Package: BalancedSampling (via r-universe)

October 28, 2026

Type Package

Title Balanced and Spatially Balanced Sampling

Version 2.1.1

Date 2024-11-15

Author Anton Grafström [aut, cre] (<https://orcid.org/0000-0002-4345-4024>), Wilmer Prentius [aut] (<https://orcid.org/0000-0002-3561-290X>), Jonathan Lisic [ctb]

Maintainer Anton Grafström <anton.grafstrom@gmail.com>

Description Select balanced and spatially balanced probability samples in multi-dimensional spaces with any prescribed inclusion probabilities. It contains fast (C++ via Rcpp) implementations of the included sampling methods. The local pivotal method by Grafström, Lundström and Schelin (2012) <doi:10.1111/j.1541-0420.2011.01699.x> and spatially correlated Poisson sampling by Grafström (2012) <doi:10.1016/j.jspi.2011.07.003> are included. Also the cube method (for balanced sampling) and the local cube method (for doubly balanced sampling) are included, see Grafström and Tillé (2013) <doi:10.1002/env.2194>.

License AGPL-3

Imports Rcpp (>= 1.0.13)

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

URL https://www.envisim.se/, https://github.com/envisim/BalancedSampling/

NeedsCompilation yes

RoxygenNote 7.3.2

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

Date/Publication 2024-11-18 14:10:02 UTC

RemoteUrl https://github.com/cran/BalancedSampling

RemoteRef HEAD

RemoteSha e812751f532af73a9547edcb6eb732bf3901196b

Contents

cube 2
genpopUniform 4
getPips 5
hlpm2 6
lcube 8
lpm 10
sb 13
scps 15
vsb 17

Index **19**

---

cube *The Cube method*

Description

Selects balanced samples with prescribed inclusion probabilities from a finite population using the fast flight Cube Method.

Usage

```
cube(prob, x, eps = 1e-12)

cubestratified(prob, x, integerStrata, eps = 1e-12)
```

Arguments

`prob`	A vector of length N with inclusion probabilities.
`x`	An N by q matrix of balancing auxiliary variables.
`eps`	A small value used to determine when an updated probability is close enough to 0.0 or 1.0.
`integerStrata`	An integer vector of length N with stratum numbers.

Details

If `prob` sum to an integer n, and `prob` is included as the first balancing variable, a fixed sized sample (n) will be produced.

Stratified cube:

For `cubestratified`, `prob` is automatically inserted as a balancing variable.

The stratified version uses the fast flight Cube method and pooling of landing phases.

Value

A vector of selected indices in 1,2,...,N.

Functions

- `cubestratified()`:

References

Deville, J. C. and Tillé, Y. (2004). Efficient balanced sampling: the cube method. Biometrika, 91(4), 893-912.

Chauvet, G. and Tillé, Y. (2006). A fast algorithm for balanced sampling. Computational Statistics, 21(1), 53-62.

Chauvet, G. (2009). Stratified balanced sampling. Survey Methodology, 35, 115-119.

See Also

Other sampling: `hlpm2()`, `lcube()`, `lpm()`, `scps()`

Examples

```
## Not run:
set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
s = cube(prob, x);
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);

set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
strata = c(rep(1L, 100), rep(2L, 200), rep(3L, 300), rep(4L, 400));
s = cubestratified(prob, x, strata);
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);

set.seed(12345);
prob = c(0.2, 0.25, 0.35, 0.4, 0.5, 0.5, 0.55, 0.65, 0.7, 0.9);
N = length(prob);
x = matrix(runif(N * 2), ncol = 2);
ep = rep(0L, N);
r = 10000L;
for (i in seq_len(r)) {
  s = cube(prob, cbind(prob, x));
  ep[s] = ep[s] + 1L;
}
```

```
print(ep / r);

## End(Not run)
```

---

genpopUniform *Generate populations*

Description

Generate uniform and poisson cluster process populations

If `from` and `to` is used with `genpopPoisson` together with `mirror`, the population will be bounded within these values. For the `genpopUniform`, these numbers represent the minimum and maximum values of the uniform distribution.

Usage

```
genpopUniform(size, dims = 2L, from = 0, to = 1)

genpopPoisson(
  parents,
  children,
  dims = 2L,
  from = 0,
  to = 1,
  distribution = function(n) rnorm(n, 0, 0.02),
  mirror = TRUE
)
```

Arguments

`size`	The size of the population
`dims`	The number of auxiliary variables
`from`	A number or a vector of size `dims` with the minimum values
`to`	A number or a vector of size `dims` with the maximum values
`parents`	The number of parent locations
`children`	A number or a vector of size `parents` with the mean number of children to be spawned.
`distribution`	A function taking a number as a variable, returning the offset from the parent location.
`mirror`	If `TRUE`, the population is mirrored to be inside `from` and `to`.

Functions

- `genpopPoisson()`: Poisson cluster process

Examples

```
## Not run:
set.seed(12345);
x = genpopUniform(120, 2L);
N = nrow(x);
n = 60;
prob = rep(n / N, N);
s = lpm2(prob, x);
b = sb(prob, x, s);

## End(Not run)

## Not run:
set.seed(12345);
x = genpopPoisson(70, 50, 2L);
N = nrow(x);
n = 60;
prob = rep(n / N, N);
s = lpm2(prob, x);
b = sb(prob, x, s);

## End(Not run)
```

---

getPips *Inclusion probabilities proportional-to-size*

Description

Computes the first-order inclusion probabilties from a vector of positive numbers, for a probabilitiy proportional-to-size design.

Usage

```
getPips(x, n)
```

Arguments

x	A vector of positive numbers
n	The wanted sample size

Value

A vector of inclusion probabilities proportional-to-size

Examples

```
## Not run:
set.seed(12345);
N = 1000;
n = 100;
x = matrix(runif(N * 2), ncol = 2);
prob = getPips(x[, 1], n);
s = lpm2(prob, x);
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);

## End(Not run)
```

---

hlpm2 *Hierarchical Local Pivotal Method 2*

Description

Selects an initial sample using the `lpm2()`, and then splits this sample into subsamples of given `sizes` using successive, hierarchical selection with the `lpm2()`. The method is used to select several subsamples, such that each subsample, and the combination (i.e. the union of all subsamples), is spatially balanced.

Usage

```
hlpm2(prob, x, sizes, type = "kdtree2", bucketSize = 50, eps = 1e-12)
```

Arguments

`prob`	A vector of length N with inclusion probabilities.
`x`	An N by p matrix of (standardized) auxiliary variables. Squared euclidean distance is used in the `x` space.
`sizes`	A vector of integers containing the sizes of the subsamples. `sum(sizes) = sum(prob)` must hold.
`type`	The method used in finding nearest neighbours. Must be one of `"kdtree0"`, `"kdtree1"`, `"kdtree2"`, and `"notree"`.
`bucketSize`	The maximum size of the terminal nodes in the k-d-trees.
`eps`	A small value used to determine when an updated probability is close enough to 0.0 or 1.0.

Details

The inclusion probabilities `prob` *must* sum to an integer n. The sizes of the subsamples `sum(sizes)` *must* sum to the same integer n.

Value

A vector of selected indices in 1,2,...,N.

A matrix with the population indices of the combined sample in the first column, and the associated subsample in the second column.

k-d-trees

The types "kdtree" creates k-d-trees with terminal node bucket sizes according to `bucketSize`.

- "kdtree0" creates a k-d-tree using a median split on alternating variables.
- "kdtree1" creates a k-d-tree using a median split on the largest range.
- "kdtree2" creates a k-d-tree using a sliding-midpoint split.
- "notree" does a naive search for the nearest neighbour.

References

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS), 3(3), 209-226.

Maneewongvatana, S., & Mount, D. M. (1999, December). It's okay to be skinny, if your friends are fat. In Center for geometric computing 4th annual workshop on computational geometry (Vol. 2, pp. 1-8).

Grafström, A., Lundström, N.L.P. & Schelin, L. (2012). Spatially balanced sampling through the Pivotal method. Biometrics 68(2), 514-520.

Lisic, J. J., & Cruze, N. B. (2016, June). Local pivotal methods for large surveys. In Proceedings of the Fifth International Conference on Establishment Surveys.

See Also

Other sampling: `cube()`, `lcube()`, `lpm()`, `scps()`

Examples

```
## Not run:
set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
sizes = c(10, 20, 30, 40);
s = hlpm2(prob, x, sizes);
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);

## End(Not run)
```

`lcube` *The Local Cube method*

Description

Selects doubly balanced samples with prescribed inclusion probabilities from a finite population using the Local Cube method.

Usage

```
lcube(prob, Xspread, Xbal, type = "kdtree2", bucketSize = 50, eps = 1e-12)

lcubestratified(
  prob,
  Xspread,
  Xbal,
  integerStrata,
  type = "kdtree2",
  bucketSize = 50,
  eps = 1e-12
)
```

Arguments

`prob`	A vector of length N with inclusion probabilities.
`Xspread`	An N by p matrix of (standardized) auxiliary variables. Squared euclidean distance is used in the `Xspread` space.
`Xbal`	An N by q matrix of balancing auxiliary variables.
`type`	The method used in finding nearest neighbours. Must be one of `"kdtree0"`, `"kdtree1"`, `"kdtree2"`, and `"notree"`.
`bucketSize`	The maximum size of the terminal nodes in the k-d-trees.
`eps`	A small value used to determine when an updated probability is close enough to 0.0 or 1.0.
`integerStrata`	An integer vector of length N with stratum numbers.

Details

If `prob` sum to an integer n, and `prob` is included as the first balancing variable, a fixed sized sample (n) will be produced.

Stratified lcube:

For `lcubestratified`, `prob` is automatically inserted as a balancing variable.

The stratified version uses the fast flight Cube method and pooling of landing phases.

Value

A vector of selected indices in 1,2,...,N.

Functions

- `lcubestratified()`:

k-d-trees

The types "kdtree" creates k-d-trees with terminal node bucket sizes according to `bucketSize`.

- "kdtree0" creates a k-d-tree using a median split on alternating variables.
- "kdtree1" creates a k-d-tree using a median split on the largest range.
- "kdtree2" creates a k-d-tree using a sliding-midpoint split.
- "notree" does a naive search for the nearest neighbour.

References

Deville, J. C. and Tillé, Y. (2004). Efficient balanced sampling: the cube method. Biometrika, 91(4), 893-912.

Chauvet, G. and Tillé, Y. (2006). A fast algorithm for balanced sampling. Computational Statistics, 21(1), 53-62.

Chauvet, G. (2009). Stratified balanced sampling. Survey Methodology, 35, 115-119.

Grafström, A. and Tillé, Y. (2013). Doubly balanced spatial sampling with spreading and restitution of auxiliary totals. Environmetrics, 24(2), 120-131

See Also

Other sampling: `cube()`, `hlpm2()`, `lpm()`, `scps()`

Examples

```
## Not run:
set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
xspr = matrix(runif(N * 2), ncol = 2);
s = lcube(prob, xspr, cbind(prob, x));
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);

set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
xspr = matrix(runif(N * 2), ncol = 2);
strata = c(rep(1L, 100), rep(2L, 200), rep(3L, 300), rep(4L, 400));
s = lcubestratified(prob, xspr, x, strata);
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);
```

```
set.seed(12345);
prob = c(0.2, 0.25, 0.35, 0.4, 0.5, 0.5, 0.55, 0.65, 0.7, 0.9);
N = length(prob);
x = matrix(runif(N * 2), ncol = 2);
xspr = matrix(runif(N * 2), ncol = 2);
ep = rep(0L, N);
r = 10000L;
for (i in seq_len(r)) {
  s = lcube(prob, xspr, cbind(prob, x));
  ep[s] = ep[s] + 1L;
}
print(ep / r);

## End(Not run)
```

`lpm` *The (Local) Pivotal Methods*

Description

Selects spatially balanced samples with prescribed inclusion probabilities from a finite population using the Local Pivotal Method 1 (LPM1).

Usage

```
lpm(prob, x, type = "kdtree2", bucketSize = 50, eps = 1e-12)

lpm1(prob, x, type = "kdtree2", bucketSize = 50, eps = 1e-12)

lpm2(prob, x, type = "kdtree2", bucketSize = 50, eps = 1e-12)

lpm1s(prob, x, type = "kdtree2", bucketSize = 50, eps = 1e-12)

spm(prob, eps = 1e-12)

rpm(prob, eps = 1e-12)
```

Arguments

`prob`	A vector of length N with inclusion probabilities, or an integer > 1. If an integer n, then the sample will be drawn with equal probabilities n / N.
`x`	An N by p matrix of (standardized) auxiliary variables. Squared euclidean distance is used in the x space.
`type`	The method used in finding nearest neighbours. Must be one of `"kdtree0"`, `"kdtree1"`, `"kdtree2"`, and `"notree"`.
`bucketSize`	The maximum size of the terminal nodes in the k-d-trees.

`eps` A small value used to determine when an updated probability is close enough to 0.0 or 1.0.

Details

If `prob` sum to an integer n, a fixed sized sample (n) will be produced. For `spm` and `rpm`, `prob` must be a vector of inclusion probabilities. If equal inclusion probabilities is wanted, this can be produced by `rep(n / N, N)`.

The available pivotal methods are:

- `lpm1`: The Local Pivotal Mehtod 1 (Grafström et al., 2012). Updates only units which are mutual nearest neighbours. Selects such a pair at random.
- `lpm2`, `lpm`: The Local Pivotal Method 2 (Grafström et al., 2012). Selects a unit at random, which competes with this units nearest neighbour.
- `lpm1s`: The Local Pivotal Method 1 search: (Prentius, 2023). Updates only units which are mutual nearest neighbours. Selects such a pair by branching the remaining units, giving higher probabilities to update a pair with a long branch. This changes the algorithm of lpm1, but makes it faster.
- `spm`: The Sequential Pivotal Method. Selects the two units with smallest indices to compete against each other. If the list is ordered, the algorithm is similar to systematic sampling.
- `rpm`: The Random Pivotal Method. Selects two units at random to compete against each other. Produces a design with high entropy.

Value

A vector of selected indices in 1,2,...,N.

Functions

- `lpm1()`:
- `lpm2()`:
- `lpm1s()`:
- `spm()`:
- `rpm()`:

k-d-trees

The `types` "kdtree" creates k-d-trees with terminal node bucket sizes according to `bucketSize`.

- "kdtree0" creates a k-d-tree using a median split on alternating variables.
- "kdtree1" creates a k-d-tree using a median split on the largest range.
- "kdtree2" creates a k-d-tree using a sliding-midpoint split.
- "notree" does a naive search for the nearest neighbour.

References

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS), 3(3), 209-226.

Deville, J.-C., & Tillé, Y. (1998). Unequal probability sampling without replacement through a splitting method. Biometrika 85, 89-101.

Maneewongvatana, S., & Mount, D. M. (1999, December). It's okay to be skinny, if your friends are fat. In Center for geometric computing 4th annual workshop on computational geometry (Vol. 2, pp. 1-8).

Chauvet, G. (2012). On a characterization of ordered pivotal sampling. Bernoulli, 18(4), 1320-1340.

Grafström, A., Lundström, N.L.P. & Schelin, L. (2012). Spatially balanced sampling through the Pivotal method. Biometrics 68(2), 514-520.

Lisic, J. J., & Cruze, N. B. (2016, June). Local pivotal methods for large surveys. In Proceedings of the Fifth International Conference on Establishment Surveys.

Prentius, W. (2023) Manuscript.

See Also

Other sampling: `cube()`, `hlpm2()`, `lcube()`, `scps()`

Examples

```
## Not run:
set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
s = lpm2(prob, x);
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);

set.seed(12345);
prob = c(0.2, 0.25, 0.35, 0.4, 0.5, 0.5, 0.55, 0.65, 0.7, 0.9);
N = length(prob);
x = matrix(runif(N * 2), ncol = 2);
ep = rep(0L, N);
r = 10000L;
for (i in seq_len(r)) {
  s = lpm2(prob, x);
  ep[s] = ep[s] + 1L;
}
print(ep / r);

set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
```

```
lpm1(prob, x);
lpm2(prob, x);
lpm1s(prob, x);
spm(prob);
rpm(prob);

## End(Not run)
```

---

sb *Spatial balance*

Description

Calculates the spatial balance of a sample.

Usage

```
sb(prob, x, sample, type = "kdtree2", bucketSize = 10)

sblb(prob, x, sample, type = "kdtree2", bucketSize = 10)
```

Arguments

prob	A vector of length N with inclusion probabilities, or an integer > 1. If an integer n, then the sample will be drawn with equal probabilities n / N.
x	An N by p matrix of (standardized) auxiliary variables. Squared euclidean distance is used in the x space.
sample	A vector of sample indices.
type	The method used in finding nearest neighbours. Must be one of "kdtree0", "kdtree1", "kdtree2", and "notree".
bucketSize	The maximum size of the terminal nodes in the k-d-trees.

Details

About voronoi and sumofsquares

Value

The balance measure of the provided sample.

Functions

- sblb(): Spatial balance using local balance

k-d-trees

The types "kdtree" creates k-d-trees with terminal node bucket sizes according to `bucketSize`.

- "kdtree0" creates a k-d-tree using a median split on alternating variables.
- "kdtree1" creates a k-d-tree using a median split on the largest range.
- "kdtree2" creates a k-d-tree using a sliding-midpoint split.
- "notree" does a naive search for the nearest neighbour.

References

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS), 3(3), 209-226.

Maneewongvatana, S., & Mount, D. M. (1999, December). It's okay to be skinny, if your friends are fat. In Center for geometric computing 4th annual workshop on computational geometry (Vol. 2, pp. 1-8).

Stevens Jr, D. L., & Olsen, A. R. (2004). Spatially balanced sampling of natural resources. Journal of the American statistical Association, 99(465), 262-278.

Grafström, A., Lundström, N.L.P. & Schelin, L. (2012). Spatially balanced sampling through the Pivotal method. Biometrics 68(2), 514-520.

Prentius, W, & Grafström A. (2023). Manuscript.

See Also

Other measure: `vsb()`

Other measure: `vsb()`

Examples

```
## Not run:
set.seed(12345);
N = 500;
n = 70;
prob = rep(n / N, N);
x = matrix(runif(N * 2), ncol = 2);
s = lpm2(prob, x);
b = sb(prob, x, s);

## End(Not run)
```

scps *Spatially Correlated Poisson Sampling*

Description

Selects spatially balanced samples with prescribed inclusion probabilities from a finite population using Spatially Correlated Poisson Sampling (SCPS).

Usage

```
scps(prob, x, rand = NULL, type = "kdtree2", bucketSize = 50, eps = 1e-12)

lcps(prob, x, type = "kdtree2", bucketSize = 50, eps = 1e-12)
```

Arguments

prob	A vector of length N with inclusion probabilities, or an integer > 1. If an integer n, then the sample will be drawn with equal probabilities n / N.
x	An N by p matrix of (standardized) auxiliary variables. Squared euclidean distance is used in the x space.
rand	A vector of length N with random numbers. If this is supplied, the decision of each unit is taken with the corresponding random number. This makes it possible to coordinate the samples.
type	The method used in finding nearest neighbours. Must be one of "kdtree0", "kdtree1", "kdtree2", and "notree".
bucketSize	The maximum size of the terminal nodes in the k-d-trees.
eps	A small value used to determine when an updated probability is close enough to 0.0 or 1.0.

Details

If prob sum to an integer n, a fixed sized sample (n) will be produced. The implementation uses the maximal weight strategy, as specified in Grafström (2012).

Coordinated SCPS:

If rand is supplied, coordinated SCPS will be performed. The algorithm for coordinated SCPS differs from the SCPS algorithm, as uncoordinated SCPS chooses a unit to update randomly, whereas coordinated SCPS traverses the units in the supplied order. This has a small impact on the efficiency of the algorithm for coordinated SCPS.

Locally Correlated Poisson Sampling (LCPS):

The method differs from SCPS as LPM1 differs from LPM2. In each step of the algorithm, the unit with the smallest updating distance is chosen as the deciding unit.

Value

A vector of selected indices in 1,2,...,N.

Functions

- `lcps()`:

k-d-trees

The `types` "kdtree" creates k-d-trees with terminal node bucket sizes according to `bucketSize`.

- "kdtree0" creates a k-d-tree using a median split on alternating variables.
- "kdtree1" creates a k-d-tree using a median split on the largest range.
- "kdtree2" creates a k-d-tree using a sliding-midpoint split.
- "notree" does a naive search for the nearest neighbour.

References

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS), 3(3), 209-226.

Maneewongvatana, S., & Mount, D. M. (1999, December). It's okay to be skinny, if your friends are fat. In Center for geometric computing 4th annual workshop on computational geometry (Vol. 2, pp. 1-8).

Grafström, A. (2012). Spatially correlated Poisson sampling. Journal of Statistical Planning and Inference, 142(1), 139-147.

Prentius, W. (2023). Locally correlated Poisson sampling. Environmetrics, e2832.

See Also

Other sampling: `cube()`, `hlpm2()`, `lcube()`, `lpm()`

Examples

```
## Not run:
set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
s = scps(prob, x);
plot(x[, 1], x[, 2]);
points(x[s, 1], x[s, 2], pch = 19);

set.seed(12345);
prob = c(0.2, 0.25, 0.35, 0.4, 0.5, 0.5, 0.55, 0.65, 0.7, 0.9);
N = length(prob);
x = matrix(runif(N * 2), ncol = 2);
ep = rep(0L, N);
r = 10000L;
for (i in seq_len(r)) {
  s = scps(prob, x);
  ep[s] = ep[s] + 1L;
}
```

```
print(ep / r);

set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
scps(prob, x);
lcps(prob, x);

## End(Not run)
```

---

vsb *Variance estimator for spatially balanced samples*

Description

Variance estimator of HT estimator of population total.

Usage

```
vsb(probs, ys, xs, k = 3L, type = "kdtree2", bucketSize = 40)
```

Arguments

probs	A vector of length n with inclusion probabilities.
ys	A vector of length n containing the target variable.
xs	An n by p matrix of (standardized) auxiliary variables.
k	The number of neighbours to construct the means around.
type	The method used in finding nearest neighbours. Must be one of "kdtree0", "kdtree1", "kdtree2", and "notree".
bucketSize	The maximum size of the terminal nodes in the k-d-trees.

Details

If k = 0L, the variance estimate is constructed by using all units that have the minimum distance.

If k > 0L, the variance estimate is constructed by using the k closest units. If multiple units are located on the border, all are used.

Value

The variance estimate.

k-d-trees

The types "kdtree" creates k-d-trees with terminal node bucket sizes according to `bucketSize`.

- "kdtree0" creates a k-d-tree using a median split on alternating variables.
- "kdtree1" creates a k-d-tree using a median split on the largest range.
- "kdtree2" creates a k-d-tree using a sliding-midpoint split.
- "notree" does a naive search for the nearest neighbour.

References

Grafström, A., & Schelin, L. (2014). How to select representative samples. Scandinavian Journal of Statistics, 41(2), 277-290.

See Also

Other measure: `sb()`

Examples

```
## Not run:
set.seed(12345);
N = 1000;
n = 100;
prob = rep(n/N, N);
x = matrix(runif(N * 2), ncol = 2);
y = runif(N);
s = lpm2(prob, x);
vsb(prob[s], y[s], x[s, ]);
vsb(prob[s], y[s], x[s, ], 0L);

## End(Not run)
```

Index

∗ **measure**
  sb, 13
  vsb, 17
∗ **sampling**
  cube, 2
  hlpm2, 6
  lcube, 8
  lpm, 10
  scps, 15
∗ **utils**
  getPips, 5

cube, 2, *7*, *9*, *12*, *16*
cubestratified *(*cube*)*, 2

genpopPoisson *(*genpopUniform*)*, 4
genpopUniform, 4
getPips, 5

hlpm2, *3*, 6, *9*, *12*, *16*

lcps *(*scps*)*, 15
lcube, *3*, *7*, 8, *12*, *16*
lcubestratified *(*lcube*)*, 8
lpm, *3*, *7*, *9*, 10, *16*
lpm1 *(*lpm*)*, 10
lpm1s *(*lpm*)*, 10
lpm2 *(*lpm*)*, 10
lpm2(), *6*

rpm *(*lpm*)*, 10

sb, 13, *18*
sblb *(*sb*)*, 13
scps, *3*, *7*, *9*, *12*, 15
spm *(*lpm*)*, 10

vsb, *14*, 17