Package: OptimalDesign (via r-universe)

August 21, 2024

Type Package

Title A Toolbox for Computing Efficient Designs of Experiments

Version 1.0.1

Author Radoslav Harman, Lenka Filova

Maintainer Lenka Filova <OptimalDesignR@gmail.com>

Description Algorithms for D-, A-, I-, and c-optimal designs. Some of the functions in this package require the 'gurobi' software and its accompanying R package. For their installation, please follow the instructions at <https://www.gurobi.com> and the file gurobi_inst.txt, respectively.

License GPL-3

URL < http://www.iam.fmph.uniba.sk/design/ >

Depends R (>= 3.1.1)

Encoding UTF-8

LazyData true

Imports grDevices, graphics, Matrix, lpSolve, matrixStats, matrixcalc, plyr, quadprog, rgl, stats, utils

Enhances gurobi, slam

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-02 08:50:07 UTC

Contents

`OptimalDesign-package` *OptimalDesign*

Description

Procedures for computing D-, A-, I-, and c-optimal approximate and exact designs of experiments on finite domains, for regression models with uncorrelated observations.

Author(s)

Radoslav Harman, Lenka Filova

`dirder` *Vector of directional derivatives*

Description

Computes the vector of derivatives at a normalized approximate design `w` of length `n` in the directions of singular designs e_i, where i ranges from `1` to `n`.

Usage

```
dirder(Fx, w, crit="D", h=NULL, echo=TRUE)
```

Arguments

- `Fx` — the `n` times `m` matrix of candidate regressors (as rows), where `n` is the number of candidate design points and `m` (where `m>=2`, `m<=n`) is the number of parameters.
- `w` — a non-negative vector of length `n` representing the design. It is normalized prior to the computation of the directional derivatives.
- `crit` — the criterion; possible values are `"D"`, `"A"`, `"I"`, `"C"` and `"c"`.
- `h` — a non-zero vector of length `m` corresponding to the coefficients of the linear parameter combination of interest. If `crit` is not `"C"` nor `"c"` then `h` is ignored. If `crit` is `"C"` or `"c"` and `h=NULL` then `h` is assumed to be `c(0,...,0,1)`.
- `echo` — Print the call of the function?

Details

The i-th directional derivative measures the increase of the criterion value provided that we infinitesimally increase the i-th design weight (and decrease other weights by the same proportion). For a concave optimality criterion, an approximate design is optimal in the class of all normalized approximate designs if and only if all its directional derivatives are non-positive. This statement can be rewritten to the form of the so-called equivalence theorem. See the reference paper at http://www.iam.fmph.uniba.sk/design/ for mathematical details.

Value

The vector of directional derivatives of the chosen criterion at `w/sum(w)` in the direction of the singular designs e_i, where i ranges from `1` to `n`.

Note

The design `w` should have a non-singular information matrix.

Author(s)

Radoslav Harman, Lenka Filova

See Also

`effbound`, `varfun`

Examples

```
## Not run:
# The directional derivatives of the D-optimal approximate design
# for a cubic regression on a square grid.

form.cube <- ~x1 + x2 + I(x1^2) + I(x2^2) + I(x1*x2) +
              I(x1^3) + I(x1^2*x2) + I(x1*x2^2) + I(x2^3)
Fx <- Fx_cube(form.cube, n.levels = c(101, 101))
w <- od_REX(Fx)$w.best

# Because w is optimal approximate, no directional derivative is positive:
```

```
boxplot(dirder(Fx, w))

# The yellow values indicate the directional derivative at each design point:
od_plot(Fx, w, Fx[, 2:3])

# An alternative view is a "projection" of the above plot:
od_plot(Fx, w, Fx[, 2], dd.pool = c("max", "min"))

## End(Not run)
```

---

effbound *Lower bound on efficiency*

Description

Computes a lower bound on the efficiency of a design w in the class of all approximate designs of the same size as w.

Usage

```
effbound(Fx, w, crit="D", h=NULL, echo=TRUE)
```

Arguments

Fx	the n times m matrix of candidate regressors (as rows), where n is the number of candidate design points and m (where m>=2, m<=n) is the number of parameters.
w	a non-negative vector of length n representing the design.
crit	the criterion; possible values are "D", "A", "I", "C" and "c".
h	a non-zero vector of length m corresponding to the coefficients of the linear parameter combination of interest. If crit is not "C" nor "c" then h is ignored. If crit is "C" or "c" and h=NULL then h is assumed to be c(0,...,0,1).
echo	Print the call of the function?

Details

The lower bounds are based on the standard methods of convex analysis. See the reference paper at http://www.iam.fmph.uniba.sk/design/ for mathematical details.

Value

A lower bound on the D-, A-, I-, c-, or C-efficiency of w in the class of all approximate designs of the same size as w at the set of candidate regressors given by Fx.

Note

The design w should have a non-singular information matrix. Occasionally, the lower bound is very conservative. The exact value of the efficiency of w is the ratio of the criterion value of w and the criterion value of the optimal design.

Author(s)

Radoslav Harman, Lenka Filova

See Also

varfun, dirder

Examples

```
# A lower bound on the D-efficiencies of the uniform designs
# for the quadratic regression on a line grid

Fx <- Fx_cube(~x1 + I(x1^2), n.levels = 101)
effbound(Fx, rep(1/101, 101))

# The precise value of the D-efficiency
# requires computing the D-optimal design:

w.opt <- od_REX(Fx)$w.best
optcrit(Fx, rep(1/101, 101)) / optcrit(Fx, w.opt)

## Not run:
# Let us do this for polynomial regressions of various degrees:

n <- 101; d.max <- 10; x <- seq(-1, 1, length = n)
effs <- matrix(0, ncol = 2, nrow = d.max)
Fx <- matrix(1, ncol = 1, nrow = n)
for(d in 1:d.max) {
  Fx <- cbind(Fx, x^d)
  effs[d, 1] <- effbound(Fx, rep(1/n, n))
  w.opt <- od_REX(Fx)$w.best
  effs[d, 2] <- optcrit(Fx, rep(1/n, n)) / optcrit(Fx, w.opt)
}
print(effs)

# We see that the lower bound becomes more and more conservative
# compared to the real efficiency which actually increases with d.


# Compute a D-optimal design for the main effects model
# on a random subsample of a 6D cube

n <- 1000000; m <- 6
Fx <- cbind(1, matrix(runif(n*m), ncol = m))
w <- od_REX(Fx, eff = 0.99)$w.best
Fx <- od_DEL(Fx, w)$Fx.keep
w <- od_REX(Fx)$w.best

# Now we will compute a lower bound on efficiency of such design
# on the entire (continuous) cube:
Fx <- rbind(Fx, Fx_cube(~x1 + x2 + x3 + x4 + x5 + x6, lower = rep(0, 6)))
w <- c(w, rep(0, 2^6))
```

```
effbound(Fx, w)

# The real D-efficiency of w on the entire cube is
optcrit(Fx, w)/od_REX(Fx)$Phi.best

## End(Not run)
```

Fx_blocks *Matrix of candidate regressors for a block size-two model*

Description

Creates the matrix of all candidate regressors of a linear regression model corresponding to the problem of the optimal block size-two design.

Usage

```
Fx_blocks(n.treats, blocks=NULL, echo=TRUE)
```

Arguments

`n.treats`	the number of "treatments" in the block experiment.
`blocks`	the `2` times `n` matrix of all permissible blocks (that is, permissible pairings of treatments). If `blocks=NULL`, `blocks` is set to `combn(n.treats, 2)`, which means that all treatment pairings are permissible.
`echo`	Print the call of the function?

Details

Creates the matrix `Fx` of artificial regressors, such that the D- and A-optimal designs for the corresponding artificial LRM are are the same as what is called the D- and A-optimal design in the original block model with blocks of size two.

Value

the `n` times `m` matrix of all candidate regressors of an auxiliary linear regression model corresponding to the problem of the optimal block size-two design (`n` is `ncol(blocks)`, `m` is `n.treats-1`).

Note

This optimal design problem is equivalent to various optimum-subgraph problems, depending on the criterion.

Author(s)

Radoslav Harman, Lenka Filova

References

Harman R, Filova, L: Computing efficient exact designs of experiments using integer quadratic programming, Computational Statistics and Data Analysis 71 (2014) 1159-1167.

Sagnol G, Harman R: Computing Exact D-optimal designs by mixed integer second-order cone programming, The Annals of Statistics 43 (2015), 2198-2224.

See Also

`Fx_cube`, `Fx_simplex`, `Fx_glm`, `Fx_dose`, `Fx_survival`

Examples

```
## Not run:
# Compute a D-efficient block size-two design
# with 15 treatments and 10 blocks of size two

Fx <- Fx_blocks(10)
w <- od_KL(Fx, 15, t.max = 5)$w.best
des <- combn(10, 2)[, as.logical(w)]
print(des)


# We can visualize the design as a graph
library(igraph)
grp <- graph_(t(des), from_edgelist(directed = FALSE))
plot(grp, layout=layout_with_graphopt)

## End(Not run)
```

`Fx_CtoA` *Transformation of candidate regressors for regularized c-optimality*

Description

Pre-transforms the matrix of all candidate regressors to the form suitable for computing regularized c-optimal designs via A-optimum algorithms.

Usage

```
Fx_CtoA(Fx, h=NULL, echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` matrix of candidate regressors (as rows), where `n` is the number of candidate design points and `m` (where `m>=2`, `m<=n`) is the number of parameters.
`h`	a non-zero vector of length `m` corresponding to the coefficients of the linear parameter combination of interest. If `crit` is not `"C"` nor `"c"` then `h` is ignored. If `crit` is `"C"` or `"c"` and `h=NULL` then `h` is assumed to be `c(0,...,0,1)`.
`echo`	Print the call of the function?

Details

The standard c-optimal designs are often singular, which may render them unsuitable for practical use. The regularized c-optimality, which we call C-optimality, is an approach to computing designs that are non-singular, but still efficient with respect to the criterion of c-optimality. See http://www.iam.fmph.uniba.sk/design/ for more details.

Value

The n times m matrix Fx.trans of all candidate regressors with the following property: The A-optimal design for the problem defined by Fx.trans is the same as the regularized c-optimal (i.e., C-optimal) design for the problem defined by Fx.

Author(s)

Radoslav Harman and Lenka Filova

See Also

Fx_ItoA

Examples

```
# We will compute a C-efficient (regularized c-optimal) design
# for estimating the mean response in x=1 for a quadratic regression
# using and algorithm for A-optimality.

Fx <- Fx_cube(~x1 + I(x1^2), n.levels=101)
Fx.trans <- Fx_CtoA(Fx, h=c(1, 1, 1))
w <- od_REX(Fx.trans, crit="A")$w.best
od_print(Fx, w, h=c(1, 1, 1))

# Compare the design to the (non-regularized) c-optimal design
w.crisp <- od_REX(Fx, crit="c", h=c(1, 1, 1))$w.best
od_print(Fx, w.crisp, h=c(1, 1, 1))

# The c-efficiency of the C-optimal design is about 0.68
# The D-efficiency of the c-optimal design is 0
# The D-efficiency of the C-optimal design is a very decent
optcrit(Fx, w) / od_REX(Fx)$Phi.best
```

Fx_cube *Matrix of candidate regressors for a model on a cuboid grid*

Description

Creates the matrix of all candidate regressors for a factor regression model on a cuboid grid (up to 9 factors).

Usage

```
Fx_cube(formula, lower=NULL, upper=NULL, n.levels=NULL, echo=TRUE)
```

Arguments

`formula`	the formula of the model. The rules for creating the formula are standard for R but: 1) the formula must not contain the dependent variable (it is one-sided); 2) the `d` factors (variables) must be labeled `x1,x2,x3,`...
`lower`	the `d`-dimensional vector of the smallest values of factors. If `lower=NULL`, the program sets `lower <- rep(-1, d)`.
`upper`	the `d`-dimensional vector of the largest values of factors. If `upper=NULL`, the program sets `upper <- rep(1, d)`.
`n.levels`	the `d`-dimensional vector of the numbers of levels of each factor. If `n.levels=NULL`, the program sets `n.levels <- rep(2, d)`.
`echo`	Print the call of the function?

Value

The `n` times `m` matrix of all candidate regressors for a factor regression model on a cuboid grid. The rows of `Fx` are the regressors `f(x)` for all candidate design points `x`.

Note

Note that `Fx` is *not* the design matrix (which is also sometimes called the regression matrix, or the model matrix). The design matrix depends on `Fx` as well as on the exact experimental design `w`. For this package, an exact experimental design is formalized as the vector of non-negative integer values corresponding to the replication of trials (observations) in individual design points. Thus, if `Fx` is the matrix of all candidate regressors and `w` is the exact design then `Fx[rep(1:nrow(Fx), w),]` is the actual design matrix for the experiment.

Author(s)

Radoslav Harman, Lenka Filova

See Also

`Fx_simplex`, `Fx_blocks`, `Fx_glm`, `Fx_survival`, `Fx_dose`

Examples

```
## Not run:
# The Fx for the cubic model on a discretized interval
Fx <- Fx_cube(~x1 + I(x1^2) + I(x1^3), lower=0, upper=2, n.levels=101)

# The D-optimal design of size 20
w <- od_KL(Fx, 20, t.max=5)$w.best
od_plot(Fx, w, Fx[, 2])
```

```
# The Fx for the full quadratic response surface model on a non-convex region
Fx <- Fx_cube(~x1 + x2 + I(x1^2) + I(x2^2) + I(x1*x2), n.levels=c(51, 51))
keep <- rep(TRUE, nrow(Fx))
for(i in 1:nrow(Fx)) if(prod(abs(Fx[i, 2:3])) > 0.2) keep[i] <- FALSE
Fx <- Fx[keep, ]

# The D-optimal design of size 29 without replications
w <- od_KL(Fx, 29, bin=TRUE, t.max=5)$w.best
od_plot(Fx, w, Fx[, 2:3])


# The Fx for the chemical weighing with 3 items and a bias term
Fx <- Fx_cube(~x1 + x2 + x3, n.levels=c(3, 3, 3))

# The D-optimal design of size 12
w <- od_KL(Fx, 12, t.max=2)$w.best
od_plot(Fx, w, Fx[, 2:4])

## End(Not run)
```

`Fx_dose` *Matrix of candidate regressors for a dose-response model*

Description

Creates the matrix of all candidate regressors for a linearization of a dose response model.

Usage

```
Fx_dose(dose.levels, theta0, dose.model="emax", echo=TRUE)
```

Arguments

`dose.levels`	the n-dimensional vector of admissible doses.
`theta0`	the 3-dimensional vector of values of the unknown parameter in which to linearize the model.
`dose.model`	the type of the dose-response model, possible values are `"emax"`, `"loglin"`, and `"exp"`.
`echo`	Print the call of the function?

Details

For mathematical details, see the referenced paper.

Value

The `n` times `3` matrix of all candidate regressors of a dose-response model linearized in `theta0`.

Author(s)

Radoslav Harman, Lenka Filova

References

Dette H, Kiss C, Bevanda M, Bretz F (2010). Optimal designs for the EMAX, log-linear and exponential models. Biometrika, 97(2), 513-518.

See Also

Fx_cube, Fx_simplex, Fx_blocks, Fx_glm, Fx_survival

Examples

```
# The loglinear model for the doses 1:150
# Localized at the values of theta0=c(0, 0.0797, 1)
Fx <- Fx_dose(1:150, c(0, 0.0797, 1), dose.model="loglin")

# The locally D-optimal approximate design
w_a <- od_REX(Fx)$w.best
od_plot(Fx, w_a, 1:150)

# The locally D-optimal exact design of size 10
w_e <- od_KL(Fx, 10, t.max=3)$w.best
od_plot(Fx, w_e, 1:150)
```

Fx_glm *Matrix of candidate regressors for a generalized linear model*

Description

Creates the matrix of all candidate regressors for a linearization of a generalized linear model.

Usage

```
Fx_glm(formula, theta0, glm.model="bin-logit", lower=NULL, upper=NULL,
       n.levels=NULL, echo=TRUE)
```

Arguments

formula	the formula of the linear part of the model. The rules for creating the formula are standard for R but: 1) the formula must not contain the dependent variable (it is one-sided); 2) the d factors (variables) must be labeled x1,x2,x3,...
theta0	the d-dimensional vector of values of the unknown parameter in which to linearize the model
glm.model	the type of the generalized linear model. Available models are "bin-logit", "bin-probit", "bin-cloglog", and Poisson-log.

lower	the d-dimensional vector of the smallest values of factors. If lower=NULL, the program sets lower <- rep(-1, d).
upper	the d-dimensional vector of the largest values of factors. If upper=NULL, the program sets upper <- rep(1, d).
n.levels	the d-dimensional vector of the numbers of levels of each factor. If n.levels=NULL, the program sets n.levels <- rep(2, d).
echo	Print the call of the function?

Details

For mathematical details, see the referenced paper.

Value

The n times m matrix of all candidate regressors of a generalized linear regression model linearized in theta0.

Author(s)

Radoslav Harman, Lenka Filova

References

Atkinson AC, Woods DC (2015). Designs for generalized linear models. Handbook of Design and Analysis of Experiments, 471-514.

See Also

Fx_cube, Fx_simplex, Fx_blocks, Fx_survival, Fx_dose

Examples

```
# The logistic model with second-order predictors x1, x2 in [-1,1]
# discretized into 21 points and theta0=c(1, 2, 2, -1, -1.5, 1.5)

form.quad <- ~ x1 + x2 + I(x1*x2) + I(x1^2) + I(x2^2)
Fx <- Fx_glm(form.quad, c(1, 2, 2, -1, -1.5, 1.5),
             glm.model="bin-logit", n.levels=c(21,21))

# The locally D-optimal approximate design
w <- od_REX(Fx)$w.best
Fx.lin <- Fx_cube(form.quad, n.levels=c(21,21)) # Just for the plot
od_plot(Fx, w, Fx.lin[, 2:3], dd.size=2)

## Not run:
#The GLM with Poisson link and 2 linear predictors x1,x2 in [-1,1]
# discretized into 21 points and theta0=c(0,2,2)
Fx <- Fx_glm(~x1+x2, c(0, 2, 2), glm.model="Poisson-log", n.levels=c(21, 21))

# The locally D-optimal exact design of size 50 without replications
```

```
w <- od_KL(Fx, 50, bin=TRUE, t.max=5)$w.best
Fx.lin <- Fx_cube(~x1+x2, n.levels=c(21, 21))
od_plot(Fx, w, Fx.lin[, 2:3], w.lim=Inf)

## End(Not run)
```

Fx_ItoA *Transformation of candidate regressors for I-optimality*

Description

Pre-transforms the matrix of all candidate regressors to the form suitable for computing I-optimal designs via A-optimum algorithms.

Usage

```
Fx_ItoA(Fx, echo=TRUE)
```

Arguments

Fx	the n times m matrix of candidate regressors (as rows), where n is the number of candidate design points and m (where m>=2, m<=n) is the number of parameters.
echo	Print the call of the function?

Details

It is simple to see that the problem of I-optimality is equivalent to the problem of A-optimality for a transformed matrix of candidate regressors. This function performs the transformation. See http://www.iam.fmph.uniba.sk/design/ for more details.

Value

The n times m matrix Fx.trans of all candidate regressors with the following property: The A-optimal design for the problem defined by Fx.trans is the same as the I-optimal design for the problem defined by Fx.

Note

It is also simple to transform the *weighted* I-optimality to A-optimality; just multiply the rows of Fx by the squares roots of weights of individual design points and transform the resulting matrix by Fx_ItoA.

Author(s)

Radoslav Harman, Lenka Filova

See Also

Fx_CtoA

Examples

```
## Not run:
# Compute an I-efficient exact size 20 design without replications
# for the Scheffe mixture model with 4 components
# using the AQUA heuristic for A-optimality.

Fx <- Fx_simplex(~x1 + x2 + x3 + x4 + I(x1*x2) + I(x1*x3) + I(x1*x4) +
                 I(x2*x3) + I(x2*x4) + I(x3*x4) - 1, 11)

w <- od_AQUA(Fx_ItoA(Fx), b3=24, bin=TRUE, crit="I", conic=FALSE)$w.best
od_plot(Fx, w, Fx[, 2:4])

## End(Not run)
```

`Fx_simplex` *Matrix of candidate regressors for a regression model on a simplex grid*

Description

Creates the matrix of all candidate regressors for a mixture regression model on a regular simplex grid (up to 9 factors).

Usage

```
Fx_simplex(formula, n.levels.mix=NULL, echo=TRUE)
```

Arguments

`formula`	the formula of the model. The rules for creating the formula are standard for R but: 1) the formula must not contain the dependent variable (it is one-sided); 2) the `d` factors (variables) must be labeled `x1,x2,x3,...`
`n.levels.mix`	the number of levels of each factor (each factor has the same number of levels). If `n.levels=NULL`, the program sets `n.levels <- 2*d + 1`.
`echo`	Print the call of the function?

Value

The `n` times `m` matrix of all candidate regressors of a mixture regression model on a regular simplex grid.

Note

Note that `Fx` is *not* the design matrix (which is also sometimes called the regression matrix, or the model matrix). The design matrix depends on `Fx` as well as on the exact experimental design `w`. For this package, an exact experimental design is formalized as the vector of non-negative integer values corresponding to the replication of trials (observations) in individual design points. Thus, if `Fx` is the matrix of all candidate regressors and `w` is the exact design then `Fx[rep(1:nrow(Fx), w),]` is the actual design matrix for the experiment.

Author(s)

Radoslav Harman, Lenka Filova

See Also

`Fx_cube`, `Fx_glm`, `Fx_dose`, `Fx_survival`, `Fx_blocks`

Examples

```
## Not run:
# The Fx of the Scheffe quadratic mixture model
# with 3 mixture components, each with 21 levels.
Fx <- Fx_simplex(~x1 + x2 + x3 + I(x1*x2) + I(x1*x3) + I(x2*x3) - 1, 21)

# The approximate I-optimal design of size 20
# bound by 1 at each design point
w <- od_MISOCP(Fx, b3=20, bin=TRUE, crit="I", type="approximate")$w.best
od_plot(Fx, w, Fx[, 2:3])

# As above, with constraints on the proportions
r <- c(); for (i in 1:nrow(Fx)) if (max(Fx[i, 2:4]) > 0.7) r <- c(r, i)
w <- od_MISOCP(Fx[-r, ], b3=20, bin=TRUE, crit="I", type="approximate")$w.best
od_plot(Fx[-r, ], w, Fx[-r, 2:3])

# Note that one must be careful when choosing a model for a mixture experiment:
# Let us compute the matrix of regressors of the simple linear mixture model
# with 4 mixture components, each with levels {0, 0.5, 1}.

Fx <- Fx_simplex(~x1 + x2 + x3 + x4, 3)

# The model has only 5 parameters and as many as 10 design points,
# but there is no design that guarantees estimability of the parameters.
# This can be shown by evaluating:
det(infmat(Fx, rep(1, 10)))

## End(Not run)
```

`Fx_survival` *Matrix of candidate regressors for a survival model*

Description

Creates the matrix of all candidate regressors for a linearization of a proportional hazards survival model.

Usage

```
Fx_survival(formula, theta0, censor.time, survival.model="phI", lower=NULL,
             upper=NULL, n.levels=NULL, echo=TRUE)
```

Arguments

`formula`	the formula of the linear part of the model. The rules for creating the formula are standard for R but: 1) the formula must not contain the dependent variable (it is one-sided); 2) the `d` factors (variables) must be labeled `x1,x2,x3,...`
`theta0`	the `d`-dimensional vector of values of the unknown parameter in which to linearize the model.
`censor.time`	the censoring time, a positive constant.
`survival.model`	the type of the survival model, can be either proportional hazards with Type I censoring (`"phI"`) or with random censoring (`"phrand"`). Both models assume a constant baseline hazard.
`lower`	the `d`-dimensional vector of the smallest values of factors. If `lower=NULL`, the program sets `lower <- rep(-1, d)`.
`upper`	the `d`-dimensional vector of the largest values of factors. If `upper=NULL`, the program sets `upper <- rep(1, d)`.
`n.levels`	the `d`-dimensional vector of the numbers of levels of each factor. If `n.levels=NULL`, the program sets `n.levels <- rep(2, d)`.
`echo`	Print the call of the function?

Details

For mathematical details, see the referenced paper.

Value

The `n` times `m` matrix of all candidate regressors of a proportional hazards model linearized in `theta0`.

Author(s)

Radoslav Harman, Lenka Filova

References

Konstantinou M, Biedermann S, Kimber A (2014). Optimal designs for two-parameter nonlinear models with application to survival models. Statistica Sinica, 24(1), 415-428.

See Also

`Fx_cube`, `Fx_simplex`, `Fx_blocks`, `Fx_glm`, `Fx_dose`

Examples

```
# The proportional hazards model with random censoring
# for three binary explanatory variables x1,x2,x3 without intercept
# censoring time 30 and parameter values theta0=c(1,1,1)
Fx <- Fx_survival(~x1 + x2 + x3 - 1, c(1, 1, 1), 30, "phrand",
      lower = c(0, 0, 0), upper = c(1, 1, 1), n.levels = c(2, 2, 2))
```

```
# The locally D-optimal approximate design
w <- od_REX(Fx, crit="D")$w.best
od_print(Fx, w, Fx)

## Not run:
# The proportional hazards model with random censoring
# for explanatory variables x1,x2,x3 in the range [0,1] discretized into 11 points
# censoring time 30 and parameter values theta0=c(1,1,1)
Fx <- Fx_survival(~x1 + x2 + x3 - 1, c(1, 1, 1), 30, "phrand",
      lower = c(0, 0, 0), upper = c(1, 1, 1), n.levels = c(11, 11, 11))

# The locally A-optimal exact design of size 50 without replications
w <- od_KL(Fx, 50, crit="A", bin=TRUE, t.max=5)$w.best
od_plot(Fx, w, Fx)

## End(Not run)
```

---

infmat — *Information matrix of a design*

Description

Computes the information matrix of a design `w` in the model determined by the matrix `Fx` of candidate regressors.

Usage

```
infmat(Fx, w, echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` matrix of candidate regressors (as rows), where `n` is the number of candidate design points and `m` (where `m>=2`, `m<=n`) is the number of parameters.
`w`	a non-negative vector of length `n` representing the design.
`echo`	Print the call of the function?

Value

The information matrix of the design `w` in the model with all candidate regresors given by the rows of `Fx`.

Note

The information matrix is standardized, i.e., it assumes that the variance of the errors is 1.

Author(s)

Radoslav Harman, Lenka Filova

See Also

optcrit

Examples

```
# Compute its information matrix for the design that is
# uniform on all the points with at most two levels equal to 1
# in the main effects model with 2 factors.

Fx <- Fx_cube(~x1 + x2 + x3 + x4 + x5, lower = rep(0, 5))
w <- rep(0, 2^5)
for (i in 1:(2^5)) if (sum(Fx[i, 2:6]) <= 2) w[i] <- 1
print(M <- infmat(Fx, w))

## Not run:
# Visualize the correlation matrix of the parameter estimators

V <- solve(M); Y <- diag(1/sqrt(diag(V)))
library(corrplot); corrplot(Y %*% V %*% Y)

## End(Not run)
```

---

mvee_REX *Minimum-volume enclosing ellipsoid*

Description

Computes the shape matrix H and the center z of the minimum-volume ellipsoid enclosing a finite set of data-points.

Usage

```
mvee_REX(Data, alg.AA="REX", eff=0.999999, it.max=Inf, t.max=60,
          picture=FALSE, echo=TRUE, track=TRUE)
```

Arguments

Data	the n times d (where d<n) matrix containing the d-dimensional data-vectors as rows.
alg.AA	the underlying computational method for approximate D-optimal design; possible values are "REX", "MUL" and "VDM".
eff	the minimum required efficiency.
it.max	a limit on the number of iterations of the underlying D-optimum approximate design algorithm.
t.max	a limit on the time of computation.
picture	Should a picture be plotted? (For the picture, the data need to be either two- or three-dimensional.)

echo	Print the call of the function?
track	Display the progress of the computation?

Details

The problem of the minimum-volume data-enclosing ellipsoid (MVEE) is computationally equivalent to the problem of D-optimal approximate design for an artificial problem based on the data. This procedure performs the computation and the proper conversion of the D-optimal approximate design to the MVEE parameters (the center and the shape matrix).

Value

Output is a list with components:

call	the call of the function
H	the shape matrix of the MVEE
z	the center of the MVEE
bpts	a set containing the boundary points of the MVEE
vol	the volume of the MVEE
eff.best	the actual precision of the result (1 is the perfect precision)
t.iter	the number of iterations of the underlying D-optimum design algorithm
t.act	the actual time of the computation

Note

Note: The affine hull of the rows of `X` should be the full space of dimension `d`. For the choice of the algorithm, see the comments in `od_REX`.

Author(s)

Radoslav Harman, Lenka Filova

References

Harman R, Filova L, Richtarik P (2019). A randomized exchange algorithm for computing optimal approximate designs of experiments. Journal of the American Statistical Association, 1-30.

See Also

`od_REX`

Examples

```
# Generate random 1000 points in a 3-dimensional space
# and compute the MVEE

Data <- matrix(rnorm(3000), ncol = 3)
mvee_REX(Data, picture = FALSE)
```

od_AQUA *Efficient exact design using a quadratic approximation*

Description

Computes an efficient exact design under general linear constraints via a quadratic approximation of the optimality criterion.

Usage

```
od_AQUA(Fx, b1=NULL, A1=NULL, b2=NULL, A2=NULL, b3=NULL, A3=NULL, w0=NULL,
      bin=FALSE, crit="D", h=NULL, M.anchor=NULL, ver.qa="+", conic=TRUE,
      t.max=120, echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` (where `m>=2`, `m<=n`) matrix containing all candidate regressors (as rows), i.e., `n` is the number of design points, and `m` (where `m>=2`) is the number of parameters
`b1, A1, b2, A2, b3, A3`	the real vectors and matrices that define the constraints on permissible designs `w` as follows: `A1 %*% w <= b1`, `A2 %*% w >= b2`, `A3 %*% w == b3`. Each of the arguments can be `NULL`, but at least one of `b1`, `b2`, `b3` must be non-`NULL`. If some `bi` is non-`NULL` and `Ai` is `NULL`, then `Ai` is set to be `matrix(1, nrow=1, ncol=n)`.
`w0`	a non-negative vector of length `n` representing the design to be augmented (i.e., the function adds the constraint `w >= w0` for permissible designs `w`). This argument can also be `NULL`; in that case, `w0` is set to the vector of zeros.
`bin`	Should each design point be used at most once?
`crit`	The optimality criterion. Possible values are `"D"`, `"A"`, `"I"`, `"C"`.
`h`	a non-zero vector of length `m` corresponding to the coefficients of the linear parameter combination of interest. If `crit` is not `"C"` then `h` is ignored. If `crit` is `"C"` and `h=NULL` then `h` is assumed to be `c(0,...,0,1)`.
`M.anchor`	the `m` times `m` information matrix of the optimal or nearly-optimal approximate design for the design problem (for the non-normalized version of the problem and including the design constraints). The argument `M.anchor` can also be `NULL`. In that case the procedure computes `M.anchor` using an appropriate approximate design procedure from the package.
`ver.qa`	version of the criterion; possible values are `"+"` and `"-"`.
`conic`	Should the conic reformulation be used?
`t.max`	the time limit for the computation.
`echo`	Print the call of the function?

Details

At least one of `b1`, `b2`, `b3` must be non-`NULL`. If `bi` is non-`NULL` and `Ai` is `NULL` for some `i` then `Ai` is set to be the vector of ones. If `bi` is `NULL` for some `i` then `Ai` is ignored.

Value

A list with the following components:

`call`	The call of the function.
`w.best`	The permissible design found, or `NULL`. The value `NULL` indicates a failed computation.
`supp`	The indices of the support of `w.best`.
`w.supp`	The weights of `w.best` on the support.
`M.best`	The information matrix of `w.best` or `NULL` if `w.best` is `NULL`.
`Phi.best`	The value of the criterion of optimality of the design `w.best`. If `w.best` has a singular information matrix or if the computation fails, the value of `Phi.best` is `0`.
`status`	The status variable of the gurobi optimization procedure; see the gurobi solver documentation for details.
`t.act`	The actual time of the computation.

Note

The function does not support the classical c-optimality, but it includes its regularized version referred to as C-optimality. The computation is generally stable, but it may fail for instance if the model is numerically singular, there is no exact design satisfying the constraints, no permissible exact design was found within the time limit, the set of feasible exact designs is unbounded and so on; see the `status` variable for more details. Note, however, that `status = "OPTIMAL"` indicates that the auxiliary integer programming problem was completely solved, which for this procedure does not guarantee that the result is a globally optimal design.

Author(s)

Radoslav Harman, Lenka Filova

References

Harman R., Filova L. (2014): Computing efficient exact designs of experiments using integer quadratic programming, Computational Statistics & Data Analysis, Volume 71, pp. 1159-1167

Filova L., Harman R. (2018). Ascent with Quadratic Assistance for the Construction of Exact Experimental Designs. arXiv preprint arXiv:1801.09124. (Submitted to Computational Statistics)

See Also

`od_KL`, `od_RC`, `od_MISOCP`

Examples

```
## Not run:
# Compute an I-efficient non-replicated exact design of size 51
# for the "special cubic" model with 3 mixture components
```

```
# Each factor has 11 levels:
form.sc <- ~x1 + x2 + x3 + I(x1*x2) + I(x1*x3) + I(x2*x3) + I(x1*x2*x3) - 1
Fx <- Fx_simplex(form.sc, 11)
w <- od_AQUA(Fx, b3 = 51, crit = "I", bin = TRUE)$w.best
od_plot(Fx, w, Fx[, 1:3])

# Each factor has 101 levels (memory intensive without the conic trick)
Fx <- Fx_simplex(form.sc, 101)
w <- od_AQUA(Fx, b3 = 51, crit = "I", bin = TRUE, t.max = 10)$w.best
od_plot(Fx, w, Fx[, 1:3])

# Find an A-efficient exact design for the spring balance model
# with 5 items and 10 weighings
Fx <- Fx_cube(~x1 + x2 + x3 + x4 + x5 - 1, lower = rep(0, 5))
w <- od_AQUA(Fx, b3 = 10, crit = "A", t.max = 10)$w.best
od_print(Fx, w)


## End(Not run)
```

---

od_DEL *Removal of redundant design points*

Description

Removes the design points (or, equivalently, candidate regressors) that cannot support an optimal approximate design.

Usage

```
od_DEL(Fx, w, crit = "D", h=NULL, echo = TRUE)
```

Arguments

Fx	the n times m (where m>=2, m<=n) matrix containing all candidate regressors (as rows), i.e., n is the number of candidate design points, and m is the number of parameters
w	a non-negative vector of length n representing the design
crit	the optimality criterion. Possible values are "D", "A", "I", "C".
h	a non-zero vector of length m corresponding to the coefficients of the linear parameter combination of interest. If crit is not "C" nor "c" then h is ignored. If crit is "C" or "c" and h=NULL then h is assumed to be c(0,...,0,1).
echo	Print the call of the function?

Value

Output is the list with components:

`call`	the call of the function
`keep`	the indices of `w` that have not been removed
`w.keep`	the approximate design on the reduced space
`Fx.keep`	the model matrix of the regressors on the reduced space

Note

The design vector `w` should have a non-singular information matrix. The procedure is valid only for the standard (size) constraint.

Author(s)

Radoslav Harman, Lenka Filova

References

Harman R, Pronzato L (2007): Improvements on removing non-optimal support points in D-optimum design algorithms, Statistics & Probability Letters 77, 90-94

Pronzato L (2013): A delimitation of the support of optimal designs for Kiefers Phi_p-class of criteria. Statistics & Probability Letters 83, 2721-2728

Examples

```
## Not run:
# Generate a model matrix for the quadratic model
# on a semi-circle with a huge number of design points
form.q <- ~x1 + x2 + I(x1^2) + I(x2^2) + I(x1*x2)
Fx <- Fx_cube(form.q, lower = c(-1, 0), n.levels = c(1001, 501))
remove <- (1:nrow(Fx))[Fx[ ,2]^2 + Fx[ ,3]^2 > 1]
Fx <- Fx[-remove, ]

# Compute an approximate design w with an efficiency of cca 0.999
w <- od_REX(Fx, eff = 0.999)$w.best

# Remove the redundant design points based on w
Fx <- od_DEL(Fx, w)$Fx.keep

# Now an almost perfect design can be computed very rapidly:
w <- od_REX(Fx, eff = 0.9999999999)$w.best

# Plotting of the relevant directional derivative is also faster:
od_plot(Fx, w, Fx[ , 2:3], dd.size = 0.1)

## End(Not run)
```

od_KL *The KL exchange algorithm for efficient exact designs*

Description

Computes an optimal or near-optimal exact design of experiments under the standard (size) constraint on the size of the experiment.

Usage

```
od_KL(Fx, N, bin=FALSE, Phi.app=NULL, crit="D", h=NULL, w1=NULL, K=NULL,
      L=NULL, rest.max=Inf, t.max=120, echo=TRUE, track=TRUE)
```

Arguments

Fx	the n times m (where m>=2, m<=n) matrix containing all candidate regressors (as rows), i.e., n is the number of candidate design points, and m (where m>=2) is the number of parameters.
N	the size of the experiment (i.e., the required number of trials).
bin	Should each design point be used at most once?
Phi.app	the optimal value of the corresponding approximate (relaxed) problem. If Phi.app = NULL, the value is pre-computed using od_REX.
crit	the optimality criterion. Possible values are "D", "A", "I", "C".
h	a non-zero vector of length m corresponding to the coefficients of the linear parameter combination of interest. If crit is not "C" nor "c" then h is ignored. If crit is "C" or "c" and h=NULL then h is assumed to be c(0,...,0,1).
w1	the initial design; it must have a non-singular information matrix and the size sum(w1) of w1 must be N. The default option w1 = NULL prompts the algorithm to generate its own initial design using od_PIN.
K, L	integer numbers (or Inf) representing parameters of the method. Various combinations of K and L lead to specific variants of the exchange method. If K = NULL or L = NULL, the algorithm automatically chooses appropriate values.
rest.max	the limit on the number of restarts of the method.
t.max	the time limit for the computation.
echo	Print the call of the function?
track	Display the progress of the computation?

Details

This implementation of the KL algorithm is loosely based on the ideas described in Atkinson et al. (2007); see the references.

The tuning parameter K is the (upper bound on the) number of "least promising" support points of the current design, for which exchanges are attempted. The tuning parameter L is the (upper bound on the) number of "most promising" candidate design points for which exchanges are attempted.

The implemented method is greedy in the sense that each improving exchange is immediately executed. If the algorithm stops in a local optimum before the allotted time elapsed, the computation is restarted with a random initial design (independent of `w1`). The final result is the best design found within all restarts.

The performance of the function depends on the problem, on the chosen parameters, and on the hardware used, but in most cases the function can compute a nearly-optimal exact design for a problem with a ten thousands design points within seconds of computing time. Because this is only a heuristic, we advise the user to verify the quality of the resulting design by comparing it to the result of an alternative method (such as `od_RC`).

Value

Output is the list with components:

`call`	the call of the function
`w.best`	the best exact design found by the algorithm
`supp`	the indices of the support of w.best
`w.supp`	the weights of w.best on the support
`M.best`	the information matrix of w.best
`Phi.best`	the criterion value of w.best
`eff.best`	a lower bound on the eff of w.best with respect to `Phi.app`
`n.rest`	number of restarts performed
`n.ex`	the total number of exchanges performed
`t.act`	the actual time of the computation

Author(s)

Radoslav Harman, Lenka Filova

References

Atkinson AC, Donev AN, Tobias RD (2007): Optimum experimental designs, with SAS. Vol. 34. Oxford: Oxford University Press.

See Also

`od_RC`, `od_AQUA`, `od_MISOCP`

Examples

```
## Not run:
# Compute a D-efficient exact design of size 27 on a unit square
# for the full quadratic model with 2 discretized factors

form.q <- ~x1 + x2 + I(x1^2) + I(x2^2) + I(x1*x2)
Fx <- Fx_cube(form.q, n.levels = c(101, 101))
w <- od_KL(Fx, 13, t.max = 8)$w.best
```

```
od_plot(Fx, w, Fx[, 2:3])
od_print(Fx, w)


# Compute an I-efficient exact design of size 100 without replications
# on a discretized L1 ball for the full quadratic model with 3 factors

form.q <- ~x1 + x2 + x3 + I(x1^2) + I(x2^2) + I(x3^2) + I(x1*x2) + I(x1*x3) + I(x2*x3)
Fx <- Fx_cube(form.q, n.levels = c(21, 21, 21))
remove <- (1:nrow(Fx))[apply(abs(Fx[, 2:4]), 1, sum) > 1 + 1e-9]
Fx <- Fx[-remove, ]
w <- od_KL(Fx, 100, bin = TRUE, crit = "I", t.max = 3)$w.best
od_plot(Fx, w, Fx[, 2:4])

# Compute a D-efficient exact design of size 20 on a 4D cube
# for the full quadratic model with 4 continuous factors

# We can begin with a crude discretization and compute
# an initial (already good) exact design using the KL algorithm

form.q <- ~x1 + x2 + x3 + x4 + I(x1^2) + I(x2^2) + I(x3^2) + I(x4^2) +
           I(x1*x2) + I(x1*x3) + I(x1*x4) + I(x2*x3) + I(x2*x4) + I(x3*x4)
Fx <- Fx_cube(form.q, n.levels = rep(11, 4))
w <- od_KL(Fx, 20, t.max = 10)$w.best
od_print(Fx, w)$design[, c(2:5, 16)]
print(paste("D-criterion value:", optcrit(Fx, w)))

# Now we can fine-tune the positions of the design points
# using any general-purpose continuous optimization method

F <- Fx[rep(1:nrow(Fx), w), ]
f <- function(x) {c(1, x, x^2, x[1]*x[2], x[1]*x[3], x[1]*x[4],
                               x[2]*x[3], x[2]*x[4], x[3]*x[4])}
obj <- function(x, M.red) {-log(det(M.red + f(x) %*% t(f(x))))}
for (i in 1:10)
  for (j in 1:20) {
    F[j, ] <- f(optim(F[j, 2:5], obj, M.red = t(F[-j, ]) %*% F[-j, ],
              method = "L-BFGS-B", lower = rep(-1, 3), upper = rep(1, 3))$par)
  }

tune <- od_pool(round(F, 4), rep(1, 20))
Fx.tune <- tune$X.unique; w.tune <- tune$val.pooled
od_print(Fx.tune, w.tune)$design[, c(2:5, 16)]
print(paste("D-criterion value:", optcrit(Fx.tune, w.tune)))

## End(Not run)
```

---

od_MISOCP *Optimal exact design using mixed integer second-order cone programming*

Description

Computes an optimal or nearly-optimal approximate or exact experimental design using mixed integer second-order cone programming.

Usage

```
od_MISOCP(Fx, b1=NULL, A1=NULL, b2=NULL, A2=NULL, b3=NULL, A3=NULL, w0=NULL,
          bin=FALSE, type="exact", crit="D", h=NULL, gap=NULL,
          t.max=120, echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` (where `m>=2`, `m<=n`) matrix containing all candidate regressors (as rows), i.e., `n` is the number of candidate design points, and `m` (where `m>=2`) is the number of parameters
`b1`, `A1`, `b2`, `A2`, `b3`, `A3`	the real vectors and matrices that define the constraints on permissible designs `w` as follows: `A1 %*% w <= b1`, `A2 %*% w >= b2`, `A3 %*% w == b3`. Each of the arguments can be `NULL`, but at least one of `b1`, `b2`, `b3` must be non-`NULL`. If some `bi` is non-`NULL` and `Ai` is `NULL`, then `Ai` is set to be `matrix(1, nrow =1, ncol = n)`.
`w0`	a non-negative vector of length `n` representing the design to be augmented (i.e., the function adds the constraint `w >= w0` for permissible designs `w`). This argument can also be `NULL`; in that case, `w0` is set to the vector of zeros.
`bin`	Should each design point be used at most once?
`type`	the type of the design. Permissible values are `"approximate"` and `"exact"`.
`crit`	the optimality criterion. Possible values are `"D"`, `"A"`, `"I"`, `"C"`, `"c"`.
`h`	a non-zero vector of length `m` corresponding to the coefficients of the linear parameter combination of interest. If `crit` is not `"C"` nor `"c"` then `h` is ignored. If `crit` is `"C"` or `"c"` and `h=NULL` then `h` is assumed to be `c(0,...,0,1)`.
`gap`	the gap for the MISOCP solver to stop the computation. If `NULL`, the default gap is used. Setting `gap=0` and `t.max=Inf` will ultimately provide the optimal exact design, but the computation may be extremely time consuming.
`t.max`	the time limit for the computation.
`echo`	Print the call of the function?

Details

At least one of `b1`, `b2`, `b3` must be non-`NULL`. If `bi` is non-`NULL` and `Ai` is `NULL` for some `i` then `Ai` is set to be the vector of ones. If `bi` is `NULL` for some `i` then `Ai` is ignored.

Value

A list with the following components:

`call`	the call of the function
`w.best`	the permissible design found, or `NULL`. The value `NULL` indicates a failed computation

supp	the indices of the support of w.best
w.supp	the weights of w.best on the support
M.best	the information matrix of w.best or NULL if w.best is NULL
Phi.best	the value of the criterion of optimality of the design w.best. If w.best has a singular information matrix or if the computation fails, the value of Phi.best is 0
status	the status variable of the gurobi optimization procedure; see the gurobi solver documentation for details
t.act	the actual time of the computation

Author(s)

Radoslav Harman, Lenka Filova

References

Sagnol G, Harman R (2015): Computing exact D-optimal designs by mixed integer second order cone programming. The Annals of Statistics, Volume 43, Number 5, pp. 2198-2224.

See Also

od_KL, od_RC, od_AQUA

Examples

```
## Not run:
# Compute an A-optimal block size two design
# for 6 treatments and 9 blocks

Fx <- Fx_blocks(6)
w <- od_MISOCP(Fx, b3 = 9, crit = "A", bin = TRUE)$w.best
des <- combn(6, 2)[, as.logical(w)]
print(des)

library(igraph)
grp <- graph_(t(des), from_edgelist(directed = FALSE))
plot(grp, layout=layout_with_graphopt)

# Compute a symmetrized D-optimal approximate design
# for the full quadratic model on a square grid
# with uniform marginal constraints

Fx <- Fx_cube(~x1 + x2 + I(x1^2) + I(x2^2) + I(x1*x2), n.levels = c(21, 21))
A3 <- matrix(0, nrow = 21, ncol = 21^2)
for(i in 1:21) A3[i, (i*21 - 20):(i*21)] <- 1
w <- od_MISOCP(Fx, b3 = rep(1, 21), A3 = A3, crit = "D", type = "approximate")$w.best
w.sym <- od_SYM(Fx, w, b3 = rep(1, 21), A3 = A3)$w.sym
od_plot(Fx, w.sym, Fx[, 2:3], dd.size = 2)

## End(Not run)
```

od_PIN *Efficient saturated exact design*

Description

Use a fast greedy method to compute an efficient saturated subset (saturated exact design).

Usage

```
od_PIN(Fx, alg.PIN="KYM", echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` (where `m>=2`, `m<=n`) matrix containing all candidate regressors (as rows), i.e., `n` is the number of candidate design points, and `m` is the number of parameters.
`alg.PIN`	the method used (either `"KYM"` or `"GKM"`). KYM is randomized, faster but provides somewhat less efficient subsets/designs. GKM is deterministic, slower, but tends to give more efficient subsets/designs.
`echo`	Print the call of the function?

Details

The function is developed with the criterion of D-optimality in mind, but it also gives reasonably efficient subset/designs with respect to other criteria. The main purpose of `od_PIN` is to initialize algorithms for computing optimal approximate and exact designs. It can also be used to verify whether a model, represented by a matrix `Fx` of candidate regressors, permits a non-singular design.

Value

Output is the list with components:

`call`	the call of the function
`w.pin`	the resulting exact design
`supp`	the indices of the support of w.pin
`M.pin`	the information matrix of w.pin
`Phi.D`	the D-criterion value of w.pin
`t.act`	the actual time of the computation

Author(s)

Radoslav Harman, Samuel Rosa, Lenka Filova

References

Harman R, Rosa S (2019): On greedy heuristics for computing D-efficient saturated subsets, (submitted to Operations Research Letters), https://arxiv.org/abs/1905.07647

Examples

```
# Compute a saturated subset of a random Fx
Fx <- matrix(rnorm(10000), ncol = 5)
w.KYM <- od_PIN(Fx)$w.pin
w.GKM <- od_PIN(Fx, alg.PIN = "GKM")$w.pin
w.REX <- 5*od_REX(Fx)$w.best
optcrit(Fx, w.KYM)
optcrit(Fx, w.GKM)
optcrit(Fx, w.REX)
```

`od_plot` *Visualization of a design*

Description

Visualizes selected aspects of an experimental design

Usage

```
od_plot(Fx, w, X=NULL, w.pool=c("sum", "0"), w.color="darkblue",
        w.size=1, w.pch=16, w.cex=0.8, w.lim=0.01, crit="D",
        h=NULL, dd.pool=c("max", "mean"), dd.color="orange",
        dd.size=1.5, dd.pch=15, asp = NA, main.lab="",
        y.lab="", return.pools=FALSE, echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` (where `m>=2`, `m<=n`) matrix containing all candidate regressors (as rows), i.e., `n` is the number of candidate design points, and `m` is the number of parameters
`w`	the vector of non-negative real numbers of length `n` representing the design
`X`	an `n` times `k` matrix of coordinates of design points
`w.pool`	a vector with components from the set `"sum"`, `"min"`, `"max"`, `"mean"`, `"median"`, `"0"` that determines various "pools" of the design weights along the projections defined by the coordinates provided by `X`
`w.color`	the color string for plotting the design weight (from the standard list of R colors)
`w.size`	the size of the characters/balls that represent the non-zero design weights
`w.pch`	the numerical code of the characters used to plot the non-zero design weights
`w.cex`	the size of the text labels representing the magnitudes of the design weights
`w.lim`	a threshold fraction of the total design weight to plot the labels
`crit`	the optimality criterion. Possible values are `"D"`, `"A"`, `"I"`, `"C"`
`h`	a non-zero vector of length `m` corresponding to the coefficients of the linear parameter combination of interest. If `crit` is not `"C"` then `h` is ignored. If `crit` is `"C"` and `h=NULL` then `h` is assumed to be `c(0,...,0,1)`

`dd.pool`	a vector with components from the set `"sum"`, `"min"`, `"max"`, `"mean"`, `"median"`, `"0"` that determines various "pools" of the directional derivatives along the projections defined by the coordinates provided by `X`
`dd.color`	the color string for plotting the directional derivatives (from the standard list of R colors)
`dd.size`	the size of the characters that represent the directional derivatives
`dd.pch`	the numerical code of the character used to plot the directional derivatives
`asp`	the aspect ration of the axes
`main.lab`	the main title of the plot
`y.lab`	the label of the y axis
`return.pools`	Return the pooled values?
`echo`	Print the call of the function?

Details

This function performs a simple visualization of some aspects of an experimental design. It visualizes (the selected pools of) the design weights and (the selected pools of) the directional derivative. The type of graph depends on the number of columns in `X`.

Value

If `return.pool` is set to `TRUE`, the procedure returns the data used to plot the figure. The data can be used to plot a different figure according to the user's needs.

Note

The labels of the axes correspond to the column names of `X`. For a large `unique(Fx)`, rendering the plot can take a considerable time. Note also that using RStudio, it may be a good idea to open an external graphical window (using the command `windows()`) before running `od_plot`.

Author(s)

Radoslav Harman, Lenka Filova

See Also

`od_pool`, `od_print`

Examples

```
# Compute a D-optimal approximate design
# for the 2nd degree Fourier regression on a partial circle
# Use several types of graphs to visualize the design

Fx <- Fx_cube(~I(cos(x1)) + I(sin(x1)) + I(cos(2*x1)) + I(sin(2*x1)),
              lower = -2*pi/3, upper = 2*pi/3, n.levels = 121)
w <- od_REX(Fx)$w.best
par(mfrow = c(2, 2))
```

```
od_plot(Fx, w, X = seq(-2*pi/3, 2*pi/3, length = 121), main = "Plot 1")
od_plot(Fx, w, X = Fx[, 2:3], asp = 1, main = "Plot 2")
od_plot(Fx, w, X = Fx[, c(2,5)], asp = 1, main = "Plot 3")
od_plot(Fx, w, X = Fx[, c(3,4)], asp = 1, main = "Plot 4")
par(mfrow = c(1, 1))

## Not run:
# Compute an I-efficient exact design of size 20 without replications
# for the Scheffe mixture model
# Use several types of graphs to visualize the design

Fx <- Fx_simplex(~x1 + x2 + x3 + I(x1*x2) + I(x1*x3) + I(x2*x3) - 1, 21)
w <- od_AQUA(Fx, b3=20, bin=TRUE, crit="I")$w.best
X <- Fx[, 1:2]
colnames(X) <- c("", "")
od_plot(Fx, w, X, asp = 1, main = "Plot 1")
od_plot(Fx, w, Fx[, 1:3], main = "Plot 2")

# Compute a symmetrized D-optimal approximate design
# for the full quadratic model with 4 factors
# Use several types of graphs to visualize the design

form.q <- ~x1 + x2 + x3 + x4 + I(x1^2) + I(x2^2) + I(x3^2) + I(x4^2) +
           I(x1*x2) + I(x1*x3) + I(x1*x4) + I(x2*x3) + I(x2*x4) + I(x3*x4)
Fx <- Fx_cube(form.q, n.levels = rep(11, 4))
w <- od_REX(Fx)$w.best
od_plot(Fx, w, Fx[, 2:3], dd.size=3)
od_plot(Fx, w, Fx[, 2:4], w.lim=Inf)


# A more complex example:

# Compute the D-optimal 17 point exact design
# for the spring-balance weighing model with 4 items

Fx <- Fx_cube(~x1 + x2 + x3 + x4 - 1, lower = rep(0, 4))
w <- od_KL(Fx, 17, t.max = 5)$w.best
od_print(Fx, w)$design
U <- eigen(diag(4) - 0.25 * rep(1, 4))

# A 2D visualization

X <- Fx[, 1:4]
X[, 2] <- -2*X[, 2]
colnames(X) <- c("V", "Number of items on the pan")
od_plot(Fx, w+0.001, X)
for(i in 1:16) for(j in 1:16)
  if(sum(abs(Fx[i,1:4]-Fx[j,1:4]))==1)
    lines(X[c(i,j),1], X[c(i,j),2])

# A 3D visualization

X <- Fx[, 1:4]
```

```
colnames(X) <- c("V1", "V2", "V3")
od_plot(Fx, w+0.001, X)
for(i in 1:16) for(j in 1:16)
  if(sum(abs(Fx[i, 1:4] - Fx[j, 1:4])) == 1)
    rgl::lines3d(X[c(i, j), 1], X[c(i, j), 2], X[c(i, j), 3])

## End(Not run)
```

`od_pool` *Pool of a vector*

Description

A function `pool.fun` is applied to all the elements of a vector `val` that appear within the groups formed by identical rows of a matrix `X`.

Usage

```
od_pool(X, val=NULL, pool.fun="sum", echo=TRUE)
```

Arguments

`X`	the `n` times `k` matrix of real values.
`val`	a real vector of length `n`.
`pool.fun`	a string denoting the function to be applied to the subgroups of elements of `val` corresponding to the identical rows of `X`. Possible values are `"sum"`, `"min"`, `"max"`, `"mean"`, `"median"` and `"0"`.
`echo`	Print the call of the function?

Details

This function is useful for plotting (and understanding) of designs of experiments with more factors than the dimension of the plot.

Value

A list with components:

`call`	the call of the function
`X.unique`	the matrix of unique rows of `X`
`val.pooled`	the vector of the length `nrows(X.unique)` containing the values of `val` pooled using `pool.fun`

Note

The function performs a non-trivial operation only if some of the rows of `X` are identical.

Author(s)

Radoslav Harman, Lenka Filova

See Also

od_plot, od_print

Examples

```
v1 <- c(1, 2, 3); v2 <- c(2, 4, 6); v3 <- c(2, 5, 3)
X <- rbind(v1, v1, v1, v1, v2, v3, v2, v3, v3)
val <- c(1, 2, 7, 9, 5, 8, 4, 3, 6)
od_pool(X, val, "sum")

# The result $val.pooled is a vector with components:
# 19 (=1+2+7+9) because the first 4 rows of X are identical
# 9 (=5+4) because the 5th and the 7th rows of X are identical
# 17 (=8+3+6) because the 6th, the 8th and the 9th rows of X are identical
```

---

od_print *Compact information about a design*

Description

Prints various characteristics of an experimental design

Usage

```
od_print(Fx, w, X=NULL, h=NULL, echo=TRUE)
```

Arguments

Fx	the n times m (where m>=2, m<=n) matrix containing all candidate regressors (as rows), i.e., n is the number of candidate design points, and m is the number of parameters
w	the vector of non-negative real numbers of length n representing the design
X	an n times k matrix of coordinates of design points
h	a non-zero vector of length m corresponding to the coefficients of the linear parameter combination of interest. If crit is not "C" then h is ignored. If crit is "C" and h=NULL then h is assumed to be c(0,...,0,1)
echo	Print the call of the function?

Value

Output is a list with components

`call`	the call of the function
`design`	a matrix with the rows of `Fx` corresponding to non-zero design weights and the non-weights themselves
`M`	the information matrix of `w`
`eigenvalues`	the eigenvalues of `M`
`D.value`	the value of the D-optimality criterion for `w`
`A.value`	the value of the A-optimality criterion for `w`
`I.value`	the value of the I-optimality criterion for `w`
`C.value`	the value of the C-optimality criterion for `w`
`c.value`	the value of the c-optimality criterion for `w`

Author(s)

Radoslav Harman, Lenka Filova

See Also

`od_plot`, `od_pool`

Examples

```
Fx <- Fx_cube(~x1 + I(x1^2), n.levels = 11)
w <- 1:11/sum(1:11)
od_print(Fx, w, Fx[, 2])
```

---

od_PUK *Efficient rounding of an approximate design*

Description

Compute the classical efficient rounding of a non-normalized approximate design `w` such that the resulting exact design has size `floor(sum(w))`.

Usage

```
od_PUK(Fx, w, echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` (where `m>=2`, `m<=n`) matrix containing all candidate regressors (as rows), i.e., `n` is the number of candidate design points, and `m` (where `m>=2`) is the number of parameters
`w`	the vector of non-negative real numbers of length `n` representing the design
`echo`	Print the call of the function?

Value

The rounded version of w

Author(s)

Radoslav Harman and Samuel Rosa

References

Pukelsheim F, Rieder S (1992) Efficient rounding of approximate designs. Biometrika, 79(4), 763–770.

Examples

```
# Compute a D-optimal approximate design
# Round it using the efficient rounding to various sizes
# Visualize the designs

Fx <- Fx_cube(~x1 + I(x1^2) + I(x1^3), lower = 0, upper = 1, n.levels = 11)
w.app <- od_REX(Fx)$w.best
Phi.app <- optcrit(Fx, w.app)

w.ex10 <- od_PUK(Fx, 10*w.app)$w.round
w.ex20 <- od_PUK(Fx, 20*w.app)$w.round
w.ex30 <- od_PUK(Fx, 30*w.app)$w.round

par(mfrow = c(2, 2))
od_plot(Fx, w.app, main.lab = "Approximate")
od_plot(Fx, w.ex10, main.lab = paste("N=10, Eff:", round(optcrit(Fx, w.ex10)/Phi.app/10, 4)))
od_plot(Fx, w.ex20, main.lab = paste("N=20, Eff:", round(optcrit(Fx, w.ex20)/Phi.app/20, 4)))
od_plot(Fx, w.ex30, main.lab = paste("N=30, Eff:", round(optcrit(Fx, w.ex30)/Phi.app/30, 4)))
par(mfrow = c(1, 1))
```

---

od_RC *Efficient exact design using the RC heuristic*

Description

Computes an efficient exact design under multiple linear resource constraints using the RC heuristic.

Usage

```
od_RC(Fx, b, A = NULL, w0 = NULL, bin = FALSE, Phi.app = NULL, crit = "D",
      h=NULL, w1 = NULL, rest.max = Inf, t.max = 120,
      echo = TRUE, track=TRUE)
```

Arguments

Fx	the n times m (where m>=2, m<=n) matrix containing all candidate regressors (as rows), i.e., n is the number of candidate design points, and m is the number of parameters.
b, A	the vector of length k with positive real components and the k times n matrix of non-negative reals numbers. Each column of A must have at least one strictly positive element. The linear constraints A%*%w<=b, w0<=w define the set of permissible designs w (where w0 is a described below.) The argument A can also be NULL; in that case b must be a positive number and A is set to the 1 times n matrix of ones.
w0	a non-negative vector of length n representing the design to be augmented (i.e., the function adds the constraint w >= w0 for permissible designs w). This argument can also be NULL; in that case, w0 is set to the vector of zeros.
bin	Should each design point be used at most once?
Phi.app	the optimal value of the corresponding approximate (relaxed) problem. If Phi.app = NULL, a very conservative upper bound on Phi.app is pre-computed.
crit	the optimality criterion. Possible values are "D", "A", "I", "C".
h	a non-zero vector of length m corresponding to the coefficients of the linear parameter combination of interest. If crit is not "C" then h is ignored. If crit is "C" and h=NULL then h is assumed to be c(0,...,0,1).
w1	an n times 1 nonnegative vector that represents the initial design. The design w1 must satisfy w0<=w1 and A*w1<=b. The argument w1 can also be NULL; in that case the procedure sets w1 to be w0.
rest.max	the maximum allowed number of restarts of the method.
t.max	the time limit for the computation.
echo	Print the call of the function?
track	Trace the computation?

Details

This is an implementation of the algorithm proposed by Harman et al. (2016); see the references. The inequalities A%*%w<=b, w0<=w with the specific properties mentioned above, form the so-called resource constraints. They encompass many practical restrictions on the design, and lead to a bounded set of feasible solutions.

The information matrix of w1 should preferably have the reciprocal condition number of at least 1e-5. Note that the floor of an optimal approximate design (computed for instance using od_MISOCP) is often a good initial design. Alternatively, the initial design can be the result of another optimal design procedure, such as od_AQUA. Even if no initial design is provided, the model should be non-singular in the sense that there *exists* an exact design w with a well conditioned information matrix, satisfying all constraints. If this requirement is not satisfied, the computation may fail, or it may produce a deficient design.

The procedure always returns a permissible design, but in some cases, especially if t.max is too small, the resulting design can be inefficient. The performance depends on the problem and on the hardware used, but in most cases the function can compute a nearly-optimal exact design for a problem with a few hundreds design points and tens of constraints within minutes of computing time. Because this is a heuristic method, we advise the user to verify the quality of the resulting design by comparing it to the result of an alternative method (such as od_AQUA and od_MISOCP) and/or by computing its efficiency relative to the corresponding optimal approximate design.

In the very special (but frequently used) case of the single constraint on the experimental size, it is generally more efficient to use the function od_KL.

Value

A list with the following components:

call	The call of the function.
w.best	The resulting exact design.
supp	The indices of the support of w.best.
w.supp	The weights of w.best on the support.
M.best	The information matrix of w.best.
Phi.best	The criterion value of w.best.
eff.best	A lower bound on the efficiency of w.best with respect to the optimal approximate design.
n.rest	The number of restarts performed.
t.act	The actual time of the computation.

Author(s)

Radoslav Harman, Alena Bachrata, Lenka Filova

References

Harman R, Bachrata A, Filova L (2016): Heuristic construction of exact experimental designs under multiple resource constraints, Applied Stochastic Models in Business and Industry, Volume 32, pp. 3-17

See Also

od_AQUA, od_MISOCP, od_KL

Examples

```
## Not run:
# A D-efficient exact design for a quadratic model with 2 factors
# constrained by the total time and the total cost of the experiment.
# The cost of a single trial in (x1, x2) is 10 + x1 + 2*x2
# The limit on the total cost is 1000
# (we do not know the number of trials in advance)

form.quad <- ~x1 + x2 + I(x1^2) + I(x2^2) + I(x1 * x2)
Fx <- Fx_cube(form.quad, lower = c(0, 0), upper = c(10, 10), n.levels = c(11, 11))
n <- nrow(Fx); A <- matrix(0, nrow = 1, ncol = n)
```

```
for(i in 1:n) A[1, i] <- 5 + Fx[i, 2] + 2*Fx[i, 3]
w <- od_RC(Fx, 1000, A, bin = TRUE, t.max = 8)$w.best
od_plot(Fx, w, Fx[, 2:3], dd.size = 3)

## End(Not run)
```

---

od_REX *Optimal approximate size-constrained design*

Description

Computes an optimal approximate design under the standard (size) constraint using one of three methods.

Usage

```
od_REX(Fx, crit="D", h=NULL, w1=NULL, alg.AA="REX",
       eff=0.999999, it.max=Inf, t.max=60, echo=TRUE, track=TRUE)
```

Arguments

Fx	the n times m (where m>=2, m<=n) matrix containing all candidate regressors (as rows), i.e., n is the number of candidate design points, and m (where m>=2) is the number of parameters
crit	the optimality criterion. Possible values are "D", "A", "I", "C" and "c".
h	a non-zero vector of length m corresponding to the coefficients of the linear parameter combination of interest. If crit is not "C" nor "c" then h is ignored. If crit is "C" or "c" and h=NULL then h is assumed to be c(0,...,0,1).
w1	a real vector of length n with non-negative components, representing the initial design. The information matrix of w1 must be nonsingular. It should have a small support (e.g., m) provided that alg.AA="REX" and it should have the full support of length n provided that alg.AA="MUL". The argument w1 can also be NULL; in that case a non-singular initial design is generated by od_PIN.
alg.AA	the computational method to be applied; possible choices are "REX", "MUL", and "VDM". For crit="c", argument alg.AA is ignored and the function uses the rapid linear programming approach (see the references).
eff	the efficiency for the stopping rule in the interval (0,1). The algorithm will be stopped if a lower bound on the efficiency of the actual design is equal or greater than eff.
it.max	the maximum allowed number of iterations of the method.
t.max	the time limit for the computation.
echo	Print the call of the function?
track	Trace the computation?

Details

The function implements three algorithms for the computation of optimal approximate designs with respect to the criteria of D-, A-, I-, and C-optimality: the standard vertex-direction method ("VDM"), the standard multiplicative method ("MUL"), and the randomized exchange method ("REX"). The first two methods are classical and the method REX is proposed in Harman et al (2019).

For the specific criterion of c-optimality, the function runs the LP-based method from Harman and Jurik (2008).

The information matrix of `w1` should have the reciprocal condition number of at least `1e-5`. Even if no initial design is provided, the model should be non-singular in the sense that there *exists* an approximate design `w` with an information matrix that is not severely ill-conditioned. If this requirement is not satisfied, the computation may fail, or it may produce a deficient design. If `w1=NULL`, the initial design is computed with `od_PIN`.

Since the result is a normalized approximate design, it only gives recommended *proportions* of trials in individual design points. To convert it to an optimal approximate design of size N (under the standard, i.e., size, constraints), just multiply `w.best` by N. To obtain an efficient exact design with N trials, `w.best` must be multiplied by N and the result should be properly rounded to the neighboring integers by, for example, `od_PUK`. However, it is often more efficient to directly use `od_KL` to obtain an efficient exact design of size N.

Value

A list with the following components:

`call`	The call of the function.
`w.best`	The resulting exact design.
`supp`	The indices of the support of `w.best`.
`w.supp`	The weights of `w.best` on the support.
`M.best`	The information matrix of `w.best`.
`Phi.best`	The criterion value of `w.best`.
`eff.best`	A lower bound on the efficiency of `w.best` with respect to the optimal approximate design.
`n.iter`	The number of iterations performed.
`t.act`	The actual time of the computation.

Note

REX is a randomized algorithm, therefore the resulting designs may differ from run to run. In case that the optimal design is unique, the fluctuation of the results are minor and can be made negligible by setting `eff` to a value very close to 1.

If the optimal design is not unique, REX provides a selection of significantly different optimal designs by running it multiple times, which can help choosing the best optimal design based on a secondary criterion.

A unique and often "symmetric" optimal design (within the possibly infinite set of optimal designs) can be computed by `od_SYM`.

Note also that the optimal *information matrix* is always unique for criteria of D-, A-, I- and C-optimality, even if the optimal design is not unique.

While the default choice is `alg.AA="REX"`, our numerical experience suggests that `alg.AA="MUL"` may be a better choice in problems with a relatively small `n` and a relatively large `m`.

The method VDM is included mostly for teaching purposes; it is only rarely competitive with REX or MUL. Its advantage is that it tends to be easy to generalize to more complex optimum design problems.

Author(s)

Radoslav Harman, Lenka Filova

References

Harman R, Jurik T (2008). Computing c-optimal experimental designs using the simplex method of linear programming. Computational Statistics and Data Analysis 53 (2008) 247-254

Harman R, Filova L, Richtarik P (2019). A randomized exchange algorithm for computing optimal approximate designs of experiments. Journal of the American Statistical Association, 1-30.

See Also

`od_KL`, `od_RC`, `od_MISOCP`, `od_AQUA`

Examples

```
## Not run:
# Note: Many small examples of od_REX are in other help files.

# Compute an essentially perfect D-optimal design
# on 10 million design points in a few seconds
n <- 10000000; m <- 5
Fx <- matrix(rnorm(n*m), ncol = m)
w <- od_REX(Fx, t.max = 10)$w.best
Fx.small <- od_DEL(Fx, w)$Fx.keep
w <- od_REX(Fx.small, eff = 0.999999999)$w.best
od_plot(Fx.small, w, Fx.small[, 1:2], dd.pch = 16, dd.size = 0.35)

## End(Not run)
```

`od_SYM` *Symmetrization of an approximate design*

Description

Attempts to "symmetrize" an approximate design `w` by minimizing its norm while keeping its information matrix.

Usage

```
od_SYM(Fx, w, b1=NULL, A1=NULL, b2=NULL, A2=NULL, b3=NULL, A3=NULL, w0=NULL,
      crit="D", h=NULL, echo=TRUE)
```

Arguments

- `Fx` — the `n` times `m` (where `m>=2`, `m<=n`) matrix containing all candidate regressors (as rows), i.e., `n` is the number of candidate design points, and `m` is the number of parameters
- `w` — a non-negative vector of length `n` representing the design
- `b1, A1, b2, A2, b3, A3` — the real vectors and matrices that define the constraints on permissible designs `w` as follows: `A1 %*% w <= b1`, `A2 %*% w >= b2`, `A3 %*% w == b3`. Each of the arguments can be `NULL`, but at least one of `b1`, `b2`, `b3` must be non-`NULL`. If some `bi` is non-`NULL` and `Ai` is `NULL`, then `Ai` is set to be `matrix(1, nrow =1, ncol = n)`.
- `w0` — a non-negative vector of length `n` representing the design to be augmented (i.e., the function adds the constraint `w >= w0` for permissible designs `w`). This argument can also be `NULL`; in that case, `w0` is set to the vector of zeros.
- `crit` — the optimality criterion. Possible values are `"D"`, `"A"`, `"I"`, `"C"`, `"c"`.
- `h` — a non-zero vector of length `m` corresponding to the coefficients of the linear parameter combination of interest. If `crit` is not `"C"` nor `"c"` then `h` is ignored. If `crit` is `"C"` or `"c"` and `h=NULL` then `h` is assumed to be `c(0,...,0,1)`.
- `echo` — Print the call of the function?

Details

For some models, the optimum approximate design is not unique (although the optimum information matrix usually *is* unique). This function uses one optimal approximate design to produce an optimal approximate design with a minimum Euclidean norm, which is unique and usually more "symmetric".

Value

A list with the following components:

- `call` — The call of the function
- `w.sym` — The resulting "symmetrized" approximate design

Author(s)

Radoslav Harman, Lenka Filova

References

Harman R, Filova L, Richtarik P (2019). A randomized exchange algorithm for computing optimal approximate designs of experiments. Journal of the American Statistical Association, 1-30. (Subsection 5.1)

Examples

```
# Compute a D-optimal approximate design using the randomized method REX.
# Visualize both the design obtained by REX and its symmetrized version.

form.q <- ~x1 + x2 + x3 + I(x1^2) + I(x2^2) + I(x3^2) + I(x1*x2) + I(x1*x3) + I(x2*x3)
Fx <- Fx_cube(form.q, n.levels = c(5, 5, 5))
w.app <- od_REX(Fx)$w.best
od_plot(Fx, w.app, X=Fx[, 2:3])
w.app.sym <- od_SYM(Fx, w.app, b3 = 1)$w.sym
od_plot(Fx, w.app.sym, X=Fx[, 2:3])
```

---

optcrit — *Criterion value of a design*

Description

Computes the criterion value of a design `w` in the model determined by the matrix `Fx` of all regressors.

Usage

```
optcrit(Fx, w, crit="D", h=NULL, echo=TRUE)
```

Arguments

`Fx`	the `n` times `m` (where `m>=2`, `m<=n`) matrix containing all candidate regressors (as rows), i.e., `n` is the number of candidate design points, and `m` (where `m>=2`) is the number of parameters.
`w`	a non-negative vector of length `n` representing the design.
`crit`	the criterion; possible values are `"D"`, `"A"`, `"I"`, `"C"` and `"c"`.
`h`	a non-zero vector of length `m` corresponding to the coefficients of the linear parameter combination of interest. If `crit` is not `"C"` nor `"c"` then `h` is ignored. If `crit` is `"C"` or `"c"` and `h=NULL` then `h` is assumed to be `c(0,...,0,1)`.
`echo`	Print the call of the function?

Details

The package works with optimality criteria as information functions, i.e., the criteria are concave, positive homogeneous and upper semicontinuous on the set of all non-negative definite matrices. The criteria are normalized such that they assign the value of 1 to any design with information matrix equal to the identity matrix.

Value

A non-negative number corresponding to the criterion value.

Note

Since the criteria are positive homogeneous, the relative efficiency of two designs is just the ratio of their criterion values.

Author(s)

Radoslav Harman, Lenka Filova

See Also

infmat

Examples

```
# The Fx matrix for the spring balance weighing model with 6 weighed items.
Fx <- Fx_cube(~x1 + x2 + x3 + x4 + x5 + x6 - 1, lower = rep(0, 6), n.levels = rep(2, 6))

# Criteria of the design of size 15 that weighs each pair of items exactly once.
w2 <- rep(0, 64); w2[apply(Fx, 1, sum) == 2] <- 1
optcrit(Fx, w2, crit = "D")
optcrit(Fx, w2, crit = "A")
optcrit(Fx, w2, crit = "I")

# Criteria for the design of size 15 that weighs each quadruple of items exactly once.
w4 <- rep(0, 64); w4[apply(Fx, 1, sum) == 4] <- 1
optcrit(Fx, w4, crit = "D")
optcrit(Fx, w4, crit = "A")
optcrit(Fx, w4, crit = "I")
```

---

varfun *Vector of variances*

Description

Computes the vector of variances (sensitivities) for a given design w.

Usage

```
varfun(Fx, w, crit="D", h=NULL, echo=TRUE)
```

Arguments

Fx	the n times m matrix of candidate regressors (as rows), where n is the number of candidate design points and m (where m>=2, m<=n) is the number of parameters.
w	a non-negative vector of length n representing the design.
crit	the criterion; possible values are "D", "A", "I", "C" and "c".

h	a non-zero vector of length m corresponding to the coefficients of the linear parameter combination of interest. If crit is not "C" nor "c" then h is ignored. If crit is "C" or "c" and h=NULL then h is assumed to be c(0,...,0,1).
echo	Print the call of the function?

Details

For D-optimality, the i-th element of the vector of variances is the variance of the best linear unbiased estimator of the mean value of observations under the experimental conditions represented by the i-th design point (where the variance of the observational errors is assumed to be 1). There is a linear transformation relation of the vector of variances and the vector of directional derivatives for the criterion of D-optimality. See the reference paper at http://www.iam.fmph.uniba.sk/design/ for mathematical details.

Value

The vector of variances (sensitivities) for a given design w.

Note

The design w should have a non-singular information matrix.

Author(s)

Radoslav Harman, Lenka Filova

See Also

effbound, dirder

Examples

```
# The values of the variance function (for crit=D)
# of D-, I-, and C-optimal approximate design

Fx <- Fx_cube(~x1 + I(x1^2), n.levels = 21)
wD <- od_REX(Fx)$w.best
wI <- od_REX(Fx, crit="I")$w.best
wC <- od_REX(Fx, crit="C", h=c(1, 0, 0))$w.best
vD <- varfun(Fx, wD)
vI <- varfun(Fx, wI)
vC <- varfun(Fx, wC)
plot(Fx[, 2], rep(0, nrow(Fx)), ylim = c(0, max(vD, vI, vC)),
     type = "n", xlab = "x", ylab = "var", lwd = 2)
grid()
lines(Fx[, 2], vD, col = "red")
lines(Fx[, 2], vI, col = "blue")
lines(Fx[, 2], vC, col = "green")

# The D-optimal approximate design minimized the maximum
# of the var. function (it is "G-optimal").
```

```
# The I-optimal approximate design minimizes the integral of the var. function.
# The C-optimal design with h=f(0) makes the var. function small around 0.
```

Index