Package 'RDS'

September 6, 2024

Type Package

Title Respondent-Driven Sampling

Version 0.9-10

Date 2024-09-05

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

Description Provides functionality for carrying out estimation
with data collected using Respondent-Driven Sampling. This includes
Heckathorn's RDS-I and RDS-II estimators as well as Gile's Sequential
Sampling estimator. The package is part of the ``RDS Analyst'' suite of
packages for the analysis of respondent-driven sampling data.
See Gile and Handcock (2010) <doi:10.1111/j.1467-9531.2010.01223.x>,
Gile and Handcock (2015) <doi:10.1111/rssa.12091> and
Gile, Beaudry, Handcock and Ott (2018) <doi:10.1146/annurev-statistics-031017-100704>.

License LGPL-2.1

URL https://hpmrg.org

Depends R (>= 2.5.1), methods

Suggests survey, sspse, testthat

Imports gridExtra, ggplot2 (>= 2.0.0), network, igraph, reshape2,
scales, anytime, Hmisc, statnet.common, ergm, isotone

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation yes

Author Mark S. Handcock [aut, cre] (<https://orcid.org/0000-0002-9985-2785>),
Krista J. Gile [aut],
Ian E. Fellows [aut],
W. Whipple Neely [ctb]

Repository CRAN

Date/Publication 2024-09-06 08:00:06 UTC

Contents

`as.char` *converts to character with minimal loss of precision for numeric variables*

Description

converts to character with minimal loss of precision for numeric variables

Usage

```
as.char(x, ...)
```

Arguments

x	the value
...	passed to either format or as.character.

`as.rds.data.frame` *Coerces a data.frame object into an rds.data.frame object.*

Description

This function converts a regular R data frame into an `rds.data.frame`. The greatest advantage of this is that it performs integrity checks and will fail if the recruitment information in the original data frame is incomplete.

Usage

```
as.rds.data.frame(
  df,
  id = if (is.null(attr(df, "id"))) "id" else attr(df, "id"),
  recruiter.id = if (is.null(attr(df, "recruiter.id"))) {
     "recruiter.id"
 } else
    attr(df, "recruiter.id"),
  network.size = if (is.null(attr(df, "network.size.variable"))) {

    "network.size.variable"
 } else attr(df, "network.size.variable"),
  population.size = if (all(is.na(get.population.size(df, FALSE)))) {
     NULL
 } else
    get.population.size(df, FALSE),
  max.coupons = if (is.null(attr(df, "max.coupons"))) {
     NULL
 } else attr(df,
    "max.coupons"),
  notes = if (is.null(attr(df, "notes"))) {
     NULL
 } else attr(df, "time"),
  time = if (is.null(attr(df, "time"))) {
     NULL
 } else attr(df, "time"),
  check.valid = TRUE
)
```

Arguments

df	A data.frame representing an RDS sample.
id	The unique identifier.
recruiter.id	The unique identifier of the recruiter of this row.
network.size	The number of alters (i.e. possible recruitees).
population.size	The size of the population from which this RDS sample has been drawn. Either a single number, or a vector of length three indicating low, mid and high estimates.
max.coupons	The number of recruitment coupons distributed to each enrolled subject (i.e. the maximum number of recruitees for any subject).
notes	Data set notes.
time	the name of the recruitment time variable. optional.
check.valid	If true, validity checks are performed to ensure that the data is well formed.

Value

An rds.data.frame object

Examples

```
dat <- data.frame(id=c(1,2,3,4,5), recruiter.id=c(2,-1,2,-1,4),
                  network.size.variable=c(4,8,8,2,3))
as.rds.data.frame(dat)
```

---

assert.valid.rds.data.frame

Does various checks and throws errors if x is not a valid rds.data.frame

---

Description

Does various checks and throws errors if x is not a valid rds.data.frame

Usage

```
assert.valid.rds.data.frame(x, ...)
```

Arguments

x	an rds.data.frame
...	unused

Details

Throws an informative message if x is malformed.

`bootstrap.contingency.test` *Performs a bootstrap test of independance between two categorical variables*

Description

Performs a bootstrap test of independance between two categorical variables

Usage

```
bootstrap.contingency.test(
  rds.data,
  row.var,
  col.var,
  number.of.bootstrap.samples = 1000,
  weight.type = c("HCG", "RDS-II", "Arithmetic Mean"),
  table.only = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`rds.data`	an rds.data.frame
`row.var`	the name of the first categorical variable
`col.var`	the name of the second categorical variable
`number.of.bootstrap.samples`	The number of simulated boootstrap populations
`weight.type`	The type of weighting to use for the contningency table. Only large sample methods are allowed.
`table.only`	only returns the weighted table, without bootstrap.
`verbose`	level of output
`...`	Additional parameters for compute_weights

Details

This function first estimates a Homophily Configuration Graph model for the underlying network under the assumption that the two variables are independant and that the population size is large. It then draws bootstrap RDS samples from this population distribution and calculates the chi.squared statistic on the weighted contingency table. Weights are calculated using the HCG estimator assuming a large population size.

Examples

```
data(faux)
bootstrap.contingency.test(rds.data=faux, row.var="X", col.var="Y",
  number.of.bootstrap.samples=50, verbose=FALSE)
```

`bootstrap.incidence` *Calculates incidence and bootstrap confidence intervals for immunoassay data collected with RDS*

Description

Calculates incidence and bootstrap confidence intervals for immunoassay data collected with RDS

Usage

```
bootstrap.incidence(
  rds.data,
  recent.variable,
  hiv.variable,
  N = NULL,
 weight.type = c("Gile's SS", "RDS-I", "RDS-I (DS)", "RDS-II", "Arithmetic Mean", "HCG"),
  mean.duration = 200,
  frr = 0.01,
  post.infection.cutoff = 730,
  number.of.bootstrap.samples = 1000,
  se.mean.duration = 0,
  se.frr = 0,
  confidence.level = 0.95,
  verbose = TRUE,
  ...
)
```

Arguments

`rds.data`	an rds.data.frame
`recent.variable`	The name of the variable indicating recent infection
`hiv.variable`	The name of the variable indicating of hiv infection
`N`	Population size
`weight.type`	A string giving the type of estimator to use. The options are `"Gile's SS"`, `"RDS-I"`, `"RDS-II"`, `"RDS-I/DS"`, and `"Arithemic Mean"`. It defaults to `"Gile's SS"`.
`mean.duration`	Estimated mean duration of recent infection (MDRI) (days)
`frr`	Estimated false-recent rate (FRR)

`post.infection.cutoff`
: Post-infection time cut-off T, separating "true-recent" from "false-recent" results (days)

`number.of.bootstrap.samples`
: The number of bootstrap samples used to construct the interval.

`se.mean.duration`
: The standard error of the mean.duration estimate

`se.frr`
: The standard error of the false recency estimate

`confidence.level`
: The level of confidence for the interval

`verbose`
: verbosity control

`...`
: additional arguments to compute.weights

Details

The recent.variable and hiv should be the names of logical variables. Otherwise they are converted to logical using as.numeric(x) > 0.5.

This function estimates incidence using RDS sampling wieghts. Confidence intervals are constucted using HCG bootstraps. See http://www.incidence-estimation.org/ for additional information on (non-RDS) incidence estimation.

Examples

```
data(faux)
faux$hiv <- faux$X == "blue"
faux$recent <- NA
faux$recent[faux$hiv] <- runif(sum(faux$hiv)) < .2
faux$recent[runif(nrow(faux)) > .5] <- NA
faux$hiv[is.na(faux$recent)][c(1,6,10,21)] <- NA
attr(faux,"time") <- "wave"
bootstrap.incidence(faux,"recent","hiv",weight.type="RDS-II", number.of.bootstrap.samples=100)
```

---

bottleneck.plot *Bottleneck Plot*

Description

Bottleneck Plot

Usage

```
bottleneck.plot(
  rds.data,
  outcome.variable,
  est.func = RDS.II.estimates,
  as.factor = FALSE,
  n.eval.points = 25,
  ...
)
```

Arguments

rds.data	An rds.data.frame.
outcome.variable	A character vector of outcome variables.
est.func	A function taking rds.data and outcome.variable as parameters and returning an rds.weighted.estimate object.
as.factor	Convert all outcome variables to factors
n.eval.points	number of evaluation points to calculate the estimates at
...	additional parameters for est.func.

References

Krista J. Gile, Lisa G. Johnston, Matthew J. Salganik *Diagnostics for Respondent-driven Sampling* eprint arXiv:1209.6254, 2012

Examples

```
data(fauxmadrona)
bottleneck.plot(fauxmadrona,"disease")
```

---

compute.weights *Compute estimates of the sampling weights of the respondent's observations based on various estimators*

Description

Compute estimates of the sampling weights of the respondent's observations based on various estimators

Usage

```
compute.weights(
  rds.data,
 weight.type = c("Gile's SS", "RDS-I", "RDS-I (DS)", "RDS-II", "Arithmetic Mean", "HCG"),
  N = NULL,
  subset = NULL,
  control = control.rds.estimates(),
  ...
)
```

Arguments

rds.data	An rds.data.frame that indicates recruitment patterns by a pair of attributes named "id" and "recruiter.id".
weight.type	A string giving the type of estimator to use. The options are "Gile's SS", "RDS-I", "RDS-II", "RDS-I/DS", and "Arithemic Mean". It defaults to "Gile's SS".
N	An estimate of the number of members of the population being sampled. If NULL it is read as the population.size.mid attribute of the rds.data frame. If that is missing, the weights will sum to 1. Note that this parameter is required for Gile's SS.
subset	An optional criterion to subset rds.data by. It is an R expression which, when evaluated, subset the data. In plain English, it can be something like subset = seed > 0 to exclude seeds. It can also be the name of a logical vector of the same length of the outcome variable where TRUE means include it in the analysis. If NULL then no subsetting is done.
control	A list of control parameters for algorithm tuning. Constructed using control.rds.estimates.
...	Additional parameters passed to the individual weighting algorithms.

Value

A vector of weights for each of the respondents. It is of the same size as the number of rows in rds.data.

See Also

rds.I.weights, gile.ss.weights, vh.weights

---

control.list.accessor *Named element accessor for ergm control lists*

Description

Utility method that overrides the standard '$' list accessor to disable partial matching for ergm control.list objects

Usage

```
## S3 method for class 'control.list'
object$name
```

Arguments

object	list-coearceable object with elements to be searched
name	literal character name of list element to search for and return

Details

Executes `getElement` instead of `$` so that element names must match exactly to be returned and partially matching names will not return the wrong object.

Value

Returns the named list element exactly matching `name`, or `NULL` if no matching elements found

Author(s)

Pavel N. Krivitsky

See Also

see `getElement`

`control.rds.estimates` *Auxiliary for Controlling RDS.bootstrap.intervals*

Description

Auxiliary function as user interface for fine-tuning RDS.bootstrap.intervals algorithm, which computes interval estimates for via bootstrapping.

Usage

```
control.rds.estimates(
  confidence.level = 0.95,
  SS.infinity = 0.01,
  lowprevalence = c(8, 14),
  discrete.cutoff = 0.8,
  useC = TRUE,
  number.of.bootstrap.samples = NULL,
  hcg.reltol = sqrt(.Machine$double.eps),
  hcg.BS.reltol = 1e+05 * sqrt(.Machine$double.eps),
  hcg.max.optim = 500,
  seed = NULL
)
```

Arguments

`confidence.level` The confidence level for the confidence intervals. The default is 0.95 for 95%.

`SS.infinity` The sample proportion, `n/N`, below which the computation of the SS weights should simplify to that of the `RDS-II` weights.

`lowprevalence` Standard confidence interval procedures can be inaccurate when the outcome expected count is close to zero. This sets conditions where alternatives to the standard are used for the `ci.type="hmg"` option. See Details for its use.

`discrete.cutoff`
The minimum proportion of the values of the outcome variable that need to be unique before the variable is judged to be continuous.

`useC` Use a C-level implementation of Gile's bootstrap (rather than the R level). The implementations should be computational equivalent (except for speed).

`number.of.bootstrap.samples`
The number of bootstrap samples to take in estimating the uncertainty of the estimator. If `NULL` it defaults to the number necessary to compute the standard error to accuracy 0.001.

`hcg.reltol` Relative convergence tolerance for the HCG estimator. The algorithm stops if it is unable to reduce the log-likelihood by a factor of `reltol * (abs(log-likelihood) + reltol)` at a step. Defaults to `sqrt(.Machine$double.eps)`, typically about `1e-8`.

`hcg.BS.reltol` Relative convergence tolerance for the bootstrap of the HCG estimator. It has the same interpretation as `hcg.reltol` except it is applied to each bootstrap sample. It is typically the same or larger than `hcg.reltol`.

`hcg.max.optim` The number of iterations on the likelihood optimization for the HCG estimator.

`seed` Seed value (integer) for the random number generator. See `set.seed`

Details

This function is only used within a call to the `RDS.bootstrap.intervals` function.

Some of the arguments are not yet fully implemented. It will evolve slower to incorporate more arguments as the package develops.

Standard confidence interval procedures can be inaccurate when the outcome expected count is close to zero. In these cases the combined Agresti-Coull and the bootstrap-t interval of Mantalos and Zografos (2008) can be used. The `lowprevalence` argument is a two vector parameter setting the conditions under which the approximation is used. The first is the penalty term on the differential activity. If the observed number of the rare group minus the product of the first parameter and the differential activity is lower than the second parameter, the low prevalence approximation is used.

Value

A list with arguments as components.

See Also

`RDS.bootstrap.intervals`

convergence.plot *Convergence Plots*

Description

This function creates diagnostic convergence plots for RDS estimators.

Usage

```
convergence.plot(
  rds.data,
  outcome.variable,
  est.func = RDS.II.estimates,
  as.factor = FALSE,
  n.eval.points = 25,
  ...
)
```

Arguments

rds.data	An rds.data.frame.
outcome.variable	A character vector of outcome variables.
est.func	A function taking rds.data and outcome.variable as parameters and returning an rds.weighted.estimate object.
as.factor	Convert all outcome variables to factors
n.eval.points	number of evaluation points to calculate the estimates at
...	additional parameters for est.func.

References

Krista J. Gile, Lisa G. Johnston, Matthew J. Salganik *Diagnostics for Respondent-driven Sampling* eprint arXiv:1209.6254, 2012

Examples

```
data(faux)
convergence.plot(faux,c("X","Y"))
```

count.transitions *Counts the number or recruiter->recruitee transitions between different levels of the grouping variable.*

Description

Counts the number or recruiter->recruitee transitions between different levels of the grouping variable.

Usage

```
count.transitions(rds.data, group.variable)
```

Arguments

rds.data	An rds.data.frame
group.variable	The name of a categorical variable in rds.data

Examples

```
data(faux)
count.transitions(faux,"X")
```

cumulative.estimate *Calculates estimates at each successive wave of the sampling process*

Description

Calculates estimates at each successive wave of the sampling process

Usage

```
cumulative.estimate(
  rds.data,
  outcome.variable,
  est.func = RDS.II.estimates,
  n.eval.points = 25,
  ...
)
```

Arguments

rds.data	An rds.data.frame
outcome.variable	The outcome
est.func	A function taking rds.data and outcome.variable as parameters and returning an rds.weighted.estimate object
n.eval.points	number of evaluation points to calculate the estimates at
...	additional parameters for est.func

---

differential.activity.estimates

Differential Activity between groups

Description

Differential Activity between groups

Usage

```
differential.activity.estimates(
  rds.data,
  outcome.variable,
  weight.type = "Gile's SS",
  N = NULL,
  subset = NULL,
  ...
)
```

Arguments

rds.data	An rds.data.frame object
outcome.variable	A character string of column names representing categorical variables.
weight.type	A string giving the type of estimator to use. The options are "Gile's SS", "RDS-I", "RDS-II", "RDS-I/DS", and "Arithemic Mean". It defaults to "Gile's SS".
N	The population size.
subset	An expression defining a subset of rds.data.
...	Additional parameters passed to compute.weights.

Details

This function estimates the ratio of the average degree of one population group divided by the average degree of those in another population group.

Examples

```
data(faux)
differential.activity.estimates(faux,"X",weight.type="RDS-II")
```

`export.rds.interval.estimate` *Convert the output of print.rds.interval.estimate from a character data.frame to a numeric matrix*

Description

Convert the output of print.rds.interval.estimate from a character data.frame to a numeric matrix

Usage

```
export.rds.interval.estimate(x, proportion = TRUE)
```

Arguments

`x`	An object, typically the result of print.rds.interval.estimate.
`proportion`	logical, Should the outcome be treated as a proportion and converted to a percentage.

`faux` *A Simulated RDS Data Set*

Description

This is a faux set used to demonstrate RDS functions and analysis. It is used is some simple examples and has categorical variables "X", "Y" and "Z".

Format

An rds.data.frame object

References

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

See Also

`fauxsycamore`, `fauxmadrona`

Examples

```
data(faux)
RDS.I.estimates(rds.data=faux,outcome.variable='X')
```

fauxmadrona *A Simulated RDS Data Set with no seed dependency*

Description

This is a faux set used to illustrate how the estimators perform under different populations and RDS schemes.

Format

An `rds.data.frame`

Details

The population had N=1000 nodes. In this case, the sample size is 500 so that there is a relatively small sample fraction (50%). There is homophily on disease status (R=5) and there is differential activity by disease status whereby the infected nodes have mean degree twice that of the uninfected (w=1.8).

In the sampling, the seeds are chosen randomly from the full population, so there is no dependency induced by seed selection.

Each sample member is given 2 uniquely identified coupons to distribute to other members of the target population in their acquaintance. Further each respondent distributes their coupons completely at random from among those they are connected to.

Here are the results for this data set and the sister `fauxsycamore` data set:

Name	City	Type	Mean	RDS I (SH)	RDS II (VH)	SS
fauxsycamore	Oxford	seed dependency, 70%	0.2408	0.1087	0.1372	0.1814
fauxmadrona	Seattle	no seed dependency, 50%	0.2592	0.1592	0.1644	0.1941

Even with only 50% sample, the VH is substantially biased , and the SS does much better.

Source

The original network is included as `fauxmadrona.network` as a `network` object.
The data set also includes the `data.frame` of the RDS data set as `fauxmadrona`.
Use `data(package="RDS")` to get a full list of datasets.

References

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

See Also

`fauxsycamore`, `faux`

`fauxsycamore` *A Simulated RDS Data Set with extreme seed dependency*

Description

This is a faux set used to demonstrate RDS functions and analysis. The population had N=715 nodes. In this case, the sample size is 500 so that there is a relatively large sample fraction (70%). There is homophily on disease status (R=5) and there is differential activity by disease status whereby the infected nodes have mean degree twice that of the uninfected (w=1.8).

Format

An rds.data.frame plus the original network as a network object

Details

In the sampling the seeds are chosen randomly from the infected population, so there is extreme dependency induced by seed selection.

Each sample member is given 2 uniquely identified coupons to distribute to other members of the target population in their acquaintance. Further each respondent distributes their coupons completely at random from among those they are connected to.

With 70% sample, the VH is substantially biased, so the SS (and presumably MA) do much better. We expect the MA to perform a bit better than the SS.

It is network 702 and its sample from YesYes on mosix. Look for "extract702.R"
The original network is included as `fauxsycamore.network` as a `network` object.
The data set also includes the `data.frame` of the RDS data set as `fauxsycamore`.
Use `data(package="RDS")` to get a full list of datasets.

References

Gile, Krista J., Handcock, Mark S., 2009. *Respondent-driven Sampling: An Assessment of Current Methodology*, *Sociological Methodology*, 40, 285-327.

See Also

`faux`, `fauxmadrona`

fauxtime *A Simulated RDS Data Set*

Description

This is a faux set used to demonstrate RDS functions and analysis.

Format

An rds.data.frame object

References

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

See Also

fauxsycamore, fauxmadrona

get.h.hat *Get Horvitz-Thompson estimator assuming inclusion probability proportional to the inverse of network.var (i.e. degree).*

Description

Get Horvitz-Thompson estimator assuming inclusion probability proportional to the inverse of network.var (i.e. degree).

Usage

```
get.h.hat(
  rds.data,
  group.variable,
  network.var = attr(rds.data, "network.size")
)
```

Arguments

rds.data	An rds.data.from
group.variable	The grouping variable.
network.var	The network.size variable.

get.id *Get the subject id*

Description

Get the subject id

Usage

```
get.id(x, check.type = TRUE)
```

Arguments

x	an rds.data.frame object
check.type	if true, x is required to be of type rds.data.frame

Details

returns the variable indicated by the 'id' attribute, coercing to a character vector

get.net.size *Returns the network size of each subject (i.e. their degree).*

Description

Returns the network size of each subject (i.e. their degree).

Usage

```
get.net.size(x, check.type = TRUE)
```

Arguments

x	the rds.data.frame
check.type	if true, x is required to be of type rds.data.frame

get.number.of.recruits *Calculates the number of (direct) recuits for each respondent.*

Description

Calculates the number of (direct) recuits for each respondent.

Usage

```
get.number.of.recruits(data)
```

Arguments

data	An rds.data.frame

Examples

```
data(fauxmadrona)
nr <- get.number.of.recruits(fauxmadrona)
#frequency of number recruited by each id
barplot(table(nr))
```

get.population.size *Returns the population size associated with the data.*

Description

Returns the population size associated with the data.

Usage

```
get.population.size(x, check.type = TRUE)
```

Arguments

x	the rds.data.frame
check.type	if true, x is required to be of type rds.data.frame

get.recruitment.time *Returns the recruitment time for each subject*

Description

Returns the recruitment time for each subject

Usage

```
get.recruitment.time(
  x,
  to.numeric = TRUE,
  wave.fallback = FALSE,
  check.type = TRUE
)
```

Arguments

x	the rds.data.frame
to.numeric	if true, time will be converted into a numeric variable.
wave.fallback	if true, subjects' recruitment times are ordered by wave and then by data.frame index if no recruitment time variable is available.
check.type	if true, x is required to be of type rds.data.frame

get.rid *Get recruiter id*

Description

Get recruiter id

Usage

```
get.rid(x, check.type = TRUE)
```

Arguments

x	an rds.data.frame object
check.type	if true, x is required to be of type rds.data.frame

Details

returns the variable indicated by the 'recruiter.id' attribute, coercing to a character vector

get.seed.id *Calculates the root seed id for each node of the recruitement tree.*

Description

Calculates the root seed id for each node of the recruitement tree.

Usage

```
get.seed.id(data)
```

Arguments

data	An rds.data.frame

Examples

```
data(fauxmadrona)
seeds <- get.seed.id(fauxmadrona)
#number recruited by each seed
barplot(table(seeds))
```

get.seed.rid *Gets the recruiter id associated with the seeds*

Description

Gets the recruiter id associated with the seeds

Usage

```
get.seed.rid(x, check.type = TRUE)
```

Arguments

x	an rds.data.frame object
check.type	if true, x is required to be of type rds.data.frame

Details

All seed nodes must have the same placeholder recruiter id.

get.stationary.distribution *Markov chain statistionary distribution*

Description

Markov chain statistionary distribution

Usage

```
get.stationary.distribution(mle)
```

Arguments

mle	The transition probabilities

Value

A vector of proportions representing the proportion in each group at the stationary distribution of the Markov chain.

get.wave *Calculates the depth of the recruitment tree (i.e. the recruitment wave) at each node.*

Description

Calculates the depth of the recruitment tree (i.e. the recruitment wave) at each node.

Usage

```
get.wave(data)
```

Arguments

data	An rds.data.frame

Examples

```
data(fauxmadrona)
#number subjects in each wave
w <- get.wave(fauxmadrona)
#number recruited in each wave
barplot(table(w))
```

gile.ss.weights *Weights using Giles SS estimator*

Description

Weights using Giles SS estimator

Usage

```
gile.ss.weights(
  degs,
  N,
  number.ss.samples.per.iteration = 500,
  number.ss.iterations = 5,
  hajek = TRUE,
  SS.infinity = 0.04,
  se = FALSE,
  ...
)
```

Arguments

degs	subjects' degrees (i.e. network sizes).
N	Population size estimate.
number.ss.samples.per.iteration	The number of samples to use to estimate inclusion probabilities in a probability proportional to size without replacement design.
number.ss.iterations	number of iterations to use in giles SS algorithm.
hajek	Should the hajek estiamtor be used. If false, the HT estimator is used.
SS.infinity	The sample proportion, n/N, below which the computation of the SS weights should simplify to that of the RDS-II weights.
se	Should covariances be included.
...	unused

has.recruitment.time *RDS data.frame has recruitment time information*

Description

RDS data.frame has recruitment time information

Usage

```
has.recruitment.time(x, check.type = TRUE)
```

Arguments

x	the rds.data.frame
check.type	if true, x is required to be of type rds.data.frame

---

hcg.replicate.weights *HCG parametric bootstrap replicate weights*

Description

HCG parametric bootstrap replicate weights

Usage

```
hcg.replicate.weights(
  rds.data,
  outcome.variable,
  number.of.bootstrap.samples = 500,
  include.sample.weights = FALSE,
  N = NULL,
  small.fraction = FALSE
)
```

Arguments

rds.data	An rds.data.frame
outcome.variable	The column name of the variable defining the groups for the homophily configuration graph
number.of.bootstrap.samples	The number of bootstrap replicate weights to be generated
include.sample.weights	If TRUE, the first column of the returned frame are the HCG weights for the sample
N	The population size
small.fraction	If TRUE, the sample size is assumed to be small compared to the population size

Details

This function generates bootstrap replicate weights which may be used to analyze RDS data in other packages or software systems (e.g. the survey package with svrepdesign).

Value

A data.frame of replicate weights. If include.sample.weights is TRUE, the first column are the HCG weights for the observed sample.

Examples

```
## Not run:
data("fauxmadrona")
set.seed(1)
# Generate replicate weights
result <- hcg.replicate.weights(fauxmadrona, "disease", 50, TRUE)
# Analyze with survey package and compare to internal function
if(require(survey)){
  set.seed(1)
  design <- svrepdesign(fauxmadrona, type = "bootstrap",
    weights= result[[1]], repweights = result[-1])
  svymean(~disease, design) |> print()
  RDS.bootstrap.intervals(fauxmadrona, "disease", "HCG", "HCG",
  number.of.bootstrap.samples = 50) |> print()
}

## End(Not run)
```

---

hcg.weights *homophily configuration graph weights*

Description

homophily configuration graph weights

Usage

```
hcg.weights(
  rds.data,
  outcome.variable,
  N = NULL,
  small.fraction = FALSE,
  reltol = sqrt(.Machine$double.eps),
  max.optim = 500,
  theta.start = NULL,
  weights.include.seeds = TRUE,
  ...
)
```

Arguments

rds.data	An rds.data.frame

outcome.variable	The variable used to base the weights on.
N	Population size
small.fraction	should a small sample fraction be assumed
reltol	Relative convergence tolerance for the HCG estimator. The algorithm stops if it is unable to reduce the log-likelihood by a factor of `reltol * (abs(log-likelihood) + reltol)` at a step. Defaults to `sqrt(.Machine$double.eps)`, typically about `1e-8`.
max.optim	The number of iterations on the likelihood optimization for the HCG estimator.
theta.start	The initial value of theta used in the likelihood optimization for the HCG estimator. If NULL, the default, it is the margin of the table of counts for the transitions.
weights.include.seeds	logical Should the weights be computed including the influence of the seeds?
...	Unused

Examples

```
data(fauxtime)
hcg.weights(fauxtime,"var1",N=3000)
fauxtime$NETWORK[c(1,100,40,82,77)] <- NA
```

---

homophily.estimates *This function computes an estimate of the population homophily and the recruitment homophily based on a categorical variable.*

Description

This function computes an estimate of the population homophily and the recruitment homophily based on a categorical variable.

Usage

```
homophily.estimates(
  rds.data,
  outcome.variable,
  weight.type = NULL,
  uncertainty = NULL,
  recruitment = FALSE,
  N = NULL,
  to.group0.variable = NULL,
  to.group1.variable = NULL,
  number.ss.samples.per.iteration = NULL,
  confidence.level = 0.95
)
```

Arguments

rds.data	An rds.data.frame that indicates recruitment patterns by a pair of attributes named "id" and "recruiter.id".
outcome.variable	A string giving the name of the variable in the rds.data that contains a categorical or numeric variable to be analyzed.
weight.type	A string giving the type of estimator to use. The options are "Gile's SS", "RDS-I", "RDS-II", "RDS-I/DS", "Good-Fellows" and "Arithemic Mean". If NULL it defaults to "Gile's SS".
uncertainty	A string giving the type of uncertainty estimator to use. The options are "Gile's SS" and "Salganik". This is usually determined by weight.type to be consistent with the estimator's origins (e.g., for "Gile's SS", "RDS-I", "RDS-II", "RDS-I/DS", and "Arithemic Mean"). Hence it's current functionality is limited. If NULL it defaults to "Gile's SS".
recruitment	A logical indicating if the homophily in the recruitment chains should be computed also. The default is FALSE.
N	An estimate of the number of members of the population being sampled. If NULL it is read as the population.size.mid attribute of the rds.data frame. If that is missing it defaults to 1000.
to.group0.variable	The number in the network of each survey respondent who have group variable value 0. Usually this is not available. The default is to not use this variable.
to.group1.variable	The number in the network of each survey respondent who have group variable value 1. Usually this is not available. The default is to not use this variable.
number.ss.samples.per.iteration	The number of samples to take in estimating the inclusion probabilites in each iteration of the sequential sampling algorithm. If NULL it is read as the number.ss.samples.per.iteration attribute of rds.data. If that is missing it defaults to 5000.
confidence.level	The confidence level for the confidence intervals. The default is 0.95 for 95%.

Value

If outcome.variable is binary then the homophily estimate of 0 verses 1 is returned, otherwise a vector of differential homophily estimates is returned.

Recruitment Homophily

The recruitment homophily is a homophily measure for the recruitment process. It addresses the question: Do respondents differential recruit people like themselves? That is, the homophily on a variable in the recruitment chains. Take as an example infection status. In this case, it is the ratio of number of recruits that have the same infection status as their recruiter to the number we would expect if there was no homophily on infection status. The difference with the Population Homophily (see below) is that this is in the recruitment chain rather than the population of social

ties. For example, of the recruitment homophily on infection status is about 1, we see little effect of recruitment homophily on infection status (as the numbers of homophilous pairs are close to what we would expect by chance).

Population Homophily

This is an estimate the homophily of a given variable in the underlying networked population. For example, consider HIV status. The population homophily is the homophily in the HIV status of two people who are tied in the underlying population social network (a "couple"). Specifically, the population homophily is the ratio of the expected number of HIV discordant couples absent homophily to the expected number of HIV discordant couples with the homophily. Hence larger values of population homophily indicate more homophily on HIV status. For example, a value of 1 means the couple are random with respect to HIV status. A value of 2 means there are twice as many HIV discordant couples as we would expect if there was no homophily in the population. This measure is meaningful across different levels of differential activity. As we do not see most of the population network, we estimate the population homophily from the RDS data. As an example, suppose the population homophily on HIV is 0.75 so there are 25% more HIV discordant couples than expected due to chance. So their is actually heterophily on HIV in the population. If the population homophily on sex is 1.1, there are 10% more same-sex couples than expected due to chance. Hence there is modest homophily on sex.

Author(s)

Mark S. Handcock with help from Krista J. Gile

References

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

Examples

```
## Not run:
data(fauxmadrona)
names(fauxmadrona)
#
# True value:
#
if(require(network)){
a=as.sociomatrix(fauxmadrona.network)
deg <- apply(a,1,sum)
dis <- fauxmadrona.network \
deg1 <- apply(a[dis==1,],1,sum)
deg0 <- apply(a[dis==0,],1,sum)
# differential activity
mean(deg1)/ mean(deg0)
p=mean(dis)
N=1000
# True homophily
p*(1-p)*mean(deg0)*mean(deg1)*N/(mean(deg)*sum(a[dis==1,dis==0]))
}
```

```
# HT based estimators using the to.group information
data(fauxmadrona)
homophily.estimates(fauxmadrona,outcome.variable="disease",
  to.group0.variable="tonondiseased", to.group1.variable="todiseased",
  N=1000)
# HT based estimators not using the to.group information
homophily.estimates(fauxmadrona,outcome.variable="disease",
  N=1000,weight.type="RDS-II")

## End(Not run)
```

`impute.degree` *Imputes missing degree values*

Description

Imputes missing degree values

Usage

```
impute.degree(
  rds.data,
  trait.variable = NULL,
  N = NULL,
  method = c("mean", "quantile"),
  quantile = 0.5,
  recruitment.lower.bound = TRUE,
  round.degree = TRUE
)
```

Arguments

`rds.data`	an rds.data.frame
`trait.variable`	the name of the variable in rds.data to stratify the imputation by
`N`	population size
`method`	If mean, the weighted mean value is imputed, otherwize a quantile is used.
`quantile`	If method is "quantile", this is the quantile that is used. Defaults to median
`recruitment.lower.bound`	If TRUE, then for each individual, the degree is taken to be the minimum of the number of recruits plus one, and the reported degree
`round.degree`	Should degrees be integer rounded.

Details

This function imputes degree values using the weighted mean or quantile values of the non-missing degrees. Weights are calcualted using Gile's SS if N is not NULL, or RDS-II if it is. If a trait variable is specified, means and quantile are calculated within the levels of the trait variable

Examples

```
data(faux)
rds.data <- faux
rds.data$network.size[c(1,2,30,52,81,101,108,111)] <- NA
impute.degree(rds.data)
impute.degree(rds.data,trait.variable="X")
impute.degree(rds.data,trait.variable="X",method="quantile")
```

`impute.visibility` *Estimates each person's personal visibility based on their self-reported degree and the number of their (direct) recruits. It uses the time the person was recruited as a factor in determining the number of recruits they produce.*

Description

Estimates each person's personal visibility based on their self-reported degree and the number of their (direct) recruits. It uses the time the person was recruited as a factor in determining the number of recruits they produce.

Usage

```
impute.visibility(
  rds.data,
  max.coupons = NULL,
  type.impute = c("median", "distribution", "mode", "mean"),
  recruit.time = NULL,
  include.tree = FALSE,
  reflect.time = FALSE,
  parallel = 1,
  parallel.type = "PSOCK",
  interval = 10,
  burnin = 5000,
  mem.optimism.prior = NULL,
  df.mem.optimism.prior = 5,
  mem.scale.prior = 2,
  df.mem.scale.prior = 10,
  mem.overdispersion = 15,
  return.posterior.sample.visibilities = FALSE,
  verbose = FALSE
)
```

Arguments

`rds.data`	An rds.data.frame
`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.

`type.impute` The type of imputation based on the conditional distribution. It can be of type `distribution`,`mode`,`median`, or `mean` with the first , the default, being a random draw from the conditional distribution.

`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).

`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).

`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".

`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.

`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.scale.prior`
scalar; A hyper parameter being the scale of the concentration of baseline negative binomial measurement error model.

`df.mem.scale.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.

`mem.overdispersion`
scalar; A parameter being the overdispersion of the negative binomial distribution that is the baseline for the measurement error model.

`return.posterior.sample.visibilities`
logical; If TRUE then return a matrix of dimension `samplesize` by `n` of posterior draws from the visibility distribution for those in the survey. The sample for the `i`th person is the `i`th column. The default is FALSE so that the vector of imputes defined by `type.impute` is returned.

`verbose` logical; if this is `TRUE`, the program will print out additional

References

McLaughlin, Katherine R.; Johnston, Lisa G.; Jakupi, Xhevat; Gexha-Bunjaku, Dafina; Deva, Edona and Handcock, Mark S. (2023) Modeling the Visibility Distribution for Respondent-Driven Sampling with Application to Population Size Estimation, *Annals of Applied Statistics*, doi:10.1093/jrsssa/qnad031

Examples

```
## Not run:
data(fauxmadrona)
# The next line fits the model for the self-reported personal
# network sizes and imputes the personal network sizes
# It may take up to 60 seconds.
visibility <- impute.visibility(fauxmadrona)
# frequency of estimated personal visibility
table(visibility)

## End(Not run)
```

`impute.visibility_mle` *Estimates each person's personal visibility based on their self-reported degree and the number of their (direct) recruits. It uses the time the person was recruited as a factor in determining the number of recruits they produce.*

Description

Estimates each person's personal visibility based on their self-reported degree and the number of their (direct) recruits. It uses the time the person was recruited as a factor in determining the number of recruits they produce.

Usage

```
impute.visibility_mle(
  rds.data,
  max.coupons = NULL,
  type.impute = c("distribution", "mode", "median", "mean"),
  recruit.time = NULL,
  include.tree = FALSE,
  unit.scale = NULL,
  unit.model = c("cmp", "nbinom"),
  optimism = FALSE,
  guess = NULL,
  reflect.time = TRUE,
  maxit = 100,
  K = NULL,
  verbose = TRUE
)
```

Arguments

rds.data	An rds.data.frame
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.
type.impute	The type of imputation based on the conditional distribution. It can be of type distribution,mode,median, or mean with the first , the default, being a random draw from the conditional distribution.
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.
unit.model	The type of distribution for the unit sizes. It can be of nbinom, meaning a negative binomial. In this case, unit.scale is the multiplier on the variance of the negative binomial compared to a Poisson of the same mean. The alternative is cmp, meaning a Conway-Maxwell-Poisson distribution. In this case, unit.scale is the scale parameter compared to a Poisson of the same mean (values less than one mean under-dispersed and values over one mean over-dispersed). The default is cmp.
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.
guess	vector; if not NULL, the initial parameter values for the MLE fitting.
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).
maxit	integer; The maximum number of iterations in the likelihood maximization. By default it is 100.
K	integer; The maximum degree. All self-reported degrees above this are recorded as being at least K. By default it is the 95th percentile of the self-reported network sizes.
verbose	logical; if this is TRUE, the program will print out additional

References

McLaughlin, K.R., M.S. Handcock, and L.G. Johnston, 2015. Inference for the visibility distribution for respondent-driven sampling. In JSM Proceedings. Alexandria, VA: American Statistical Association. 2259-2267.

Examples

```
## Not run:
data(fauxmadrona)
# The next line fits the model for the self-reported personal
# network sizes and imputes the personal network sizes
# It may take up to 60 seconds.
visibility <- impute.visibility(fauxmadrona)
# frequency of estimated personal visibility
table(visibility)

## End(Not run)
```

---

is.rds.data.frame *Is an instance of rds.data.frame*

Description

Is an instance of rds.data.frame

Usage

```
is.rds.data.frame(x)
```

Arguments

x	An object to be tested.

---

is.rds.interval.estimate *Is an instance of rds.interval.estimate*

Description

Is an instance of rds.interval.estimate

Usage

```
is.rds.interval.estimate(x)
```

Arguments

x	An object to be tested.

is.rds.interval.estimate.list

Is an instance of rds.interval.estimate.list This is a (typically time ordered) sequence of RDS estimates of a comparable quantity

Description

Is an instance of rds.interval.estimate.list This is a (typically time ordered) sequence of RDS estimates of a comparable quantity

Usage

```
is.rds.interval.estimate.list(x)
```

Arguments

x	An object to be tested.

LRT.trend.test

Compute a test of trend in prevalences based on a likelihood-ratio statistic

Description

This function takes a series of point estimates and their associated standard errors and computes the p-value for the test of a monotone decrease in the population prevalences (in sequence order). The p-value for a monotone increase is also reported. An optional plot of the estimates and the null distribution of the test statistics is provided. More formally, let the K population prevalences in sequence order be $p_1, \ldots, p_K$. We test the null hypothesis:

$$H_0 : p_1 = \ldots = p_K$$

vs

$$H_1 : p_1 \geq p_2 \ldots \geq p_K$$

with at least one equality strict. The alternatie hypothesis is for a monotone decreasing trend. A likelihood ratio statistic for this test has been derived (Bartholomew 1959). The null distribution of the likelihood ratio statistic is very complex but can be determined by a simple Monte Carlo process.
Alternatively, we can test the null hypothesis:

$$H_0 : p_1 \geq p_2 \ldots \geq p_K$$

vs

$$H_1 : \overline{H_0}$$

The null distribution of the likelihood ratio statistic is very complex but can be determined by a simple Monte Carlo process. In both cases we also test for:

$$H : p_1 \leq p_2 \ldots \leq p_K$$

that is, a monotonically increasing trend. The function requires the isotone library.

Usage

```
LRT.trend.test(
  data,
  variables = colnames(data),
  null = "monotone",
  confidence.level = 0.95,
  number.of.bootstrap.samples = 5000,
  plot = NULL,
  seed = 1
)
```

Arguments

`data`	A two row matrix or data.frame of prevalence estimates and their standard errors. The first row is the prevalence estimates and the second are the standard errors. The column are the comparison groups in the order (e.g., time) there are to be assessed. The row names of `data` should be "estimate" and "sigma". This is
`variables`	A character vector of column names it select from `data`.
`null`	A character string indicating the null hypothesis to use. The value `"monotone"` uses the various monotone hypotheses as the nulls. If not `"monotone"`, the null is chosen to be that of equality of the means over all periods.
`confidence.level`	The confidence level for the confidence intervals. The default is 0.95 for 95%.
`number.of.bootstrap.samples`	The number of Monte Carlo draws to determine the null distribution of the likelihood ratio statistic.
`plot`	A character vector of choices, a subset of `estimates`, `distributions`. If `estimates` is given then a plot of the estimates and nominal 95% confidence bands (as error bars) is produced. If `distributions` is given then a plot is produced of the null distributions of the likelihood ratio statistic with the observed likelihood ratio statistics plotted as a vertical dashed line.
`seed`	The value of the random number seed. Preset by default to allow reproducibility.

Value

A list with components

- `pvalue.increasing`: The p-value for the test of a monotone increase in population prevalence.

- pvalue.decreasing: The p-value for the test of a monotone decrease in population prevalence.
- L: The value of the likelihood-ratio statistic.
- x: The passed vector of prevalence estimates in the order (e.g., time).
- sigma The passed vector of standard error estimates corresponding to x.

Author(s)

Mark S. Handcock

References

Bartholomew, D. J. (1959). A test of homogeneity for ordered alternatives. Biometrika 46 36-48.

Examples

```
d <- t(data.frame(estimate=c(0.16,0.15,0.3), sigma=c(0.04,0.04,0.1)))
colnames(d) <- c("time_1","time_2","time_3")
LRT.trend.test(d,number.of.bootstrap.samples=1000)
```

LRT.value.trend *Compute a test of trend in prevalences based on a likelihood-ratio statistic*

Description

This function takes a series of point estimates and their associated standard errors and computes the p-value for the test of a monotone decrease in the population prevalences (in sequence order). The p-value for a monotone increase is also reported. More formally, let the K population prevalences in sequence order be $p_1, \ldots, p_K$. We test the null hypothesis:

$$H_0 : p_1 = \ldots = p_K$$

vs

$$H_1 : p_1 \geq p_2 \ldots \geq p_K$$

with at least one equality strict. A likelihood ratio statistic for this test has been derived (Bartholomew 1959). The null distribution of the likelihood ratio statistic is very complex but can be determined by a simple Monte Carlo process.
We also test the null hypothesis:

$$H_0 : p_1 \geq p_2 \ldots \geq p_K$$

vs

$$H_1 : \overline{H_0}$$

The null distribution of the likelihood ratio statistic is very complex but can be determined by a simple Monte Carlo process. The function requires the isotone library.

Usage

```
LRT.value.trend(x, sigma)
```

Arguments

x	A vector of prevalence estimates in the order (e.g., time).
sigma	A vector of standard error estimates corresponding to x.

Value

A list with components

- pvalue.increasing: The p-value for the test of a monotone increase in population prevalence.
- pvalue.decreasing: The p-value for the test of a monotone decrease in population prevalence.
- L: The value of the likelihood-ratio statistic.
- x: The passed vector of prevalence estimates in the order (e.g., time).
- sigma The passed vector of standard error estimates corresponding to x.

Author(s)

Mark S. Handcock

References

Bartholomew, D. J. (1959). A test of homogeneity for ordered alternatives. Biometrika 46 36-48.

Examples

```
## Not run:
x <- c(0.16,0.15,0.3)
sigma <- c(0.04,0.04,0.1)
LRT.value.trend(x,sigma)

## End(Not run)
```

MA.estimates *MA Estimates*

Description

This function computes the sequential sampling (MA) estimates for a categorical variable or numeric variable.

Usage

```
MA.estimates(
  rds.data,
  trait.variable,
  seed.selection = "degree",
  number.of.seeds = NULL,
  number.of.coupons = NULL,
  number.of.iterations = 3,
  N = NULL,
  M1 = 25,
  M2 = 20,
  seed = 1,
  initial.sampling.probabilities = NULL,
  MPLE.samplesize = 50000,
  SAN.maxit = 5,
  SAN.nsteps = 2^19,
  sim.interval = 10000,
  number.of.cross.ties = NULL,
  max.degree = NULL,
  parallel = 1,
  parallel.type = "PSOCK",
  full.output = FALSE,
  verbose = TRUE
)
```

Arguments

`rds.data` An `rds.data.frame` that indicates recruitment patterns by a pair of attributes named "id" and "recruiter.id".

`trait.variable` A string giving the name of the variable in the `rds.data` that contains a categorical or numeric variable to be analyzed.

`seed.selection` An estimate of the mechanism guiding the choice of seeds. The choices are

- **"allwithtrait"** indicating that all the seeds had the trait;
- **"random"** meaning they were, as if, a simple random sample of individuals from the population;
- **"sample"** indicating that the seeds are taken as those in the sample (and resampled for the population with that composition if necessary);
- **"degree"** is proportional to the degree of the individual;
- **"allwithtraitdegree"** indicating that all the seeds had the trait and the probability of being a seed is proportional to the degree of the respondent.

`number.of.seeds` The number of seeds chosen to initiate the sampling.

`number.of.coupons` The number of coupons given to each respondent.

`number.of.iterations` The number of iterations used at the core of the algorithm.

`N` An estimate of the number of members of the population being sampled. If `NULL` it is read as the `pop.size.mid` attribute of the `rds.data` frame. If that is missing it defaults to 1000.

`M1` The number of networked populations generated at each iteration.

`M2` The number of (full) RDS samples generated for each networked population at each iteration.

`seed` The random number seed used to initiate the computations.

`initial.sampling.probabilities`
Initialize sampling probabilities for the algorithm. If missing, they are taken as proportional to degree, and this is almost always the best starting values.

`MPLE.samplesize`
Number of samples to take in the computation of the maximum pseudolikelihood estimator (MPLE) of the working model parameter. The default is almost always sufficient.

`SAN.maxit` A ceiling on the number of simulated annealing iterations.

`SAN.nsteps` Number of MCMC proposals for all the annealing runs combined.

`sim.interval` Number of MCMC steps between each of the M1 sampled networks per iteration.

`number.of.cross.ties`
The expected number of ties between those with the trait and those without. If missing, it is computed based on the respondent's reports of the number of ties they have to population members who have the trait (i.e. `ties.to.trait.variable`) and do not have the trait (i.e. `ties.not.to.trait.variable`).

`max.degree` Impose ceiling on degree size.

`parallel` Number of processors to use in the computations. The default is 1, that is no parallel processing.

`parallel.type` The type of cluster to start. e.g. 'PSOCK', 'MPI', etc.

`full.output` More verbose output

`verbose` Should verbose diagnostics be printed while the algorithm is running.

Value

If `trait.variable` is numeric then the model-assisted estimate of the mean is returned, otherwise a vector of proportion estimates is returned. If `full.output=TRUE` this leads to:

If `full.output=FALSE` this leads to an object of class `rds.interval.estimate` which is a list with component

estimate the numerical point estimate of proportion of the`trait.variable`.

interval a matrix with size columns and one row per category of `trait.variable`:

point estimate The HT estimate of the population mean.

95% Lower Bound Lower 95% confidence bound

95% Upper Bound Upper 95% confidence bound

Design Effect The design effect of the RDS

s.e. standard error

n count of the number of sample values with that value of the trait

rds.data an `rds.data.frame` that indicates recruitment patterns by a pair of attributes named "id" and "recruiter.id".

N an estimate of the number of members of the population being sampled. If `NULL` it is read as the `pop.size.mid` attribute of the `rds.data` frame. If that is missing it defaults to 1000.

M1 the number of networked populations generated at each iteration.

M2 the number of (full) RDS populations generated for each networked population at each iteration.

seed the random number seed used to initiate the computations.

seed.selection an estimate of the mechanism guiding the choice of seeds. The choices are

"allwithtrait" indicating that all the seeds had the trait;

"random" meaning they were, as if, a simple random sample of individuals from the population;

"sample" indicating that the seeds are taken as those in the sample (and resampled for the population with that composition if necessary);

"degree" is proportional to the degree of the individual;

"allwithtraitdegree" indicating that all the seeds had the trait and the probability of being a seed is proportional to the degree of the respondent.

number.of.seeds The number of seeds chosen to initiate the sampling.

number.of.coupons The number of coupons given to each respondent.

number.of.iterations The number of iterations used at the core of the algorithm.

outcome.variable The name of the outcome variable

weight.type The type of weighting used (i.e. MA)

uncertainty The type of weighting used (i.e. MA)

details A list of other diagnostic output from the computations.

varestBS Output from the bootstrap procedure. A list with two elements: `var` is the bootstrap variance, and `BSest` is the vector of bootstrap estimates themselves.

coefficient estimate of the parameter of the ERGM for the network.

Author(s)

Krista J. Gile with help from Mark S. Handcock

References

Gile, Krista J. 2011 Improved Inference for Respondent-Driven Sampling Data with Application to HIV Prevalence Estimation, Journal of the American Statistical Association, 106, 135-146.

Gile, Krista J., Handcock, Mark S., 2010. Respondent-driven Sampling: An Assessment of Current Methodology, Sociological Methodology, 40, 285-327. <doi:10.1111/j.1467-9531.2010.01223.x>

Gile, Krista J., Beaudry, Isabelle S. and Handcock, Mark S., 2018 Methods for Inference from Respondent-Driven Sampling Data, Annual Review of Statistics and Its Application <doi:10.1146/annurev-statistics-031017-100704>.

See Also

RDS.I.estimates, RDS.I.estimates

Examples

```
## Not run:
data(faux)
MA.estimates(rds.data=faux,trait.variable='X')

## End(Not run)
```

`plot.rds.data.frame` *Diagnostic plots for the RDS recruitment process*

Description

Diagnostic plots for the RDS recruitment process

Usage

```
## S3 method for class 'rds.data.frame'
plot(
  x,
  plot.type = c("Recruitment tree", "Network size by wave", "Recruits by wave",
    "Recruits per seed", "Recruits per subject"),
  stratify.by = NULL,
  ...
)
```

Arguments

`x`	An rds.data.frame object.
`plot.type`	the type of diagnostic.
`stratify.by`	A factor used to color or stratify the plot elements.
`...`	Additional arguments for the underlying plot function if applicable.

Details

Several types of diagnostics are supported by the plot.type argument. 'Recruitment tree' displays a network plot of the RDS recruitment process. 'Network size by wave' monitors systematic changes is network size based on how far subjects are from the seed 'Recruits by wave' displays counts of subjects based on how far they rare from their seed. 'Recruit per seed' shows the total tree size for each seed. 'Recruits per subject' shows counts of how many subjects are recruited by each subject who are non-terminal.

Value

Either nothing (for the recruitment tree plot), or a ggplot2 object.

Examples

```
data(fauxmadrona)
## Not run:
plot(fauxmadrona)

## End(Not run)
plot(fauxmadrona, plot.type='Recruits by wave')
plot(fauxmadrona, plot.type='Recruits per seed')
plot(fauxmadrona, plot.type='Recruits per subject')

plot(fauxmadrona, plot.type='Recruits by wave', stratify.by='disease')
plot(fauxmadrona, plot.type='Recruits per seed', stratify.by='disease')
plot(fauxmadrona, plot.type='Recruits per subject', stratify.by='disease')
```

---

print.differential.activity.estimate

Prints an differential.activity.estimate object

Description

Prints an differential.activity.estimate object

Usage

```
## S3 method for class 'differential.activity.estimate'
print(x, ...)
```

Arguments

x	an differential.activity.estimate object
...	unused

---

print.pvalue.table

Displays a pvalue.table

Description

Displays a pvalue.table

Usage

```
## S3 method for class 'pvalue.table'
print(x, ...)
```

Arguments

x	a pvalue.table object
...	additional parameters passed to print.data.frame.

---

print.rds.contin.bootstrap *Displays an rds.contin.bootstrap*

Description

Displays an rds.contin.bootstrap

Usage

```
## S3 method for class 'rds.contin.bootstrap'
print(x, show.table = FALSE, ...)
```

Arguments

x	an rds.contin.bootstrap object
show.table	Display weighted contingency table
...	additional parameters passed to print.matrix.

---

print.rds.data.frame *Displays an rds.data.frame*

Description

Displays an rds.data.frame

Usage

```
## S3 method for class 'rds.data.frame'
print(x, ...)
```

Arguments

x	an rds.data.frame object
...	additional parameters passed to print.data.frame.

print.rds.interval.estimate

Prints an rds.interval.estimate *object*

Description

Prints an rds.interval.estimate object

Usage

```
## S3 method for class 'rds.interval.estimate'
print(x, as.percentage = NULL, ...)
```

Arguments

x	an rds.interval.estimate object
as.percentage	logical. Print the interval estimates as percentages (as distinct from proportions). The default, NULL, means that it will determine if the variable is discrete or continuous and only print them as percentages if they are discrete.
...	unused

print.summary.svyglm.RDS

Summarizing Generalized Linear Model Fits with Odds Ratios

Description

print.summary.svyglm.RDS is a version of print.summary.svyglm that reports odds-ratios in place of coefficients in the summary table. This only applies for the binomial family. Otherwise it is identical to print.summary.svyglm. The default in
print.summary.svyglm is to display the log-odds-ratios and this displays the exponetiated from and a 95 p-values are still displayed.

Usage

```
## S3 method for class 'summary.svyglm.RDS'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  symbolic.cor = x$symbolic.cor,
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

x	an object of class "summary.svyglm.RDS", usually, a result of a call to RDS::summary.svyglm.
digits	the number of significant digits to use when printing.
symbolic.cor	logical. If TRUE, print the correlations in a symbolic form (see symnum) rather than as numbers.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.
...	further arguments passed to or from other methods.

See Also

svyglm, summary.svyglm.

Examples

```
## For examples see example(svyglm)
```

RDS.bootstrap.intervals *RDS Bootstrap Interval Estimates*

Description

This function computes an interval estimate for one or more categorical variables. It optionally uses attributes of the RDS data set to determine the type of estimator and type of uncertainty estimate to use.

Usage

```
RDS.bootstrap.intervals(
  rds.data,
  outcome.variable,
  weight.type = NULL,
  uncertainty = NULL,
  N = NULL,
  subset = NULL,
  confidence.level = 0.95,
  number.of.bootstrap.samples = NULL,
  fast = TRUE,
  useC = TRUE,
  ci.type = "t",
  control = control.rds.estimates(),
  to.factor = FALSE,
  cont.breaks = 3,
  ...
)
```

Arguments

rds.data	An rds.data.frame that indicates recruitment patterns by a pair of attributes named “id” and “recruiter.id”.
outcome.variable	A string giving the name of the variable in the rds.data that contains a categorical or numeric variable to be analyzed.
weight.type	A string giving the type of estimator to use. The options are "Gile's SS", "RDS-I", "RDS-II", "RDS-I (DS)", and "Arithemic Mean". If NULL it defaults to "Gile's SS".
uncertainty	A string giving the type of uncertainty estimator to use. The options are "SRS", "Gile" and "Salganik". This is usually determined by weight.type to be consistent with the estimator's origins. The estimators RDS-I, RDS-I (DS), and RDS-II default to "Salganik", "Arithmetic Mean" defaults to "SRS" and "Gile's SS" defaults to the "Gile" bootstrap.
N	An estimate of the number of members of the population being sampled. If NULL it is read as the population.size.mid attribute of the rds.data frame. If that is missing it defaults to 1000.
subset	An optional criterion to subset rds.data by. It is a character string giving an R expression which, when evaluated, subset the data. In plain English, it can be something like "seed > 0" to exclude seeds. It can be the name of a logical vector of the same length of the outcome variable where TRUE means include it in the analysis. If NULL then no subsetting is done.
confidence.level	The confidence level for the confidence intervals. The default is 0.95 for 95%.
number.of.bootstrap.samples	The number of bootstrap samples to take in estimating the uncertainty of the estimator. If NULL it defaults to the number necessary to compute the standard error to accuracy 0.001. outcome.variable. Otherwise it will compute the population frequencies of each value of the outcome.variable.
fast	Use a fast bootstrap where the weights are reused from the estimator rather than being recomputed for each bootstrap sample.
useC	Use a C-level implementation of Gile's bootstrap (rather than the R level). The implementations should be a computational equivalent estimator (except for speed).
ci.type	Type of confidence interval to use, if possible. If "t", use lower and upper confidence interval values based on the standard deviation of the bootstrapped values and a t multiplier. If "pivotal", use lower and upper confidence interval values based on the basic bootstrap (also called the pivotal confidence interval). If "quantile", use lower and upper confidence interval values based on the quantiles of the bootstrap sample. If "proportion", use the "t" unless the estimated proportion is less than 0.15 or the bounds are outside [0,1 . In this case, try the "quantile" and constrain the bounds to be compatible with [0,1].
control	A list of control parameters for algorithm tuning. Constructed using control.rds.estimates.
to.factor	force variable to be a factor

cont.breaks	For continuous variates, some bootstrap proceedures require categorical data. In these cases, in order to contruct each bootstrap replicate, the outcome variable is split into cont.breaks categories.
...	Additional arguments for RDS.*.estimates.

Value

An object of class rds.interval.estimate summarizing the inference. The confidence interval and standard error are based on the bootstrap procedure. In additon, the object has attribute bsresult which provides details of the bootstrap procedure. The contents of the bsresult attribute depends on the uncertainty used. If uncertainty=="Salganik" then bsresult is a vector of standard deviations of the bootstrap samples. If uncertainty=="Gile's SS" then bsresult is a list with components for the bootstrap point estimate, the bootstrap samples themselves and the standard deviations of the bootstrap samples. If uncertainty=="SRS" then bsresult is NULL.

References

Gile, Krista J. 2011 *Improved Inference for Respondent-Driven Sampling Data with Application to HIV Prevalence Estimation*, *Journal of the American Statistical Association*, 106, 135-146.

Gile, Krista J., Handcock, Mark S., 2010. Respondent-driven Sampling: An Assessment of Current Methodology, Sociological Methodology, 40, 285-327. <doi:10.1111/j.1467-9531.2010.01223.x>

Gile, Krista J., Beaudry, Isabelle S. and Handcock, Mark S., 2018 Methods for Inference from Respondent-Driven Sampling Data, Annual Review of Statistics and Its Application <doi:10.1146/annurev-statistics-031017-100704>.

Examples

```
## Not run:
data(fauxmadrona)
RDS.bootstrap.intervals(rds.data=fauxmadrona,weight.type="RDS-II",
     uncertainty="Salganik",
outcome.variable="disease",N=1000,number.of.bootstrap.samples=50)

data(fauxtime)
RDS.bootstrap.intervals(rds.data=fauxtime,weight.type="HCG",
     uncertainty="HCG",
outcome.variable="var1",N=1000,number.of.bootstrap.samples=10)

## End(Not run)
```

RDS.compare.proportions

Compares the rates of two variables against one another.

Description

Compares the rates of two variables against one another.

Usage

```
RDS.compare.proportions(first.interval, second.interval, M = 10000)
```

Arguments

first.interval	An rds.interval.estimate object fit with either "Gile" or "Salganik" uncertainty.
second.interval	An rds.interval.estimate object fit with either "Gile" or "Salganik" uncertainty.
M	The number of bootstrap resamplings to use

Details

This function preforms a bootstrap test comparing the the rates of two variables against one another.

Examples

```
## Not run:
data(faux)
int1 <- RDS.bootstrap.intervals(faux, outcome.variable=c("X"),
weight.type="RDS-II", uncertainty="Salganik", N=1000,
number.ss.samples.per.iteration=1000,
confidence.level=0.95, number.of.bootstrap.samples=100)
int2 <- RDS.bootstrap.intervals(faux, outcome.variable=c("Y"),
weight.type="RDS-II", uncertainty="Salganik", N=1000,
number.ss.samples.per.iteration=1000,
confidence.level=0.95, number.of.bootstrap.samples=100)
RDS.compare.proportions(int1,int2)

## End(Not run)
```

RDS.compare.two.proportions

Compares the rates of two variables against one another.

Description

Compares the rates of two variables against one another.

Usage

```
RDS.compare.two.proportions(
  data,
  variables,
  confidence.level = 0.95,
  number.of.bootstrap.samples = 5000,
```

```
  plot = FALSE,
  seed = 1
)
```

Arguments

data	An object of class `rds.interval.estimates.list` with attribute `variables` containing a character vector of names of objects of class `rds.interval.estimate`.
variables	A character vector of column names to select from `data`.
confidence.level	The confidence level for the confidence intervals. The default is 0.95 for 95%.
number.of.bootstrap.samples	The number of Monte Carlo draws to determine the null distribution of the likelihood ratio statistic.
plot	Logical, if TRUE then a plot is produces of the null distribution of the likelihood ratio statistic with the observed statistics plotted as a vertical dashed line.
seed	The value of the random number seed. Preset by default to allow reproducability.

Value

An object of class `pvalue.table` containing the cross-tabulation of p-values for comparing the two classes

---

RDS.HCG.estimates *Homophily Configuration Graph Estimates*

Description

This function computes the Homophily Configuration Graph type estimates for a categorical variable.

Usage

```
RDS.HCG.estimates(
  rds.data,
  outcome.variable,
  N = NULL,
  subset = NULL,
  small.fraction = FALSE,
  empir.lik = TRUE,
  to.factor = FALSE,
  cont.breaks = 3
)
```

Arguments

rds.data	An rds.data.frame with recruitment time set.
outcome.variable	A string giving the name of the variable in the rds.data that contains a categorical variable to be analyzed.
N	Population size to be used to calculate the empirical likelihood interval. If NULL, this value is taken to be the population.size.mid attribute of the data and if that is not set, no finite population correction is used.
subset	An optional criterion to subset rds.data by. It is an R expression which, when evaluated, subset the data. In plain English, it can be something like subset = seed > 0 to exclude seeds. It can also be the name of a logical vector of the same length of the outcome variable where TRUE means include it in the analysis. If NULL then no subsetting is done.
small.fraction	Should a small sample fraction be assumed
empir.lik	Should confidence intervals be estimated using empirical likelihood.
to.factor	force variable to be a factor
cont.breaks	If variable is numeric, how many discretization points should be used in the calculation of the weights.

Value

If the empir.lik is true, an object of class rds.interval.estimate is returned. This is a list with components

- estimate: The numerical point estimate of proportion of the trait.variable.
- interval: A matrix with six columns and one row per category of trait.variable:
  - point estimate: The HT estimate of the population mean.
  - 95% Lower Bound: Lower 95% confidence bound.
  - 95% Upper Bound: Upper 95% confidence bound.
  - Design Effect: The design effect of the RDS.
  - s.e.: Standard error.
  - n: Count of the number of sample values with that value of the trait.

Otherwise an object of class rds.HCG.estimate object is returned.

Author(s)

Ian E. Fellows

See Also

RDS.I.estimates, RDS.II.estimates, RDS.SS.estimates

Examples

```
data(fauxtime)
RDS.HCG.estimates(rds.data=fauxtime,outcome.variable='var1')
```

`RDS.I.estimates` *Compute RDS-I Estimates*

Description

This function computes the RDS-I type estimates for a categorical variable. It is also referred to as the Salganik-Heckathorn estimator.

Usage

```
RDS.I.estimates(
  rds.data,
  outcome.variable,
  N = NULL,
  subset = NULL,
  smoothed = FALSE,
  empir.lik = TRUE,
  to.factor = FALSE,
  cont.breaks = 3
)
```

Arguments

`rds.data`	An `rds.data.frame` that indicates recruitment patterns by a pair of attributes named "id" and "recruiter.id".
`outcome.variable`	A string giving the name of the variable in the `rds.data` that contains a categorical variable to be analyzed.
`N`	Population size to be used to calculate the empirical likelihood interval. If NULL, this value is taken to be the population.size.mid attribute of the data and if that is not set, no finite population correction is used.
`subset`	An optional criterion to subset `rds.data` by. It is an R expression which, when evaluated, subset the data. In plain English, it can be something like `subset = seed > 0` to exclude seeds. It can also be the name of a logical vector of the same length of the outcome variable where TRUE means include it in the analysis. If `NULL` then no subsetting is done.
`smoothed`	Logical, if TRUE then the "data smoothed" version of RDS-I is used, where it is assumed that the observed Markov process is reversible.
`empir.lik`	Should confidence intervals be estimated using empirical likelihood.
`to.factor`	force variable to be a factor
`cont.breaks`	The number of categories used for the RDS-I adjustment when the variate is continuous.

Value

If the `empir.lik` is true, an object of class `rds.interval.estimate` is returned. This is a list with components

- `estimate`: The numerical point estimate of proportion of the `trait.variable`.
- `interval`: A matrix with six columns and one row per category of `trait.variable`:
  - `point estimate`: The HT estimate of the population mean.
  - `95% Lower Bound`: Lower 95% confidence bound.
  - `95% Upper Bound`: Upper 95% confidence bound.
  - `Design Effect`: The design effect of the RDS.
  - `s.e.`: Standard error.
  - `n`: Count of the number of sample values with that value of the trait.

Otherwise an object of class `rds.I.estimate` object is returned.

Author(s)

Mark S. Handcock and W. Whipple Neely

References

Gile, Krista J., Handcock, Mark S., 2010. Respondent-driven Sampling: An Assessment of Current Methodology, Sociological Methodology, 40, 285-327. <doi:10.1111/j.1467-9531.2010.01223.x>

Gile, Krista J., Beaudry, Isabelle S. and Handcock, Mark S., 2018 Methods for Inference from Respondent-Driven Sampling Data, Annual Review of Statistics and Its Application <doi:10.1146/annurev-statistics-031017-100704>.

Neely, W. W., 2009. *Bayesian methods for data from respondent driven sampling*. Dissertation in-progress, Department of Statistics, University of Wisconsin, Madison.

Salganik, M., Heckathorn, D. D., 2004. *Sampling and estimation in hidden populations using respondent-driven sampling*. Sociological Methodology 34, 193-239.

Volz, E., Heckathorn, D., 2008. *Probability based estimation theory for Respondent Driven Sampling*. The Journal of Official Statistics 24 (1), 79-97.

See Also

`RDS.II.estimates`, `RDS.SS.estimates`

Examples

```
data(faux)
RDS.I.estimates(rds.data=faux,outcome.variable='X')
RDS.I.estimates(rds.data=faux,outcome.variable='X',smoothed=TRUE)
```

rds.I.weights *RDS-I weights*

Description

RDS-I weights

Usage

```
rds.I.weights(rds.data, outcome.variable, N = NULL, smoothed = FALSE, ...)
```

Arguments

rds.data	An rds.data.frame
outcome.variable	The variable used to base the weights on.
N	Population size
smoothed	Should the data smoothed RDS-I weights be computed.
...	Unused

RDS.II.estimates *RDS-II Estimates*

Description

This function computes the RDS-II estimates for a categorical variable or the RDS-II estimate for a numeric variable.

Usage

```
RDS.II.estimates(
  rds.data,
  outcome.variable,
  N = NULL,
  subset = NULL,
  empir.lik = TRUE,
  to.factor = FALSE
)
```

Arguments

rds.data	An rds.data.frame that indicates recruitment patterns by a pair of attributes named "id" and "recruiter.id".
outcome.variable	A string giving the name of the variable in the rds.data that contains a categorical or numeric variable to be analyzed.
N	Population size to be used to calculate the empirical likelihood interval. If NULL, this value is taken to be the population.size.mid attribute of the data and if that is not set, no finite population correction is used.
subset	An optional criterion to subset rds.data by. It is an R expression which, when evaluated, subset the data. In plain English, it can be something like subset = seed > 0 to exclude seeds. It can also be the name of a logical vector of the same length of the outcome variable where TRUE means include it in the analysis. If NULL then no subsetting is done.
empir.lik	If true, and outcome.variable is numeric, standard errors based on empirical likelihood will be given.
to.factor	force variable to be a factor

Value

If outcome.variable is numeric then the RDS-II estimate of the mean is returned, otherwise a vector of proportion estimates is returned. If the empir.lik is true, an object of class rds.interval.estimate is returned. This is a list with components

- estimate: The numerical point estimate of proportion of the trait.variable.
- interval: A matrix with six columns and one row per category of trait.variable:
  - point estimate: The HT estimate of the population mean.
  - 95% Lower Bound: Lower 95% confidence bound.
  - 95% Upper Bound: Upper 95% confidence bound.
  - Design Effect: The design effect of the RDS.
  - s.e.: Standard error.
  - n: Count of the number of sample values with that value of the trait.

Otherwise, an object of class rds.II.estimate is returned.

Author(s)

Mark S. Handcock and W. Whipple Neely

References

Gile, Krista J., Handcock, Mark S., 2010. Respondent-driven Sampling: An Assessment of Current Methodology, Sociological Methodology, 40, 285-327. <doi:10.1111/j.1467-9531.2010.01223.x>

Gile, Krista J., Beaudry, Isabelle S. and Handcock, Mark S., 2018 Methods for Inference from Respondent-Driven Sampling Data, Annual Review of Statistics and Its Application <doi:10.1146/annurev-statistics-031017-100704>.

Salganik, M., Heckathorn, D. D., 2004. *Sampling and estimation in hidden populations using respondent-driven sampling*. Sociological Methodology 34, 193-239.

Volz, E., Heckathorn, D., 2008. *Probability based estimation theory for Respondent Driven Sampling*. The Journal of Official Statistics 24 (1), 79-97.

See Also

RDS.I.estimates, RDS.SS.estimates

Examples

```
data(faux)
RDS.II.estimates(rds.data=faux,outcome.variable='X')
RDS.II.estimates(rds.data=faux,outcome.variable='X',subset= Y!="blue")
```

---

rds.interval.estimate *An object of class rds.interval.estimate*

Description

This function creates an object of class rds.interval.estimate.

Usage

```
rds.interval.estimate(
  estimate,
  outcome.variable,
  weight.type,
  uncertainty,
  weights,
  N = NULL,
  conf.level = 0.95,
  csubset = ""
)
```

Arguments

estimate	The numerical point estimate of proportion of the trait.variable.
outcome.variable	A string giving the name of the variable in the rds.data that contains a categorical variable to be analyzed.
weight.type	A string giving the type of estimator to use. The options are "Gile's SS", "RDS-I", "RDS-II", "RDS-I (DS)", and "Arithemic Mean". If NULL it defaults to "Gile's SS".

uncertainty	A string giving the type of uncertainty estimator to use. The options are "SRS", "Gile" and "Salganik". This is usually determined by weight.type to be consistent with the estimator's origins. The estimators "RDS-I", "RDS-I (DS)", "RDS-II" default to "Salganik", "Arithmetic Mean" defaults to "SRS" and "Gile's SS" defaults to the "Gile" bootstrap.
weights	A numerical vector of sampling weights for the sample, in order of the sample. They should be inversely proportional to the first-order inclusion probabilites, although this is not assessed or inforced.
N	An estimate of the number of members of the population being sampled. If NULL it is read as the pop.size.mid attribute of the rds.data frame. If that is missing it defaults to 1000.
conf.level	The confidence level for the confidence intervals. The default is 0.95 for 95%.
csubset	A character string representing text to add to the output label. Typically this will be the expression used it define the subset of the data used for the estimate.

Value

An object of class rds.interval.estimate is returned. This is a list with components

- estimate: The numerical point estimate of proportion of the trait.variable.
- interval: A matrix with six columns and one row per category of trait.variable:
  - point estimate: The HT estimate of the population mean.
  - 95% Lower Bound: Lower 95% confidence bound.
  - 95% Upper Bound: Upper 95% confidence bound.
  - Design Effect: The design effect of the RDS.
  - s.e.: Standard error.
  - n: Count of the number of sample values with that value of the trait.

Author(s)

Mark S. Handcock

References

Gile, Krista J., Handcock, Mark S., 2010. Respondent-driven Sampling: An Assessment of Current Methodology, Sociological Methodology, 40, 285-327. <doi:10.1111/j.1467-9531.2010.01223.x>

Gile, Krista J., Beaudry, Isabelle S. and Handcock, Mark S., 2018 Methods for Inference from Respondent-Driven Sampling Data, Annual Review of Statistics and Its Application <doi:10.1146/annurev-statistics-031017-100704>.

Salganik, M., Heckathorn, D. D., 2004. *Sampling and estimation in hidden populations using respondent-driven sampling*. Sociological Methodology 34, 193-239.

Volz, E., Heckathorn, D., 2008. *Probability based estimation theory for Respondent Driven Sampling*. The Journal of Official Statistics 24 (1), 79-97.

Examples

```
data(faux)
RDS.I.estimates(rds.data=faux,outcome.variable='X',smoothed=TRUE)
```

`RDS.SS.estimates` *Gile's SS Estimates*

Description

This function computes the sequential sampling (SS) estimates for a categorical variable or numeric variable.

Usage

```
RDS.SS.estimates(
  rds.data,
  outcome.variable,
  N = NULL,
  subset = NULL,
  number.ss.samples.per.iteration = 500,
  number.ss.iterations = 5,
  control = control.rds.estimates(),
  hajek = TRUE,
  empir.lik = TRUE,
  to.factor = FALSE
)
```

Arguments

`rds.data`	An `rds.data.frame` that indicates recruitment patterns by a pair of attributes named "id" and "recruiter.id".
`outcome.variable`	A string giving the name of the variable in the `rds.data` that contains a categorical or numeric variable to be analyzed.
`N`	An estimate of the number of members of the population being sampled. If `NULL` it is read as the `population.size.mid` attribute of the `rds.data` frame. If that is missing it defaults to 1000.
`subset`	An optional criterion to subset `rds.data` by. It is an R expression which, when evaluated, subset the data. In plain English, it can be something like `subset = seed > 0` to exclude seeds. It can also be the name of a logical vector of the same length of the outcome variable where TRUE means include it in the analysis. If `NULL` then no subsetting is done.
`number.ss.samples.per.iteration`	The number of samples to take in estimating the inclusion probabilites in each iteration of the sequential sampling algorithm. If `NULL` it is read as the eponymous attribute of `rds.data`. If that is missing it defaults to 5000.

`number.ss.iterations`
: The number of iterations of the sequential sampling algorithm. If that is missing it defaults to 5.

`control`
: A list of control parameters for algorithm tuning. Constructed using `control.rds.estimates`.

`hajek`
: logical; Use the standard Hajek-type estimator of Gile (2011) or the standard Hortitz-Thompson. The default is TRUE.

`empir.lik`
: If true, and outcome.variable is numeric, standard errors based on empirical likelihood will be given.

`to.factor`
: force variable to be a factor

Value

If `outcome.variable` is numeric then the Gile SS estimate of the mean is returned, otherwise a vector of proportion estimates is returned. If the `empir.lik` is true, an object of class `rds.interval.estimate` is returned. This is a list with components

- `estimate`: The numerical point estimate of proportion of the `trait.variable`.
- `interval`: A matrix with six columns and one row per category of `trait.variable`:
  - `point estimate`: The HT estimate of the population mean.
  - `95% Lower Bound`: Lower 95% confidence bound.
  - `95% Upper Bound`: Upper 95% confidence bound.
  - `Design Effect`: The design effect of the RDS.
  - `s.e.`: Standard error.
  - `n`: Count of the number of sample values with that value of the trait.

Otherwise, an object of class `rds.SS.estimate` is returned.

Author(s)

Krista J. Gile with help from Mark S. Handcock

References

Gile, Krista J. 2011 *Improved Inference for Respondent-Driven Sampling Data with Application to HIV Prevalence Estimation*, *Journal of the American Statistical Association*, 106, 135-146.

Gile, Krista J., Handcock, Mark S., 2010. Respondent-driven Sampling: An Assessment of Current Methodology, Sociological Methodology, 40, 285-327. <doi:10.1111/j.1467-9531.2010.01223.x>

Gile, Krista J., Beaudry, Isabelle S. and Handcock, Mark S., 2018 Methods for Inference from Respondent-Driven Sampling Data, Annual Review of Statistics and Its Application <doi:10.1146/annurev-statistics-031017-100704>.

Gile, Krista J., Handcock, Mark S., 2015 *Network Model-Assisted Inference from Respondent-Driven Sampling Data*, Journal of the Royal Statistical Society, A. <doi:10.1111/rssa.12091>.

Salganik, M., Heckathorn, D. D., 2004. *Sampling and estimation in hidden populations using respondent-driven sampling*. Sociological Methodology 34, 193-239.

Volz, E., Heckathorn, D., 2008. *Probability based estimation theory for Respondent Driven Sampling*. The Journal of Official Statistics 24 (1), 79-97.

See Also

RDS.I.estimates, RDS.II.estimates

Examples

```
data(fauxmadrona)
RDS.SS.estimates(rds.data=fauxmadrona,outcome.variable="disease",N=1000)
```

---

rdssampleC *Create RDS samples with given characteristics*

Description

Create RDS samples with given characteristics

Usage

```
rdssampleC(
  net,
  nnodes = network.size(net),
  nsamp0,
  fixinitial,
  nsamp,
  replace,
  coupons,
  select = NULL,
  bias = NULL,
  rds.samp = NULL,
  seed.distribution = NULL,
  attrall = FALSE,
  trait.variable = "disease",
  nsims = 1,
  seeds = NULL,
  prob.network.recall = 1,
  verbose = TRUE
)
```

Arguments

net	the network object from which to draw a sample
nnodes	the number of nodes in the network [at least as default]
nsamp0	the number of seeds to be drawn (i.e. the size of the 0th wave of sampling)
fixinitial	a variable that indicates the distribution from which to draw the initial seeds, if the seeds variable is NULL and the seed.distribution variable is NULL
nsamp	number of individuals in each RDS sample

replace	sampling with replacement
coupons	number of coupons
select	not used
bias	not used
rds.samp	not used
seed.distribution	a variable [what kind?] that indicates the distribution from which to draw the initial seeds
attrall	Whether all the information about the sample should be returned [??]
trait.variable	attribute of interest
nsims	number of RDS samples to draw
seeds	an array of seeds. Default is NULL, in which case the function draws the seeds from the nodes of the network.
prob.network.recall	simulates the probability that an individual will remember any particular link
verbose	Print verbose output

Value

A list with the following elements: nsample: vector of indices of sampled nodes wsample: vector of waves of each sampled node degsample: vector of degrees of sampled nodes attrsample: vector of attrs of sampled nodes toattr: vector of numbers of referrals to attrsd nodes tonoattr: vector of number of referrans to unattrsd nominators: recruiter of each sample

---

read.rdsat *Import data from the 'RDSAT' format as an* rds.data.frame

---

Description

This function imports RDSAT data files as rds.data.frame objects.

Usage

```
read.rdsat(file, delim = c("<auto>", "\t", " ", ","), N = NULL)
```

Arguments

file	the name of the file which the data are to be read from. If it does not contain an _absolute_ path, the file name is _relative_ to the current working directory, 'getwd()'. Tilde-expansion is performed where supported. As from R 2.10.0 this can be a compressed file (see 'file')
delim	The seperator defining columns. <auto> will guess the delimitor based on the file.
N	The population size (Optional).

Examples

```
fn <- paste0(path.package("RDS"),"/extdata/nyjazz.rdsat")
rd <- read.rdsat(fn)
plot(rd)
```

`read.rdsobj` *Import data saved using write.rdsobj*

Description

Import data saved using write.rdsobj

Usage

```
read.rdsobj(file)
```

Arguments

`file`	the name of the file which the data are to be read from. If it does not contain an _absolute_ path, the file name is _relative_ to the current working directory, 'getwd()'. Tilde-expansion is performed where supported. As from R 2.10.0 this can be a compressed file (see 'file')

`reingold.tilford.plot` *Plots the recruitment network using the Reingold Tilford algorithm.*

Description

Plots the recruitment network using the Reingold Tilford algorithm.

Usage

```
reingold.tilford.plot(
  x,
  vertex.color = NULL,
  vertex.color.scale = hue_pal(),
  vertex.size = 2,
  vertex.size.range = c(1, 5),
  edge.arrow.size = 0,
  vertex.label.cex = 0.2,
  vertex.frame.color = NA,
  vertex.label = get.id(x),
  show.legend = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

x	An rds.data.frame
vertex.color	The name of the categorical variable in x to color the points with.
vertex.color.scale	The scale to create the color palette.
vertex.size	The size of the vertex points. either a number or the name of a column of x.
vertex.size.range	If vertex.size represents a variable, vertex.size.range is a vector of length 2 representing the minimum and maximum cex for the points.
edge.arrow.size	The size of the arrow from recruiter to recruitee.
vertex.label.cex	The size expansion factor for the vertex.labels.
vertex.frame.color	the color of the outside of the vertex.points.
vertex.label	The name of a variable to use as vertex labels. NA implies no labels.
show.legend	If true and either vertex.color or vertex.size represent variables, legends will be displayed at the bottom of the plot.
plot	Logical, if TRUE then a plot is produced of recruitment tree. ratio statistic with the observed statistics plotted as a vertical dashed line.
...	Additional parameters passed to plot.igraph.

Value

A two-column vector of the positions of the nodes in the recruitment tree.

Examples

```
## Not run:
data(fauxmadrona)
data(faux)
reingold.tilford.plot(faux)
reingold.tilford.plot(fauxmadrona,vertex.color="disease")

## End(Not run)
```

---

rid.from.coupons *Determines the recruiter.id from recruitment coupon information*

Description

Determines the recruiter.id from recruitment coupon information

Usage

```
rid.from.coupons(
  data,
  subject.coupon = NULL,
  coupon.variables,
  subject.id = NULL,
  seed.id = "seed"
)
```

Arguments

data	a data.frame
subject.coupon	The variable representing the coupon returned by subject
coupon.variables	The variable representing the coupon ids given to the subject
subject.id	The variable representing the subject's id
seed.id	The recruiter.id to assign to seed subjects.

Examples

```
fpath <- system.file("extdata", "nyjazz.csv", package="RDS")
dat <- read.csv(fpath)
dat$recruiter.id <- rid.from.coupons(dat,"own.coupon",
                      paste0("coupon.",1:7),"id")

#create and rds.data.frame
rds <- as.rds.data.frame(dat,network.size="network.size")
```

---

set.control.class *Set the class of the control list*

Description

This function sets the class of the control list, with the default being the name of the calling function.

Usage

```
set.control.class(
  myname = as.character(RDS::ult(sys.calls(), 2)[[1L]]),
  control = get("control", pos = parent.frame())
)
```

Arguments

myname	Name of the class to set.
control	Control list. Defaults to the control variable in the calling function.

Value

The control list with class set.

See Also

check.control.class, print.control.list

`show.rds.data.frame` *Displays an rds.data.frame*

Description

Displays an rds.data.frame

Usage

```
show.rds.data.frame(x, ...)
```

Arguments

`x`	an rds.data.frame object.
`...`	additional parameters passed to print.data.frame.

`summary.svyglm.RDS` *Summarizing Generalized Linear Model Fits with Odds Ratios for Survey Data*

Description

`RDS::summary.svyglm.RDS` is a version of `summary.svyglm` that reports odds-ratios in place of coefficients in the summary table. This only applies for the `binomial` family. Otherwise it is identical to `summary.svyglm`. The default in `summary.svyglm` is to display the log-odds-ratios and this displays the exponetiated from and a 95 p-values are still displayed.

Usage

```
## S3 method for class 'svyglm.RDS'
summary(object, correlation = FALSE, df.resid = NULL, odds = TRUE, ...)
```

Arguments

`object`	an object of class `"svyglm"`, usually, a result of a call to `svyglm`.
`correlation`	logical; if `TRUE`, the correlation matrix of the estimated parameters is returned and printed.
`df.resid`	Optional denominator degrees of freedom for Wald tests.
`odds`	logical; Should the coefficients be reported as odds (rather than log-odds)?
`...`	further arguments passed to or from other methods.

Details

svyglm fits a generalised linear model to data from a complex survey design, with inverse-probability weighting and design-based standard errors.

There is no anova method for svyglm as the models are not fitted by maximum likelihood.

See the manual page on svyglm for detail of that function.

Value

RDS::summary.svyglm returns an object of class "summary.svyglm.RDS", a list with components

call	the component from object.
family	the component from object.
deviance	the component from object.
contrasts	the component from object.
df.residual	the component from object.
null.deviance	the component from object.
df.null	the component from object.
deviance.resid	the deviance residuals: see residuals.svyglm.
coefficients	the matrix of coefficients, standard errors, z-values and p-values. Aliased coefficients are omitted.
aliased	named logical vector showing if the original coefficients are aliased.
dispersion	either the supplied argument or the inferred/estimated dispersion if the latter is NULL.
df	a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of coefficients (including aliased ones).
cov.unscaled	the unscaled (dispersion = 1) estimated covariance matrix of the estimated coefficients.
cov.scaled	ditto, scaled by dispersion.
correlation	(only if correlation is true.) The estimated correlations of the estimated coefficients.
symbolic.cor	(only if correlation is true.) The value of the argument symbolic.cor.
odds	Are the coefficients reported as odds (rather than log-odds)?

See Also

svyglm, summary.

Examples

```
## For examples see example(svyglm)
```

transition.counts.to.Markov.mle

calculates the mle. i.e. the row proportions of the transition matrix

Description

calculates the mle. i.e. the row proportions of the transition matrix

Usage

```
transition.counts.to.Markov.mle(transition.counts)
```

Arguments

transition.counts	a matrix or table of transition counts

Details

depreicated. just use prop.table(transition.counts,1)

ult

*Extract or replace the *ult*imate (last) element of a vector or a list, or an element counting from the end.*

Description

Extract or replace the *ult*imate (last) element of a vector or a list, or an element counting from the end.

Usage

```
ult(x, i = 1L)
```

Arguments

x	a vector or a list.
i	index from the end of the list to extract or replace (where 1 is the last element, 2 is the penultimate element, etc.).

Value

An element of ‘x‘.

Examples

```
x <- 1:5
(last <- ult(x))
(penultimate <- ult(x, 2)) # 2nd last.
```

`vh.weights` *Volz-Heckathorn (RDS-II) weights*

Description

Volz-Heckathorn (RDS-II) weights

Usage

```
vh.weights(degs, N = NULL)
```

Arguments

`degs`	The degrees (i.e. network sizes) of the sample units.
`N`	Population size

`write.graphviz` *writes an rds.data.frame recruitment tree as a GraphViz file*

Description

writes an rds.data.frame recruitment tree as a GraphViz file

Usage

```
write.graphviz(x, file)
```

Arguments

`x`	An rds.data.frame.
`file`	A character vector representing the file

write.netdraw *Writes out the RDS tree in NetDraw format*

Description

Writes out the RDS tree in NetDraw format

Usage

```
write.netdraw(x, file = NULL, by.seed = FALSE)
```

Arguments

x	An rds.data.frame.
file	a character vector representing a file.
by.seed	If true, seperate files will be created for each seed.

Details

If by.seed is false, two files are created using 'file' as a base name. `paste0(file,".DL")` contains the edge information, and `paste0(file,".vna")` contains the nodal attributes

write.rdsat *Writes out the RDS tree in RDSAT format*

Description

Writes out the RDS tree in RDSAT format

Usage

```
write.rdsat(x, file = NULL)
```

Arguments

x	An rds.data.frame.
file	a character vector representing a file.

write.rdsobj *Export an rds.data.frame to file*

Description

Export an rds.data.frame to file

Usage

```
write.rdsobj(x, file)
```

Arguments

x	The rds.data.frame to export
file	The name of the file to create.

[.rds.data.frame *indexing*

Description

indexing

Usage

```
## S3 method for class 'rds.data.frame'
x[i, j, ..., drop, warn = TRUE]
```

Arguments

x	object
i	indices
j	indices
...	unused
drop	drop
warn	Warn if any new seeds are created

Details

Subsetting of RDS recruitment trees does not always yield a full RDS tree. In this case, subjects whose recruiter is no longer in the dataset are considered seeds. is issued if the 'warn' parameter is TRUE. dat <- data.frame(id=c(1,2,3,4,5), recruiter.id=c(2,-1,2,-1,4), network.size.variable=c(4,8,8,2,3)) r <- as.rds.data.frame(dat) r[1:3,] # A valid pruning of the RDS tree. r[c(1,5),warn=FALSE] # recruiter.id of last row set to -1 (i.e. a seed) to maintain validity of tree

`[<-.rds.data.frame` *indexing*

Description

indexing

Usage

```
## S3 replacement method for class 'rds.data.frame'
 x[i, j] <- value
```

Arguments

`x`	object
`i`	indices
`j`	indices
`value`	value

Details

Indexed assignment. If the result is not a valid rds.data.frame, an error is emitted.

Index