Package ‘drpop’

November 5, 2021

Title Efficient and Doubly Robust Population Size Estimation

Version 0.0.3

Description Estimation of the total population size from capture-recapture data efficiently and with low bias implementing the methods from Das M, Kennedy EH, and Jewell NP (2021) <arXiv:2104.14091>. The estimator is doubly robust against errors in the estimation of the intermediate nuisance parameters. Users can choose from the flexible estimation models provided in the package, or use any other preferred model.

Depends stats

Imports gam, janitor, reshape2, stringr, tidyr, dplyr, SuperLearner, ggplot2, nnet, nnls, parallel, ranger

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Suggests testthat (>= 2.0.0)

Config/testthat/edition 2

NeedsCompilation no

Author Manjari Das [aut, cre] (<https://orcid.org/0000-0002-6781-6368>), Edward H. Kennedy [aut] (<https://orcid.org/0000-0002-1510-8175>)

Maintainer Manjari Das <manjari8d@gmail.com>

Repository CRAN

Date/Publication 2021-11-05 21:10:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>informat</td>
<td>2</td>
</tr>
<tr>
<td>plotci</td>
<td>3</td>
</tr>
<tr>
<td>popsize</td>
<td>4</td>
</tr>
<tr>
<td>popsize_cond</td>
<td>6</td>
</tr>
<tr>
<td>popsize_simul</td>
<td>8</td>
</tr>
<tr>
<td>qhat_gam</td>
<td>10</td>
</tr>
<tr>
<td>qhat_logit</td>
<td>11</td>
</tr>
</tbody>
</table>
informat

A function to check whether a given data table/matrix/data frame is in the appropriate for drpop.

Description

A function to check whether a given data table/matrix/data frame is in the appropriate for drpop.

Usage

informat(data, K = 2)

Arguments

data The data table/matrix/data frame which is to be checked.
K The number of lists (optional).

Value

A boolean for whether data is in the appropriate format.

Examples

data = matrix(sample(c(0,1), 2000, replace = TRUE), ncol = 2)
x = matrix(rnorm(nrow(data)*3, 2,1), nrow = nrow(data))
informat(data = data)
#this returns TRUE

data = cbind(data, x)
informat(data = data)
#this returns TRUE

informat(data = data, K = 3)
#this returns FALSE
Plot estimated confidence interval of total population size from object of class popsize or popsize_cond.

Description
Plot estimated confidence interval of total population size from object of class popsize or popsize_cond.

Usage
plotci(object, tsize = 12, ...)

Arguments
- object: An object of class popsize or popsize_cond.
- tsize: The text size for the plots.
- ...: Any extra arguments passed into the function.

Value
A ggplot object fig with population size estimates and the 95% confidence intervals.

References

Examples
```r
data = simuldata(n = 10000, l = 1)$data_xstar
p = popsize(data = data, funcname = c("logit", "gam"))
plotci(p)

data = simuldata(n = 10000, l = 1, categorical = TRUE)$data_xstar
p = popsize_cond(data = data, condvar = 'catcov')
plotci(p)
```
popsieze

**Estimate total population size and capture probability using user provided set of models or user provided nuisance estimates.**

**Description**

Estimate total population size and capture probability using user provided set of models or user provided nuisance estimates.

**Usage**

```r
popsieze(
data, K = 2, j, k, margin = 0.005, filterrows = FALSE, nfolds = 5, funcname = c("rangerlogit"), sl.lib = c("SL.gam", "SL.glm", "SL.glm.interaction", "SL.ranger", "SL.glmnet"), getnuis, q1mat, q2mat, q12mat, idfold, TMLE = TRUE, PLUGIN = TRUE, Nmin = 100, ... )
```

**Arguments**

- **data**
  - The data frame in capture-recapture format with K lists for which total population is to be estimated. The first K columns are the capture history indicators for the K lists. The remaining columns are covariates in numeric format.
- **K**
  - The number of lists that are present in the data.
- **j**
  - The first list to be used for estimation.
- **k**
  - The second list to be used in the estimation.
- **margin**
  - The minimum value the estimates can attain to bound them away from zero.
- **filterrows**
  - A logical value denoting whether to remove all rows with only zeroes.
- **nfolds**
  - The number of folds to be used for cross fitting.
- **funcname**
  - The vector of estimation function names to obtain the population size.
Algorithm library for \texttt{qhat\_sl()}. Default library includes "gam", "glm", "glmnet", "glm.interaction", "ranger".

getnuis

A list object with the nuisance function estimates and the fold assignment of the rows for cross-fitting or a data.frame with the nuisance estimates.

q1mat

A dataframe with capture probabilities for the first list.

q2mat

A dataframe with capture probabilities for the second list.

q12mat

A dataframe with capture probabilities for both the lists simultaneously.

idfold

The fold assignment of each row during estimation.

TMLE

The logical value to indicate whether TMLE has to be computed.

PLUGIN

The logical value to indicate whether the plug-in estimates are returned.

Nmin

The cutoff for minimum sample size to perform doubly robust estimation. Otherwise, Petersen estimator is returned.

... Any extra arguments passed into the function. See \texttt{qhat\_rangerlogit()}, \texttt{qhat\_sl()}, \texttt{tmle()}.

Value

A list of estimates containing the following components for each list-pair, model and method (PI = plug-in, DR = doubly-robust, TMLE = targeted maximum likelihood estimate):

result

A dataframe of the below estimated quantities.

- \( \psi \) The estimated capture probability.
- \( \sigma \) The efficiency bound.
- \( n \) The estimated population size \( n \).
- \( \sigma_n \) The estimated standard deviation of the population size.
- \( \text{cin.l} \) The estimated lower bound of a 95\% confidence interval of \( n \).
- \( \text{cin.u} \) The estimated upper bound of a 95\% confidence interval of \( n \).

N

The number of data points used in the estimation after removing rows with missing data.

ifvals

The estimated influence function values for the observed data.

nuis

The estimated nuisance functions (q12, q1, q2) for each element in funcname.

nuistmle

The estimated nuisance functions (q12, q1, q2) from \texttt{tmle} for each element in funcname.

idfold

The division of the rows into sets (folds) for cross-fitting.

References


Examples

```r
data = simuldata(1000, l = 3)$data
qhat = popsize(data = data, funcname = c("logit", "gam"), nfolds = 2, margin = 0.005)
psin_estimate = popsize(data = data, getnuis = qhat$nuis, idfold = qhat$idfold)

data = simuldata(n = 6000, l = 3)$data
psin_estimate = popsize(data = data[,1:2])
#this returns the basic plug-in estimate since covariates are absent.

psin_estimate = popsize(data = data, funcname = c("gam", "rangerlogit"))
```

---

**popsize_cond**

*Estimate total population size and capture probability using user provided set of models conditioned on an attribute.*

---

**Description**

Estimate total population size and capture probability using user provided set of models conditioned on an attribute.

**Usage**

```r
popsize_cond(
  data,
  K = 2,
  filterrows = FALSE,
  funcname = c("rangerlogit"),
  condvar,
  nfolds = 2,
  margin = 0.005,
  sl.lib = c("SL.gam", "SL.glm", "SL.glm.interaction", "SL.ranger", "SL.glmnet"),
  TMLE = TRUE,
  PLUGIN = TRUE,
  Nmin = 100,
  ...
)
```
Arguments

data: The data frame in capture-recapture format for which total population is to be estimated. The first K columns are the capture history indicators for the K lists. The remaining columns are covariates in numeric format.

K: The number of lists in the data. Typically the first K rows of data.

filterrows: A logical value denoting whether to remove all rows with only zeroes.

funcname: The vector of estimation function names to obtain the population size.

condvar: The covariate for which conditional estimates are required.

nfold: The number of folds to be used for cross fitting.

margin: The minimum value the estimates can attain to bound them away from zero.

sl.lib: Algorithm library for qhat_sl(). See SuperLearner::listWrappers(). Default library includes "gam", "glm", "glmnet", "glm.interaction", "ranger".

TMLE: The logical value to indicate whether TMLE has to be computed.

PLUGIN: The logical value to indicate whether the plug-in estimates are returned.

Nmin: The cutoff for minimum sample size to perform doubly robust estimation. Otherwise, Petersen estimator is returned.

... Any extra arguments passed into the function. See qhat_rangerlogit(), qhat_sl(), tmle().

Value

A list of estimates containing the following components for each list-pair, model and method (PI = plug-in, DR = doubly-robust, TMLE = targeted maximum likelihood estimate):

result: A dataframe of the below estimated quantities.

• psi: The estimated capture probability.
• sigma: The efficiency bound.
• n: The estimated population size n.
• sigman: The estimated standard deviation of the population size.
• cin.l: The estimated lower bound of a 95% confidence interval of n.
• cin.u: The estimated upper bound of a 95% confidence interval of n.

N: The number of data points used in the estimation after removing rows with missing data.

ifvals: The estimated influence function values for the observed data.

nuis: The estimated nuisance functions (q12, q1, q2) for each element in funcname.

nuistmle: The estimated nuisance functions (q12, q1, q2) from tmle for each element in funcname.

idfold: The division of the rows into sets (folds) for cross-fitting.

References

**popsize_simul**

Estimate the total population size and capture probabilities using perturbed true nuisance functions.

**Description**

Estimate the total population size and capture probabilities using perturbed true nuisance functions.

**Usage**

```r
popsize_simul(
  data,
  n,
  K = 2,
  nfolds = 5,
  pi1,
  pi2,
  omega,
  alpha,
  margin = 0.005,
  iter = 100,
  twolist = TRUE
)
```

**Arguments**

- `data` The data frame in capture-recapture format for which total population is to be estimated. The first K columns are the capture history indicators for the K lists. The remaining columns are covariates in numeric format.
- `n` The true population size. Required to calculate the added error.
- `K` The number of lists in the data. Typically the first K rows of data.
- `nfolds` The number of folds to be used for cross fitting.

**Examples**

```r
data = simuldata(n = 10000, l = 2, categorical = TRUE)$data

psin_estimate = popsize_cond(data = data, funcname = c("logit", "gam"),
  condvar = 'catcov', PLUGIN = TRUE, TMLE = TRUE)
# this returns the plug-in, the bias-corrected and the tmle estimate for the
# two models conditioned on column catcov
```

**See Also**

`popsize`
The function to calculate the conditional capture probabilities of list 1 using covariates.

The function to calculate the conditional capture probabilities of list 2 using covariates.

The standard deviation from zero of the added error.

The rate of convergence. Takes values in (0, 1].

The minimum value the estimates can attain to bound them away from zero.

An integer denoting the maximum number of iterations allowed for targeted maximum likelihood method.

The logical value of whether targeted maximum likelihood algorithm fits only two modes when K = 2.

A list of estimates containing the following components:

A matrix of the estimated capture probability for each list pair, model and method combination. In the absence of covariates, the column represents the standard plug-in estimate. The rows represent the list pair which is assumed to be independent conditioned on the covariates. The columns represent the model and method combinations (PI = plug-in, DR = bias-corrected, TMLE = targeted maximum likelihood estimate) indicated in the columns.

A matrix of the efficiency bound \( \sigma^2 \) in the same format as psi.

A matrix of the estimated population size \( n \) in the same format as psi.

A matrix of the variance for population size estimate in the same format as psi.

The number of data points used in the estimation after removing rows with missing data.


### Examples

```r
simulresult = simuldata(n = 2000, l = 2)
data = simulresult$data

psin_estimate = popsize_simul(data = data,
pi1 = simulresult$pi1, pi2 = simulresult$pi2,
alpha = 0.25, omega = 1)
```
**qhat_gam**

Estimate marginal and joint distribution of lists \( j \) and \( k \) using generalized additive models.

**Description**

Estimate marginal and joint distribution of lists \( j \) and \( k \) using generalized additive models.

**Usage**

`qhat_gam(List.train, List.test, K = 2, j = 1, k = 2, margin = 0.005, ...)`

**Arguments**

- `List.train`: The training data matrix used to estimate the distribution functions.
- `List.test`: The data matrix on which the estimator function is applied.
- `K`: The number of lists in the data.
- `j`: The first list that is conditionally independent.
- `k`: The second list that is conditionally independent.
- `margin`: The minimum value the estimates can attain to bound them away from zero.
- `...`: Any extra arguments passed into the function.

**Value**

A list of the marginal and joint distribution probabilities \( q_1 \), \( q_2 \) and \( q_{12} \).

**References**


**Examples**

```r
## Not run:
qhat = qhat_gam(List.train = List.train, List.test = List.test, margin = 0.005)
q1 = qhat$q1
q2 = qhat$q2
q12 = qhat$q12

## End(Not run)
```
qhat_logit

Estimate marginal and joint distribution of lists j and k using logistic regression.

Description

Estimate marginal and joint distribution of lists j and k using logistic regression.

Usage

qhat_logit(List.train, List.test, K = 2, j = 1, k = 2, margin = 0.005, ...)

Arguments

- List.train: The training data matrix used to estimate the distribution functions.
- List.test: The data matrix on which the estimator function is applied.
- K: The number of lists in the data.
- j: The first list that is conditionally independent.
- k: The second list that is conditionally independent.
- margin: The minimum value the estimates can attain to bound them away from zero.
- ...: Any extra arguments passed into the function.

Value

A list of the marginal and joint distribution probabilities q1, q2 and q12.

Examples

```r
## Not run:
qhat = qhat_logit(List.train = List.train, List.test = List.test, margin = 0.005)
q1 = qhat$q1
q2 = qhat$q2
q12 = qhat$q12

## End(Not run)
```
qhat_mlogit

Estimate marginal and joint distribution of lists j and k using multinomial logistic model.

Description

Estimate marginal and joint distribution of lists j and k using multinomial logistic model.

Usage

qhat_mlogit(List.train, List.test, K = 2, j = 1, k = 2, margin = 0.005, ...)

Arguments

- **List.train**: The training data matrix used to estimate the distribution functions.
- **List.test**: The data matrix on which the estimator function is applied.
- **K**: The number of lists in the data.
- **j**: The first list that is conditionally independent.
- **k**: The second list that is conditionally independent.
- **margin**: The minimum value the estimates can attain to bound them away from zero.
- **...**: Any extra arguments passed into the function.

Value

A list of the marginal and joint distribution probabilities q1, q2 and q12.

References


Examples

```r
## Not run:
qhat = qhat_mlogit(List.train = List.train, List.test = List.test, margin = 0.005)
q1 = qhat$q1
q2 = qhat$q2
q12 = qhat$q12

## End(Not run)
```
qhat_ranger

Estimate marginal and joint distribution of lists j and k using ranger.

Description

Estimate marginal and joint distribution of lists j and k using ranger.

Usage

qhat_ranger(List.train, List.test, K = 2, j = 1, k = 2, margin = 0.005, ...)

Arguments

List.train The training data matrix used to estimate the distribution functions.
List.test The data matrix on which the estimator function is applied.
K The number of lists in the data.
j The first list that is conditionally independent.
k The second list that is conditionally independent.
margin The minimum value the estimates can attain to bound them away from zero.
... Any extra arguments passed into the function.

Value

A list of the marginal and joint distribution probabilities q1, q2 and q12.

References


Examples

## Not run:
qhat = qhat_ranger(List.train = List.train, List.test = List.test, margin = 0.005)
q1 = qhat$q1
q2 = qhat$q2
q12 = qhat$q12

## End(Not run)
qhat_rangerlogit

Estimate marginal and joint distribution of lists j and k using ensemble of ranger and logit.

Description

Estimate marginal and joint distribution of lists j and k using ensemble of ranger and logit.

Usage

qhat_rangerlogit(
  List.train,
  List.test,
  K = 2,
  j = 1,
  k = 2,
  margin = 0.005,
  ...
)

Arguments

List.train The training data matrix used to estimate the distribution functions.
List.test The data matrix on which the estimator function is applied.
K The number of lists in the data.
j The first list that is conditionally independent.
k The second list that is conditionally independent.
margin The minimum value the estimates can attain to bound them away from zero.
... Any extra arguments passed into the function.

Value

A list of the marginal and joint distribution probabilities q1, q2 and q12.

References


## Examples

```r
## Not run:
qhat = qhat_ranger(List.train = List.train, List.test = List.test, margin = 0.005)
q1 = qhat$q1
q2 = qhat$q2
q12 = qhat$q12

## End(Not run)
```

### qhat_sl

Estimate marginal and joint distribution of lists j and k using super learner.

**Description**

Estimate marginal and joint distribution of lists j and k using super learner.

**Usage**

```r
qhat_sl(
  List.train,
  List.test,
  K = 2,
  j = 1,
  k = 2,
  margin = 0.005,
  sl.lib = c("SL.glm", "SL.gam", "SL.glm.interaction", "SL.ranger", "SL.glmnet"),
  num_cores = NA,
  ...
)
```

**Arguments**

- `List.train`  The training data matrix used to estimate the distribution functions.
- `List.test`  The data matrix on which the estimator function is applied.
- `K`  The number of lists in the data.
- `j`  The first list that is conditionally independent.
- `k`  The second list that is conditionally independent.
- `margin`  The minimum value the estimates can attain to bound them away from zero.
- `sl.lib`  The functions from the SuperLearner library to be used for model fitting. See `SuperLearner::listWrappers()`.
- `num_cores`  The number of cores to be used for parallelization in Super Learner.
- `...`  Any extra arguments passed into the function.
Value
A list of the marginal and joint distribution probabilities q1, q2 and q12.

References

Examples
```r
## Not run:
qhat = qhat_sl(List.train = List.train, List.test = List.test, margin = 0.005, num_cores = 1)
q1 = qhat$q1
q2 = qhat$q2
q12 = qhat$q12
## End(Not run)
```

reformat
A function to reorder the columns of a data table/matrix/data frame and to change factor variables to numeric.

Description
A function to reorder the columns of a data table/matrix/data frame and to change factor variables to numeric.

Usage
```r
reformat(data, capturelists)
```

Arguments
- `data`: The data table/matrix/data frame which is to be checked.
- `capturelists`: The vector of column names or locations for the capture history list columns.

Value
/data With reordered columns so that the capture history columns are followed by the rest.

Examples
```r
data = matrix(sample(c(0,1), 2000, replace = TRUE), ncol = 2)
x = matrix(rnorm(nrow(data)*3, 2, 1), nrow = nrow(data))
data = cbind(x, data)
result<- reformat(data = data, capturelists = c(4,5))
```
simuldata

A function to reorder the columns of a data table/matrix/data frame and to change factor variables to numeric.

Description
A function to reorder the columns of a data table/matrix/data frame and to change factor variables to numeric.

Usage
simuldata(n, l, categorical = FALSE, ep = 0, K = 2)

Arguments
- n: The size of the population.
- l: The number of continuous covariates.
- categorical: A logical value of whether to include a categorical column.
- ep: A numeric value to change the list probabilities.
- K: The number of lists. Default value is 2. Maximum value is 3.

Value
A list of estimates containing the following components:
- data: A dataframe in with K list capture histories and covariates from a population if true size n with only observed rows.
- data_xstar: A dataframe in with two list capture histories and transformed covariates from a population if true size n with only observed rows.
- psi0: The empirical capture probability for the set-up used.
- pi1: The conditional capture probabilities for list 1.
- pi2: The conditional capture probabilities for list 2.
- pi3: The conditional capture probabilities for list 3 when K = 3.

References

Examples
data = simuldata(n = 1000, l = 2)$data
psi0 = simuldata(n = 10000, l = 2)$psi0
tmle

Returns the targeted maximum likelihood estimates for the nuisance functions

Description

Returns the targeted maximum likelihood estimates for the nuisance functions

Usage

```r
tmle(
  datmat,
  iter = 250,
  margin = 0.005,
  stop_margin = 0.005,
  twolist = FALSE,
  K = 2,
  ...
)
```

Arguments

- `datmat`: The data frame containing columns `yj`, `yk`, `yjk`, `q10`, `q02` and `q12`.
- `iter`: An integer denoting the maximum number of iterations allowed for targeted maximum likelihood method. Default value is 100.
- `margin`: The minimum value the estimates can attain to bound them away from zero.
- `stop_margin`: The minimum value the estimates can attain to bound them away from zero.
- `twolist`: The logical value of whether targeted maximum likelihood algorithm fits only two modes when `K = 2`.
- `K`: The number of lists in the original data.
- `...`: Any extra arguments passed into the function.

Value

A list of estimates containing the following components:

- `error`: An indicator of whether the algorithm ran and converged. Returns FALSE, if it ran correctly and FALSE otherwise.
- `datmat`: A data frame returning `datmat` with the updated estimates for the nuisance functions `q10`, `q02` and `q12`. This is returned only if `error` is FALSE.

References


Examples

data = matrix(sample(c(0,1), 2000, replace = TRUE), ncol = 2)
xmat = matrix(runif(nrow(data)*3, 0, 1), nrow = nrow(data))
datmat = cbind(data, data[,1]*data[,2], xmat)
colnames(datmat) = c("yj", "yk", "yjk", "q10", "q02", "q12")
datmat = as.data.frame(datmat)
result = tmle(datmat, margin = 0.005, stop_margin = 0.00001, twolist = TRUE)
Index

informat, 2
plotci, 3
popszie, 4, 8
popszie_cond, 6
popszie_simul, 8
qhat_gam, 10
qhat_logit, 11
qhat_mlogit, 12
qhat_ranger, 13
qhat_rangerlogit, 14
qhat_rangerlogit(), 5, 7
qhat_sl, 15
qhat_sl(), 5, 7
reformat, 16
simuldata, 17
SuperLearner::listWrappers(), 5, 7, 15
tmle, 18
tmle(), 5, 7