Package ‘maximin’

November 1, 2019

Title Space-Filling Design under Maximin Distance

Version 1.0-3

Date 2019-10-31

Author Furong Sun <furongs@vt.edu>, Robert B. Gramacy <rbg@vt.edu>

Depends R (>= 3.5.0)

Imports plgp

Suggests lhs

Description Constructs a space-filling design under the criterion of maximin distance. Both discrete and continuous versions are provided.

Maintainer Furong Sun <furongs@vt.edu>

NeedsCompilation no

Repository CRAN

Date/Publication 2019-11-01 05:30:03 UTC

R topics documented:

lola_kn 1
maximin 2

Index **6**

`lola_kn` *spatial locations of 1535 weather stations*

Description

The dataset contains spatial locations of 1535 weather stations for measuring solar irradiance across the continental United States.

Format

A data frame containing 1535 observations and 2 variables

Source

https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11414

References

F. Sun, R.B. Gramacy, B. Haaland, S.Y. Lu, and Y. Hwang (2019) *Synthesizing Simulation and Field Data of Solar Irradiance*, Statistical Analysis and Data Mining, 12(4), 311-324; preprint on arXiv:1806.05131.

`maximin` *Space-filling design under the criterion of maximin distance*

Description

Generates a space-filling design under the criterion of maximin distance; both discrete and continuous searches are provided.

Usage

```
maximin.cand(n, Xcand, Tmax, Xorig=NULL, init=NULL, verb=FALSE, tempfile=NULL)
maximin(n, p, T, Xorig=NULL, Xinit=NULL, verb=FALSE, plot=FALSE, boundary=FALSE)
```

Arguments

`n`	the number of space-filling locations
`Xcand`	the candidate set, from which each space-filling location is selected
`Tmax`	the number of iterations; `Tmax` <= `nrow(Xcand)`; to be safe, set `Tmax` = `nrow(Xcand)`.
`Xorig`	the existing design; `ncol(Xorig)` = `ncol(Xcand)`
`init`	the initial indices of `X`; it can be randomly selected from `Xcand` or introduced from a previous experiment.
`verb`	progress indicator — every tenth iteration is printed out; by default `verb` = `FALSE`.
`tempfile`	the name of a temporary file given the progress is saved with each iteration; by default `tempfile` = `NULL`
`p`	the dimensionality of input space
`T`	the number of iterations; $T > n$; setting $T = 10 * n$ is a good starting point.
`Xinit`	the (initial) design introduced from a previous experiment
`plot`	if `plot` = `TRUE`, then the search space and the "start location –> new location" with each iteration is plotted; if $p > 2$, then TWO input coordinates are RANDOMLY chosen for plotting; it is worth noticing that the search space only VISUALLY makes sense when $p = 2$.

boundary — if boundary = TRUE, then for each iteration, the "to-be-swapped-in" location will be away from the boundary in addition to being away from other X locations and Xorig; how far is it? $min(d, 4 * d.bound)$, where d is the Euclidean distance between the "to-be-swapped-in" location and other X locations as well as Xorig, while d.bound is the minimum Euclidean distance between the "to-be-swapped-in" location and the boundaries.

Details

Constructing a space-filling design under the criterion of maximin distance is quite useful in computer experiments and related fields. Previously, researchers would construct such a design in a random accept-reject way, i.e., randomly propose a location within the study region to replace a randomly selected row from the initial design. If such a proposal increases the minimum pairwise Euclidean distance, then accept the replacement; otherwise keep the original design location. By repeatedly proposing (and accept-rejecting) in this way one is able to construct an (approximately) space-filling design. However the algorithm is inefficient computationally. The reason is that the proposals are not optimized in any way.

In this package, we provide an alternative to build up a well-defined space-filling design more efficiently. There are two versions, one is with discrete search, while the other is with continuous search. For the former, each iteration proposes to swap out the row from the initial design with the minimum distance, and swap in another location from a candidate set. For the latter, the core idea is the same, but instead of working with a candidate set, optim is used to maximize the distance between the "to-be-swapped-in" location and other design locations as well as to the existing design, Xorig. Several heuristics are deployed for situations where the search becomes stuck in a local mode. One involves moving to a location with non-minimum distance. Or, if there is no progress during the past $0.3 \times n$ iterations, we jump to a location which has the maximum minimum distance.

For a visualization of applying maximin.cand in a real-life problem on solar irradiance, see Sun et al. (2019).

maximin.cand returns the indices of Xcand, which makes the final space-filling design, the minimum pairwise Euclidean distance with each iteration and the corresponding number

maximin returns the combined existing design and the space-filling design, together with the minimum pairwise Euclidean distance with each iteration

Value

maximin.cand returns

inds	the indices of Xcand, which makes the final space-filling design
mis	the minimum distance with each iteration; length(mis) = Tmax + 1
mislen	the number of mis with each iteration; length(mislen) = Tmax + 1

maximin returns

Xf	dim(Xf) = (nrow(Xorig) + n) * p
mi	the minimum distance with each iteration; length(mi) = T + 1

Author(s)

Furong Sun <furongs@vt.edu> and Robert B. Gramacy <rbg@vt.edu>

References

F. Sun, R.B. Gramacy, B. Haaland, S.Y. Lu, and Y. Hwang (2019) *Synthesizing Simulation and Field Data of Solar Irradiance*, Statistical Analysis and Data Mining, 12(4), 311-324; preprint on arXiv:1806.05131.

M.H.Y. Tan (2013) *Minimax Designs for Finite Design Regions*, Technometrics, 55(3), 346-358.

M.E. Johnson, L.M. Moore, and D. Yivisaker (1990) *Minimax and Maximin Distance Designs*, Journal of Statistical Planning and Inference, 26(2), 131-148.

Examples

```
 ## Not run:
   ## maximin.cand
   # a generic function to expand grids at higher dimension
   expand.grids <- function(x,d) expand.grid(replicate(d, x, simplify=FALSE))

   # generate the design
   library(lhs)
   n <- 100
   p <- 2
   Xorig <- randomLHS(10, p)
   x1 <- seq(0, 1, length.out=n)
   Xcand <- expand.grids(x1, p)
   names(Xcand) <- paste0("x", 1:2)
   T <- nrow(Xcand)
   Xsparse <- maximin.cand(n=n, Xcand=Xcand, Tmax=T, Xorig=Xorig,
                           init=NULL, verb=FALSE, tempfile=NULL)

   maxmd <- as.numeric(format(round(max(na.omit(Xsparse$mis)), 5), nsmall=5))

   # visualization
   par(mfrow=c(1, 2))
   X <- Xcand[Xsparse$inds,]
   plot(X$x1, X$x2, xlab=expression(x[1]), ylab=expression(x[2]),
        xlim=c(0, 1), ylim=c(0, 1),
        main=paste0("n=", n, "_p=", p, "_maximin=", maxmd))
   points(Xorig, col=2, pch=20)
   abline(h=c(0, 1), v=c(0, 1), lty=2, col=2)
   if(!is.null(Xorig))
   {
     legend("topright", "Xorig", xpd=TRUE, horiz=TRUE,
            inset=c(-0.03, -0.05), pch=20, col=2, bty="n")
   }
   plot(log(na.omit(Xsparse$mis)), type="b",
        xlab="iteration", ylab="log(minimum distance)",
        main="progress on minimum distance")
   abline(v=n, lty=2)
   mtext(paste0("design size=", n), at=n, cex=0.6)

## End(Not run)
```

```
## maximin
# generate the design
library(lhs)
n <- 10
p <- 2
T <- 10*n
Xorig <- randomLHS(10, p)
Xsparse <- maximin(n=n, p=p, T=T, Xorig=Xorig, Xinit=NULL,
                   verb=FALSE, plot=FALSE, boundary=FALSE)
maxmd <- as.numeric(format(round(Xsparse$mi[T+1], 5), nsmall=5))

# visualization
par(mfrow=c(1,2))
plot(Xsparse$Xf[,1], Xsparse$Xf[,2], xlab=expression(x[1]), ylab=expression(x[2]),
     xlim=c(0, 1), ylim=c(0, 1),
     main=paste0("n=", n, "_p=", p, "_T=", T, "_maximin=", maxmd))
points(Xorig, col=2, pch=20)
abline(h=c(0,1), v=c(0,1), lty=2, col=2)
if(!is.null(Xorig)) legend("topright", "Xorig", xpd=TRUE, horiz=TRUE,
   inset=c(-0.03, -0.05), pch=20, col=2, bty="n")
plot(log(Xsparse$mi), type="b", xlab="iteration", ylab="log(minimum distance)",
     main="progress on minimum distance")
abline(v=n, lty=2)
mtext(paste0("design size=", n), at=n, cex=0.6)
abline(v=T, lty=2)
mtext(paste0("max.md=", maxmd), at=T, cex=0.6)
```

Index

∗Topic **computer experiment**
maximin, 2
∗Topic **datasets**
lola_kn, 1
∗Topic **maximin**
maximin, 2
∗Topic **space-filling design**
maximin, 2

lola_kn, 1

maximin, 2

optim, *3*