Package 'clhs'

July 22, 2025

Type Package

Title Conditioned Latin Hypercube Sampling

Version 0.9.0

Date 2021-10-14

Maintainer Pierre Roudier <roudierp@landcareresearch.co.nz>

URL https://github.com/pierreroudier/clhs/

BugReports https://github.com/pierreroudier/clhs/issues

Description Conditioned Latin hypercube sampling, as published by Minasny and McBratney (2006) <DOI:10.1016/j.cageo.2005.12.009>. This method proposes to stratify sampling in presence of ancillary data. An extension of this method, which propose to associate a cost to each individual and take it into account during the optimisation process, is also proposed (Roudier et al., 2012, <DOI:10.1201/b12728>).

Depends R (>= 2.14.0)

Imports utils, methods, grid, ggplot2, sp, sf, raster, reshape2, plyr, cluster, Rcpp

LinkingTo RcppArmadillo, Rcpp

License GPL (>= 2)

Encoding UTF-8

LazyLoad yes

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 7.1.2

Collate 'RcppExports.R' 'clhs-internal.R' 'clhs-data.frame.R' 'clhs-package.R' 'clhs-raster.R' 'utils.R' 'clhs.R' 'clhs-sf.R' 'clhs-sp.R' 'plot.R' 'similarity.R'

NeedsCompilation yes

Author Pierre Roudier [aut, cre], Colby Brugnard [ctb], Dylan Beaudette [ctb],

Benjamin Louis [ctb],
Kiri Daust [ctb],
David Clifford [ctb]

Repository CRAN

Date/Publication 2021-10-14 04:30:10 UTC

Contents

`clhs-package` *Conditioned Latin Hypercube Sampling*

Description

This package implements the conditioned Latin hypercube sampling, as published by Minasny and McBratney (2006) and the DLHS variant method (Minasny and McBratney, 2010).. This method proposes to stratify sampling in presence of ancillary data.

Details

An extension of this method, which propose to associate a cost to each individual and take it into account during the optimisation process, is also proposed (Roudier et al., 2012).

Author(s)

Pierre Roudier

References

* For the initial cLHS method:

Minasny, B. and McBratney, A.B. 2006. A conditioned Latin hypercube method for sampling in the presence of ancillary information. Computers and Geosciences, 32:1378-1388.

*For the DLHS variant method:

Minasny, B. and A. B. McBratney, A.B.. 2010. Conditioned Latin Hypercube Sampling for Calibrating Soil Sensor Data to Soil Properties. In: Proximal Soil Sensing, Progress in Soil Science, pages 111-119.

* For the cost-constrained implementation:

Roudier, P., Beaudette, D.E. and Hewitt, A.E. 2012. A conditioned Latin hypercube sampling algorithm incorporating operational constraints. In: Digital Soil Assessments and Beyond. Proceedings of the 5th Global Workshop on Digital Soil Mapping, Sydney, Australia.

* For the similarity buffer prediction:

Brungard, C. and Johanson, J. 2015. The gate's locked! I can't get to the exact sampling spot... can I sample nearby? Pedometron, 37:8–10.

See Also

`sample`

`clhs` *Conditioned Latin Hypercube Sampling*

Description

Implementation of the conditioned Latin hypercube sampling, as published by Minasny and McBratney (2006) and the DLHS variant method (Minasny and McBratney, 2010). These methods propose to stratify sampling in presence of ancillary data. An extension of this method, which propose to associate a cost to each individual and take it into account during the optimisation process, is also proposed (Roudier et al., 2012).

Usage

```
clhs(
  x,
  size,
  must.include,
  can.include,
  cost,
  iter,
  use.cpp,
  temp,
  tdecrease,
  weights,
  eta,
  obj.limit,
  length.cycle,
  simple,
  progress,
  track,
  use.coords,
  ...
)
```

Arguments

x	A data.frame, SpatialPointsDataFrame, sf, or Raster object.
size	A non-negative integer giving the total number of items to select
must.include	A numeric vector giving the indices of the rows from x that must be included in the selected items. For the cost-constrained cLHS method, cost of these mandatory samples is set to 0. If NULL (default), all data are randomly chosen according to the classic cLHS method. If must.include is not NULL, argument size must include the total size of the final sample i.e. the size of mandatory samples given by must.include plus the size of the randomly chosen samples to pick.
can.include	A numeric vector giving indices of the rows from x that are allowed to be sampled from. The algorithm will use all of x as the reference distribution, but will only select samples from possible.sample. The option is only available in the C++ version; if use.cpp == FALSE, this parameter will be ignored.
cost	A character giving the name or an integer giving the index of the attribute in x that gives a cost that can be use to constrain the cLHS sampling. If NULL (default), the cost-constrained implementation is not used.
iter	A positive number, giving the number of iterations for the Metropolis-Hastings annealing process. Defaults to 10000.
use.cpp	TRUE or FALSE. If set to TRUE, annealing process uses C++ code. This is ~ 150 times faster than the R version, but is less stable and currently doesn't accept track or obj.limit parameters. Default to TRUE.
temp	The initial temperature at which the simulated annealing begins. Defaults to 1.
tdecrease	A number between 0 and 1, giving the rate at which temperature decreases in the simulated annealing process. Defaults to 0.95.
weights	A list a length 3, giving the relative weights for continuous data, categorical data, and correlation between variables. Defaults to list(numeric = 1, factor = 1, correlation = 1).
eta	Either a number equal 1 to perform a classic cLHS or a constrained cLHS or a matrix to perform a cLHS that samples more on the edge of the distibutions (DLHS, see details)
obj.limit	The minimal value at which the optimisation is stopped. Defaults to -Inf.
length.cycle	The duration (number of iterations) of the isotemperature steps. Defaults to 10.
simple	TRUE or FALSE. If set to TRUE, only the indices of the selected samples are returned, as a numeric vector. If set to FALSE, a cLHS_result object is returned (takes more memory but allows to make use of cLHS_results methods such as plot.cLHS_result).
progress	TRUE or FALSE, displays a progress bar.
track	A character giving the name or an integer giving the index of the attribute in x that gives a cost associated with each individual. However, this method will only track the cost - the sampling process will not be constrained by this attribute. If NULL (default), this option is not used.
use.coords	Logical, if TRUE the spatial coordinates of supported spatial objects (either a 'SpatialPointsDataFrame' object if using 'sp', or a 'sf' object if using 'sf') are included in the Latin hypercube calculations. Defaults to FALSE.
...	additional parameters passed to clhs

Details

For the DLHS method, the original paper defines parameter b as the importance of the edge of the distributions. A matrix eta (size N x K, where N is the size of the final sample and K the number of continuous variables) is defined, to compute the objective function of the algorithm, where each column equal the vector (b, 1, ..., 1, b) in order to give the edge of the distribution a probability b times higher to be sampled. In our function, instead of define the b parameter, users can defined their own eta matrix so that they can give more complex probability design of sampling each strata of the distribution instead of just be able to give more importance to both edges of the distribution.

Value

* If the simple option is set to TRUE (default behaviour): A numeric vector containing the indices of the selected samples is returned

* If the simple option is set to FALSE: An object of class cLHS_result, with the following elements:

index_samples	a vector giving the indices of the chosen samples.
sampled_data	the sampled data.frame.
obj	a vector giving the evolution of the objective function throughout the Metropolis-Hastings iterations.
cost	a vector giving the evolution of the cost function throughout the Metropolis-Hastings iterations (if available).

Author(s)

Pierre Roudier

References

*For the initial cLHS method:

Minasny, B. and McBratney, A.B. 2006. A conditioned Latin hypercube method for sampling in the presence of ancillary information. Computers and Geosciences, 32:1378-1388.

*For the DLHS method:

Minasny, B. and A. B. McBratney, A.B.. 2010. Conditioned Latin Hypercube Sampling for Calibrating Soil Sensor Data to Soil Properties. In: Proximal Soil Sensing, Progress in Soil Science, pages 111-119.

*For the cost-constrained implementation:

Roudier, P., Beaudette, D.E. and Hewitt, A.E. 2012. A conditioned Latin hypercube sampling algorithm incorporating operational constraints. In: Digital Soil Assessments and Beyond. Proceedings of the 5th Global Workshop on Digital Soil Mapping, Sydney, Australia.

See Also

plot.cLHS_result

Examples

```
df <- data.frame(
  a = runif(1000),
  b = rnorm(1000),
  c = sample(LETTERS[1:5], size = 1000, replace = TRUE)
)

# Returning the indices of the sampled points
res <- clhs(df, size = 50, progress = FALSE, simple = TRUE)
str(res)

# Returning a cLHS_result object for plotting using C++
res <- clhs(df, size = 50, use.cpp = TRUE, iter = 5000, progress = FALSE, simple = FALSE)
str(res)
plot(res)

# Method DLHS with a linear increase of the strata weight (i.e. probability to be sampled)
# from 1 for the middle starta to 3 for the edge of the distribution
linear_increase <- 1+(2/24)*0:24
eta <- matrix(c(rev(linear_increase), linear_increase), ncol = 2, nrow = 50)
set.seed(1)
res <- clhs(df, size = 50, iter = 100, eta = eta, progress = FALSE, simple = FALSE)
str(res)
plot(res)
```

---

cLHS_result *Conditioned Latin Hypercube Sampling result*

Description

A S3 class describing a cLHS result.

Value

An object of class `cLHS_result` contains the following slots:

`index_samples`	a vector giving the indices of the chosen samples.
`sampled_data`	the sampled data.frame.
`obj`	a vector giving the evolution of the objective function throughout the Metropolis-Hastings iterations.
`cost`	a vector giving the evolution of the cost function throughout the Meropolis-Hastings iterations, if available, otherwise NULL.

Author(s)

Pierre Roudier

See Also

clhs

CppLHS — *This is the internal Cpp function used to run the metropolis hasting algorithm if use.cpp = T. In general, it shouldn't be used as a stand alone function, because some preprocessing is done in R*

Description

This is the internal Cpp function used to run the metropolis hasting algorithm if use.cpp = T. In general, it shouldn't be used as a stand alone function, because some preprocessing is done in R

Arguments

xA	matrix of data - must be numeric (factors are converted to numeric in R)
cost	cost vector (0 if no cost)
strata	matrix of continuous strata
include	matrix of included data
idx	integer vector of rows from which sampling is allowed
factors	boolean factor flag
i_fact	indices of factors in xA
nsample	number of samples
cost_mode	bool cost flag
iter	number of iterations
wCont	continuous weight
wFact	factor weights
wCorr	correlation weights
etaMat	eta matrix - either all 1, or user input
temperature	initial temperature
tdecrease	temperature decrease every length_cycle iterations
length_cycle	number of iterations between temperature decrease

Value

list with sampled data, indices, objective values, cost value, and final continuous weights for each sample

`plot.cLHS_result` *Plot cLHS results*

Description

Produces a plot illustrating the result of a cLHS sampling procedure.

Usage

```
## S3 method for class 'cLHS_result'
plot(x, modes = "obj", ...)
```

Arguments

`x`	Object of class "cLHS_result".
`modes`	A character vector describing the plot to produce (see Details)
`...`	Other ggplot2 plotting parameters.

Details

The subplots to be included in the final illustration are controlled by the `mode` option: - `"obj"` adds the evolution of the objective function over the iterations - `"cost"` adds the evolution of the cost function over the iterations (if available in `x`) - `"hist"` adds the comparison of the distributions of each variables in both the original object and the sampled result using histogram plots (for continuous variables). - `"dens"` adds the comparison of the distributions of each variables in both the original object and the sampled result using density plots (for continuous variables). - `"box"` adds the comparison of the distributions of each variables in both the original object and the sampled result using boxplots (for continuous variables).

Author(s)

Pierre Roudier

See Also

`clhs`

Examples

```
df <- data.frame(
  a = runif(1000),
  b = rnorm(1000),
  c = sample(LETTERS[1:5], size = 1000, replace = TRUE)
)

res <- clhs(df, size = 50, iter = 1000, use.cpp = FALSE, progress = FALSE, simple = FALSE)

# You can plot only the objective function
```

```
plot(res, mode = "obj")

# Or you can compare the distribution in the original object
# and in the sampled result
plot(res, mode = c("obj", "box"))
```

`similarity_buffer` *Gower similarity analysis*

Description

Calculates Gower's similarity index for every pixel within an given radius buffer of each sampling point

Usage

```
similarity_buffer(
  covs,
  pts,
  buffer,
  fac = NA,
  metric = "gower",
  stand = FALSE,
  ...
)
```

Arguments

`covs`	raster stack of environmental covariates
`pts`	sampling points, object of class SpatialPointsDataframe
`buffer`	Radius of the disk around each point that similarity will be calculated
`fac`	numeric, can be > 1, (e.g., fac = c(2,3)). Raster layer(s) which are categorical variables. Set to NA if no factor is present
`metric`	character string specifying the similarity metric to be used. The currently available options are "euclidean", "manhattan" and "gower" (the default). See `daisy` from the `cluster` package for more details
`stand`	logical flag: if TRUE, then the measurements in x are standardized before calculating the dissimilarities.
`...`	passed to plyr::llply

Value

a RasterStack

Author(s)

Colby Brungard

References

Brungard, C. and Johanson, J. 2015. The gate's locked! I can't get to the exact sampling spot... can I sample nearby? Pedometron, 37:8–10.

Examples

```
library(raster)
library(sp)

data(meuse.grid)
coordinates(meuse.grid) = ~x+y
proj4string(meuse.grid) <- CRS("+init=epsg:28992")
gridded(meuse.grid) = TRUE
ms <- stack(meuse.grid)

suppressWarnings(RNGversion("3.5.0"))
set.seed(1)
pts <- clhs(ms, size = 3, iter = 100, progress = FALSE, simple = FALSE)
gw <- similarity_buffer(ms, pts$sampled_data, buffer = 500)
plot(gw)
```

Index