Package ‘rodd’

February 20, 2015

Type Package
Title Optimal Discriminating Designs
Version 0.1-1
Date 2014-10-28
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) 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 2014-12-14 01:05:16

R topics documented:

rodd-package ................................................................. 2
plot ................................................................. 2
print ................................................................. 4
summary ................................................................. 4
tpopt ................................................................. 5

Index 13
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 the package provides the routine `tpopt` for the construction of $T_P$-optimal designs and several auxiliary procedures to represent the results. Function `tpopt` is based on the algorithms that were recently developed by Holger Dette, Viatcheslav B. Melas and Roman Guchenko in [7] (see the references for more details).

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

**References**


**Description**

Plots the $\Psi(x, \xi)$ function for resulting approximation $\xi^*$ 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

```r
## 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 $\xi^{**}$ (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

```r
# List of models
eta.1 = function(x, theta.1)

eta.2 = function(x, theta.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)```
print  

*Short information about the input*

**Description**

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

**Usage**

```r
## 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  

*Detailed information about the input*

**Description**

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

**Usage**

```r
## 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`, `plotNtpopt`, `printNtpopt`

---

**tpopt**

*Calculation of optimal discriminating design*

**Description**

Calculates an approximation $\xi^{**}$ of the $T_P$-optimal design $\xi^*$ for discrimination between a given list of models $\{\eta_i(x, \theta_i), i = 1, \ldots, \nu\}$. 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

$$T_P(\xi) = \nu \sum_{i,j=1}^{\nu} p_{i,j} \inf_{\theta_{i,j}} \int_{\mathcal{X}} \left[ \eta_i(x, \bar{\theta}_i) - \eta_j(x, \theta_{i,j}) \right]^2 \xi(dx),$$

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

It was also shown in [7] that calculation of Bayesian $T_B$-optimal design, which maximizes more complicated criterion

$$T_B(\xi) = \nu \sum_{i,j=1}^{\nu} p_{i,j} \int_{\Theta_{i,j}} \inf_{\theta_{i,j}} \int_{\mathcal{X}} \left[ \eta_i(x, \lambda_i) - \eta_j(x, \theta_{i,j}) \right]^2 \xi(dx) P_i(d\lambda_i),$$

can be reduced to calculation of ordinary $T_P$-optimal design, when distributions $P_i$ are discrete. That is why in this case the current function is also suitable for calculation of Bayesian designs.

**Usage**

```r
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 \( \mathbb{R} \).

\( w \)  
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 a one-dimensional variable from \( \mathcal{X} \) and \( \theta_i \) is a vector of corresponding model parameters. We will refer to length of this list as \( \nu \).

\( \theta_{\text{fix}} \)  
A list of fixed model parameters \( \theta_{\text{fix}} \) from the functional \( T_P \). This list should have the same length as the list of models.

\( \theta_{\text{var}} \)  
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 \( i \)’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.

\( x_{\text{lb}} \)  
A left bound for support points. If it is not specified, then minimal value from input vector \( x \) is taken.

\( x_{\text{rb}} \)  
A right bound for support points. If it is not specified, then maximal value from input vector \( x \) is taken.

\( \text{opt} \)  
A list of options containing such named fields:

\textbf{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.

\textbf{max.iter}  
Maximum number of iterations for the main loop. Reaching this number of iterations is one of the possible stopping conditions.

\textbf{des.eff}  
Desired efficiency for resulted approximation of optimal design. Reaching efficiency of more than \( \text{des.eff} \) 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.

\textbf{derivative.epsilon}  
A value that is used for numerical computation of first and second order derivatives.

\textbf{support.epsilon}  
A value that is used for support points exclusion, if corresponding weight’s value is less than \( \text{support.epsilon} \).

\textbf{weights.evaluation.epsilon}  
A value that is used in the inner loop for weights evaluation.
Details

Firstly, let's define

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

The simplified algorithm schema is as follows:

Let \( \xi_s \) denotes the design obtained on the \( s \)'th iteration of the algorithm. Then

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

**Step 2.** Weights of the new design \( \xi_{s+1} \) are calculated so that the functional \( T_{\nu}(\xi) \) 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 \( T_{\nu} \)-optimal design.
- **w** the numeric vector of weights for resulting approximation of \( T_{\nu} \)-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 \( T_{\nu} \) 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


See Also

plot.tpopt, summary.tpopt, print.tpopt

Examples

```r
### Auxiliary libraries for examples
library(mvtnorm)
### EMAX vs MM
# List of models
eta.1 <- function(x, theta.1)

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,      
    1, 0,      
    0, 0,      
  ), 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)

eta.2 <- function(x, theta.2)
```

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)

eta.2 <- function(x, theta.2)

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)

# 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 --- 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],
)
etag <- 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,1])
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 <- 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)
eta.2 <- function(x, theta.2)
eta.3 <- function(x, theta.3)
eta.4 <- function(x, theta.4)
etag <- 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),
)
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)
Index

*Topic auxiliary
  plot, 2
  print, 4
  summary, 4

*Topic discriminative designs
  tpopt, 5

*Topic package
  rodd-package, 2

plot, 2
plot.tpopt, 4, 5, 8
print, 4
print.tpopt, 3, 5, 8

rodd (rodd-package), 2
rodd-package, 2

summary, 4
summary.tpopt, 3, 4, 8

tpopt, 2–5, 5