Package ‘minimaxdesign’

May 7, 2019

Type Package

Title Minimax and Minimax Projection Designs

Version 0.1.4

Author Simon Mak

Maintainer Simon Mak <smak6@gatech.edu>

Description Provides two main functions, minimax() and miniMaxPro(), for computing minimax and minimax projection designs using the minimax clustering algorithm in Mak and Joseph (2018) <DOI:10.1080/10618600.2017.1302881>. Current design region options include the unit hypercube (``hypercube''), the unit simplex (``simplex''), the unit ball (``ball''), as well as user-defined constraints on the unit hypercube (``custom''). Minimax designs can also be computed on user-provided images using the function minimax.map(). Design quality can be assessed using the function mMdist(), which computes the minimax (fill) distance of a design.

License GPL (>= 2)

LazyData TRUE

Imports Rcpp (>= 0.12.4), randtoolbox, DiceDesign, MaxPro, doParallel, doSNOW, gtools, nloptr, foreach, jpeg, gmp, conf.design, pdist, DoE.base, FrF2

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-05-06 22:30:03 UTC

R topics documented:

`minimaxdesign-package` *An R package for computing Minimax and Minimax Projection Designs*

Description

The 'minimaxdesign' package provides functions for generating minimax designs and minimax projection designs.

Details

Package:	minimaxdesign
Type:	Package
Version:	0.1.4
Date:	2019-05-03
License:	GPL (>= 2)

Provides two main functions, `minimax()` and `miniMaxPro()`, for computing minimax and minimax projection designs using the minimax clustering algorithm in Mak and Joseph (2018) <DOI:10.1080/10618600.2017.1302881 Current design region options include the unit hypercube ("hypercube"), the unit simplex ("simplex"), the unit ball ("ball"), as well as user-defined constraints on the unit hypercube ("custom"). Minimax designs can also be computed on user-provided images using the function `minimax.map()`. Design quality can be assessed using the function `mMdist()`, which computes the minimax (fill) distance of a design.

Author(s)

Simon Mak

Maintainer: Simon Mak <smak6@gatech.edu>

References

Mak, S. and Joseph, V. R. (2018). Minimax and minimax projection designs using clustering. *Journal of Computational and Graphical Statistics*, 27(1):166-178.

Examples

```
## Not run:
  #20-point minimax design on the hypercube [0,1]^2
  D <- minimax(N=20,p=2)
  plot(NULL,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2") #set up plot
  polygon(c(0,0,1,1),c(0,1,1,0),col="gray") #design space
  points(D,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2",pch=16) #design points
  mM <- mMdist(D)
```

```
mM$dist #minimax (fill) distance
lines(rbind(mM$far.pt,mM$far.despt),col="red",lty=2,lwd=2) #plot farthest point

#20-point minimax design on custom design space (inequalities on [0,1]^2)
ineqs <- function(xx){ #user-defined inequalities
  bool.vec <- rep(TRUE,length(xx))
  bool.vec[1] <- (xx[2]<=2-2*xx[1]) #inequality 1: x2 <= 2 - 2*x1
  bool.vec[2] <- (xx[1]>=xx[2]) #inequality 2: x1 >= x2
  return(all(bool.vec))
}
D <- minimax(N=20,p=2,region="custom",const=ineqs)
plot(NULL,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2") #set up plot
polygon(c(0,2/3,1),c(0,2/3,0),col="gray") #design space
points(D,pch=16) #design points
mM <- mMdist(D,region="custom",const=ineqs)
mM$dist #minimax (fill) distance
lines(rbind(mM$far.pt,mM$far.despt),col="red",lty=2,lwd=2) #plot farthest point

## End(Not run)
```

CtoA *Inverse Rosenblatt transformation from the unit hypercube to the unit simplex*

Description

CtoA maps points on the unit hypercube in p-dimensions, $C_p = [0,1]^p$, to points on the unit simplex in p-dimensions, A_p.

Usage

```
CtoA(D, by=ifelse(ncol(D)>2,1e-3,-1), num_proc=parallel:::detectCores())
```

Arguments

D	An N-by-p matrix representing N points in p-dimensions.
by	Step-size used for approximating integrals.
num_proc	Number of processors to use.

Value

An N-by-p matrix for the inverse-Rosenblatt mapping of D onto the unit simplex A_p.

Examples

```
## Not run:
  # Map the first 100 points of the Sobol' sequence in 3D
  #   onto the unit simplex in 3D
  library(randtoolbox)
  des <- sobol(100,3)
  des_simp <- CtoA(des)

  pairs(des_simp,xlim=c(0,1),ylim=c(0,1),pch=16)


## End(Not run)
```

CtoB — *Inverse Rosenblatt transformation from the unit hypercube to the unit ball*

Description

`CtoB` maps points on the unit hypercube in p-dimensions, $C_p = [0,1]^p$, to points on the unit simplex in p-dimensions, B_p.

Usage

```
CtoB(D, by=ifelse(ncol(D)>2,1e-3,-1), num_proc=parallel:::detectCores())
```

Arguments

`D`	An N-by-p matrix representing N points in p-dimensions.
`by`	Step-size used for approximating integrals.
`num_proc`	Number of processors to use.

Value

An N-by-p matrix for the inverse-Rosenblatt mapping of D onto the unit ball B_p.

Examples

```
## Not run:
  # Map the first 100 points of the Sobol' sequence in 3D
  #   onto the unit ball in 3D
  library(randtoolbox)
  des <- sobol(100,3)
  des_ball <- CtoB(des)

  pairs(des_ball,xlim=c(-1,1),ylim=c(-1,1),pch=16)


## End(Not run)
```

minimax *Compute minimax designs using clustering on constrained design regions*

Description

minimax computes minimax designs using the minimax clustering algorithm in Mak and Joseph (2018). Current design region options include the unit hypercube ("hypercube"), the unit simplex ("simplex"), the unit ball ("ball"), and user-defined constraints on the unit hypercube ("custom").

Usage

```
minimax(N,p,q=10,region="hypercube",ini=NA,const=NA,
        params_pso=list(w=0.72,c1=1.49,c2=1.49),
        npart=5,nclust=1e5,neval=nclust,
        itmax_pso=50,itmax_pp=100,itmax_inn=1e4,jit=0.1/sqrt(N))
```

Arguments

N	Number of design points.
p	Dimension of design region.
q	Power parameter for approximating the minimax criterion (see paper for details). Larger values of q give a better approximation, but may cause numerical instability.
region	String for desired design region. Current options include "hypercube", "simplex", "ball", and "custom".
ini	String for initial design. Current choices include NA (automatic choice), "sobol" (Sobol' sequence), and "ff" (fractional factorial).
const	Function for desired constraints (inequalities) on design space. See examples for implementation.
params_pso	Particle swarm optimization parameters (particle momentum (w), local-best velocity (c1) and global-best velocity (c2)).
npart	Number of particles for particle swarm optimization.
nclust,neval	Number of sample points for minimax clustering and post-processing.
itmax_pso,itmax_pp,itmax_inn	Maximum number of iterations for minimax clustering, post-processing and inner optimization.
jit	Jitter radius for post-processing.

Value

An N-by-p matrix for the minimax design.

Examples

```
## Not run:
#20-point minimax design on the hypercube [0,1]^2
D <- minimax(N=20,p=2)
plot(NULL,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2") #set up plot
polygon(c(0,0,1,1),c(0,1,1,0),col="gray") #design space
points(D,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2",pch=16) #design points
mM <- mMdist(D)
mM$dist #minimax (fill) distance
lines(rbind(mM$far.pt,mM$far.despt),col="red",lty=2,lwd=2) #plot farthest point

#20-point minimax design on the simplex A_2
D <- minimax(N=20,p=2,region="simplex")
plot(NULL,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2") #set up plot
polygon(c(0,0,1),c(0,1,1),col="gray") #design space
points(D,pch=16) #design points
mM <- mMdist(D,region="simplex")
mM$dist #minimax (fill) distance
lines(rbind(mM$far.pt,mM$far.despt),col="red",lty=2,lwd=2) #plot farthest point

#20-point minimax design on the ball B_2
library(plotrix)
D <- minimax(N=20,p=2,region="ball")
plot(NULL,xlim=c(-1,1),ylim=c(-1,1),xlab="x1",ylab="x2") #set up plot
draw.circle(0,0,1,col="gray") #design space
points(D,pch=16) #design points
mM <- mMdist(D,region="ball")
mM$dist #minimax (fill) distance
lines(rbind(mM$far.pt,mM$far.despt),col="red",lty=2,lwd=2) #plot farthest point

#20-point minimax design on custom design space (inequalities on [0,1]^2)
ineqs <- function(xx){ #user-defined inequalities
  bool.vec <- rep(TRUE,length(xx))
  bool.vec[1] <- (xx[2]<=2-2*xx[1]) #inequality 1: x2 <= 2 - 2*x1
  bool.vec[2] <- (xx[1]>=xx[2]) #inequality 2: x1 >= x2
  return(all(bool.vec))
}
D <- minimax(N=20,p=2,region="custom",const=ineqs)
plot(NULL,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2") #set up plot
polygon(c(0,2/3,1),c(0,2/3,0),col="gray") #design space
points(D,pch=16) #design points
mM <- mMdist(D,region="custom",const=ineqs)
mM$dist #minimax (fill) distance
lines(rbind(mM$far.pt,mM$far.despt),col="red",lty=2,lwd=2) #plot farthest point

#30-point minimax design on the hypercube [0,1]^6
D <- minimax(N=30,p=6)
mMdist(D)$dist #minimax (fill) distance


## End(Not run)
```

minimax.map *Compute minimax designs using clustering on a user-provided image*

Description

minimax.map computes minimax designs on a user-provided binary (0-1) image, using the minimax clustering algorithm in Mak and Joseph (2018).

Usage

```
minimax.map(N,img,p=2,q=10,
            params_pso=list(w=0.72,c1=1.49,c2=1.49),
            npart=5,nclust=1e5,neval=nclust,
            itmax_pso=50,itmax_pp=100,itmax_inn=1e4,jit=0.1/sqrt(N))
```

Arguments

N	Number of design points.
img	A binary 0-1 matrix, with 1 indicating the desired design region.
p	Dimension of design region.
q	Power parameter for approximating the minimax criterion (see paper for details). Larger values of q give a better approximation, but may cause numerical instability.
params_pso	Particle swarm optimization parameters (particle momentum (w), local-best velocity (c1) and global-best velocity (c2)).
npart	Number of particles for particle swarm optimization.
nclust,neval	Number of sample points for minimax clustering and post-processing.
itmax_pso,itmax_pp,itmax_inn	Maximum number of iterations for minimax clustering, post-processing and inner optimization.
jit	Jitter radius for post-processing.

Value

An N-by-p matrix for the minimax design.

Examples

```
## Not run:
  #20-point minimax design on the hypercube [0,1]^2
  library(jpeg)
  n <- 25
  img <- readJPEG(system.file("img", "gamap.jpg", package="minimaxdesign"))[,,1]
  image(t(img)[,nrow(img):1],col=gray.colors(12,start=0.6),main="Georgia")
  img <- t(img)[,nrow(img):1] #Invert image due to reading distortion
  des <- minimax.map(n,img)
```

```
    points(des,pch=16)

## End(Not run)
```

---

miniMaxPro — *Compute minimax projection designs using clustering on constrained design regions*

Description

miniMaxPro.

Usage

```
miniMaxPro(N,p,mMdes=NA, mMtol=1e-3*p,
           neval=1e5, itmax_refine=100, ...)
```

Arguments

N	Number of design points.
p	Dimension of design region.
mMdes	Minimax design from minimax().
mMtol	Tolerance for fill distance increase (actual increase may be slightly higher).
neval	Number of sample points for refinement.
itmax_refine	Maximum number of iterations for refinement.
...	Parameters for minimax().

Value

A list with two objects:

minimax	An N-by-p matrix for the minimax design.
miniMaxPro	An N-by-p matrix for the minimax projection design.

Examples

```
## Not run:
#30-point miniMaxPro design on the hypercube [0,1]^6
D <- minimax(N=30,p=6)
D <- miniMaxPro(N=30,p=6,mMdes=D)
mMdist(D$minimax)$dist
mMdist(D$miniMaxPro)$dist #slightly higher fill distance
pairs(D$minimax,xlim=c(0,1),ylim=c(0,1),pch=16)
pairs(D$miniMaxPro,xlim=c(0,1),ylim=c(0,1),pch=16) #... but better projections


## End(Not run)
```

mMdist *Computes the minimax (fill) distance of a design*

Description

mMdist computes the minimax (fill) distance of a design (see Mak and Joseph (2018) for definition). Current design region options include the unit hypercube ("hypercube"), the unit simplex ("simplex"), the unit ball ("ball"), and user-defined constraints on the unit hypercube ("custom").

Usage

```
mMdist(D,neval=1e5,method="lattice",region="hypercube",const=NA)
```

Arguments

D	An N-by-p design matrix.
neval	Number of sample points for approximation.
method	Method for generating approximation points. Current options include "lattice" (lattice rule) and "sobol" (Sobol' sequence).
region	String for desired design region. Current options include "hypercube", "simplex", "ball", and "custom".
const	Function for desired constraints (inequalities) on design space.

Value

A list with two objects:

dist	Minimax (fill) distance.
dist.vec	Minimax (fill) distance for each design point.
far.pt	A p-vector for the farthest point in design region to design.
far.despt	A p-vector for the design point closest to far.pt.

Examples

```
  ## Not run:
    #20-point minimax design on the hypercube [0,1]^2
    D <- minimax(N=20,p=2)
    plot(NULL,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2") #set up plot
    polygon(c(0,0,1,1),c(0,1,1,0),col="gray") #design space
    points(D,xlim=c(0,1),ylim=c(0,1),xlab="x1",ylab="x2",pch=16) #design points
    mM <- mMdist(D)
    mM$dist #minimax (fill) distance
    lines(rbind(mM$far.pt,mM$far.despt),col="red",lty=2,lwd=2) #plot farthest point

## End(Not run)
```

Index