Package ‘sspse’

August 26, 2018

Type Package

Version 0.6

Date 2018-08-24

Title Estimating Hidden Population Size using Respondent Driven
Sampling Data

Maintainer Mark S. Handcock <handcock@stat.ucla.edu>

Depends methods, parallel

Suggests network, testthat

Description Estimate the size of a networked population based on
respondent-driven sampling data. The package is part of the ``RDS Analyst''
suite of packages for the analysis of respondent-driven sampling data.
See Handcock, Gile and Mar (2014) <doi:10.1214/14-EJS923> and
Handcock, Gile and Mar (2015) <doi:10.1111/biom.12255>.

License GPL-3 + file LICENSE

URL http://www.hpmrg.org

Imports RDS, scam, coda

RoxygenNote 6.1.0

Encoding UTF-8

NeedsCompilation yes

Author Mark S. Handcock [aut, cre, cph],
Krista J. Gile [aut, cph]

Repository CRAN

Date/Publication 2018-08-26 17:20:02 UTC

R topics documented:

`sspse-package` *Estimating Hidden Population Size using Respondent Driven Sampling Data*

Description

An integrated set of tools to estimate the size of a networked population based on respondent-driven sampling data. The "sspse" packages is part of the "RDS Analyst" suite of packages for the analysis of respondent-driven sampling data. For a list of functions type: help(package='sspse')

Details

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

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

Gile, Krista J. and Handcock, Mark S. (2014) **sspse**: Estimating Hidden Population Size using Respondent Driven Sampling Data R package, Los Angeles, CA. Version 0.5, `http://hpmrg.org`.

All programs derived from this package must cite it. For complete citation information, use `citation(package="sspse")`.

The package can also be accessed via graphical user interface provided by the `RDS Analyst` software. `RDS Analyst` software was designed to help researchers visualize and analyze data collected via respondent-driven sampling designs. It has a broad range of estimation and visualization capabilities.

For detailed information on how to download and install the software, go to the Hard-to-Reach Population Methods Research Group website: `http://hpmrg.org`. A tutorial, support newsgroup, references and links to further resources are provided there.

Author(s)

Krista J. Gile <`gile@math.umass.edu`>,
Mark S. Handcock <`handcock@stat.ucla.edu`>

Maintainer: Mark S. Handcock <`handcock@stat.ucla.edu`>

References

Gile, Krista J. (2008) *Inference from Partially-Observed Network Data*, Ph.D. Thesis, Department of Statistics, University of Washington.

Gile, Krista J. and Handcock, Mark S. (2010) *Respondent-Driven Sampling: An Assessment of Current Methodology*, Sociological Methodology 40, 285-327.

Gile, Krista J. and Handcock, Mark S. (2014) **sspse**: Estimating Hidden Population Size using Respondent Driven Sampling Data R package, Los Angeles, CA. Version 0.5, `http://hpmrg.org`.

Handcock MS (2003). **degreenet**: Models for Skewed Count Distributions Relevant to Networks. Statnet Project, Seattle, WA. Version 1.2, `http://statnetproject.org`.

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2014) *Estimating Hidden Population Size using Respondent-Driven Sampling Data*, Electronic Journal of Statistics, 8, 1, 1491-1521

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2015) *Estimating the Size of Populations at High Risk for HIV using Respondent-Driven Sampling Data*, Biometrics.

`dsizeprior` *Prior distributions for the size of a hidden population*

Description

`dsizeprior` computes the prior distribution of the population size of a hidden population. The prior is intended to be used in Bayesian inference for the population size based on data collected by Respondent Driven Sampling, but can be used with any Bayesian method to estimate population size.

Usage

```
dsizeprior(n, type = c("beta", "nbinom", "pln", "flat", "continuous",
  "supplied"), mean.prior.size = NULL, sd.prior.size = NULL,
  mode.prior.sample.proportion = NULL,
  median.prior.sample.proportion = NULL, median.prior.size = NULL,
  mode.prior.size = NULL, quartiles.prior.size = NULL,
  effective.prior.df = 1, alpha = NULL, beta = NULL, maxN = NULL,
  log = FALSE, maxbeta = 120, maxNmax = 2e+05, supplied = list(maxN =
  maxN), verbose = TRUE)
```

Arguments

`n` count; the sample size.

`type` character; the type of parametric distribution to use for the prior on population size. The options are `"beta"` (for a Beta-type prior on the sample proportion (i.e. n/N), `"nbinom"` (Negative-Binomial), `"pln"` (Poisson-log-normal), `"flat"` (uniform), `continuous` (the continuous version of the Beta-type prior on the sample proportion). The last option is `"supplied"` which enables a numeric prior to be specified. See the argument `supplied` for the format of the information. The default `type` is `beta`.

`mean.prior.size`
scalar; A hyperparameter being the mean of the prior distribution on the population size.

`sd.prior.size` scalar; A hyperparameter being the standard deviation of the prior distribution on the population size.

`mode.prior.sample.proportion`
scalar; A hyperparameter being the mode of the prior distribution on the sample proportion n/N.

`median.prior.sample.proportion`
scalar; A hyperparameter being the median of the prior distribution on the sample proportion n/N.

`median.prior.size`
scalar; A hyperparameter being the mode of the prior distribution on the population size.

`mode.prior.size`
scalar; A hyperparameter being the mode of the prior distribution on the population size.

`quartiles.prior.size`
vector of length 2; A pair of hyperparameters being the lower and upper quartiles of the prior distribution on the population size. For example, `quartiles.prior.size=c(1000,4000)` corresponds to a prior where the lower quartile (25%) is 1000 and the upper (75%) is 4000.

`effective.prior.df`
scalar; A hyperparameter being the effective number of samples worth of information represented in the prior distribution on the population size. By default this is 1, but it can be greater (or less!) to allow for different levels of uncertainty.

`alpha` scalar; A hyperparameter being the first parameter of the Beta prior model for the sample proportion. By default this is NULL, meaning that 1 is chosen. it can be any value at least 1 to allow for different levels of uncertainty.

`beta` scalar; A hyperparameter being the second parameter of the Beta prior model for the sample proportion. By default this is NULL, meaning that 1 is chosen. it can be any value at least 1 to allow for different levels of uncertainty.

`maxN` integer; maximum possible population size. By default this is determined from an upper quantile of the prior distribution.

`log` logical; return the prior or the the logarithm of the prior.

`maxbeta` integer; maximum beta in the prior for population size. By default this is determined to ensure numerical stability.

`maxNmax` integer; maximum possible population size. By default this is determined to ensure numerical stability.

`supplied` list; If the argument `type="supplied"` then this should be a list object, typically of class `sspse`. It is primarily used to pass the posterior sample from a separate `size` call for use as the prior to this call. Essentially, it must have two components named `maxN` and `sample`. `maxN` is the maximum population envisaged and `sample` is random sample from the prior distribution.

`verbose` logical; if this is `TRUE`, the program will print out additional information, including goodness of fit statistics.

Value

`dsizeprior` returns a list consisting of the following elements:

`x` vector; vector of degrees `1:N` at which the prior PMF is computed.

`lpriorm` vector; vector of probabilities corresponding to the values in `x`.

`N` scalar; a starting value for the population size computed from the prior.

`maxN` integer; maximum possible population size. By default this is determined from an upper quantile of the prior distribution.

`mean.prior.size`
scalar; A hyperparameter being the mean of the prior distribution on the population size.

`mode.prior.size`
scalar; A hyperparameter being the mode of the prior distribution on the population size.

`effective.prior.df`
scalar; A hyperparameter being the effective number of samples worth of information represented in the prior distribution on the population size. By default this is 1, but it can be greater (or less!) to allow for different levels of uncertainty.

`mode.prior.sample.proportion`
scalar; A hyperparameter being the mode of the prior distribution on the sample proportion n/N.

`median.prior.size`
scalar; A hyperparameter being the mode of the prior distribution on the population size.

`beta` scalar; A hyperparameter being the second parameter of the Beta distribution that is a component of the prior distribution on the sample proportion n/N.

`type` character; the type of parametric distribution to use for the prior on population size. The possible values are `beta` (for a Beta prior on the sample proportion (i.e. n/N), `nbinom` (Negative-Binomial), `pln` (Poisson-log-normal), `flat` (uniform), and `continuous` (the continuous version of the Beta prior on the sample proportion. The default is `beta`.

Details on priors

The best way to specify the prior is via the hyperparameter `mode.prior.size` which specifies the mode of the prior distribution on the population size. You can alternatively specify the hyperparameter `median.prior.size` which specifies the median of the prior distribution on the population size, or `mode.prior.sample proportion` which specifies the mode of the prior distribution on the proportion of the population size in the sample.

References

Gile, Krista J. (2008) *Inference from Partially-Observed Network Data*, Ph.D. Thesis, Department of Statistics, University of Washington.

Gile, Krista J. and Handcock, Mark S. (2010) *Respondent-Driven Sampling: An Assessment of Current Methodology*, Sociological Methodology 40, 285-327.

Gile, Krista J. and Handcock, Mark S. (2014) **sspse**: Estimating Hidden Population Size using Respondent Driven Sampling Data R package, Los Angeles, CA. Version 0.5, `http://hpmrg.org`.

Handcock MS (2003). **degreenet**: Models for Skewed Count Distributions Relevant to Networks. Statnet Project, Seattle, WA. Version 1.2, `http://statnetproject.org`.

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2014) *Estimating Hidden Population Size using Respondent-Driven Sampling Data*, Electronic Journal of Statistics, 8, 1, 1491-1521

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2015) *Estimating the Size of Populations at High Risk for HIV using Respondent-Driven Sampling Data*, Biometrics.

See Also

network, statnet, degreenet

Examples

```
prior <- dsizeprior(n=100,
                  type="beta",
                  mode.prior.size=1000)
```

---

plot.sspse   *Plot Summary and Diagnostics for Population Size Estimation Model Fits*

Description

This is the plot method for class "sspse". Objects of this class encapsulate the estimate of the posterior distribution of the population size based on data collected by Respondent Driven Sampling. The approach approximates the RDS via the Sequential Sampling model of Gile (2008). As such, it is referred to as the Sequential Sampling - Population Size Estimate (SS-PSE). It uses the order of selection of the sample to provide information on the distribution of network sizes over the population members.

Usage

```
## S3 method for class 'sspse'
plot(x, xlim = NULL, data = NULL, support = 1000,
  HPD.level = 0.9, N = NULL, ylim = NULL, mcmc = FALSE, type = "both",
  main = "posterior for population size", smooth = 4, include.tree = TRUE,
  ...)
```

Arguments

x	an object of class "plot.sspse", usually, a result of a call to plot.sspse.
xlim	the (optional) x limits (x1, x2) of the plot of the posterior of the population size.
data	Optionally, the vector of degrees from the RDS in order they are recorded and as passed to posteriorsize.
support	the number of equally-spaced points to use for the support of the estimated posterior density function.
HPD.level	numeric; probability level of the highest probability density interval determined from the estimated posterior.

`N` Optionally, an estimate of the population size to mark on the plots as a reference point.

`ylim` the (optional) vertical limits (y1, y2) of the plot of the posterior of the population size. A vertical axis is the probability density scale.

`mcmc` logical; If TRUE, additionally create simple diagnostic plots for the MCMC sampled statistics produced from the fit.

`type` character; This controls the types of plots produced. If `"N"`, a density plot of the posterior for population size is produced. and the prior for population size is overlaid. If `"others"`, a density plot of the prior for population size, a density plot of the posterior for mean network size in the population, the posterior for standard deviation of the network size, and a density plot of the posterior mean network size distribution with sample histogram overlaid is produced. If `"both"`, then all plots for `"N"` and `"others"` are produced.

`main` an overall title for the posterior plot.

`smooth` the (optional) smoothing parameter for the density estimate.

`include.tree` logical; If `TRUE`, augment the reported network size by the number of recruits and one for the recruiter (if any). This reflects a more accurate value for the visibility, but is not the self-reported degree. In particular, it typically produces a positive visibility (compared to a possibility zero self-reported degree).

`...` further arguments passed to or from other methods.

Details

By default it produces a density plot of the posterior for population size and the prior for population size is overlaid. It also produces a density plot of the posterior for mean network size in the population, the posterior for standard deviation of the network size, and a density plot of the posterior mean network size distribution with sample histogram overlaid.

References

Gile, Krista J. (2008) *Inference from Partially-Observed Network Data*, Ph.D. Thesis, Department of Statistics, University of Washington.

Gile, Krista J. and Handcock, Mark S. (2010) *Respondent-Driven Sampling: An Assessment of Current Methodology*, Sociological Methodology 40, 285-327.

Gile, Krista J. and Handcock, Mark S. (2014) **sspse**: Estimating Hidden Population Size using Respondent Driven Sampling Data R package, Los Angeles, CA. Version 0.5, `http://hpmrg.org`.

Handcock MS (2003). **degreenet**: Models for Skewed Count Distributions Relevant to Networks. Statnet Project, Seattle, WA. Version 1.2, `http://statnetproject.org`.

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2014) *Estimating Hidden Population Size using Respondent-Driven Sampling Data*, Electronic Journal of Statistics, 8, 1, 1491-1521

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2015) *Estimating the Size of Populations at High Risk for HIV using Respondent-Driven Sampling Data*, Biometrics.

See Also

The model fitting function posteriorsize, plot.

Function coef will extract the matrix of coefficients with standard errors, t-statistics and p-values.

Examples

```
N0 <- 200
n <- 100
K <- 10

# Create probabilities for a Waring distribution
# with scaling parameter 3 and mean 5, but truncated at K=10.
probs <- c(0.33333333,0.19047619,0.11904762,0.07936508,0.05555556,
           0.04040404,0.03030303,0.02331002,0.01831502,0.01465201)
probs <- probs / sum(probs)

#
# Create a sample
#
set.seed(1)
pop<-sample(1:K, size=N0, replace = TRUE, prob = probs)
s<-sample(pop, size=n, replace = FALSE, prob = pop)

# Here interval=1 so that it will run faster. It should be higher in a
# real application.
out <- posteriorsize(s=s,interval=1)
plot(out, HPD.level=0.9,data=pop[s])
summary(out, HPD.level=0.9)
# Let's look at some MCMC diagnostics
plot(out, HPD.level=0.9,mcmc=TRUE)
```

---

posize_warning *Warning message for posteriorsize fit failure*

Description

posteriorsize computes the posterior distribution of the population size based on data collected by Respondent Driven Sampling. This function returns the warning message if it fails. It enables packages that call posteriorsize to use a consistent error message.

Usage

```
posize_warning()
```

Value

posize_warning returns a character string witn the warning message.

See Also

posteriorsize

`posteriorsize` *Estimating hidden population size using RDS data*

Description

`posteriorsize` computes the posterior distribution of the population size based on data collected by Respondent Driven Sampling. The approach approximates the RDS via the Sequential Sampling model of Gile (2008). As such, it is referred to as the Sequential Sampling - Population Size Estimate (SS-PSE). It uses the order of selection of the sample to provide information on the distribution of network sizes over the population members.

Usage

```
posteriorsize(s, s2 = NULL, rc = rep(FALSE, length(s2)),
  median.prior.size = NULL, interval = 10, burnin = 5000, maxN = NULL,
  K = NULL, samplesize = 1000, quartiles.prior.size = NULL,
  mean.prior.size = NULL, mode.prior.size = NULL,
  priorsizedistribution = c("beta", "flat", "nbinom", "pln", "supplied"),
  effective.prior.df = 1, sd.prior.size = NULL,
  mode.prior.sample.proportion = NULL, alpha = NULL,
  degreedistribution = c("cmp", "nbinom", "pln"), mean.prior.degree = NULL,
  sd.prior.degree = NULL, max.sd.prior.degree = 4, df.mean.prior = 1,
  df.sd.prior = 3, beta0.mean.prior = -3, beta1.mean.prior = 0,
  beta0.sd.prior = 10, beta1.sd.prior = 10, mem.optimism.prior = 1,
  df.mem.optimism.prior = 5, mem.sd.prior = 5, df.mem.sd.prior = 3,
  visibility = FALSE, type.impute = c("mode", "distribution", "median",
  "mean"), Np = 0, nk = NULL, n = NULL, n2 = length(s2),
  muproposal = 0.1, sigmaproposal = 0.15, beta0proposal = 0.2,
  beta1proposal = 0.001, memmuproposal = 0.1, memsdproposal = 0.15,
  burnintheta = 500, burninbeta = 50, parallel = 1,
  parallel.type = "PSOCK", seed = NULL, maxbeta = 120,
  supplied = list(maxN = maxN), max.coupons = NULL, recruit.time = NULL,
  include.tree = TRUE, unit.scale = FALSE, optimism = TRUE,
  reflect.time = TRUE, verbose = TRUE)
```

Arguments

`s` either a vector of integers or an `rds.data.frame` providing network size information. If a `rds.data.frame` is passed and `visibility=TRUE`, the default, then the measurement error model is to used, whereby latent visibilities are used in place of the reported network sizes as the size variable. If a vector of integers is passed these are the network sizes in sequential order of recording.

`s2` vector of integers; optionally, the vector of degrees from a second RDS, subsequent to the first RDS recorded in s. These are also in the order they are recorded.

`rc` vector of logicals; optionally, a vector of the same length as $s2$ indicating if the corresponding unit was sampled in the first RDS.

`median.prior.size`
scalar; A hyperparameter being the mode of the prior distribution on the population size.

`interval` count; the number of proposals between sampled statistics.

`burnin` count; the number of proposals before any MCMC sampling is done. It typically is set to a fairly large number.

`maxN` integer; maximum possible population size. By default this is determined from an upper quantile of the prior distribution.

`K` count; the maximum degree for an individual. This is usually calculated as `round(stats::quantile(s,0.80))`.

`samplesize` count; the number of Monte-Carlo samples to draw to compute the posterior. This is the number returned by the Metropolis-Hastings algorithm.The default is 1000.

`quartiles.prior.size`
vector of length 2; A pair of hyperparameters being the lower and upper quartiles of the prior distribution on the population size. For example, `quartiles.prior.size=c(1000,4000)` corresponds to a prior where the lower quartile (25%) is 1000 and the upper (75%) is 4000.

`mean.prior.size`
scalar; A hyperparameter being the mean of the prior distribution on the population size.

`mode.prior.size`
scalar; A hyperparameter being the mode of the prior distribution on the population size.

`priorsizedistribution`
character; the type of parametric distribution to use for the prior on population size. The options are `beta` (for a Beta prior on the sample proportion (i.e. n/N)), `flat` (uniform), `nbinom` (Negative-Binomial), and `pln` (Poisson-log-normal). The default is `beta`.

`effective.prior.df`
scalar; A hyperparameter being the effective number of samples worth of information represented in the prior distribution on the population size. By default this is 1, but it can be greater (or less!) to allow for different levels of uncertainty.

`sd.prior.size` scalar; A hyperparameter being the standard deviation of the prior distribution on the population size.

`mode.prior.sample.proportion`
scalar; A hyperparameter being the mode of the prior distribution on the sample proportion n/N.

`alpha` scalar; A hyperparameter being the first parameter of the beta prior model for the sample proportion. By default this is NULL, meaning that 1 is chosen. it can be any value at least 1 to allow for different levels of uncertainty.

`degreedistribution`
count; the parametric distribution to use for the individual network sizes (i.e., degrees). The options are `cmp`, `nbinom`, and `pln`. These correspond to the Conway-Maxwell-Poisson, Negative-Binomial, and Poisson-log-normal. The default is `cmp`.

`mean.prior.degree`
scalar; A hyper parameter being the mean degree for the prior distribution for a randomly chosen person. The prior has this mean.

`sd.prior.degree`
scalar; A hyper parameter being the standard deviation of the degree for a randomly chosen person. The prior has this standard deviation.

`max.sd.prior.degree`
scalar; The maximum allowed value of `sd.prior.degree`. If the passed or computed value is higher, it is reduced to this value. This is done for numerical stability reasons.

`df.mean.prior`
scalar; A hyper parameter being the degrees-of-freedom of the prior for the mean. This gives the equivalent sample size that would contain the same amount of information inherent in the prior.

`df.sd.prior`
scalar; A hyper parameter being the degrees-of-freedom of the prior for the standard deviation. This gives the equivalent sample size that would contain the same amount of information inherent in the prior for the standard deviation.

`beta0.mean.prior`
scalar; A hyper parameter being the mean of the beta0 parameter distribution in the model for the number of recruits.

`beta1.mean.prior`
scalar; A hyper parameter being the mean of the beta1 parameter distribution in the model for the number of recruits.

`beta0.sd.prior`
scalar; A hyper parameter being the standard deviation of the beta0 parameter distribution in the model for the number of recruits.

`beta1.sd.prior`
scalar; A hyper parameter being the standard deviation of the beta0 parameter distribution in the model for the number of recruits.

`mem.optimism.prior`
scalar; A hyper parameter being the mean of the distribution of the optimism parameter.

`df.mem.optimism.prior`
scalar; A hyper parameter being the degrees-of-freedom of the prior for the optimism parameter. This gives the equivalent sample size that would contain the same amount of information inherent in the prior.

`mem.sd.prior`
scalar; A hyper parameter being the mean of the distribution of the dispersion parameter in the visibility model.

`df.mem.sd.prior`
scalar; A hyper parameter being the degrees-of-freedom of the prior for the standard deviation of the dispersion parameter in the visibility model. This gives the equivalent sample size that would contain the same amount of information inherent in the prior for the standard deviation.

`visibility` logical; Indicate if the measurement error model is to be used, whereby latent visibilities are used in place of the reported network sizes as the unit size variable. If `TRUE` then a `rds.data.frame` need to be passed to provide the RDS information needed for the measurement error model.

`type.impute` The type of imputation to use for the summary visibilities (returned in the component `visibilities`. The imputes are based on the posterior draws of the visibilities. It can be of type `distribution`,`mode`,`median`, or `mean` with `mode` the default, being the posterior mode of the visibility for that person.

`Np` integer; The overall degree distribution is a mixture of the `Np` rates for `1:Np` and a parametric degree distribution model truncated below `Np`. Thus the model fits the proportions of the population with degree `1:Np` each with a separate parameter. This should adjust for an lack-of-fit of the parametric degree distribution model at lower degrees, although it also changes the model away from the parametric degree distribution model.

`nk` vector; the vector of counts for the number of people in the sample with degree k. This is usually computed from s automatically as `tabulate(s,nbins=K)` and not usually specified by the user.

`n` integer; the number of people in the sample. This is usually computed from s automatically and not usually specified by the user.

`n2` integer; If $s2$ is specified, this is the number of people in the second sample. This is usually computed from s automatically and not usually specified by the user.

`muproposal` scalar; The standard deviation of the proposal distribution for the mean degree.

`sigmaproposal` scalar; The standard deviation of the proposal distribution for the standard deviation of the degree.

`beta0proposal` scalar; The standard deviation of the proposal distribution for the beta0 parameter of the recruit model.

`beta1proposal` scalar; The standard deviation of the proposal distribution for the beta1 parameter of the recruit model.

`memmuproposal` scalar; The standard deviation of the proposal distribution for the log of the optimism parameter (that is, gamma).

`memsdproposal` scalar; The standard deviation of the proposal distribution for the log of the s.d. in the optimism model.

`burnintheta` count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters (θ) before any MCMC sampling is done. It typically is set to a modestly large number.

`burninbeta` count; the number of proposals in the Metropolis-Hastings sub-step for the visibility distribution parameters (β) before any MCMC sampling is done. It typically is set to a modestly large number.

`parallel` count; the number of parallel processes to run for the Monte-Carlo sample. This uses MPI or PSOCK. The default is 1, that is not to use parallel processing.

`parallel.type` The type of parallel processing to use. The options are "PSOCK" or "MPI". This requires the corresponding type to be installed. The default is "PSOCK".

`seed` integer; random number integer seed. Defaults to `NULL` to use whatever the state of the random number generator is at the time of the call.

`maxbeta` scalar; The maximum allowed value of the `beta` parameter. If the implied or computed value is higher, it is reduced to this value. This is done for numerical stability reasons.

`supplied` list; If supplied, is a list with components `maxN` and `sample`. In this case `supplied` is a matrix with a column named `N` being a sample from a prior distribution for the population size. The value `maxN` specifies the maximum value of the population size, a priori.

`max.coupons` The number of recruitment coupons distributed to each enrolled subject (i.e. the maximum number of recruitees for any subject). By default it is taken by the attribute or data, else the maximum recorded number of coupons.

`recruit.time` vector; An optional value for the data/time that the person was interviewed. It needs to resolve as a numeric vector with number of elements the number of rows of the data with non-missing values of the network variable. If it is a character name of a variable in the data then that variable is used. If it is NULL then the sequence number of the recruit in the data is used. If it is NA then the recruitment is not used in the model. Otherwise, the recruitment time is used in the model to better predict the visibility of the person.

`include.tree` logical; If `TRUE`, augment the reported network size by the number of recruits and one for the recruiter (if any). This reflects a more accurate value for the visibility, but is not the self-reported degree. In particular, it typically produces a positive visibility (compared to a possibility zero self-reported degree).

`unit.scale` numeric; If not `NULL` it sets the numeric value of the scale parameter of the distribution of the unit sizes. For the negative binomial, it is the multiplier on the variance of the negative binomial compared to a Poisson (via the Poisson-Gamma mixture representation). Sometimes the scale is unnaturally large (e.g. 40) so this give the option of fixing it (rather than using the MLE of it). The model is fit with the parameter fixed at this passed value.

`optimism` logical; If `TRUE` then add a term to the model allowing the (proportional) inflation of the self-reported degrees relative to the unit sizes.

`reflect.time` logical; If `FALSE` then the `recruit.time` is the time before the end of the study (instead of the time since the survey started or chronological time).

`verbose` logical; if this is `TRUE`, the program will print out additional information, including goodness of fit statistics.

Value

`posteriorsize` returns a list consisting of the following elements:

`pop` vector; The final posterior draw for the degrees of the population. The first n are the sample in sequence and the reminder are non-sequenced.

`K` count; the maximum degree for an individual. This is usually calculated as twice the maximum observed degree.

`n` count; the sample size.

`samplesize` count; the number of Monte-Carlo samples to draw to compute the posterior. This is the number returned by the Metropolis-Hastings algorithm.The default is 1000.

`burnin` count; the number of proposals before any MCMC sampling is done. It typically is set to a fairly large number.

`interval` count; the number of proposals between sampled statistics.

`mu` scalar; The hyper parameter `mean.prior.degree` being the mean degree for the prior distribution for a randomly chosen person. The prior has this mean.

`sigma` scalar; The hyper parameter `sd.prior.degree` being the standard deviation of the degree for a randomly chosen person. The prior has this standard deviation.

`df.mean.prior` scalar; A hyper parameter being the degrees-of-freedom of the prior for the mean. This gives the equivalent sample size that would contain the same amount of information inherent in the prior.

`df.sd.prior` scalar; A hyper parameter being the degrees-of-freedom of the prior for the standard deviation. This gives the equivalent sample size that would contain the same amount of information inherent in the prior for the standard deviation.

`Np` integer; The overall degree distribution is a mixture of the `1:Np` rates and a parametric degree distribution model truncated below Np. Thus the model fits the proportions of the population with degree `1:Np` each with a separate parameter. This should adjust for an lack-of-fit of the parametric degree distribution model at lower degrees, although it also changes the model away from the parametric degree distribution model.

`muproposal` scalar; The standard deviation of the proposal distribution for the mean degree.

`sigmaproposal` scalar; The standard deviation of the proposal distribution for the standard deviation of the degree.

`N` vector of length 5; summary statistics for the posterior population size.

MAP maximum aposteriori value of N

Mean AP mean aposteriori value of N

Median AP median aposteriori value of N

P025 the 2.5th percentile of the (posterior) distribution for the N. That is, the lower point on a 95% probability interval.

P975 the 97.5th percentile of the (posterior) distribution for the N. That is, the upper point on a 95% probability interval.

`maxN` integer; maximum possible population size. By default this is determined from an upper quantile of the prior distribution.

`sample` matrix of dimension `samplesize`× `10` matrix of summary statistics from the posterior. This is also an object of class `mcmc` so it can be plotted and summarized via the `mcmc.diagnostics` function in the `ergm` package (and also the `coda` package). The statistics are:

N population size.

mu scalar; The mean degree for the prior distribution for a randomly chosen person. The prior has this mean.

sigma scalar; The standard deviation of the degree for a randomly chosen person. The prior has this standard deviation.

degree1 scalar; the number of nodes of degree 1 in the population (it is assumed all nodes have degree 1 or more).

lambda scalar; This is only present for the `cmp` model. It is the λ parameter in the standard parameterization of the Conway-Maxwell-Poisson model for the degree distribution.

nu scalar; This is only present for the `cmp` model. It is the ν parameter in the standard parameterization of the Conway-Maxwell-Poisson model for the degree distribution.

`sample` matrix of dimension `samplesize`$\times$ `n` matrix of posterior.draws from the unit size distribution for those in the survey. The sample for the `i`th person is the `i`th column.

`lpriorm` vector; the vector of (log) prior probabilities on each value of $m = N - n$ - that is, the number of unobserved members of the population. The values are `n:(length(lpriorm)-1+n)`.

`burnintheta` count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters (θ) before any MCMC sampling is done. It typically is set to a modestly large number.

`verbose` logical; if this is `TRUE`, the program printed out additional information, including goodness of fit statistics.

`predictive.degree.count`

vector; a vector of length the maximum degree (`K`) (by default `K=2*max(sample degree)`). The `k`th entry is the posterior predictive number persons with degree `k`. That is, it is the posterior predictive distribution of the number of people with each degree in the population.

`predictive.degree`

vector; a vector of length the maximum degree (`K`) (by default `K=2*max(sample degree)`). The `k`th entry is the posterior predictive proportion of persons with degree `k`. That is, it is the posterior predictive distribution of the proportion of people with each degree in the population.

`MAP` vector of length 6 of MAP estimates corresponding to the output `sample`. These are:

N population size.

mu scalar; The mean degree for the prior distribution for a randomly chosen person. The prior has this mean.

sigma scalar; The standard deviation of the degree for a randomly chosen person. The prior has this standard deviation.

degree1 scalar; the number of nodes of degree 1 in the population (it is assumed all nodes have degree 1 or more).

lambda scalar; This is only present for the `cmp` model. It is the λ parameter in the standard parameterization of the Conway-Maxwell-Poisson model for the degree distribution.

nu scalar; This is only present for the `cmp` model. It is the ν parameter in the standard parameterization of the Conway-Maxwell-Poisson model for the degree distribution.

`mode.prior.sample.proportion`
: scalar; A hyperparameter being the mode of the prior distribution on the sample proportion n/N.

`median.prior.size`
: scalar; A hyperparameter being the mode of the prior distribution on the population size.

`mode.prior.size`
: scalar; A hyperparameter being the mode of the prior distribution on the population size.

`mean.prior.size`
: scalar; A hyperparameter being the mean of the prior distribution on the population size.

`quartiles.prior.size`
: vector of length 2; A pair of hyperparameters being the lower and upper quartiles of the prior distribution on the population size.

`degreedistribution`
: count; the parametric distribution to use for the individual network sizes (i.e., degrees). The options are `cmp`, `nbinom`, and `pln`. These correspond to the Conway-Maxwell-Poisson, Negative-Binomial, and Poisson-log-normal. The default is `cmp`.

`priorsizedistribution`
: character; the type of parametric distribution to use for the prior on population size. The options are `beta` (for a Beta prior on the sample proportion (i.e. n/N), `nbinom` (Negative-Binomial), `pln` (Poisson-log-normal), `flat` (uniform), and `continuous` (the continuous version of the Beta prior on the sample proportion. The default is `beta`.

Details on priors

The best way to specify the prior is via the hyperparameter `mode.prior.size` which specifies the mode of the prior distribution on the population size. You can alternatively specify the hyperparameter `median.prior.size` which specifies the median of the prior distribution on the population size, or `mean.prior.sample proportion` which specifies the mean of the prior distribution on the proportion of the population size in the sample or `mode.prior.sample proportion` which specifies the mode of the prior distribution on the proportion of the population size in the sample. Finally, you can specify `quartiles.prior.size` as a vector of length 2 being the pair of lower and upper quartiles of the prior distribution on the population size.

References

Gile, Krista J. (2008) *Inference from Partially-Observed Network Data*, Ph.D. Thesis, Department of Statistics, University of Washington.

Gile, Krista J. and Handcock, Mark S. (2010) *Respondent-Driven Sampling: An Assessment of Current Methodology*, Sociological Methodology 40, 285-327.

Gile, Krista J. and Handcock, Mark S. (2014) **sspse**: Estimating Hidden Population Size using Respondent Driven Sampling Data R package, Los Angeles, CA. Version 0.5, `http://hpmrg.org`.

Handcock MS (2003). **degreenet**: Models for Skewed Count Distributions Relevant to Networks. Statnet Project, Seattle, WA. Version 1.2, `http://statnetproject.org`.

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2014) *Estimating Hidden Population Size using Respondent-Driven Sampling Data*, Electronic Journal of Statistics, 8, 1, 1491-1521

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2015) *Estimating the Size of Populations at High Risk for HIV using Respondent-Driven Sampling Data*, Biometrics.

See Also

network, statnet, degreenet

Examples

```
N0 <- 200
n <- 100
K <- 10

# Create probabilities for a Waring distribution
# with scaling parameter 3 and mean 5, but truncated at K=10.
probs <- c(0.33333333,0.19047619,0.11904762,0.07936508,0.05555556,
           0.04040404,0.03030303,0.02331002,0.01831502,0.01465201)
probs <- probs / sum(probs)

#
# Create a sample
#
set.seed(1)
pop<-sample(1:K, size=N0, replace = TRUE, prob = probs)
s<-sample(pop, size=n, replace = FALSE, prob = pop)

# Here interval=1 so that it will run faster. It should be higher in a
# real application.
out <- posteriorsize(s=s,interval=1)
plot(out, HPD.level=0.9,data=pop[s])
summary(out, HPD.level=0.9)
# Let's look at some MCMC diagnostics
plot(out, HPD.level=0.9,mcmc=TRUE)
```

`print.summary.sspse` *Summarizing Population Size Estimation Model Fits*

Description

This is the `print` method for the `summary` class method for class `"sspse"` objects. These objects encapsulate an estimate of the posterior distribution of the population size based on data collected by Respondent Driven Sampling. The approach approximates the RDS via the Sequential Sampling model of Gile (2008). As such, it is referred to as the Sequential Sampling - Population Size Estimate (SS-PSE). It uses the order of selection of the sample to provide information on the distribution of network sizes over the population members.

Usage

```
## S3 method for class 'summary.sspse'
print(x, digits = max(3, getOption("digits") - 3),
  correlation = FALSE, covariance = FALSE,
  signif.stars = getOption("show.signif.stars"), eps.Pvalue = 1e-04, ...)
```

Arguments

`x` an object of class "summary.sspse", usually, a result of a call to summary.sspse.

`digits` the number of significant digits to use when printing.

`correlation` logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed.

`covariance` logical; if TRUE, the covariance matrix of the estimated parameters is returned and printed.

`signif.stars` logical. If TRUE, 'significance stars' are printed for each coefficient.

`eps.Pvalue` number; indicates the smallest p-value. printCoefmat.

`...` further arguments passed to or from other methods.

Details

print.summary.sspse tries to be smart about formatting the coefficients, standard errors, etc. and additionally gives 'significance stars' if signif.stars is TRUE.

Aliased coefficients are omitted in the returned object but restored by the print method.

Correlations are printed to two decimal places (or symbolically): to see the actual correlations print summary(object)$correlation directly.

Value

The function summary.sspse computes and returns a two row matrix of summary statistics of the prior and estimated posterior distributions. The rows correspond to the Prior and the Posterior, respectively. The rows names are Mean, Median, Mode, 25%, 75%, and 90%. These correspond to the distributional mean, median, mode, lower quartile, upper quartile and 90% quantile, respectively.

References

Gile, Krista J. (2008) *Inference from Partially-Observed Network Data*, Ph.D. Thesis, Department of Statistics, University of Washington.

Gile, Krista J. and Handcock, Mark S. (2010) *Respondent-Driven Sampling: An Assessment of Current Methodology*, Sociological Methodology 40, 285-327.

Gile, Krista J. and Handcock, Mark S. (2014) **sspse**: Estimating Hidden Population Size using Respondent Driven Sampling Data R package, Los Angeles, CA. Version 0.5, http://hpmrg.org.

Handcock MS (2003). **degreenet**: Models for Skewed Count Distributions Relevant to Networks. Statnet Project, Seattle, WA. Version 1.2, http://statnetproject.org.

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2014) *Estimating Hidden Population Size using Respondent-Driven Sampling Data*, Electronic Journal of Statistics, 8, 1, 1491-1521

Handcock, Mark S., Gile, Krista J. and Mar, Corinne M. (2015) *Estimating the Size of Populations at High Risk for HIV using Respondent-Driven Sampling Data*, Biometrics.

See Also

The model fitting function `posteriorsize`, `summary`.

Function `coef` will extract the matrix of coefficients with standard errors, t-statistics and p-values.

Examples

```
N0 <- 200
n <- 100
K <- 10

# Create probabilities for a Waring distribution
# with scaling parameter 3 and mean 5, but truncated at K=10.
probs <- c(0.33333333,0.19047619,0.11904762,0.07936508,0.05555556,
           0.04040404,0.03030303,0.02331002,0.01831502,0.01465201)
probs <- probs / sum(probs)

#
# Create a sample
#
set.seed(1)
pop<-sample(1:K, size=N0, replace = TRUE, prob = probs)
s<-sample(pop, size=n, replace = FALSE, prob = pop)

# Here interval=1 so that it will run faster. It should be higher in a
# real application.
out <- posteriorsize(s=s,interval=1)
plot(out, HPD.level=0.9,data=pop[s])
summary(out, HPD.level=0.9)
# Let's look at some MCMC diagnostics
plot(out, HPD.level=0.9,mcmc=TRUE)
```

`summary.sspse` *Summarizing Population Size Estimation Model Fits*

Description

This is the `summary` method for class `"sspse"` objects. These objects encapsulate an estimate of the posterior distribution of the population size based on data collected by Respondent Driven Sampling. The approach approximates the RDS via the Sequential Sampling model of Gile (2008). As such, it is referred to as the Sequential Sampling - Population Size Estimate (SS-PSE). It uses the order of selection of the sample to provide information on the distribution of network sizes over the population members. `summary` method for class `"sspse"`. posterior distribution of the population size based on data collected by Respondent Driven Sampling. The approach approximates the RDS via the Sequential Sampling model of Gile (2008). As such, it is referred to as the Sequential Sampling - Population Size Estimate (SS-PSE). It uses the order of selection of the sample to provide information on the distribution of network sizes over the population members.

Usage

```
## S3 method for class 'sspse'
summary(object, support = 1000, HPD.level = 0.95, ...)
```

Arguments

`object`	an object of class `"sspse"`, usually, a result of a call to `posteriorsize`.
`support`	the number of equally-spaced points to use for the support of the estimated posterior density function.
`HPD.level`	numeric; probability level of the highest probability density interval determined from the estimated posterior.
`...`	further arguments passed to or from other methods.

Details

`print.summary.sspse` tries to be smart about formatting the coefficients, standard errors, etc. and additionally gives ‘significance stars’ if `signif.stars` is `TRUE`.

Aliased coefficients are omitted in the returned object but restored by the `print` method.

Correlations are printed to two decimal places (or symbolically): to see the actual correlations print `summary(object)$correlation` directly.

Value

The function `summary.sspse` computes and returns a two row matrix of summary statistics of the prior and estimated posterior distributions. The rows correspond to the `Prior` and the `Posterior`, respectively. The rows names are `Mean`, `Median`, `Mode`, `25%`, `75%`, and `90%`. These correspond to the distributional mean, median, mode, lower quartile, upper quartile and 90% quantile, respectively.

See Also

The model fitting function `posteriorsize`, `summary`.

Examples

```
N0 <- 200
n <- 100
K <- 10

# Create probabilities for a Waring distribution
# with scaling parameter 3 and mean 5, but truncated at K=10.
probs <- c(0.33333333,0.19047619,0.11904762,0.07936508,0.05555556,
           0.04040404,0.03030303,0.02331002,0.01831502,0.01465201)
probs <- probs / sum(probs)
```

```
#
# Create a sample
#
set.seed(1)
pop<-sample(1:K, size=N0, replace = TRUE, prob = probs)
s<-sample(pop, size=n, replace = FALSE, prob = pop)

# Here interval=1 so that it will run faster. It should be higher in a
# real application.
out <- posteriorsize(s=s,interval=1)
plot(out, HPD.level=0.9,data=pop[s])
summary(out, HPD.level=0.9)
# Let's look at some MCMC diagnostics
plot(out, HPD.level=0.9,mcmc=TRUE)
```

Index