Package 'rodd'

July 23, 2025

Type Package

Title Optimal Discriminating Designs

Version 0.2-1

Depends R (>= 3.0.0)

Imports numDeriv, quadprog, Matrix, rootSolve, matrixcalc

Suggests mvtnorm

Description

A collection of functions for numerical construction of optimal discriminating designs. At the current moment T-optimal designs (which maximize the lower bound for the power of F-test for regression model discrimination), KL-optimal designs (for lognormal errors) and their robust analogues can be calculated with the package.

License GPL (>= 2)

NeedsCompilation no

Author Roman Guchenko [aut, cre]

Maintainer Roman Guchenko <RomanGuchenko@yandex.ru>

Repository CRAN

Date/Publication 2016-01-12 00:30:32

Contents

`rodd-package` *Optimal Discriminating Designs*

Description

This package provides several functions suitable for efficient numerical construction of optimal discriminative designs.

Details

At the current state this package provides the routine `tpopt` for the construction of T_{P}-optimal designs, the routine `KLopt.lnorm` for the calculation of KL-optimal designs (for lognormal errors) and several auxiliary procedures to represent the results. Function `tpopt` is based on the algorithms that were developed in [7]. Function `KLopt.lnorm` is based on the methodology proposed in [8]. See the references for more details.

It is planned to add several new routines for different types of discriminative designs.

References

[1] Atkinson A.C., Fedorov V.V. (1975) *The design of experiments for discriminating between two rival models*. Biometrika, vol. 62(1), pp. 57–70.

[2] Atkinson A.C., Fedorov V.V. (1975) *Optimal design: Experiments for discriminating between several models*. Biometrika, vol. 62(2), pp. 289–303.

[3] Dette H., Pepelyshev A. (2008) *Efficient experimental designs for sigmoidal growth models*. Journal of statistical planning and inference, vol. 138, pp. 2–17.

[4] Dette H., Melas V.B., Shpilev P. (2013) *Robust T-optimal discriminating designs*. Annals of Statistics, vol. 41(4), pp. 1693–1715.

[5] Braess D., Dette H. (2013) *Optimal discriminating designs for several competing regression models*. Annals of Statistics, vol. 41(2), pp. 897–922.

[6] Braess D., Dette H. (2013) *Supplement to "Optimal discriminating designs for several competing regression models"*. Annals of Statistics, online supplementary material.

[7] Dette H., Melas V.B., Guchenko R. (2014) *Bayesian T-optimal discriminating designs*. ArXiv link.

[8] Dette H., Guchenko R., Melas V.B. (2015) *Efficient computation of Bayesian optimal discriminating designs*. ArXiv link.

KLopt.lnorm *Calculation of KL-optimal discriminating design for lognormal errors*

Description

Calculates an approximation ξ^{**} of the KL-optimal design (in case of lognormal errors) ξ^{*} for discrimination between a given list of error densities $\{f_i(x, \theta_i),\ i = 1, \ldots, \nu\}$. This procedure is based on the work [8]. This function mimics `tpopt` almost entirely. It is planed to combine `tpopt` and `KLopt.lnorm` in the future. See `tpopt` for the detailed description of the arguments marked with "-//-".

Usage

```
KLopt.lnorm(    x,
                w = rep(1, length(x)) / length(x),
                eta,
                sq.var,
                theta.fix,
                theta.var = NULL,
                p,
                x.lb = min(x),
                x.rb = max(x),
                opt = list())
```

Arguments

`x`	-//-
`w`	-//-
`eta`	a list of means for the error densities $\{f_i(x, \theta_i),\ i = 1, \ldots, \nu\}$ between which proposed optimization should be performed. Every function from this list should be defined in the form of $\eta_i(x, \theta_i)$, where x is one dimensional variable from $\mathcal{X}$ and θ_i is a vector of corresponding model parameters. We will refer to length of this list as ν.
`sq.var`	a list of variances for the error densities $\{f_i(x, \theta_i),\ i = 1, \ldots, \nu\}$ between which proposed optimization should be performed. Every function from this list should be defined in the form of $v_i^2(x, \theta_i)$. This list also has the length equal to ν.
`theta.fix`	-//-
`theta.var`	-//-
`p`	-//-
`x.lb`	-//-
`x.rb`	-//-
`opt`	-//-

Value

Object of class "KLopt.lnorm" which contains the following fields:

x, w, efficiency, functional -//-

eta a list of means from the input.

sq.var a list of variances from the input.

theta.fix, theta.var, p, x.lb, x.rb, max.iter, done.iter, des.eff, time -//-

See Also

plot.KLopt.lnorm, summary.KLopt.lnorm, print.KLopt.lnorm

Examples

```
## Not run:
### Examples from [8]
### Cases 1 and 3 are presented here; case 2 can be computed using the
### function tpopt (see the description of this function for exact example)

library(mvtnorm)

### Example 1 from [8]; EMAX vs MM

#List of models
eta.1 <- function(x, theta.1)
    theta.1[1] * x + theta.1[2] * x / (x + theta.1[3])

eta.2 <- function(x, theta.2)
    theta.2[1] * x / (x + theta.2[2])

eta <- list(eta.1, eta.2)

#List of fixed parameters
theta.1 <- c(1, 1, 1)
theta.2 <- c(1, 1)
theta.fix <- list(theta.1, theta.2)

#Comparison table
p <- matrix(
    c(
        0,1,
        0,0
    ), c(length(eta), length(eta)), byrow = TRUE)

### Case 1

#List of variances
sq.var.1 <- function(x, theta.1)
    1

sq.var.2 <- function(x, theta.2)
```

```
    1

sq.var <- list(sq.var.1, sq.var.2)

#Case 1, method 1
res <- KLopt.lnorm(
    x = seq(0.1, 5, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 1)
)
plot(res)
summary(res)

#Case 1, method 2
res <- KLopt.lnorm(
    x = seq(0.1, 5, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 2)
)
plot(res)
summary(res)

### case 3
#List of variances
sq.var.1 <- function(x, theta.1)
    exp(eta.1(x, theta.1))

sq.var.2 <- function(x, theta.2)
    exp(eta.2(x, theta.2))

sq.var <- list(sq.var.1, sq.var.2)

#Case 3, method 1
res <- KLopt.lnorm(
    x = seq(0.1, 5, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 1)
)
plot(res)
summary(res)

#Case 3, method 2
res <- KLopt.lnorm(
    x = seq(0.1, 5, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 2)
)
plot(res)
summary(res)

### Example 2 from [8]; sigmoidal

#List of models
```

```
eta.1 = function(x, theta.1)
    theta.1[1] - theta.1[2] * exp(-theta.1[3] * x ^ theta.1[4])

eta.2 <- function(x, theta.2)
    theta.2[1] - theta.2[2] * exp(-theta.2[3] * x)

#List of fixed parameters
theta.1.mean <- c(2, 1, 0.8, 1.5)
sigma <- 0.3
theta.1.sigma <- matrix(
    c(
        sigma,0,
        0,sigma
    ), c(2, 2), byrow = TRUE)
grid <- expand.grid(
    theta.1.mean[1],
    theta.1.mean[2],
    seq(theta.1.mean[3] - sqrt(sigma), theta.1.mean[3] + sqrt(sigma), length.out = 5),
    seq(theta.1.mean[4] - sqrt(sigma), theta.1.mean[4] + sqrt(sigma), length.out = 5)
)

theta.2 <- c(2,1,1)

theta.fix <- list()
for(i in 1:length(grid[,1]))
    theta.fix[[length(theta.fix)+1]] <- as.numeric(grid[i,])
theta.fix[[length(theta.fix)+1]] <- theta.2

density.on.grid <- dmvnorm(grid[,3:4], mean = theta.1.mean[3:4], sigma = theta.1.sigma)
density.on.grid <- density.on.grid / sum(density.on.grid)

eta <- list()
for(i in 1:length(grid[,1]))
    eta <- c(eta, eta.1)
eta <- c(eta, eta.2)

#Comparison table
p <- rep(0,length(eta))
for(i in 1:length(grid[,1]))
    p <- rbind(p, c(rep(0,length(eta)-1), density.on.grid[i]))
p <- rbind(p, rep(0,length(eta)))
p <- p[-1,]

### Case 1

sq.var.1 <- function(x, theta.1)
    1

sq.var.2 <- function(x, theta.2)
    1

sq.var <- list()
for(i in 1:length(grid[,1]))
```

```
    sq.var <- c(sq.var, sq.var.1)
sq.var <- c(sq.var, sq.var.2)

#Case 1, method 1
res <- KLopt.lnorm(
    x = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 1)
)
plot(res)
summary(res)

#Case 1, method 2
res <- KLopt.lnorm(
    x = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 2)
)
plot(res)
summary(res)

### Case 3

sq.var.1 <- function(x, theta.1)
    exp(eta.1(x, theta.1))

sq.var.2 <- function(x, theta.2)
    exp(eta.2(x, theta.2))

sq.var <- list()
for(i in 1:length(grid[,1]))
    sq.var <- c(sq.var, sq.var.1)
sq.var <- c(sq.var, sq.var.2)

#Case 3, method 1
res <- KLopt.lnorm(
    x = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 1)
)
plot(res)
summary(res)

#Case 3, method 2
res <- KLopt.lnorm(
    x = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(method = 2)
)
plot(res)
summary(res)

### Example 3 from [8]; dose response
```

```
#List of models
eta.1 <- function(x, theta.1)
    theta.1[1] + theta.1[2] * x

eta.2 <- function(x, theta.2)
    theta.2[1] + theta.2[2] * x * (theta.2[3] - x)

eta.3 <- function(x, theta.3)
    theta.3[1] + theta.3[2] * x / (theta.3[3] + x)

eta.4 <- function(x, theta.4)
    theta.4[1] + theta.4[2] / (1 + exp((theta.4[3] - x) / theta.4[4]))

#List of fixed parameters
theta.1 <- c(60, 0.56)
theta.2 <- c(60, 7 / 2250, 600)
theta.3 <- c(60, 294, 25)
theta.4.mean <- c(49.62, 290.51, 150, 45.51)
a <- 45
b <- 20
grid <- expand.grid(
        c(theta.4.mean[1] - b, theta.4.mean[1], theta.4.mean[1] + a),
        c(theta.4.mean[2] - b, theta.4.mean[2], theta.4.mean[2] + a),
        c(theta.4.mean[3] - b, theta.4.mean[3], theta.4.mean[3] + a),
        c(theta.4.mean[4] - b, theta.4.mean[4], theta.4.mean[4] + a)
        )

eta <- list()
eta <- c(eta, eta.1, eta.2, eta.3)
for(i in 1:length(grid[,1]))
    eta <- c(eta, eta.4)

theta.fix <- list(theta.1, theta.2, theta.3)
for(i in 1:length(grid[,1]))
    theta.fix[[length(theta.fix) + 1]] <- as.numeric(grid[i,])

density.on.grid <- rep(1,length(grid[,1]))
density.on.grid <- density.on.grid / sum(density.on.grid)

#Comparison table
p <- rep(0, length(eta))
p <- rbind(p, c(1, rep(0, length(eta) - 1)))
p <- rbind(p, c(1, 1, rep(0,length(eta) - 2)))
for(i in 1:length(grid[,1]))
    p <- rbind(p, c(rep(density.on.grid[i], 3), rep(0, length(eta) - 3)))

### Case 1

#List of variances
sq.var.1 <- function(x, theta.1)
    1
```

```
sq.var.2 <- function(x, theta.2)
    1

sq.var.3 <- function(x, theta.3)
    1

sq.var.4 <- function(x, theta.4)
    1

sq.var <- list()
sq.var <- c(sq.var, sq.var.1, sq.var.2, sq.var.3)
for(i in 1:length(grid[,1]))
    sq.var <- c(sq.var, sq.var.4)

#Case 1, method 1

#Design estimation
res <- KLopt.lnorm(
    x = seq(0, 500, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(max.iter = 10)
)
plot(res)
summary(res)

#Case 1, method 2

#Design estimation
res <- KLopt.lnorm(
    x = seq(0, 500, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(
        method = 2,
        max.iter = 10,
        weights.evaluation.max.iter = 50,
        support.epsilon = 1e-4
    )
)
plot(res)
summary(res)

### Case 3

#List of variances
sq.var.1 <- function(x, theta.1)
    exp(1e-2 * eta.1(x,theta.1))

sq.var.2 <- function(x, theta.2)
    exp(1e-2 * eta.2(x,theta.2))

sq.var.3 <- function(x, theta.3)
    exp(1e-2 * eta.3(x,theta.3))
```

```
sq.var.4 <- function(x, theta.4)
    exp(1e-2 * eta.4(x,theta.4))

sq.var <- list()
sq.var <- c(sq.var, sq.var.1, sq.var.2, sq.var.3)
for(i in 1:length(grid[,1]))
    sq.var <- c(sq.var, sq.var.4)

#Case 3, method 1

#Design estimation
res <- KLopt.lnorm(
    x = seq(0, 500, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(max.iter = 10)
)
plot(res)
summary(res)

#Case 3, method 2

eta.2 <- function(x, theta.2)
    theta.2[1] + theta.2[2] * x - theta.2[3] * x * x

theta.2 <- c(60, 7 * 600 / 2250, 7 / 2250)

eta <- list()
eta <- c(eta, eta.1, eta.2, eta.3)
for(i in 1:length(grid[,1]))
    eta <- c(eta, eta.4)

theta.fix <- list(theta.1, theta.2, theta.3)
for(i in 1:length(grid[,1]))
    theta.fix[[length(theta.fix) + 1]] <- as.numeric(grid[i,])

#Design estimation
res <- KLopt.lnorm(
    x = seq(0, 500, length.out = 10),
    eta = eta, sq.var = sq.var, theta.fix = theta.fix, p = p,
    opt = list(max.iter = 6, method = 2)
)
plot(res)
summary(res)

## End(Not run)
```

`plot.KLopt.lnorm` *Plot of Ψ function for resulting design*

Description

Plots the function from the formulation of the equivalence theorem for resulting approximation ξ^{**} of the KL-optimal design achieved with the help of `KLopt.lnorm`.

Usage

```
## S3 method for class 'KLopt.lnorm'
plot(x, ...)
```

Arguments

`x`	an object of type "tpopt".
`...`	additional graphical parameters.

See Also

`tpopt`, `summary.tpopt`, `print.tpopt`

`plot.tpopt` *Plot of Ψ function for resulting design*

Description

Plots the $\Psi(x, \xi)$ function for resulting approximation ξ^{**} of the T_{P}-optimal design achieved with the help of `tpopt`. The definition of $\Psi(x, \xi)$ can be found in the "details" section of function's `tpopt` specifications.

Usage

```
## S3 method for class 'tpopt'
plot(x, ...)
```

Arguments

`x`	an object of type "tpopt".
`...`	additional graphical parameters.

Details

We are interested in the shape of function $\Psi(x, \xi^{**})$ when we want to ensure the convergence of the algorithm. If algorithm had converged, then support points of ξ^{**} (which are represented by dots) will be near local maximums of the mentioned function. Furthermore, at all local maximums $\Psi(x, \xi^{**})$ should have the same value. Otherwise something went wrong and the algorithm should be restarted with another parameters.

See Also

`tpopt`, `summary.tpopt`, `print.tpopt`

Examples

```
#List of models
eta.1 = function(x, theta.1)
    theta.1[1] + theta.1[2] * x + theta.1[3] * (x ^ 2) +
    theta.1[4] * (x ^ 3) + theta.1[5] * (x ^ 4)

eta.2 = function(x, theta.2)
    theta.2[1] + theta.2[2] * x + theta.2[3] * (x ^ 2)

eta <- list(eta.1, eta.2)

#List of fixed parameters
theta.1 <- c(1,1,1,1,1)
theta.2 <- c(1,1,1)
theta.fix <- list(theta.1, theta.2)

#Comparison table
p <- matrix(
    c(
        0, 1,
        0, 0
    ), c(length(eta), length(eta)), byrow = TRUE)

x <- seq(-1, 1, 0.1)
opt.1 <- list(method = 1, max.iter = 1)
opt.2 <- list(method = 1, max.iter = 2)
opt.3 <- list(method = 1)

res.1 <- tpopt(x = x, eta = eta, theta.fix = theta.fix, p = p, opt = opt.1)
res.2 <- tpopt(x = x, eta = eta, theta.fix = theta.fix, p = p, opt = opt.2)
res.3 <- tpopt(x = x, eta = eta, theta.fix = theta.fix, p = p, opt = opt.3)

plot(res.1)
plot(res.2)
plot(res.3)
```

---

print.KLopt.lnorm *Short information about the input*

Description

Prints short information about the input object of class “KLopt.lnorm”.

Usage

```
## S3 method for class 'KLopt.lnorm'
print(x, ...)
```

Arguments

x	an object of type "KLopt.lnorm".
...	further arguments.

Details

List of models, list of fixed parameters and resulting design are displayed.

See Also

KLopt.lnorm, summary.KLopt.lnorm, plot.KLopt.lnorm

---

print.tpopt *Short information about the input*

Description

Prints short information about the input object of class “tpopt”.

Usage

```
## S3 method for class 'tpopt'
print(x, ...)
```

Arguments

x	an object of type "tpopt".
...	further arguments.

Details

List of models, list of fixed parameters and resulting design are displayed.

See Also

tpopt, summary.tpopt, plot.tpopt

summary.KLopt.lnorm *Detailed information about the input*

Description

Prints detailed information about the input object of class “KLopt.lnorm”.

Usage

```
## S3 method for class 'KLopt.lnorm'
summary(object, ...)
```

Arguments

object	an object of type "KLopt.lnorm".
...	further arguments.

Details

Call, list of models, list of fixed parameters, resulting design, efficiency by iteration and overall execution time are displayed.

See Also

KLopt.lnorm, plot.KLopt.lnorm, print.KLopt.lnorm

summary.tpopt *Detailed information about the input*

Description

Prints detailed information about the input object of class “tpopt”.

Usage

```
## S3 method for class 'tpopt'
summary(object, ...)
```

Arguments

object	an object of type "tpopt".
...	further arguments.

Details

Call, list of models, list of fixed parameters, resulting design, efficiency by iteration and overall execution time are displayed.

See Also

tpopt, plot.tpopt, print.tpopt

---

tpopt *Calculation of optimal discriminating design*

Description

Calculates an approximation ξ^{**} of the T_{P}-optimal design ξ^{*} for discrimination between a given list of models $\{\eta_i(x,\theta_i),\ i=1,\dots,\nu\}$. This procedure is based on the algorithms developed by Holger Dette, Viatcheslav B. Melas and Roman Guchenko in [7]. T_{P}-optimal design is a probability measure, which maximizes the functional

$$T_{\mathrm{P}}(\xi)=\sum_{i,j=1}^{\nu} p_{i,j} \inf_{\theta_{i,j}\in\Theta_j} \int_{\mathcal{X}} \Big[\eta_i(x,\overline{\theta}_i)-\eta_j(x,\theta_{i,j})\Big]^2 \xi(dx),$$

where ξ is an arbitrary design on $\mathcal{X}$ (it is presumed here, that $\mathcal{X}$ is an interval from $\mathbf{R}$), $\mathrm{P}=\{p_{i,j}\}_{i,j=1}^{\nu}$ is a table of non-negative weights with zeros on the diagonal (comparison table) and $\overline{\theta}_i$ are predefined fixed parameters.

It was also shown in [7] that calculation of Bayesian T_{P}-optimal design, which maximizes more complicated criterion

$$T_{\mathrm{P}}^{\mathrm{B}}(\xi)=\sum_{i,j=1}^{\nu} p_{i,j} \int_{\Theta_i} \inf_{\theta_{i,j}\in\Theta_j} \int_{\mathcal{X}} \Big[\eta_i(x,\lambda_i)-\eta_j(x,\theta_{i,j})\Big]^2 \xi(dx)\mathcal{P}_i(d\lambda_i),$$

can be reduced to calculation of ordinary T_{P}-optimal design, when distributions $\mathcal{P}_i$ are discrete. That is why in this case the current function is also suitable for calculation of Bayesian designs.

Usage

```
tpopt(  x,
        w = rep(1, length(x)) / length(x),
        eta,
        theta.fix,
        theta.var = NULL,
        p,
        x.lb = min(x),
        x.rb = max(x),
        opt = list())
```

Arguments

`x` a numeric vector specifying support points from $\mathcal{X}$ for initial design. Current algorithm operates under the assumption, that $\mathcal{X}$ is an interval from $\mathbf{R}$

.

`w` a numeric vector specifying weights for initial design. This vector should have the same length as vector of support points. Furthermore, the weights of the design should sum to 1. If this vector is not specified, then the weights are presumed to be equal.

`eta` a list of models between which proposed optimization should be performed. Every function from this list should be defined in the form of $\eta_i(x, \theta_i)$, where x is one dimensional variable from $\mathcal{X}$ and θ_i is a vector of corresponding model parameters. We will refer to length of this list as ν.

`theta.fix` a list of fixed model parameters $\overline{\theta}_i$ from the functional T_{P}. This list should have the same length as the list of models.

`theta.var` an array with two dimensions specifying initial values for parameter vectors $\theta_{i,j}$. The default value here is NULL, which means that initial guess is calculated automatically.

`p` a $\nu \times \nu$ square table (R-matrix) containing non-negative weights for comparison. The diagonal values of this table should all be zeros. If one want to include comparison of the i'th model with fixed parameters against j'th model with variable parameters into optimization, then he/she should place non-negative weight $p_{i,j}$ into the table; otherwise this weight should be zero.

`x.lb` a left bound for support points. If it is not specified, then minimal value from input vector x is taken.

`x.rb` a right bound for support points. If it is not specified, then maximal value from input vector x is taken.

`opt` a list of options containing such named fields:

method a variable specifying the method to be used in inner weight optimization step. See details section for more info. The value "1" stands for quadratic programming based procedure and "2" stands for specific gradient method. See [7] for more details on that methods.

max.iter maximum number of iterations for the main loop. Reaching this number of iterations is one of the possible stopping conditions.

des.eff desired efficiency for resulted approximation of optimal design. Reaching efficiency of more than des.eff is another stopping condition (to be exact, efficiency lower bound is calculated on each iteration of the algorithm instead of efficiency). See details section for exact definition of efficiency.

derivative.epsilon a value that is used for numerical computation of first and second order derivatives.

support.epsilon a value that is used for support points exclusion, if corresponding weight's value is less then support.epsilon.

weights.evaluation.epsilon a value that is used in the inner loop for weights evaluation.

Details

Firstly, lets define

$$\Psi(x,\xi) = \sum_{i,j=1}^{\nu} p_{i,j}\left[\eta_i(x,\overline{\theta}_i) - \eta_j(x,\widehat{\theta}_{i,j})\right]^2, \widehat{\theta}_{i,j} = \arg\inf_{\theta_{i,j}\in\Theta_j} \int_{\mathcal{X}} \left[\eta_i(x,\overline{\theta}_i) - \eta_j(x,\theta_{i,j})\right]^2 \xi(dx).$$

The simplified algorithm schema is as follows:

Let ξ_s denotes the design obtained on the s'th iteration of the algorithm. Then

Step 1. Support of the new design ξ_{s+1} consists of all local maximums of function $\Psi(x,\xi_s)$ on $\mathcal{X}$ united with the support of current design ξ_s.

Step 2. Weights of the new design ξ_{s+1} are calculated so that the functional $T_{\mathrm{P}}(\xi)$ achieves its maximum in the class of all designs with support from previous step.

Value

Object of class "tpopt" which contains the following fields:

x the numeric vector of support points from $\mathcal{X}$ for resulting approximation of T_{P}-optimal design.

w the numeric vector of weights for resulting approximation of T_{P}-optimal design. The values of this vector sum to 1.

efficiency the numeric vector containing efficiency lower bound values by iteration. See details section for definition.

functional the numeric vector containing values of functional T_{P} by iteration.

eta the list of models, which is exactly the same as one from the arguments list.

theta.fix the list of fixed model parameters. It goes to the result without any changes too.

theta.var the array with two dimensions specifying calculated values for parameter vectors $\theta_{i,j}$ according to resulting design.

p, x.lb, x.rb same as in input.

max.iter max.iter from options list.

done.iter number of iterations done.

des.eff desired efficiency from options list.

time overall execution time.

References

[1] Atkinson A.C., Fedorov V.V. (1975) *The design of experiments for discriminating between two rival models*. Biometrika, vol. 62(1), pp. 57–70.

[2] Atkinson A.C., Fedorov V.V. (1975) *Optimal design: Experiments for discriminating between several models*. Biometrika, vol. 62(2), pp. 289–303.

[3] Dette H., Pepelyshev A. (2008) *Efficient experimental designs for sigmoidal growth models*. Journal of statistical planning and inference, vol. 138, pp. 2–17.

[4] Dette H., Melas V.B., Shpilev P. (2013) *Robust T-optimal discriminating designs*. Annals of Statistics, vol. 41(4), pp. 1693–1715.

[5] Braess D., Dette H. (2013) *Optimal discriminating designs for several competing regression models*. Annals of Statistics, vol. 41(2), pp. 897–922.

[6] Braess D., Dette H. (2013) *Supplement to "Optimal discriminating designs for several competing regression models"*. Annals of Statistics, online supplementary material.

[7] Dette H., Melas V.B., Guchenko R. (2014) *Bayesian T-optimal discriminating designs*. ArXiv link.

See Also

plot.tpopt, summary.tpopt, print.tpopt

Examples

```
### Auxiliary libraries for examples
library(mvtnorm)
### EMAX vs MM
#List of models
eta.1 <- function(x, theta.1)
    theta.1[1] + theta.1[2] * x / (x + theta.1[3])

eta.2 <- function(x, theta.2)
    theta.2[1] * x / (x + theta.2[2])

eta <- list(eta.1, eta.2)

#List of fixed parameters
theta.1 <- c(1, 1, 1)
theta.2 <- c(1, 1)
theta.fix <- list(theta.1, theta.2)

#Comparison table
p <- matrix(
    c(
        0, 1,
        0, 0
    ), c(2, 2), byrow = TRUE)

#Design estimation
res <- tpopt(x = c(1.2, 1.5, 1.7), eta = eta, theta.fix = theta.fix, p = p,
    x.lb = 1, x.rb = 2)

plot(res)
summary(res)

### Sigmoidal second
#List of models
eta.1 <- function(x, theta.1)
    theta.1[1] / (1 + theta.1[2] * exp(-theta.1[3] * x)) ^ theta.1[4]

eta.2 <- function(x, theta.2)
    theta.2[1] / (1 + theta.2[2] * exp(-theta.2[3] * x))
```

```
eta <- list(eta.1, eta.2)

#List of fixed parameters
theta.1 <- c(2, 5, 1, 2)
theta.2 <- c(3, 5, 0.7)
theta.fix <- list(theta.1, theta.2)

#Comparison table
p <- matrix(
    c(
        0, 1,
        0, 0
    ), c(2, 2), byrow = TRUE)

#Design estimation
res <- tpopt(x = seq(0, 10), eta = eta, theta.fix = theta.fix, p = p)

plot(res)
summary(res)

### Sigmoidal first
#List of models
eta.1 <- function(x, theta.1)
    theta.1[1] - theta.1[2] * exp(-theta.1[3] * x ^ theta.1[4])

eta.2 <- function(x, theta.2)
    theta.2[1] - theta.2[2] * exp(-theta.2[3] * x)

eta <- list(eta.1, eta.2)

#List of fixed parameters
theta.1 <- c(2, 1, 0.8, 1.5)
theta.2 <- c(2, 1, 1)
theta.fix <- list(theta.1, theta.2)

#Comparision table
p <- matrix(
    c(
        0, 1,
        0, 0
    ), c(2, 2), byrow = TRUE)

#Design estimation
res <- tpopt(x = seq(0, 10), eta = eta, theta.fix = theta.fix, p = p)

plot(res)
summary(res)

### Sigmoidal first --- Bayes

#List of fixed parameters
sigma <- sqrt(0.3)
theta.1.sigma <- matrix(
```

```
    c(
        sigma^2, 0,
        0, sigma^2
    ), c(2, 2), byrow = TRUE)
grid <- expand.grid(
    theta.1[1],
    theta.1[2],
    seq(theta.1[3] - sigma, theta.1[3] + sigma, length.out = 5),
    seq(theta.1[4] - sigma, theta.1[4] + sigma, length.out = 5)
    )

eta <- c(replicate(length(grid[,1]), eta.1, simplify = FALSE), eta.2)

theta.fix <- list()
for(i in 1:length(grid[,1]))
    theta.fix[[length(theta.fix) + 1]] <- as.numeric(grid[i,])
theta.fix[[length(theta.fix) + 1]] <- theta.2

density.on.grid <- dmvnorm(grid[,3:4], mean = theta.1[3:4], sigma = theta.1.sigma)
density.on.grid <- density.on.grid / sum(density.on.grid)

#Comparison table
p <- rep(0,length(eta))
for(i in 1:length(grid[,1]))
    p <- rbind(p, c(rep(0,length(eta) - 1), density.on.grid[i]))
p <- rbind(p, rep(0,length(eta)))
p <- p[-1,]

res <- tpopt(x = seq(0, 10), eta = eta, theta.fix = theta.fix, p = p)

plot(res)
summary(res)

### Dose response study
#List of models
eta.1 <- function(x, theta.1)
    theta.1[1] + theta.1[2] * x

eta.2 <- function(x, theta.2)
    theta.2[1] + theta.2[2] * x * (theta.2[3] - x)

eta.3 <- function(x, theta.3)
    theta.3[1] + theta.3[2] * x / (theta.3[3] + x)

eta.4 <- function(x, theta.4)
    theta.4[1] + theta.4[2] / (1 + exp((theta.4[3] - x) / theta.4[4]))

eta <- list(eta.1, eta.2, eta.3, eta.4)

#List of fixed parameters
theta.1 <- c(60, 0.56)
theta.2 <- c(60, 7/2250, 600)
theta.3 <- c(60, 294, 25)
```

```
theta.4 <- c(49.62, 290.51, 150, 45.51)

theta.fix <- list(theta.1, theta.2, theta.3, theta.4)

#Comparison table
p <- matrix(
    c(
        0, 0, 0, 0,
        1, 0, 0, 0,
        1, 1, 0, 0,
        1, 1, 1 ,0
    ), c(4, 4), byrow = TRUE)

#Design estimation
res <- tpopt(x = seq(0, 500, 100), eta = eta, theta.fix = theta.fix, p = p)

plot(res)
summary(res)

### Dose response study --- Bayes

#List of fixed parameters
sigma <- 37
theta.4.sigma <- matrix(
    c(
        sigma^2, 0, 0, 0,
        0, sigma^2, 0, 0,
        0, 0, sigma^2, 0,
        0, 0, 0, sigma^2
    ), c(4, 4), byrow = TRUE)
grid <- expand.grid(
    seq(theta.4[1] - sigma, theta.4[1] + sigma, length.out = 3),
    seq(theta.4[2] - sigma, theta.4[2] + sigma, length.out = 3),
    seq(theta.4[3] - sigma, theta.4[3] + sigma, length.out = 3),
    seq(theta.4[4] - sigma, theta.4[4] + sigma, length.out = 3)
    )

eta <- c(eta.1, eta.2, eta.3, replicate(length(grid[,1]), eta.4, simplify = FALSE))

theta.fix <- list(theta.1, theta.2, theta.3)
for(i in 1:length(grid[,1]))
    theta.fix[[length(theta.fix) + 1]] <- as.numeric(grid[i,])

density.on.grid <- dmvnorm(grid, mean = theta.4, sigma = theta.4.sigma)
density.on.grid <- density.on.grid / sum(density.on.grid)

#Comparison table
p <- rbind(
    rep(0, length(eta)),
    c(1, rep(0, length(eta) - 1)),
    c(1, 1, rep(0,length(eta) - 2))
    )
for(i in 1:length(grid[,1]))
```

```
    p <- rbind(p, c(rep(density.on.grid[i], 3), rep(0, length(eta) - 3)))

#Design estimation
## Not run:
res <- tpopt(x = seq(0, 500, 100), eta = eta, theta.fix = theta.fix, p = p)
## End(Not run)

plot(res)
summary(res)

## Not run:
### Example from [8]
### An example of how case 2 can be computed for example 1 in [8] with tpopt function

#List of models
eta.1 <- function(x, theta.1)
    log(theta.1[1] * x + theta.1[2] * x / (x + theta.1[3]))

eta.2 <- function(x, theta.2)
    log(theta.2[1] * x / (x + theta.2[2]))

eta <- list(eta.1, eta.2)

#List of fixed parameters
theta.1 <- c(1, 1, 1)
theta.2 <- c(1, 1)
theta.fix <- list(theta.1, theta.2)

#Comparison table
p <- matrix(
    c(
        0,1,
        0,0
    ), c(length(eta), length(eta)), byrow = TRUE)

#Case 2, method 1
#Design estimation
res <- tpopt(
    x = seq(0.1, 5, length.out = 10),
    eta = eta, theta.fix = theta.fix, p = p, x.lb = 0.1, x.rb = 5,
    opt = list(method = 1)
)
plot(res)
summary(res)

#Case 2, method 2
#Design estimation
res <- tpopt(
    x = seq(0.1, 5, length.out = 10),
    eta = eta, theta.fix = theta.fix, p = p, x.lb = 0.1, x.rb = 5,
    opt = list(method = 2)
)
plot(res)
```

```
summary(res)

## End(Not run)
```

Index