Package: surveybootstrap (via r-universe)

October 22, 2024

Title Bootstrap with Survey Data

Version 0.1.0.90000

Description Implements different kinds of bootstraps to estimate sampling variation from survey data with complex designs. Includes the rescaled bootstrap described in Rust and Rao (1996) <doi:10.1177/096228029600500305> and Rao and Wu (1988) <doi:10.1080/01621459.1988.10478591>.

License MIT + file LICENSE

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, stringr, dplyr, plyr, purrr, functional

Depends R (>= 2.10)

Suggests knitr, testthat

RoxygenNote 7.2.2

Encoding UTF-8

Roxygen list(markdown = TRUE)

LazyData true

Repository https://dfeehan.r-universe.dev

RemoteUrl https://github.com/dfeehan/surveybootstrap

RemoteRef HEAD

RemoteSha 74c153c48c9c44a8b157ee14dd5a4a7c012aea75

Contents

`bootstrap.estimates` *bootstrap.estimates*

Description

Use a given bootstrap method to estimate sampling uncertainty from a given estimator.

Usage

```
bootstrap.estimates(
  survey.data,
  survey.design,
  bootstrap.fn,
  estimator.fn,
  num.reps,
  weights = NULL,
  ...,
  summary.fn = NULL,
  verbose = TRUE,
  parallel = FALSE,
  paropts = NULL
)
```

Arguments

`survey.data`	The dataset to use
`survey.design`	A formula describing the design of the survey (see Details below)
`bootstrap.fn`	Name of the method to be used to take bootstrap resamples

estimator.fn	The name of a function which, given a dataset like survey.data and arguments in ..., will produce an estimate of interest
num.reps	The number of bootstrap replication samples to draw
weights	Weights to use in estimation (or NULL, if none)
...	additional arguments which will be passed on to estimator.fn
summary.fn	(Optional) Name of a function which, given the set of estimates produced by estimator.fn, summarizes them. If not specified, all of the estimates are returned in a list.
verbose	If TRUE, produce lots of feedback about what is going on
parallel	If TRUE, use the plyr library's .parallel argument to produce bootstrap resamples and estimates in parallel
paropts	If not NULL, additional arguments to pass along to the parallelization routine

Details

The formula describing the survey design should have the form ~ psu_v1 + psu_v2 + ... + strata(strata_v1 + strata_v2 + ...), where psu_v1, ... are the variables identifying primary sampling units (PSUs) and strata_v1, ... identifies the strata

Value

If summary.fn is not specified, then return the list of estimates produced by estimator.fn; if summary.fn is specified, then return its output

Examples

```
# example using a simple random sample

survey <- MU284.surveys[[1]]

estimator <- function(survey.data, weights) {
  plyr::summarise(survey.data,
                  T82.hat = sum(S82 * weights))
}

ex.mu284 <- bootstrap.estimates(
   survey.design = ~1,
   num.reps = 10,
   estimator.fn = estimator,
   weights='sample_weight',
   bootstrap.fn = 'srs.bootstrap.sample',
   survey.data=survey)

## Not run:
idu.est <- bootstrap.estimates(
  ## this describes the sampling design of the
  ## survey; here, the PSUs are given by the
  ## variable cluster, and the strata are given
  ## by the variable region
```

```
  survey.design = ~ cluster + strata(region),
  ## the number of bootstrap resamples to obtain
  num.reps=1000,
  ## this is the name of the function
  ## we want to use to produce an estimate
  ## from each bootstrapped dataset
  estimator.fn="our.estimator",
  ## these are the sampling weights
  weights="indweight",
  ## this is the name of the type of bootstrap
  ## we wish to use
  bootstrap.fn="rescaled.bootstrap.sample",
  ## our dataset
  survey.data=example.survey,
  ## other parameters we need to pass
  ## to the estimator function
  d.hat.vals=d.hat,
  total.popn.size=tot.pop.size,
  y.vals="clients",
  missing="complete.obs")


## End(Not run)
```

`chain.data` *Get a dataset from a chain*

Description

Take the data for each member of the given chain and assemble it together in a dataset.

Usage

```
chain.data(chain)
```

Arguments

`chain`	The chain to build a dataset from

Value

A dataset comprised of all of the chain's members' data put together. The order of the rows in the dataset is not specified.

chain.size *Get the size of a chain*

Description

Count the total number of respondents in the chain and return it

Usage

```
chain.size(chain)
```

Arguments

chain	The chain object

Value

The number of respondents involved in the chain

chain.vals *Get all of the values of the given variable found among members of a chain*

Description

Get all of the values of the given variable found among members of a chain

Usage

```
chain.vals(chain, qoi.var = "uid")
```

Arguments

chain	The chain to get values from
qoi.var	The name of the variable to get from each member of the chain

Value

A vector with all of the values of `qoi.var` found in this chain. (Currently, the order of the values in the vector is not guaranteed.)

`estimate.degree.distns` *Estimate degree distributions by trait*

Description

Break down RDS degree distributions by trait, and return an object which has the degrees for each trait as well as functions to draw degrees from each trait.

Usage

```
estimate.degree.distns(survey.data, d.hat.vals, traits, keep.vars = NULL)
```

Arguments

`survey.data`	The respondent info
`d.hat.vals`	The variable that contains the degrees for each respondent
`traits`	A vector of the names of the columns of `survey.data` which refer to the traits
`keep.vars`	Additional vars to return along with degrees

Details

One of the items returned as a result is a function, `draw.degrees.fn`, which takes one argument, `traits`. This is a vector of traits and, for each entry in this vector, `draw.degress.fn` returns a draw from the empirical distribution of degrees among respondents with that trait. So, `draw.degrees.fn(c("0.0", "0.1", "0.1")` would return a degree drawn uniformly at random from among the observed degrees of respondents with trait "0.0" and then two degrees from respondents with trait "0.1"

Value

An object with

- `distns` a list with one entry per trait value; each
- `draw.degrees.fn` a function which gets called with one
- `keep.vars` the name of the other vars that are kept (if any)

estimate.mixing *Construct a mixing model from GoC/RDS data*

Description

Given a dataset with the respondents and a dataset on the parents (in many cases the same individuals), and a set of relevant traits, estimate mixing parameters and return a markov model.

Usage

```
estimate.mixing(survey.data, parent.data, traits)
```

Arguments

survey.data	The respondent info
parent.data	The parent info
traits	The names of the traits to build the model on

Value

A list with entries:

- mixing.df the data used to estimate the mixing function
- choose.next.state.fn a function which can be passed a vector of states and will return a draw of a subsequent state for each entry in the vector
- mixing.df a dataframe (long-form) representation of the transition counts used to estimate the transition probabilities
- states a list with an entry for each state. within each state's entry are
  - trans.probs a vector of estimated transition probabilities
  - trans.fn a function which, when called, randomly chooses a next state with probabilities given by the transition probs.

is.child.ct *Determine whether or not one id is a parent of another*

Description

This function allows us to determine which ids are directly descended from which other ones. It is the only part of the code that relies on the ID format used by the Curitiba study (see Details); by modifying this function, it should be possible to adapt this code to another study.

Usage

```
is.child.ct(id, seed.id)
```

Arguments

id	the id of the potential child
seed.id	the id of the potential parent

Details

See:

- Salganik, M. J., Fazito, D., Bertoni, N., Abdo, A. H., Mello, M. B., & Bastos, F. I. (2011). Assessing network scale-up estimates for groups most at risk of HIV/AIDS: evidence from a multiple-method study of heavy drug users in Curitiba, Brazil. *American journal of epidemiology*, 174(10), 1190-1196.

Value

TRUE if id is the direct descendant of seed.id and FALSE otherwise

---

make.chain *Build an RDS seed's chain from the dataset*

Description

Note that this assumes that the chain is a tree (no loops)

Usage

```
make.chain(seed.id, survey.data, is.child.fn = is.child.ct)
```

Arguments

seed.id	The id of the seed whose chain we wish to build from the dataset
survey.data	The dataset
is.child.fn	A function which takes two ids as arguments; it is expected to return TRUE if the second argument is the parent of the first, and FALSE otherwise. it defaults to is.child.ct()

Value

info

max.depth *Get the height (maximum depth) of a chain*

Description

Get the height (maximum depth) of a chain

Usage

```
## S3 method for class 'depth'
max(chain)
```

Arguments

chain	The chain object

Value

The maximum depth of the chain

mc.sim *Run a markov model*

Description

Run a given markov model for n time steps, starting at a specified state.

Usage

```
mc.sim(mm, start, n)
```

Arguments

mm	The markov model object returned by estimate.mixing()
start	The name of the state to start in
n	The number of time-steps to run through

Details

This uses the markov model produced by estimate.mixing()

Value

A vector with the state visited at each time step. the first entry has the starting state

MU284 *The MU284 Population dataset*

Description

A dataset containing information about Sweden's 284 municipalities. This dataset comes from Model-Assisted Survey Sampling by Sarndal, Swensson, and Wretman (2003, ISBN:0387406204). The columns are:

Format

A data frame with 284 rows and 11 columns:

LABEL ID

P85 Population in 1985

P75 Population in 1975

RMT85 Municipal tax revenue in 1985

CS82 Number of Conservative seats in municipal council

SS82 Number of Social-Democratic seats in municipal council

S82 Total number of seats in municipal council

ME84 Number of municipal employees

REV84 Real estate values according to 1984 assessment

REG Geographic location indicator

CL Cluster indicator (neighboring municipalities are clustered together)

Source

'Model Assisted Survey Sampling' by Sarndal, Swensson, and Wretman (2003, ISBN:0387406204)

MU284.boot.res.summ *Benchmarks for unit tests*

Description

Benchmark results to use in unit tests; these are based on MU284.complex.surveys.

Format

A list with 10 data frames, each with 15 rows and 11 columns:

mean.TS82.hat, ..., sd.R.RMT85.P85.hat summaries for each estimand

MU284.complex.surveys *Simulated sample surveys drawn from the MU284 Population using a complex design*

Description

A list with 10 sample surveys with sample size 15 drawn from the MU284 dataset using a complex sampling design.

Format

A list with 10 data frames, each with 15 rows and 11 columns:

LABEL, ..., CL Same as MU284 dataset

sample_weight The sampling weight for the row

Details

The sampling design comes from Ex. 4.3.2 (pg 142-3) of 'Model Assisted Survey Sampling' by Sarndal, Swensson, and Wretman (2003, ISBN:0387406204).

The design is a two-stage sample:

- stage I: the primary sampling units (PSUs) are the standard clusters from MU284; we take a simple random sample without replacement of `n_I = 5` out of `N_I = 50` of these
- stage II: within each sampled PSU, we take a simple random sample without replacement of `n_i = 3` out of `N_i` municipalities

MU284.estimator.fn *MU284.estimator.fn*

Description

Produce estimates from a simulated sample survey of the MU284 population. Used in package tests and examples.

Usage

```
MU284.estimator.fn(survey.data, weights)
```

Arguments

`survey.data`	the survey dataset
`weights`	a vector with the survey weights

Value

a data.frame with one row and two columns:

- `TS82.hat` - the estimated total of `S82`
- `R.RMT85.P85.hat` - the estimated ratio of `RMT85` / `P85`

`MU284.estimator.summary.fn` *MU284.estimator.summary.fn*

Description

Summarize results from `MU284.estimator.fn()` applied to many surveys. (This is a dummy function, used for tests)

Usage

```
MU284.estimator.summary.fn(res)
```

Arguments

`res`	a dataframe whose rows are the results of calling `MU284.estimator.fn()`

Value

the same dataframe

`MU284.surveys` *Simulated sample surveys drawn from the MU284 Population*

Description

A list with 10 sample surveys with sample size 15 drawn from the MU284 dataset using simple random sampling with replacment.

Format

A list with 10 data frames, each with 15 rows and 11 columns:

LABEL, ..., CL Same as MU284 dataset

sample_weight The sampling weight for the row

rds.boot.draw.chain *Draw RDS bootstrap resamples for one chain*

Description

This function uses the algorithm described in the supporting online material for Weir et al 2012 (see Details) to take bootstrap resamples of one chain from an RDS dataset.

Usage

```
rds.boot.draw.chain(chain, mm, dd, parent.trait, idvar = "uid")
```

Arguments

chain	The chain to draw resamples for
mm	The mixing model to use
dd	The degree distns to use
parent.trait	A vector whose length is the number of bootstrap reps we want
idvar	The name of the variable used to label the columns of the output (presumably some id identifying the row in the original dataset they come from)

Details

See

- Weir, Sharon S., et al. "A comparison of respondent-driven and venue-based sampling of female sex workers in Liuzhou, China." *Sexually transmitted infections* 88.Suppl 2 (2012): i95-i101.

Value

A list of dataframes with one entry for each respondent in the chain. each dataframe has one row for each bootstrap replicate. so if we take 10 bootstrap resamples of a chain of length 50, there will be 50 entries in the list that is returned. each entry will be a dataframe with 10 rows.

rds.chain.boot.draws *Draw RDS bootstrap resamples*

Description

Draw bootstrap resamples for an RDS dataset, using the algorithm described in the supporting online material of Weir et al 2012 (see `rds.boot.draw.chain()`).

Usage

```
rds.chain.boot.draws(chains, mm, dd, num.reps, keep.vars = NULL)
```

Arguments

chains	A list whose entries are the chains we want to resample
mm	The mixing model
dd	The degree distributions
num.reps	The number of bootstrap resamples we want
keep.vars	If not NULL, then the names of variables from the original dataset we want appended to each bootstrap resampled dataset (default is NULL)

Value

A list of length num.reps; each entry in the list has one bootstrap-resampled dataset

---

rds.mc.boot.draws *Draw RDS bootstrap resamples using the algorithm in Salganik 2006 (see Details below)*

Description

This algorithm picks a respondent from the survey to be a seed uniformly at random. it then generates a bootstrap draw by simulating the markov process forward for n steps, where n is the size of the draw required.

If you wish the bootstrap dataset to end up with variables from the original dataset other than the traits and degree, then you must specify this when you construct dd using the 'estimate.degree.distns function.

Usage

```
rds.mc.boot.draws(chains, mm, dd, num.reps)
```

Arguments

chains	A list with the chains constructed from the survey using make.chain
mm	The mixing model
dd	The degree distributions
num.reps	The number of bootstrap resamples we want

Details

See:

- Salganik, Matthew J. "Variance estimation, design effects, and sample size calculations for respondent-driven sampling." *Journal of Urban Health* 83.1 (2006): 98-112.

Value

A list of length num.reps; each entry in the list has one bootstrap-resampled dataset

rescaled.bootstrap.sample *rescaled.bootstrap.sample*

Description

Given a survey dataset and a description of the survey design (ie, which combination of variables determines primary sampling units, and which combination of variables determines strata), take a bunch of bootstrap samples for the rescaled bootstrap estimator (see Details).

Usage

```
rescaled.bootstrap.sample(
  survey.data,
  survey.design,
  parallel = FALSE,
  paropts = NULL,
  num.reps = 1
)
```

Arguments

survey.data	The dataset to use
survey.design	A formula describing the design of the survey (see Details)
parallel	If TRUE, use parallelization (via plyr)
paropts	An optional list of arguments passed on to plyr to control details of parallelization
num.reps	The number of bootstrap replication samples to draw

Details

survey.design is a formula of the form

weight ~ psu_vars + strata(strata_vars)

where:

- weight is the variable with the survey weights
- psu_vars has the form psu_v1 + psu_v2 + ..., where primary sampling units (PSUs) are determined by psu_v1, etc
- strata_vars has the form strata_v1 + strata_v2 + ..., which determine strata

Note that we assume that the formula uniquely specifies PSUs. This will always be true if the PSUs were selected without replacement. If they were selected with replacement, then it will be necessary to make each realization of a given PSU in the sample a unique id. The code below assumes that all observations within each PSU (as identified by the design formula) are from the same draw of the PSU.

The rescaled bootstrap technique works by adjusting the estimation weights based on the number of times each row is included in the resamples. If a row is never selected, it is still included in the returned results, but its weight will be set to 0. It is therefore important to use estimators that make use of the estimation weights on the resampled datasets.

We always take m_i = n_i - 1, according to the advice presented in Rao and Wu (1988) and Rust and Rao (1996).

(This is a C++ version; a previous version, written in pure R, is called `rescaled.bootstrap.sample.pureR()`)

References:

- Rust, Keith F., and J. N. K. Rao. "Variance estimation for complex surveys using replication techniques." *Statistical methods in medical research* 5.3 (1996): 283-310.
- Rao, Jon NK, and C. F. J. Wu. "Resampling inference with complex survey data." *Journal of the American Statistical Association* 83.401 (1988): 231-241.

Value

A list with `num.reps` entries. Each entry is a dataset which has at least the variables `index` (the row index of the original dataset that was resampled) and `weight.scale` (the factor by which to multiply the sampling weights in the original dataset).

Examples

```
survey <- MU284.complex.surveys[[1]]
boot_surveys <- rescaled.bootstrap.sample(survey.data = survey,
                                          survey.design = ~ CL,
                                          num.reps = 2)
```

`rescaled.bootstrap.sample.pureR`
rescaled.bootstrap.sample.pureR

Description

(this is the pure R version; it has been supplanted by `rescaled.bootstrap.sample`, which is partially written in C++)

Usage

```
rescaled.bootstrap.sample.pureR(
  survey.data,
  survey.design,
  parallel = FALSE,
  paropts = NULL,
  num.reps = 1
)
```

Arguments

survey.data	the dataset to use
survey.design	a formula describing the design of the survey (see below - TODO)
parallel	if TRUE, use parallelization (via plyr)
paropts	an optional list of arguments passed on to plyr to control details of parallelization
num.reps	the number of bootstrap replication samples to draw

Details

given a survey dataset and a description of the survey design (ie, which combination of vars determines primary sampling units, and which combination of vars determines strata), take a bunch of bootstrap samples for the rescaled bootstrap estimator (see, eg, Rust and Rao 1996).

Note that we assume that the formula uniquely specifies PSUs. This will always be true if the PSUs were selected without replacement. If they were selected with replacement, then it will be necessary to make each realization of a given PSU in the sample a unique id. Bottom line: the code below assumes that all observations within each PSU (as identified by the design formula) are from the same draw of the PSU.

The rescaled bootstrap technique works by adjusting the estimation weights based on the number of times each row is included in the resamples. If a row is never selected, it is still included in the returned results, but its weight will be set to 0. It is therefore important to use estimators that make use of the estimation weights on the resampled datasets.

We always take m_i = n_i - 1, according to the advice presented in Rao and Wu (1988) and Rust and Rao (1996).

survey.design is a formula of the form
weight ~ psu_vars + strata(strata_vars), where weight is the variable with the survey weights and psu is the variable denoting the primary sampling unit

Value

a list with num.reps entries. each entry is a dataset which has at least the variables index (the row index of the original dataset that was resampled) and weight.scale (the factor by which to multiply the sampling weights in the original dataset).

---

rescaled.bootstrap.weights

rescaled.bootstrap.weights

Description

This function creates a dataset with rescaled bootstrap weights; it can be a helpful alternative to bootstrap.estimates in some situations

Usage

```
rescaled.bootstrap.weights(
  survey.data,
  survey.design,
  num.reps,
  weights = NULL,
  idvar,
  verbose = TRUE,
  parallel = FALSE,
  paropts = NULL
)
```

Arguments

survey.data	The dataset to use
survey.design	A formula describing the design of the survey (see Details in bootstrap.estimates() help page)
num.reps	the number of bootstrap replication samples to draw
weights	weights to use in estimation (or NULL, if none)
idvar	the name of the column in survey.data that has the respondent id
verbose	if TRUE, produce lots of feedback about what is going on
parallel	if TRUE, use the plyr library's .parallel argument to produce bootstrap resamples and estimates in parallel
paropts	if not NULL, additional arguments to pass along to the parallelization routine

Details

The formula describing the survey design should have the form ~ psu_v1 + psu_v2 + ... + strata(strata_v1 + strata_v2 + ...), where psu_v1, ... are the variables identifying primary sampling units (PSUs) and strata_v1, ... identify the strata

Value

if no summary.fn is specified, then return the list of estimates produced by estimator.fn; if summary.fn is specified, then return its output

Examples

```
survey <- MU284.complex.surveys[[1]]
rescaled.bootstrap.weights(survey.data = survey,
                           survey.design = ~ CL,
                           weights='sample_weight',
                           idvar='LABEL',
                           num.reps = 2)


## Not run:
```

```
bootweights <- rescaled.bootstrap.weights(
                                        # formula describing survey design:
                                        # psu and strata
                                        survey.design = ~ psu +
                                                        stratum(stratum_analysis),
                                        num.reps=10000,
                                        # column with respondent ids
                                        idvar='caseid',
                                        # column with sampling weight
                                        weights='wwgt',
                                        # survey dataset
                                        survey.data=mw.ego)


## End(Not run)
```

`srs.bootstrap.sample` *srs.bootstrap.sample*

Description

Given a survey dataset and a description of the survey design (ie, which combination of vars determines primary sampling units, and which combination of vars determines strata), take a bunch of bootstrap samples under a simple random sampling (with repetition) scheme

Usage

```
srs.bootstrap.sample(
  survey.data,
  num.reps = 1,
  parallel = FALSE,
  paropts = NULL,
  ...
)
```

Arguments

`survey.data`	The dataset to use
`num.reps`	The number of bootstrap replication samples to draw
`parallel`	If `TRUE`, use parallelization (via `plyr`)
`paropts`	An optional list of arguments passed on to `plyr` to control details of parallelization
`...`	Ignored, but useful because it allows params like `survey.design` which are used in other bootstrap designs, to be passed in without error

Value

A list with `num.reps` entries. Each entry is a dataset which has at least the variables `index` (the row index of the original dataset that was resampled) and `weight.scale` (the factor by which to multiply the sampling weights in the original dataset).

Examples

```
survey <- MU284.surveys[[1]]
boot_surveys <- srs.bootstrap.sample(survey, num.reps = 2)
```

`surveybootstrap` *Survey bootstrap variance estimators*

Description

`surveybootstrap` has methods for analyzing data that were collected using network reporting techniques. It includes estimators appropriate for the simple bootstrap and the rescaled bootstrap.

Index