Latin Hypercube Sampling with a Genetic Algorithm

Description

Draws a Latin Hypercube Sample from a set of uniform distributions for use in creating a Latin Hypercube Design. This function attempts to optimize the sample with respect to the S optimality criterion through a genetic type algorithm.

Usage

```
geneticLHS(n=10, k=2, pop=100, gen=4, pMut=.1, verbose=FALSE)
```

Arguments

`n`	The number of partitions (simulations or design points)
`k`	The number of replications (variables)
`pop`	The number of designs in the initial population
`gen`	The number of generations over which the algorithm is applied
`pMut`	The probability with which a mutation occurs in a column of the progeny

`verbose` Print informational messages

Details

Latin hypercube sampling (LHS) was developed to generate a distribution of collections of parameter values from a multidimensional distribution. A square grid containing possible sample points is a Latin square iff there is only one sample in each row and each column. A Latin hypercube is the generalisation of this concept to an arbitrary number of dimensions. When sampling a function of `k` variables, the range of each variable is divided into `n` equally probable intervals. `n` sample points are then drawn such that a Latin Hypercube is created. Latin Hypercube sampling generates more efficient estimates of desired parameters than simple Monte Carlo sampling.

This program generates a Latin Hypercube Sample by creating random permutations of the first `n` integers in each of `k` columns and then transforming those integers into n sections of a standard uniform distribution. Random values are then sampled from within each of the n sections. Once the sample is generated, the uniform sample from a column can be transformed to any distribution by using the quantile functions, e.g. qnorm(). Different columns can have different distributions.

S-optimality seeks to maximize the mean distance from each design point to all the other points in the design, so the points are as spread out as possible.

Genetic Algorithm:

1. Generate `pop` random latin hypercube designs of size `n` by `k`
2. Calculate the S optimality measure of each design
3. Keep the best design in the first position and throw away half of the rest of the population
4. Take a random column out of the best matrix and place it in a random column of each of the other matricies, and take a random column out of each of the other matricies and put it in copies of the best matrix thereby causing the progeny

5. For each of the progeny, cause a genetic mutation `pMut` percent of the time. The mutation is accomplished by swtching two elements in a column

Value

An `n` by `k` Latin Hypercube Sample matrix with values uniformly distributed on [0,1]

Author(s)

Rob Carnell

References

Stocki, R. (2005) A method to improve design reliability using optimal Latin hypercube sampling *Computer Assisted Mechanics and Engineering Sciences* **12**, 87–105.

Stein, M. (1987) Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics*. **29**, 143–151.

See Also

`randomLHS`, `improvedLHS`, `maximinLHS`, and `optimumLHS` to generate Latin Hypercube Samples. `optAugmentLHS`, `optSeededLHS`, and `augmentLHS` to modify and augment existing designs.

Examples

```
geneticLHS(4, 3, 50, 5, .25)
```