

# Package ‘jipApprox’

August 26, 2023

**Title** Approximate Inclusion Probabilities for Survey Sampling

**Version** 0.1.5

**Date** 2023-08-26

**Description** Approximate joint-inclusion probabilities in Unequal Probability Sampling, or compute Monte Carlo approximations of the first and second-order inclusion probabilities of a general sampling design as in Fattorini (2006) <[doi:10.1093/biomet/93.2.269](https://doi.org/10.1093/biomet/93.2.269)>.

**Depends** R (>= 4.0.0)

**License** GPL-3

**Encoding** UTF-8

**BugReports** <https://github.com/rhobis/jipApprox/issues>

**RoxygenNote** 7.2.3

**Imports** sampling

**NeedsCompilation** no

**Author** Roberto Sichera [aut, cre]

**Maintainer** Roberto Sichera <[rob.sichera@gmail.com](mailto:rob.sichera@gmail.com)>

**Repository** CRAN

**Date/Publication** 2023-08-26 08:40:02 UTC

## R topics documented:

jipApprox-package	2
HTvar	3
jipDFtoM	4
jipMtoDF	5
jip_approx	5
jip_MonteCarlo	7

<b>Index</b>	<b>11</b>
--------------	-----------

## Description

Approximate joint-inclusion probabilities in Unequal Probability Sampling, or compute Monte Carlo approximations of the first and second-order inclusion probabilities of a general sampling design as in Fattorini (2006) <doi:10.1093/biomet/93.2.269>.

## Approximation of Joint-inclusion probabilities

Function `jip_approx` provides a number of approximations of the second-order inclusion probabilities that require only the first-order inclusion probabilities. These approximations may be employed in unequal probability sampling design with high entropy. A more flexible approximation may be obtained by using function `jip_MonteCarlo`, which estimates inclusion probabilities through a Monte Carlo simulation.

The variance of the Horvitz-Thompson total estimator may be then estimated by plugging the approximated joint probabilities into the Horvitz-Thompson or Sen-Yates-Grundy variance estimator using function `HTvar`.

## Author(s)

**Maintainer:** Roberto Sichera <rob.sichera@gmail.com>

## References

Matei, A.; Tillé, Y., 2005. Evaluation of variance approximations and estimators in maximum entropy sampling with unequal probability and fixed sample size. *Journal of Official Statistics* 21 (4), 543-570.

Haziza, D.; Mecatti, F.; Rao, J.N.K. 2008. Evaluation of some approximate variance estimators under the Rao-Sampford unequal probability sampling design. *Metron* LXVI (1), 91-108.

Fattorini, L. 2006. Applying the Horvitz-Thompson criterion in complex designs: A computer-intensive perspective for estimating inclusion probabilities. *Biometrika* 93 (2), 269-278

## See Also

Useful links:

- Report bugs at <https://github.com/rhobis/jipApprox/issues>

HTvar

*Variance of the Horvitz-Thompson estimator***Description**

Compute or estimate the variance of the Horvitz-Thompson total estimator by the Horvitz-Thompson or Sen-Yates-Grundy variance estimators.

**Usage**

```
HTvar(y, pikl, sample = TRUE, method = "HT")
```

**Arguments**

**y** numeric vector representing the variable of interest

**pikl** matrix of second-order (joint) inclusion probabilities; the diagonal must contain the first-order inclusion probabilities.

**sample** logical value indicating whether sample or population values are provided. If `sample=TRUE`, the function returns a sample estimate of the variance, while if `sample=FALSE`, the Variance is computed over all population units. Default is `TRUE`.

**method** string, indicating if the Horvitz-Thompson ("HT") or the Sen-Yates-Grundy ("SYG") estimator should be computed.

**Details**

The Horvitz-Thompson variance is defined as

$$\sum_{i \in U} \sum_{j \in U} \frac{(\pi_{ij} - \pi_i \pi_j)}{\pi_i \pi_j} y_i y_j$$

which is estimated by

$$\sum_{i \in U} \sum_{j \in U} \frac{(\pi_{ij} - \pi_i \pi_j)}{\pi_i \pi_j \pi_{ij}} y_i y_j$$

The Sen-Yates-Grundy variance is obtained from the Horvitz-Thompson variance by conditioning on the sample size  $n$ , and is therefore only applicable to fixed size sampling designs:

$$\sum_{i \in U} \sum_{j > i} (\pi_i \pi_j - \pi_{ij}) \left( \frac{y_i}{\pi_i} - \frac{y_j}{\pi_j} \right)^2$$

Its estimator is

$$\sum_{i \in U} \sum_{j > i} \frac{(\pi_i \pi_j - \pi_{ij})}{\pi_{ij}} \left( \frac{y_i}{\pi_i} - \frac{y_j}{\pi_j} \right)^2$$

**Examples**

```

### Generate population data ---
N <- 500; n <- 50

set.seed(0)
x <- rgamma(500, scale=10, shape=5)
y <- abs( 2*x + 3.7*sqrt(x) * rnorm(N) )

pik <- n * x/sum(x)
pikl <- jip_approx(pik, method='Hajek')

### Dummy sample ---
s <- sample(N, n)

### Compute Variance ---
HTvar(y=y, pikl=pikl, sample=FALSE, method="HT")
HTvar(y=y, pikl=pikl, sample=FALSE, method="SYG")

### Estimate Variance ---
#' HTvar(y=y[s], pikl=pikl[s,s], sample=TRUE, method="HT")
#' HTvar(y=y[s], pikl=pikl[s,s], sample=TRUE, method="SYG")

```

---

jipDFtoM

---

*Transform a Joint-Inclusion Probability data.frame to a matrix*


---

**Description**

Transform a Joint-Inclusion Probability data.frame to a matrix

**Usage**

```
jipDFtoM(jip, symmetric = TRUE)
```

**Arguments**

jip	vector or data.frame containing the joint-inclusion probabilities
symmetric	boolean, if TRUE, returns a symmetric matrix, otherwise, an upper triangular matrix

**Value**

a symmetric matrix of joint-inclusion probabilities if TRUE, otherwise, an upper triangular matrix

---

jipMtoDF	<i>Transform a matrix of Joint-Inclusion Probabilities to a data.frame</i>
----------	--

---

**Description**

Transform a matrix of Joint-Inclusion Probabilities to a data.frame

**Usage**

```
jipMtoDF(jip, id = NULL)
```

**Arguments**

jip	a square matrix of joint-inclusion probabilities, symmetric or upper-triangular
id	optional, vector of id labels, its length should be equal to <code>ncol(jip)</code> and <code>nrow(jip)</code>

---

jip_approx	<i>Approximate Joint-Inclusion Probabilities</i>
------------	--

---

**Description**

Approximations of joint-inclusion probabilities by means of first-order inclusion probabilities.

**Usage**

```
jip_approx(pik, method)
```

**Arguments**

pik	numeric vector of first-order inclusion probabilities for all population units.
method	string representing one of the available approximation methods.

**Details**

Available methods are "Hajek", "HartleyRao", "Tille", "Brewer1", "Brewer2", "Brewer3", and "Brewer4". Note that these methods were derived for high-entropy sampling designs, therefore they could have low performance under different designs.

Hájek (1964) approximation [method="Hajek"] is derived under Maximum Entropy sampling design and is given by

$$\tilde{\pi}_{ij} = \pi_i \pi_j \frac{1 - (1 - \pi_i)(1 - \pi_j)}{d}$$

where  $d = \sum_{i \in U} \pi_i (1 - \pi_i)$

Hartley and Rao (1962) proposed the following approximation under randomised systematic sampling [method="HartleyRao"]:

$$\begin{aligned}\tilde{\pi}_{ij} &= \frac{n-1}{n}\pi_i\pi_j + \frac{n-1}{n^2}(\pi_i^2\pi_j + \pi_i\pi_j^2) - \frac{n-1}{n^3}\pi_i\pi_j \sum_{i \in U} \pi_j^2 \\ &+ \frac{2(n-1)}{n^3}(\pi_i^3\pi_j + \pi_i\pi_j^3 + \pi_i^2\pi_j^2) - \frac{3(n-1)}{n^4}(\pi_i^2\pi_j + \pi_i\pi_j^2) \sum_{i \in U} \pi_i^2 \\ &+ \frac{3(n-1)}{n^5}\pi_i\pi_j \left( \sum_{i \in U} \pi_i^2 \right)^2 - \frac{2(n-1)}{n^4}\pi_i\pi_j \sum_{i \in U} \pi_j^3\end{aligned}$$

Tillé (1996) proposed the approximation  $\tilde{\pi}_{ij} = \beta_i\beta_j$ , where the coefficients  $\beta_i$  are computed iteratively through the following procedure [method="Tille"]:

1.  $\beta_i^{(0)} = \pi_i, \forall i \in U$
2.  $\beta_i^{(2k-1)} = \frac{(n-1)\pi_i}{\beta^{(2k-2)} - \beta_i^{(2k-2)}}$
3.  $\beta_i^{2k} = \beta_i^{(2k-1)} \left( \frac{n(n-1)}{(\beta^{(2k-1)})^2 - \sum_{i \in U} (\beta_i^{(2k-1)})^2} \right)^{1/2}$

with  $\beta^{(k)} = \sum_{i \in U} \beta_i^k, k = 1, 2, 3, \dots$

Finally, Brewer (2002) and Brewer and Donadio (2003) proposed four approximations, which are defined by the general form

$$\tilde{\pi}_{ij} = \pi_i\pi_j(c_i + c_j)/2$$

where the  $c_i$  determine the approximation used:

- Equation (9) [method="Brewer1"]:

$$c_i = (n-1)/(n-\pi_i)$$

- Equation (10) [method="Brewer2"]:

$$c_i = (n-1)/\left(n - n^{-1} \sum_{i \in U} \pi_i^2\right)$$

- Equation (11) [method="Brewer3"]:

$$c_i = (n-1)/\left(n - 2\pi_i + n^{-1} \sum_{i \in U} \pi_i^2\right)$$

- Equation (18) [method="Brewer4"]:

$$c_i = (n-1)/\left(n - (2n-1)(n-1)^{-1}\pi_i + (n-1)^{-1} \sum_{i \in U} \pi_i^2\right)$$

**Value**

A symmetric matrix of inclusion probabilities, which diagonal is the vector of first-order inclusion probabilities.

**References**

Hartley, H.O.; Rao, J.N.K., 1962. Sampling With Unequal Probability and Without Replacement. *The Annals of Mathematical Statistics* 33 (2), 350-374.

Hájek, J., 1964. Asymptotic Theory of Rejective Sampling with Varying Probabilities from a Finite Population. *The Annals of Mathematical Statistics* 35 (4), 1491-1523.

Tillé, Y., 1996. Some Remarks on Unequal Probability Sampling Designs Without Replacement. *Annals of Economics and Statistics* 44, 177-189.

Brewer, K.R.W.; Donadio, M.E., 2003. The High Entropy Variance of the Horvitz-Thompson Estimator. *Survey Methodology* 29 (2), 189-196.

**Examples**

```
### Generate population data ---
N <- 20; n<-5

set.seed(0)
x <- rgamma(N, scale=10, shape=5)
y <- abs( 2*x + 3.7*sqrt(x) * rnorm(N) )

pik <- n * x/sum(x)

### Approximate joint-inclusion probabilities ---
pik1 <- jip_approx(pik, method='Hajek')
pik1 <- jip_approx(pik, method='HartleyRao')
pik1 <- jip_approx(pik, method='Tille')
pik1 <- jip_approx(pik, method='Brewer1')
pik1 <- jip_approx(pik, method='Brewer2')
pik1 <- jip_approx(pik, method='Brewer3')
pik1 <- jip_approx(pik, method='Brewer4')
```

**Description**

Approximate first and second-order inclusion probabilities by means of Monte Carlo simulation. Estimates are obtained as proportion of the number of occurrences of each unit or couple of units over the total number of replications. One unit is added to both numerator and denominator to assure strict positivity of estimates (Fattorini, 2006).

**Usage**

```
jip_MonteCarlo(
  x,
  n,
  replications = 1e+06,
  design,
  units,
  seed = NULL,
  as_data_frame = FALSE,
  design_pars,
  write_on_file = FALSE,
  filename,
  path,
  by = NULL,
  progress_bar = TRUE
)
```

**Arguments**

<code>x</code>	size measure or first-order inclusion probabilities, a vector or single-column data.frame
<code>n</code>	sample size (for fixed-size designs), or expected sample size (for Poisson sampling)
<code>replications</code>	numeric value, number of independent Monte Carlo replications
<code>design</code>	sampling procedure to be used for sample selection. Either a string indicating the name of the sampling design or a function; see section "Details" for more information.
<code>units</code>	indices of units for which probabilities have to be estimated. Optional, if missing, estimates are produced for the whole population
<code>seed</code>	a valid seed value for reproducibility
<code>as_data_frame</code>	logical, should output be in a data.frame form? if FALSE, a matrix is returned
<code>design_pars</code>	only used when a function is passed to argument <code>design</code> , named list of parameters to pass to the sampling design function.
<code>write_on_file</code>	logical, should output be written on a text file?
<code>filename</code>	string indicating the name of the file to create on disk, must include the .txt extension; only applies if <code>write_on_file = TRUE</code> .
<code>path</code>	string indicating the path to the directory where the output file should be created; only applies if <code>write_on_file = TRUE</code> .
<code>by</code>	optional; integer scalar indicating every how many replications a partial output should be saved
<code>progress_bar</code>	logical, indicating whether a progress bar is desired

## Details

Argument `design` accepts either a string indicating the sampling design to use to draw samples or a function. Accepted designs are "brewer", "tille", "maxEntropy", "poisson", "sampford", "systematic", "randomSystematic". The user may also pass a function as argument; such function should take as input the parameters passed to argument `design_pars` and return either a logical vector or a vector of 0s and 1s, where TRUE or 1 indicate sampled units and FALSE or 0 indicate non-sample units. The length of such vector must be equal to the length of `x` if `units` is not specified, otherwise it must have the same length of `units`.

When `write_on_file = TRUE`, specifying a value for argument `by` will produce intermediate files with approximate inclusion probabilities every by number of replications. E.g., if `replications=1e06` and `by=5e05`, two output files will be created: one with estimates at 5e05 and one at 1e06 replications. This option is particularly useful to assess convergence of the estimates.

## Value

A matrix of estimated inclusion probabilities if `as_data_frame=FALSE`, otherwise a data.frame with three columns: the first two indicate the ids of the the couple of units, while the third one contains the joint-inclusion probability values. Please, note that when `as_data_frame=TRUE`, first-order inclusion probabilities are not returned.

## References

Fattorini, L. 2006. Applying the Horvitz-Thompson criterion in complex designs: A computer-intensive perspective for estimating inclusion probabilities. *Biometrika* 93 (2), 269–278

## Examples

```
### Generate population data ---
N <- 20; n<-5

set.seed(0)
x <- rgamma(N, scale=10, shape=5)
y <- abs( 2*x + 3.7*sqrt(x) * rnorm(N) )

pik <- n * x/sum(x)

### Approximate joint-inclusion probabilities
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "brewer")
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "tille")
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "maxEntropy")
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "randomSystematic")
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "systematic")
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "sampford")
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "poisson")

#Use an external function to draw samples
pik1 <- jip_MonteCarlo(x=pik, n=n, replications=100,
                      design = sampling::UPmidzuno, design_pars = list(pik=pik))
#Write output on file after 50 and 100 replications
pik1 <- jip_MonteCarlo(x=pik, n = n, replications = 100, design = "brewer",
```

```
write_on_file = TRUE, filename="test.txt", path=tempdir(), by = 50 )
```

# Index

HTvar, [2](#), [3](#)

jip\_approx, [2](#), [5](#)

jip\_MonteCarlo, [2](#), [7](#)

jipApprox (jipApprox-package), [2](#)

jipApprox-package, [2](#)

jipDFtoM, [4](#)

jipMtoDF, [5](#)