Package 'FSSF'

January 20, 2025

Type Package

Title Generate Fully-Sequential Space-Filling Designs Inside a Unit Hypercube

Version 0.1.1

Date 2020-02-03

Author Boyang Shang and Daniel W. Apley

Maintainer Boyang Shang <boyangshang2015@u.northwestern.edu>

Description Provides three methods proposed by Shang and Apley (2019) <doi:10.1080/00224065.2019.1705207> to generate fully-sequential space-filling designs inside a unit hypercube. A 'fully-sequential space-filling design' means a sequence of nested designs (as the design size varies from one point up to some maximum number of points) with the design points added one at a time and such that the design at each size has good space-filling properties. Two methods target the minimum pairwise distance criterion and generate maximin designs, among which one method is more efficient when design size is large. One method targets the maximum hole size criterion and uses a heuristic to generate what is closer to a minimax design.

License GPL-2

Depends R (>= 3.3)

Imports Rcpp (>= 0.12.10)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-02-05 12:10:07 UTC

Contents

`FSSF-package` *Large-Scale Fully-Sequential Space-filling Algorithms for Computer Experiments*

Description

Provides large-scale computer experimental design tools to generate fully-sequential (a nested sequence of designs with points added one at a time) space-filling designs inside a unit hypercube (Shang and Apley, 2019).

Details

Fully-sequential (i.e., with design points added one at a time) space-filling designs are useful for global surrogate modeling of expensive computer experiments when the number of design points required to achieve a suitable accuracy is unknown in advance. We provide three fully-sequential space-filling (FSSF) design algorithms that are conceptually simple and computationally efficient and that achieve much better space-filling properties than alternative methods such as Sobol sequences and more complex batch-sequential methods based on sliced or nested optimal Latin hypercube designs (LHDs).

Brief descriptions of the main functions are provided below:

`fssf_f()` generates maximin designs using a similar idea with Kennard and Stone (1969). The "f" stands for "forward", since the algorithm begins with the smallest design and adds points one at a time.

`fssf-b()` generates large-size maximin designs efficiently. The "b" stands for "backward", since the algorithm begins with the largest design and removes points one at a time.

`fssf-fr()` uses a heuristic to generate what is closer to a minimax design. The "fr" stands for "forward reflected", since the algorithm uses a modification of the criterion used in the fssf-f algorithm.

Author(s)

Boyang Shang <boyangshang2015@u.northwestern.edu>
Daniel W. Apley <apley@northwestern.edu>

References

Shang, B. and Apley, D.W. (2019), "Large-Scale Fully-Sequential Space-filling Algorithms for Computer Experiments", Journal of Quality Technology (in press). doi:10.1080/00224065.2019.1705207.

Kennard, R.W. and Stone, L.A. (1969). "Computer aided design of experiments". Technometrics 11.1, pp. 137-148.

Examples

```
#
# See the examples in the help pages for the main functions mentioned above.
#
```

`fssf_b` *Generate fully-sequential maximin designs inside a unit hypercube.*

Description

Produces a random fully-sequential design (a nested sequence of designs with points added one at a time) inside a unit hypercube such that the design points are as far away from each other as possible (Shang and Apley, 2019). The "b" stands for "backward", since the algorithm begins with the largest design and removes points one at a time. `fssf_b` is much faster than `fssf_f` when the design size is large.

Usage

```
fssf_b(d, nMax, N=-1, eps=0.5)
```

Arguments

`d`	The dimension of the design space.
`nMax`	The largest design size required by the user.
`N`	Size of the candidate set used to generate the design points. -1 corresponds to the default setting, and the candidate set size will be calculated as $1000 \times d + 2 \times nMax$. Using large `N` will make the design more space-filling, but will slow down the program.
`eps`	The error bound for approximate nearest neighbor searching. Default value is 0.5 and `eps` must be greater or equal to 0. Using large `eps` will make the program run faster, but will cause worse space-filling performances.

Details

The `fssf_b` function calls portions of the Approximate Nearest Neighbor Library, version 1.1.2, written by David M. Mount and Sunil Arya to do the neareast neighbor search. Some changes have been made to the original ANN library to suit the needs of the `fssf_b` function. More information about the ANN library can be found in the ANN Programming Manual at http://www.cs.umd.edu/~mount/ANN

Value

A $nMax \times d$ matrix with the i^{th} row corresponding to the i^{th} design point.

Author(s)

Boyang Shang <boyangshang2015@u.northwestern.edu>
Daniel W. Apley <apley@northwestern.edu>

References

Shang, B. and Apley, D.W. (2019), "Large-Scale Fully-Sequential Space-filling Algorithms for Computer Experiments", Journal of Quality Technology (in press). doi:10.1080/00224065.2019.1705207.

Kennard, R.W. and Stone, L.A. (1969). "Computer aided design of experiments". Technometrics 11.1, pp. 137-148.

Examples

```
##Generate a design using the fssf_b function.
Design <- fssf_b(d=2, nMax = 320)
plot(Design[,1], Design[,2])
```

`fssf_f` *Generate fully-sequential maximin designs inside a unit hypercube.*

Description

Produces a random fully-sequential design (a nested sequence of designs with points added one at a time) inside a unit hypercube such that the design points are as far away from each other as possible (Shang and Apley, 2019). The "f" stands for "forward", since the algorithm begins with the smallest design and adds points one at a time.

Usage

```
fssf_f(d, nMax, N=-1, ScaleVector = NULL, Dinit = NULL)
```

Arguments

`d`	The dimension of the design space.
`nMax`	The largest design size required by the user.
`N`	Size of the candidate set used to generate the design points. -1 corresponds to the default setting, and the candidate set size will be calculated as $1000 \times d + 2 \times nMax$. Using large `N` will make the design more space-filling, but will slow down the program.
`ScaleVector`	Array of the lengthscale parameters of different inputs. Default is NULL, which corresponds to the `ScaleVector` being a unit vector of length `d`. When `ScaleVector` is not NULL, for instance, `ScaleVector` is $(\theta_1, \cdots, \theta_d)$, the distance between point $(x_1, \cdots, x_d)$ and point $(y_1, \cdots, y_d)$ will be computed as $\sum_{j=1}^{d} \left(\frac{y_j - x_j}{\theta_j}\right)^2$
`Dinit`	Numerical Matrix with n_{init} rows and `d` columns, where n_{init} is a user-specified parameter. This is an optional initial design with size n_{init} provided by the user. Default is NULL, which corresponds to no initial design. If `Dinit` is not NULL, then the algorithm will select `nMax` additional design points taking into account of this initial design.

Details

The `fssf_f` uses a similar idea as proposed by Kennard and Stone(1969); modifications have been made to improve the space-filling performance as well as efficiency.

Value

A $nMax \times d$ matrix with the i^{th} row corresponding to the i^{th} design point.

Author(s)

Boyang Shang <boyangshang2015@u.northwestern.edu>
Daniel W. Apley <apley@northwestern.edu>

References

Shang, B. and Apley, D.W. (2019), "Large-Scale Fully-Sequential Space-filling Algorithms for Computer Experiments", Journal of Quality Technology (in press). doi:10.1080/00224065.2019.1705207.

Kennard, R.W. and Stone, L.A. (1969). "Computer aided design of experiments". Technometrics 11.1, pp. 137-148.

Examples

```
##Generate a design using the fssf_f function with no scaling vector and no initial design.
Design <- fssf_f(d=2, nMax = 320)
plot(Design[,1], Design[,2])

##Generate a design using the fssf_f function with scaling vector and no initial design.
d = 2
n = 100
ScaleVector = c(1.0, 20.0)*0.5
Design = fssf_f(d = d, nMax = n, ScaleVector = ScaleVector)
plot(Design[,1], Design[,2])

##Generate a design using the fssf_f function with a scaling vector and with an initial design
d = 2
n = 100
Dinit =  fssf_f(d=2, nMax = 40)
ScaleVector = c(1.0, 20.0)*0.5
Design = fssf_f(d = d, nMax = n, ScaleVector = ScaleVector, Dinit = Dinit)
plot(Design[,1], Design[,2])
points(Dinit[,1], Dinit[,2], col="red")
```

`fssf_fr` *Generate fully-sequential minimax designs inside a unit hypercube using a heuristic.*

Description

Produces a random fully-sequential design (a nested sequence of designs with points added one at a time) inside a unit hypercube such that the largest distance between any point inside the unit hypercube to its closest design point is as small as possible (Shang and Apley, 2019). The "fr" stands for "forward reflected", since the algorithm uses a modification of the criterion used in the fssf-f algorithm.

Usage

```
fssf_fr(d, nMax, N=-1, Preference = "minimax", ScaleVector = NULL, Dinit = NULL)
```

Arguments

`d`	The dimension of the design space.
`nMax`	The largest design size required by the user.
`N`	Size of the candidate set used to generate the design points. -1 corresponds to the default setting, and the candidate set size will be calculated as $1000 \times d + 2 \times nMax$. Using large `N` will make the design more space-filling, but will slow down the program.
`Preference`	Choosing `Preference` as "minimax" will produce a design that leaves small holes in the design space, especially for the early design points, with the cost that design points may be closer to each other than the other option. Choosing `Preference` as "maximin" will produce a design that leaves a little larger holes than the other option, but the design points will be further away from each other.
`ScaleVector`	Array of the lengthscale parameters of different inputs. Default is NULL, which corresponds to the `ScaleVector` being a unit vector of length `d`. When `ScaleVector` is not NULL, for instance, `ScaleVector` is $(\theta_1, \cdots, \theta_d)$, the distance between point $(x_1, \cdots, x_d)$ and point $(y_1, \cdots, y_d)$ will be computed as $\sum_{j=1}^{d} \left(\frac{y_j - x_j}{\theta_j}\right)^2$
`Dinit`	Numerical Matrix with n_{init} rows and `d` columns, where n_{init} is a user-specified parameter. This is an optional initial design with size n_{init} provided by the user. Default is NULL, which corresponds to no initial design. If `Dinit` is not NULL, then the algorithm will select `nMax` additional design points taking into account of this initial design.

Value

A $nMax \times d$ matrix with the i^{th} row corresponding to the i^{th} design point.

Author(s)

Boyang Shang <boyangshang2015@u.northwestern.edu>
Daniel W. Apley <apley@northwestern.edu>

References

Shang, B. and Apley, D.W. (2019), "Large-Scale Fully-Sequential Space-filling Algorithms for Computer Experiments", Journal of Quality Technology (in press). doi:10.1080/00224065.2019.1705207.

Kennard, R.W. and Stone, L.A. (1969). "Computer aided design of experiments". Technometrics 11.1, pp. 137-148.

Examples

```
##Generate a design using the fssf_fr function.
Design <- fssf_fr(d=2, nMax = 320)
plot(Design[,1], Design[,2])


##Generate a design using the fssf_fr function with scaling vector and no initial design.
d = 2
n = 100
ScaleVector = c(1.0, 20.0)*0.5
Design = fssf_fr(d = d, nMax = n, ScaleVector = ScaleVector)
plot(Design[,1], Design[,2])

##Generate a design using the fssf_fr function with a scaling vector and with an initial design
d = 2
n = 100
Dinit =  fssf_fr(d=2, nMax = 40)
ScaleVector = c(1.0, 20.0)*0.5
Design = fssf_fr(d = d, nMax = n, ScaleVector = ScaleVector, Dinit = Dinit)
plot(Design[,1], Design[,2])
points(Dinit[,1], Dinit[,2], col="red")
```

Index