Package 'SamplingBigData'

July 21, 2025

Type Package

Title Sampling Methods for Big Data

Version 1.0.0

Author Jonathan Lisic, Anton Grafström

Maintainer Jonathan Lisic <jlisic@gmail.com>

Description Select sampling methods for probability samples using large data sets. This includes spatially balanced sampling in multi-dimensional spaces with any prescribed inclusion probabilities. All implementations are written in C with efficient data structures such as k-d trees that easily scale to several million rows on a modern desktop computer.

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/jlisic/SamplingBigData

NeedsCompilation yes

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2018-09-03 11:20:06 UTC

Contents

SamplingBigData-package

Sampling Methods for Big Data

Description

Select sampling methods for probability samples using large data sets. This includes spatially balanced sampling in multi-dimensional spaces with any prescribed inclusion probabilities. All implementations are written in C using efficient data structures such as k-d trees that easily scale to several million rows on a modern desktop computer.

Author(s)

Jonathan Lisic, Anton Grafström

Maintainer: Jonathan Lisic <jlisic@gmail.com>

Webpage: https://github.com/jlisic/SamplingBigData

References

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

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

Grafström, A. and Lundström, N.L.P. (2013). Why well spread probability samples are balanced. Open Journal of Statistics, 3(1).

Grafström, A. and Schelin, L. (2014). How to select representative samples. Scandinavian Journal of Statistics.

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

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

Lisic, L. and Cruze, N. (2016). Local Pivotal Methods for Large Surveys. In proceedings, ICES V, Geneva Switzerland 2016.

Examples

```
# ********************************************************
# check inclusion probabilities
# ********************************************************
set.seed(1234567);
p = c(0.2, 0.25, 0.35, 0.4, 0.5, 0.5, 0.55, 0.65, 0.7, 0.9);
N = length(p);
X = cbind(runif(N),runif(N));
p1 = p2 = p3 = p4 = rep(0,N);
nrs = 1000; # increase for more precision
```

```
for(i in seq(nrs) ){

  # lpm2 kdtree
  s = lpm2_kdtree(p,X);
  p1[s]=p1[s]+1;

  # pivotal method
  s = split_sample(p);
  p2[s]=p2[s]+1;

}
print(p);
print(p1/nrs);
print(p2/nrs);
```

`lpm2_kdtree` *Local Pivotal Method*

Description

The local pivotal method provides a way to perform balanced sampling. This implementation replace linear searches in lpm2, with k-d trees. K-d trees are binary trees used to effectively search high dimensional spaces, and reduce the average computational complexity of lpm2 from O(N^2) to O(N log(N)). Both nearest neighbor and approximate nearest neighbor searching algorithms are provided.

Usage

```
lpm2_kdtree(
   prob,
   x,
   m=40,
   algorithm = "kdtree",
   maxCheck = 4,
   termDist = 0.1,
   inOrder = FALSE,
   resample = 1,
   probTree = FALSE,
   returnTree = FALSE,
   returnBounds = FALSE
   )
```

Arguments

`prob`	An array of length N such that the sum of prob is equal to the sample size, where the N is the number of rows of `x`.
`x`	A matrix of N rows and p columns, each row is assumed to be a sampling unit.

`m`	Max leaf size used as a terminal condition for building the k-d tree. When probTree is `FALSE`, `m` is the number of rows of `x` held within each leaf node. When probTree is `TRUE`, `m` is the sum of the probability held within each node.
`algorithm`	The algorithm used to search the k-d tree. The algorithms include "kdtree", "kdtree-count", and "kdtree-dist". The "kdtree" algorithm reproduces the lpm2 using a k-d tree for nearest neighbor search. "kdtree-count" and "kdtree-dist" use approximate nearest neighor searches based on number of nodes to check and minimal sufficient distance respectfully.
`maxCheck`	A positive integer scalar parameter only used when the algorithm "kdtree-count" is specified. This parameter is the maximum number of non-empty leaf nodes to check for a nearest neighbor.
`termDist`	A positive valued scalar parameter only used when the algorithm "kdtree-dist" is specified. This parameter specifies a minimal sufficient distance to be considered a nearest neighbor. No tie handling is performed; the first nearest neighbor found will be used.
`inOrder`	A boolean value, `TRUE` will return results in order of selection. `FALSE` will return in order of index.
`resample`	The number of samples to return. Resampling builds the k-d tree exactly once for all samples. Each sample will be a distinct column in a matrix.
`probTree`	A boolean value, `TRUE` will split the k-d tree based on the weighted median, using the values in `prob`.
`returnTree`	A boolean value, `TRUE` will return the node assigment. This assignment will be appended to a returned list..
`returnBounds`	A boolean value, `TRUE` will return the bounds for each node assigment. These bounds will be appended to a returned list.

Value

A vector of selected indexes from the matrix `x`. If using default values for `inOrder`, `resample`, and `algorithm`, the results identical to the lpm2 function when no ties exist in the distance function exist. `inOrder=TRUE` will return results in order of selection, and `resample > 1` will return a matrix with each set of samples returned as a column vector.

A list is returned includiing this vector if returnTree or returnBounds is set to TRUE.

Author(s)

Jonathan Lisic

References

Lisic, L. and Cruze, N. (2016). Local Pivotal Methods for Large Surveys. In proceedings, ICES V, Geneva Switzerland 2016.

Examples

```
N <- 1000
n <- 100
x <- cbind( runif(N), runif(N))

set.seed(100)
Cprog <- proc.time()
sampled <- lpm2_kdtree( rep(n/N,N), x)
print("lpm2_kdtree running time")
print(proc.time() - Cprog)
```

`split_sample` *Split Sample*

Description

This is a fast implementation of the pivotal method.

Usage

```
split_sample(
   prob,
   delta  = exp(-16)
   )
```

Arguments

`prob`	An array of length N such that the sum of prob is equal to the sample size.
`delta`	A small real value that is used for tolerance in determining if the value is included or excluded from the sample.

Value

An array of indexes from prob. Indexes with this list are sampled.

Author(s)

Jonathan Lisic

References

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

Examples

```
N <- 1000
n <- 100
runif(N)

set.seed(100)
Cprog <- proc.time()
sampled <- split_sample( rep(n/N,N))
print(proc.time() - Cprog)
```

Index