically Sampled Data*

Description

A wrapper around the `ergm` to fit an ERGM to an `egor`.

Usage

```
ergm.ego(
  formula,
  popsize = 1,
  offset.coef = NULL,
  constraints = ~.,
  ...,
  control = control.ergm.ego(),
  na.action = na.fail,
  na.rm = FALSE,
  do.fit = TRUE
)
```

Arguments

`formula` An `formula` object, of the form `e ~ <model terms>`, where `e` is a `egor` object. See `ergm` for details and examples.

For a list of currently implemented egocentric terms for the RHS, see `ergm.ego-terms`.

`popsize` The size $|N|$ of the finite population network from which the egocentric sample was taken; only affects the shift in the coefficients of the terms modeling the overall propensity to have ties. Setting it to 1 (the default) essentially uses the $-\log|N'|$ offset on the edges term. Passing 0 disables network size adjustment and uses the egocentric sample size; passing `I(N)` uses the specified size `N` (though can be overridden by the ppop `control.ergm.ego()` option) and disables network size adjustment.

`offset.coef` A vector of coefficients for the offset terms.

`constraints` A one-sided formula `formula` giving the sample space constraints. See `ergm` for details and examples.

`...` Additional arguments passed to `ergm`.

`control` A `control.ergm.ego` control list.

`na.action` How to handle missing actor attributes in egos or alters, when the terms need them for models that scale.

`na.rm` How to handle missing actor attributes in egos or alters, when the terms need them for models that do not scale.

`do.fit` Whether to actually call `ergm`

Value

An object of class `ergm.ego` inheriting from `ergm`, with the following additional or overridden elements:

`"v"` Variance-covariance matrix of the estimate of the sufficient statistics

`"m"` Estimate of the sufficient statistics

`"egor"` The `egor` object passed

`"popsize"` Population network size used

`"ppopsize"` Pseudopopulation size used, see `control.ergm.ego`

`"coef"` The coefficients, along with the network size adjustment `netsize.adj` coefficient.

`"covar"` Pseudo-MLE estimate of the variance-covariance matrix of the parameter estimates under repeated egocentric sampling

`"ergm.covar"` The variance-covariance matrix of parameter estimates under the ERGM superpopulation process (without incorporating sampling).

`"DtDe"` Estimated Jacobian of the expectation of the sufficient statistics with respect to the model parameters

Author(s)

Pavel N. Krivitsky

References

Pavel N. Krivitsky and Martina Morris (2017). "Inference for social network models from egocentrically sampled data, with application to understanding persistent racial disparities in HIV prevalence in the US." *Annals of Applied Statistics*, 11(1): 427–455. doi:10.1214/16AOAS1010

Pavel N. Krivitsky, Martina Morris, and Michał Bojanowski (2019). "Inference for Exponential-Family Random Graph Models from Egocentrically-Sampled Data with Alter–Alter Relations." NIASRA Working Paper 08-19. `https://www.uow.edu.au/niasra/publications/`

Pavel N. Krivitsky, Michał Bojanowski, and Martina Morris (2020). "Impact of survey design on estimation of exponential-family random graph models from egocentrically-sampled data." *Social Networks*, to appear. doi:10.1016/j.socnet.2020.10.001

Pavel N. Krivitsky, Mark S. Handcock, and Martina Morris (2011). "Adjusting for Network Size and Composition Effects in Exponential-Family Random Graph Models." *Statistical Methodology*, 8(4): 319–339. doi:10.1016/j.stamet.2011.01.005

See Also

`ergm()`

Examples

```
data(faux.mesa.high)
fmh.ego <- as.egor(faux.mesa.high)

head(fmh.ego)

egofit <- ergm.ego(fmh.ego~edges+degree(0:3)+nodefactor("Race")+nodematch("Race")
                   +nodefactor("Sex")+nodematch("Sex")+absdiff("Grade")+gwesp(0,fix=TRUE),
                             popsize=network.size(faux.mesa.high))

# Run convergence diagnostics
mcmc.diagnostics(egofit)

# Estimates and standard errors
summary(egofit)
```

`ergm.ego-terms` `ergm` *Terms Implemented for* `egor`

Description

This page describes the `ergm` terms (and hence network statistics) for which inference based on egocentrically sampled data is implemented in `ergm.ego` package. Other packages may add their own terms. These functions should not be called by the end-user.

Details

The current recommendation for any package implementing additional egocentric calculator terms is to create a help file with a name or alias `ergm.ego-terms`, so that `help("ergm.ego-terms")` will list egocentric ERGM terms available from all loaded packages.

Currently implemented egocentric statistics

For each of these, please see their respective package's `ergm-terms` help for meaning and parameters. The simplest way to do this is usually via `? TERM`.

Special-purpose terms: netsize.adj(edges=+1, mutual=0, transitiveties=0, cyclicalties=0) A special-purpose term equivalent to a linear combination of `edges-ergmTerm`, `mutual-ergmTerm`, `transitiveties-ergmTerm`, and `cyclicalties-ergmTerm`, to house the network-size adjustment offset. This term is added to the model automatically and should not be used in the model formula directly.

ergm:

- `offset`
- `edges`
- `nodecov`
- `nodefactor`
- `nodematch`
- `nodemix`
- `absdiff`
- `degree`
- `degrange`
- `concurrent`
- `concurrentties`
- `degree1.5`
- `transitiveties`
- `cyclicalties`
- `esp`
- `gwesp`
- `gwdegree`
- `mm`
- `meandeg*`

tergm:

- `mean.age*`

Starred terms are *nonscaling*, in that while they can be evaluated, some inferential results and standard error calculation methods may not be applicable.

See Also

`ergm-terms`

fmhfit *Fitted ergm.ego model object*

Description

This is an object with a fitted model to faux.mesa.high data using the code shown below in the Examples section.

Format

An object of class ergm.ego.

Examples

```
## Not run:
data(faux.mesa.high)
fmh.ego <- egor::as.egor(faux.mesa.high)
fmhfit <- ergm.ego(
  fmh.ego ~ edges + degree(0:3) +
    nodefactor("Race") + nodematch("Race")
  + nodefactor("Sex")+nodematch("Sex")
  + absdiff("Grade") + gwesp(0, fix=TRUE),
  popsize = network.size(faux.mesa.high),
  control = control.ergm.ego(
    ergm = control.ergm(parallel=2)
  )
)

## End(Not run)
```

gof.ergm.ego *Conduct Goodness-of-Fit Diagnostics on a Exponential Family Random Graph Model fit to Egocentrically Sampled Data*

Description

gof.ergm.ego implements the gof method for ergm.ego fit objects.

An enhanced plotting method is also provided, giving uncertainty bars for the observed statistics as well.

Usage

```
## S3 method for class 'ergm.ego'
gof(
  object,
  ...,
  GOF = c("model", "degree", "espartners"),
  control = control.gof.ergm(),
  verbose = FALSE
)

## S3 method for class 'gof.ergm.ego'
plot(x, ..., ego.conf.level = 0.95)
```

Arguments

`object` An `ergm.ego` fit.

`...` Additional arguments. Unused by `gof.ergm.ego()`, passed to `ergm::plot.gof()` by `plot.gof.ergm.ego()`

`GOF` A string specifying the statistics whose goodness of fit is to be evaluated. Currently, only “degree”, “espartners” and “model” are implemented; see `gof` documentation for details.

`control` A list to control parameters, constructed using `control.gof.formula` or `control.gof.ergm` (which have different defaults).

`verbose` Provide verbose information on the progress of the simulation.

`x` an object returned by `gof.ergm.ego()`.

`ego.conf.level` confidence level for the observed statistic estimates as well.

Value

An object of class `gof.ergm.ego`, inheriting from `gof.ergm`.

Author(s)

Pavel N. Krivitsky

References

- David R. Hunter, Steven M. Goodreau, and Mark S. Handcock (2008). "Goodness of Fit of Social Network Models." *Journal of the American Statistical Association*, 103:481: 248–258. doi:10.1198/016214507000000446

See Also

For examples, see `ergm.ego`.

Examples

```
data(faux.mesa.high)
fmh.ego <- as.egor(faux.mesa.high)

head(fmh.ego)

egofit <- ergm.ego(fmh.ego~edges+degree(0:3)+nodefactor("Race")+nodematch("Race")
                         +nodefactor("Sex")+nodematch("Sex")+absdiff("Grade"),
                          popsize=network.size(faux.mesa.high))

# Check whether the model "converged":
(modelgof <- gof(egofit, GOF="model"))
plot(modelgof)

# Check whether the model reconstructs the degree distribution:
(deggof <- gof(egofit, GOF="degree"))
plot(deggof)
```

`mixingmatrix.egor` *Summarizing the mixing among groups in an egocentric dataset*

Description

A `mixingmatrix` method for `egor` objects, to return counts of how often a ego of each group nominates an alter of each group.

Usage

```
## S3 method for class 'egor'
mixingmatrix(object, attrname, rowprob = FALSE, weight = TRUE, ...)
```

Arguments

`object`	A `egor` object.
`attrname`	A character vector containing the name of the network attribute whose mixing matrix is wanted.
`rowprob`	Whether the counts should be normalized by row sums. That is, whether they should be proportions conditional on the ego's group.
`weight`	Whether sampling weights should be incorporated into the calculation (`TRUE`, the default) or ignored (`FALSE`).
`...`	Additional arguments, currently unused.

Value

A matrix with a row and a column for each level of `attrname`.

Note that, unlike `mixingmatrix`, what is counted are *nominations*, not ties. This means that under an egocentric census, the diagonal of `mixingmatrix.egor` will be twice that returned by `mixingmatrix` for the original undirected network.

See Also

`mixingmatrix`, `nodemix-ergmTerm`, `summary` method for egocentric data

Examples

```
data(faux.mesa.high)
fmh.ego <- as.egor(faux.mesa.high)

(mm <- mixingmatrix(faux.mesa.high,"Grade"))
(mm.ego <- mixingmatrix(fmh.ego,"Grade"))
```

nodal_attributes-API *Helper functions for specifying nodal attribute levels*

Description

These functions are meant to be used in EgoStat and other implementations to provide the user with a way to extract nodal attributes and select their levels in standardized and flexible ways. They are intended to parallel ergm::nodal_attributes-API of ergm package.

ergm.ego_get_vattr extracts and processes the specified nodal attribute vector. It is strongly recommended that check.ErgmTerm()'s corresponding vartype="function,formula,character" (using the ERGM_VATTR_SPEC constant).

ergm.ego_attr_levels filters the levels of the attribute. It is strongly recommended that check.ErgmTerm()'s corresponding vartype="function,formula,character,numeric,logical,AsIs,NULL" (using the ERGM_LEVELS_SPEC constant).

Usage

```
ergm.ego_get_vattr(
  object,
  df,
  accept = "character",
  multiple = if (accept == "character") "paste" else "stop",
  ...
)

## S3 method for class 'character'
ergm.ego_get_vattr(
  object,
  df,
  accept = "character",
  multiple = if (accept == "character") "paste" else "stop",
  ...
)

## S3 method for class '`function`'
ergm.ego_get_vattr(
  object,
  df,
  accept = "character",
  multiple = if (accept == "character") "paste" else "stop",
  ...
)

## S3 method for class 'formula'
ergm.ego_get_vattr(
  object,
  df,
  accept = "character",
  multiple = if (accept == "character") "paste" else "stop",
  ...
)

ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class 'numeric'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class 'logical'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class 'AsIs'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class 'character'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class '`NULL`'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class 'matrix'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class '`function`'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

## S3 method for class 'formula'
ergm.ego_attr_levels(object, attr, egor, levels = sort(unique(attr)), ...)

COLLAPSE_SMALLEST(object, n, into)
```

Arguments

object	An argument specifying the nodal attribute to select or which levels to include.
df	Table of egos or of alters.
accept	A character vector listing permitted data types for the output. See the Details section for the specification.

`multiple` Handling of multiple attributes or matrix or data frame output. See the Details section for the specification.

`...` Additional argument to the functions of network or to the formula's environment.

`attr` A vector of length equal to the number of nodes, specifying the attribute vector.

`egor` An `egor` object.

`levels` Starting set of levels to use; defaults to the sorted list of unique attributes.

`n, into` see `ergm::COLLAPSE_SMALLEST()`.

Details

The `accept` argument is meant to allow the user to quickly check whether the output is of an *acceptable* class or mode. Typically, if a term accepts a character (i.e., categorical) attribute, it will also accept a numeric one, treating each number as a category label. For this reason, the following outputs are defined:

`"character"` Accept any mode or class (since it can beconverted to character).

`"numeric"` Accept real, integer, or logical.

`"logical"` Accept logical.

`"integer"` Accept integer or logical.

`"natural"` Accept a strictly positive integer.

`"0natural"` Accept a nonnegative integer or logical.

`"nonnegative"` Accept a nonnegative number or logical.

`"positive"` Accept a strictly positive number or logical.

`"paste"` Paste together with dot as the separator.

`"stop"` Fail with an error message.

`"matrix"` Construct and/or return a matrix whose rows correspond to vertices.

Value

`ergm.ego_get_vattr` returns a vector of length equal to the number of nodes giving the selected attribute function. It may also have an attribute `"name"`, which controls the suggested name of the attribute combination.

`ergm.ego_attr_levels` returns a vector of levels to use and their order.

Functions

- `COLLAPSE_SMALLEST`: A version of `ergm::COLLAPSE_SMALLEST()` that can handle both `network` and `egodata` objects.

Examples

```
data(florentine)
flomego <- as.egor(flomarriage)
ergm.ego_get_vattr("priorates", flomego)
ergm.ego_get_vattr(~priorates, flomego)
ergm.ego_get_vattr(c("wealth","priorates"), flomego)
ergm.ego_get_vattr(~priorates>30, flomego)
(a <- ergm.ego_get_vattr(~cut(priorates,c(-Inf,0,20,40,60,Inf),label=FALSE)-1, flomego))
ergm.ego_attr_levels(NULL, a, flomego)
ergm.ego_attr_levels(-1, a, flomego)
ergm.ego_attr_levels(1:2, a, flomego)
ergm.ego_attr_levels(I(1:2), a, flomego)
```

`predict.ergm.ego` *ERGM-based predicted tie probabilities for the pseudo-population network*

Description

ERGM-based predicted tie probabilities for the pseudo-population network

Usage

```
## S3 method for class 'ergm.ego'
predict(object, ...)
```

Arguments

`object`	model fit as returned by `ergm.ego()`
`...`	other arguments passed to/from other methods

Value

See `ergm::predict.ergm()`

`sample` *Draw random egocentric subsamples*

Description

Implementations of the `base::sample()` function for `egor::egor()` data.

Usage

```
sample(x, size, replace = FALSE, prob = NULL, ...)

## Default S3 method:
sample(x, ...)

## S3 method for class 'egor'
sample(x, size, replace = FALSE, prob = NULL, ...)
```

Arguments

`x, size, replace, prob`
: see `base::sample()`.

`...`
: extra arguments, currently unused.

Value

An `egor::egor()` object whose egos have been resampled in accordance with the arguments. Note that its `egor::ego_design()` information is overwritten in favor of the selection probabilities used in the sampling.

Note

A reimplementation of sample as a generic was necessary because `base::sample()` is not a generic and cannot take data-frame-alikes as arguments.

Examples

```
data(faux.mesa.high)
fmh.ego <- as.egor(faux.mesa.high)

# Create a tiny weighted sample:
(s3 <- sample(fmh.ego, 3, replace=TRUE, prob=1:nrow(fmh.ego$ego)))
# Resampling with prob=weights(egor) creates a self-weighted
# sample:
(sample(s3, 3, replace=TRUE, prob=weights(s3)))

# Create a large weighted sample, oversampling 12th-graders:
p <- ifelse(as_tibble(fmh.ego$ego)$Grade==12, 2, 1)
s2000 <- sample(fmh.ego, 2000, replace=TRUE, prob=p)

# Summary function adjusts for weights:
(summ.net <- summary(faux.mesa.high ~ edges + nodematch("Grade") +
                     nodefactor("Race") + gwesp(0,fix=TRUE)))
(summ.ego <- summary(s2000 ~ edges + nodematch("Grade") +
                     nodefactor("Race") + gwesp(0,fix=TRUE),
                     scaleto=network.size(faux.mesa.high)))
```

simulate.ergm.ego *Simulate from a ergm.ego fit.*

Description

A wrapper around simulate.formula to simulate networks from an ERGM fit using ergm.ego.

Usage

```
## S3 method for class 'ergm.ego'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  constraints = object$constraints,
  popsize = if (object$popsize == 1 || object$popsize == 0 || is(object$popsize,
    "AsIs")) object$ppopsize else object$popsize,
  control = control.simulate.ergm.ego(),
  output = c("network", "stats", "edgelist", "pending_update_network", "ergm_state"),
  ...,
  verbose = FALSE
)
```

Arguments

`object` An ergm.ego fit.

`nsim` Number of realizations to simulate.

`seed` Random seed.

`constraints, ...` Additional arguments passed to san and simulate.formula.

`popsize` Either network size to which to scale the model for simulation or a data.frame with at least those ego attributes required to estimate the model, to simulate over a specific set of actors.

`control` A control.simulate.ergm.ego control list.

`output` one of "network", "stats", "edgelist", "pending_update_network", or, for future compatibility, "ergm_state". See help for simulate.ergm() for explanation.

`verbose` Verbosity of output.

Value

The ouput has the same format (with the same options) as simulate.formula. If output="stats" is passed, an additional attribute, "ppopsize" is set, giving the actual size of the network reconstructed, when the pop.wt control parameter is set to "round" and "popsize" is not a multiple of the egocentric sample size or the sampling weights.

Author(s)

Pavel N. Krivitsky

References

- Pavel N. Krivitsky and Martina Morris (2017). "Inference for social network models from egocentrically sampled data, with application to understanding persistent racial disparities in HIV prevalence in the US." *Annals of Applied Statistics*, 11(1): 427–455. doi:10.1214/16-AOAS1010
- Pavel N. Krivitsky, Martina Morris, and Michał Bojanowski (2019). "Inference for Exponential-Family Random Graph Models from Egocentrically-Sampled Data with Alter–Alter Relations." NIASRA Working Paper 08-19. https://www.uow.edu.au/niasra/publications/
- Pavel N. Krivitsky, Mark S. Handcock, and Martina Morris (2011). "Adjusting for Network Size and Composition Effects in Exponential-Family Random Graph Models." *Statistical Methodology*, 8(4): 319–339. doi:10.1016/j.stamet.2011.01.005

See Also

simulate.formula, simulate.ergm

Examples

```
data(faux.mesa.high)
fmh.ego <- as.egor(faux.mesa.high)
data(fmhfit)
colMeans(egosim <- simulate(fmhfit, popsize=300,nsim=50,
                        output="stats", control=control.simulate.ergm.ego(
                        simulate=control.simulate.formula(MCMC.burnin=2e6))))
colMeans(egosim)/attr(egosim,"ppopsize")*network.size(faux.mesa.high)
summary(faux.mesa.high~edges+degree(0:3)+nodefactor("Race")+nodematch("Race")
           +nodefactor("Sex")+nodematch("Sex")+absdiff("Grade"))
```

snctrl *Statnet Control*

Description

A utility to facilitate argument completion of control lists, reexported from statnet.common.

Currently recognised control parameters

This list is updated as packages are loaded and unloaded.

See Also

statnet.common::snctrl()

summary_formula.egor *Calculation of ERGM-style summary statistics for egor objects.*

Description

Used to calculate the specified network statistics inferred from a egor object.

Usage

```
## S3 method for class 'egor'
summary_formula(object, ..., basis = NULL, individual = FALSE, scaleto = NULL)

## S3 method for class 'ergm.ego_svystat'
x * y
```

Arguments

object	An ergm-style formula with a egor object as the LHS. For a list of currently implemented egocentric terms for the RHS, see ergm.ego-terms.
...	Not used at this time.
basis	An optional egor object relative to which the statistics should be calculated.
individual	If FALSE (the default), calculate the estimated per-capita statistics, weighted according to the ego weights, then scale them up to a network of size scaleto. If TRUE, calculate each ego's individual contribution to the specified network statistics.
scaleto	Size of a hypothetical network to which to scale the statistics. Defaults to the number of egos in the dataset.
x, y	see *.svystat.

Value

If individual==FALSE, an ergm.ego_svystat object, which is a subclass of svystat—effectively a named vector of statistics. If individual==TRUE, a matrix with a row for each ego, giving that ego's contribution to the network statistic.

Functions

- *.ergm.ego_svystat: A multiplication method that takes into account which statistics are scalable.

Author(s)

Pavel N. Krivitsky

References

- Pavel N. Krivitsky and Martina Morris (2017). "Inference for social network models from egocentrically sampled data, with application to understanding persistent racial disparities in HIV prevalence in the US." *Annals of Applied Statistics*, 11(1): 427–455. doi:10.1214/16-AOAS1010
- Pavel N. Krivitsky, Mark S. Handcock, and Martina Morris (2011). "Adjusting for Network Size and Composition Effects in Exponential-Family Random Graph Models." *Statistical Methodology*, 8(4): 319–339. doi:10.1016/j.stamet.2011.01.005

See Also

summary_formula, summary_formula.ergm

Examples

```
data(faux.mesa.high)
fmh.ego <- as.egor(faux.mesa.high)
(nw.summ <- summary(faux.mesa.high~edges+degree(0:3)+nodematch("Race")+
                    nodematch("Sex")+absdiff("Grade")+nodemix("Grade")))

(ego.summ <- summary(fmh.ego~edges+degree(0:3)+nodematch("Race")+nodematch("Sex")+
                     absdiff("Grade")+nodemix("Grade"),
                     scaleto=network.size(faux.mesa.high)))

stopifnot(isTRUE(all.equal(as.vector(nw.summ),as.vector(ego.summ))))


(ego.summ2 <- summary(fmh.ego ~ edges + meandeg + degree(0:2)))
vcov(ego.summ2)

ego.summ2 * 2 # edges and degrees scales, meandeg doesn't
vcov(ego.summ2 * 2)
```

template_network *Construct an Empty "Template" Network Consistent with an Egocentric Sample*

Description

Taking a egor object, constructs a network object with no edges whose vertices have the attributes of the egos in the dataset, replicating the egos as needed, and taking into accounts their sampling weights.

Usage

```
template_network(x, N, scaling = c("round", "sample"), ...)
```

Arguments

x
: A egor object.

N
: The target number of vertices the output network should have.

scaling
: If egor contains weights or N is not a multiple of number of egos in the sample, it may not be possible, for a finite N to represent each ego exactly according to its relative weight, and scaling controls how the fractional egos are allocated:

  "round" (the default) Rather than treating N as a hard setting, calculate $Nw_i/w.$ for each ego i and round it to the nearest integer. Then, the N actually used will be the sum of these rounded freqencies.

  "sample" Resample in proportion to w_i.

...
: Additional arguments, currently unused.

Value

A network object.

Author(s)

Pavel N. Krivitsky

See Also

as.egor.network, which performs the inverse operation.

Examples

```
data(faux.mesa.high)
summary(faux.mesa.high, print.adj = FALSE)

fmh.ego <- as.egor(faux.mesa.high)

# Same actor attributes
fmh.template <- template_network(fmh.ego, N=network.size(faux.mesa.high))
summary(fmh.template, print.adj = FALSE)

# Twice the actors, same distribution
fmh2.template <- template_network(fmh.ego, N=2*network.size(faux.mesa.high))
summary(fmh2.template, print.adj = FALSE)
```

Index