Package ‘sspse’

April 22, 2015

Type Package
Version 0.5-1
Date 2015-04-16
Title Estimating Hidden Population Size using Respondent Driven Sampling Data
Maintainer Mark S. Handcock <handcock@stat.ucla.edu>
Depends parallel
Suggests network, testthat
Description An integrated set of tools to 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.
License GPL-3 + file LICENSE
URL http://www.hpmrg.org
Imports coda
NeedsCompilation yes
Author Mark S. Handcock [aut, cre, cph], Krista J. Gile [aut, cph]
Repository CRAN
Date/Publication 2015-04-22 08:08:39

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>sspse-package</td>
<td>2</td>
</tr>
<tr>
<td>beginparallel</td>
<td>3</td>
</tr>
<tr>
<td>dsizemprior</td>
<td>4</td>
</tr>
<tr>
<td>plot.sspse</td>
<td>7</td>
</tr>
<tr>
<td>posteriorsize</td>
<td>9</td>
</tr>
<tr>
<td>print.summary.sspse</td>
<td>16</td>
</tr>
<tr>
<td>summary.sspse</td>
<td>18</td>
</tr>
</tbody>
</table>

Index 21
sspse-package  

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" package 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:


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


Parallel Processing in the *sspse* Package

**Description**

As the estimation requires MCMC, *sspse* can take advantage of multiple CPUs or CPU cores on the system on which it runs, as well as computing clusters. It uses package *parallel* and *snow* to facilitate this, and supports MPI cluster type and likely PSOCK.

**Usage**

```r
type <- "mpi"
seed <- NULL
packagenames <- c("sspse")
verbose <- TRUE
```

**Arguments**

- **parallel**: scale; Number of threads in which to run the sampling. Defaults to 1 (no parallelism).
- **type**: API to use for parallel processing. Supported values are "MPI" and "PSOCK". Defaults to using the *parallel* package with MPI clusters.
- **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.
- **packagenames**: Names of packages in which load to get the package to run functions in addition to those autodetected. This argument should not be needed outside of very strange setups.
- **verbose**: logical; if this is `TRUE`, the program will print out additional information.

**Details**

The number of nodes used and the parallel API are controlled using the `parallel` and `type` arguments.

**PSOCK clusters**

The *parallel* package is used with PSOCK clusters by default, to utilize multiple cores on a system. The number of cores on a system can be determined with the `detectCores` function.

This method works with the base installation of R on all platforms, and does not require additional software.
Examples

```r
## Not run:
# Uses 2 SOCK clusters for MCMLE estimation
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)

# Look at the degree distribution for the prior
# Plot these if you want
# plot(x=1:K,y=probs,type="l")
# points(x=1:K,y=probs)
#
# Create a sample
#
set.seed(1)
popc <- sample(1:K, size=N0, replace = TRUE, prob = probs)
s <- sample(popc, size=n, replace = FALSE, prob = popc)

out <- posteriorsize(s=s, interval=10, parallel=2)
plot(out, HPD.level=0.9, data=popc[s])
summary(out, HPD.level=0.9)
# Let's look at some MCMC diagnostics
plot(out, HPD.level=0.9, mcmc=TRUE)

## End(Not run)
```

### 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,
dsizelprior

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 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), nb (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


See Also

network, statnet, degreenet

Examples

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

```r
## 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",
     ...)
```

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.
- **...**: 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


**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**

```r
## Not run:
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(0.3333333, 0.19047619, 0.11904762, 0.07936508, 0.05555556,
          0.04040404, 0.03030303, 0.02333333, 0.01831502, 0.01465201)
probs <- probs / sum(probs)

# Look at the degree distribution for the prior
# Plot these if you want
# plot(x=1:K, y=probs, type="l")
# points(x=1:K, y=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)

out <- posteriorsize(s=s, interval=10)
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)

## End(Not run)
```

---

**posteriorsize**

*Estimating hidden population size using RDS data*
**Description**

`posterior.size` 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**

```r
posterior.size(s, median.prior.size = NULL, interval = 10, burnin = 5000, maxN = NULL, K = max(s, na.rm = TRUE), 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, Np = 0, nk = NULL, n = length(s), muproposal = 0.1, sigmaproposal = 0.15, burnintheta = 500, parallel = 1, parallel.type = "MPI", seed = NULL, maxbeta = 120, dispersion = 0, supplied = list(maxN = maxN), verbose = TRUE)
```

**Arguments**

- `s`: vector of integers; the vector of degrees from the RDS in order they are recorded.
- `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(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.

degree.distribution

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 hyperparameter being the mean degree for the prior distribution for a randomly chosen person. The prior has this mean.

sd.prior.degree

scalar; A hyperparameter 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 hyperparameter 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 hyperparameter 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 \( Np \) 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.
**posterior size**

**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** vector; the vector of counts for the number of people in the sample with degree k. 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.

**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.

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

**parallel.type** The type of parallel processing to use. The options are "PVM" or "MPI". This requires the corresponding type to be installed.

**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.

**dispersion** scalar; dispersion to use in the reported network size compared to the actual network size.

**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.

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

**Value**

**posterior size** 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 parametrization of the Conway-Maxwell-Poisson model for the degree distribution.
nu scalar; This is only present for the cmp model. It is the $\nu$ parameter in the standard parametrization of the Conway-Maxwell-Poisson model for the degree distribution.

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 $n: \text{length}(\text{lpriorm}) - 1 + n$.

burnintheta count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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 $K=2*\text{max}(\text{sample degree})$). The $k$th entry is the posterior predictive number of persons with degree $k$. That is, it is the posterior predictive distribution of the number of people with each degree in the population.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

count; the number of proposals in the Metropolis-Hastings sub-step for the degree distribution parameters ($\theta$) 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.

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


See Also

network, statnet, degreenet
Examples

```
## Not run:
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(0.333333333, 0.19047619, 0.11904762, 0.07936508, 0.05555556,
          0.04040404, 0.03030303, 0.02331002, 0.01831502, 0.01465201)
probs <- probs / sum(probs)

# Look at the degree distribution for the prior
# Plot these if you want
# plot(x=1:K,y=probs,type="l")
# points(x=1:K,y=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)

out <- posteriorSize(s=s,interval=10)
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)
```

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


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

```r
## Not run:
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(0.3333333, 0.19047619, 0.11904762, 0.07936508, 0.05555556,
          0.04040404, 0.03030303, 0.02331002, 0.01831502, 0.01465201)
probs <- probs / sum(probs)

# Look at the degree distribution for the prior
# Plot these if you want
# plot(x=1:K,y=probs,type="l")
# points(x=1:K,y=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)

out <- posteriorsize(s=s,interval=10)
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)

## End(Not run)
```

**summary.sspse**

**Summary 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. 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

```r
## 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`, `LQ`, `UQ`, and `90P`. 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

```r
## Not run:
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(0.333333333, 0.19047619, 0.11904762, 0.07936508, 0.05555556,
          0.04040404, 0.03030303, 0.02331002, 0.01831502, 0.01465201)
probs <- probs / sum(probs)

# Look at the degree distribution for the prior
# Plot these if you want
# plot(x=1:K,y=probs,type="l")
# points(x=1:K,y=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)

out <- posteriorsize(s=s, interval=10)
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)

## End(Not run)
Index

*Topic **hplot**
  plot.sspse, 7

*Topic **models**
  dsizeprior, 4
  posteriorsize, 9
  print.summary.sspse, 16
  sspse-package, 2
  summary.sspse, 18

*Topic **package**
  sspse-package, 2

beginparallel, 3

coef, 9, 17

dsizeprior, 4, 4, 6

endparallel (beginparallel), 3

plot, 9
plot.sspse, 7
posteriorsize, 8, 9, 10, 12, 17, 19
print.summary.sspse, 16
printCoefmat, 17

sspse, 3
sspse-package, 2
summary, 17, 19
summary.sspse, 18