Package 'optedr'

January 20, 2022

Title Calculating Optimal and D-Augmented Designs

Version 1.0.1

Description Calculates D-, Ds-, A- and I-optimal designs for non-linear models, via an implementation of the cocktail algorithm (Yu, 2011, <doi:10.1007/s11222-010-9183-2>). Compares designs via their efficiency, and D-augments any design with a controlled efficiency. An efficient rounding function has been provided to transform approximate designs to exact designs.

License GPL-3

Encoding UTF-8

URL https://github.com/kezrael/optedr

BugReports https://github.com/kezrael/optedr/issues

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.0), mockery, markdown, tidyverse

Imports ggplot2, purrr, rlang, crayon, cli, magrittr, dplyr, nleqslv, shiny, DT, shinydashboard, shinyalert, plotly, hrbrthemes, shinyjs, orthopolynom

Config/testthat/edition 3

NeedsCompilation no

Author Carlos de la Calle-Arroyo [aut, cre] (<https://orcid.org/0000-0002-5099-888X>), Jesús López-Fidalgo [aut] (<https://orcid.org/0000-0001-7502-8188>), Licesio J. Rodríguez-Aragón [aut] (<https://orcid.org/0000-0003-4970-3877>)

Maintainer Carlos de la Calle-Arroyo <carlos.calle.arroyo@gmail.com>

Repository CRAN

Date/Publication 2022-01-20 09:52:42 UTC

R topics documented:

add_design *Add two designs*

Description

Add two designs

Usage

```
add_design(design_1, design_2, alpha)
```

Arguments

design_1	A dataframe with 'Point' and 'Weight' as column that represent the first design to add
design_2	A dataframe with 'Point' and 'Weight' as column that represent the second design to add
alpha	Weight of the first design

Value

A design as a dataframe with the weighted addition of the two designs

add_points *Update design given crosspoints and alpha*

Description

Given a set of points, a weight and the design, the function adds these points to the new design with uniform weight, and combined weight alpha

Usage

```
add_points(points, alpha, design)
```

Arguments

points	Points to be added to the design
alpha	Combined weight of the new points to be added to the design
design	A design as a dataframe with "Point" and "Weight" columns

Value

A design as a dataframe with "Point" and "Weight" columns that is the addition of the design and the new points

augment_design *Augment Design*

Description

D-Augments a design. The user gives an initial design for which he would like to add points and specifies the weight of the new points. Then he is prompted to choose a minimum efficiency. After that, the candidate points region is calculated and the user can choose the points and weights to add.

Usage

```
augment_design(
  init_design,
  alpha,
  model,
  parameters,
  par_values,
  design_space,
  calc_optimal_design,
  weight_fun = function(x) 1
)
```

Arguments

init_design	A dataframe with "Point" and "Weight" columns that represents the initial design to augment
alpha	Combined weight of the new points
model	Formula that represent the model with x as the unknown
parameters	Character vector with the unknown parameters of the model to estimate
par_values	Numeric vector with the initial values of the unknown parameters
design_space	Numeric vector with the extremes of the space of the design
calc_optimal_design	Boolean parameter, if TRUE, the optimal design is calculated and efficiencies of the initial and augmented design are given
weight_fun	Optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response

Value

A dataframe that represents the D-augmented design

Examples

```
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
augment_design(init_des, 0.25, y ~ 10^(a-b/(c+x)), c("a","b","c"),
  c(8.07131,  1730.63, 233.426), c(1, 100), TRUE)
augment_design(init_des, 0.25, y ~ 10^(a-b/(c+x)), c("a","b","c"),
  c(8.07131,  1730.63, 233.426), c(1, 100), FALSE)
```

`check_inputs` *Check Inputs*

Description

Function to check that the inputs given to the function `opt_des` are correct. If not, throws the correspondent error message.

Usage

```
check_inputs(
  Criterion,
  model,
  parameters,
  par_values,
  design_space,
  init_design,
  join_thresh,
  delete_thresh,
  delta,
  tol,
  tol2,
  par_int,
  matB,
  reg_int,
  desired_output,
  weight_fun
)
```

Arguments

`Criterion`	character with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality'
`model`	formula describing the model to calculate the optimal design. Must use x as the variable.
`parameters`	character vector with the parameters of the models, as written in the `formula`.

par_values	numeric vector with the parameters nominal values, in the same order as given in parameters.
design_space	numeric vector of length 2, first component with the minimum of the space of the design and second component with the maximum.
init_design	optimal dataframe with the initial design for the algorithm. A dataframe with two columns: • Point contains the support points of the design. • Weight contains the corresponding weights of the Points.
join_thresh	optional numeric value that states how close, in real units, two points must be in order to be joined together by the join heuristic.
delete_thresh	optional numeric value with the minimum weight, over 1 total, that a point needs to have in order to not be deleted from the design.
delta	optional numeric value in (0, 1), parameter of the algorithm.
tol	optional numeric value for the convergence of the weight optimizing algorithm.
tol2	optional numeric value for the stop criterion: difference between maximum of sensitivity function and optimality criterion.
par_int	optional numeric vector with the index of the parameters of interest.
matB	optional matrix of dimensions k x k, integral of the information matrix of the model over the interest region.
reg_int	optional numeric vector of two components with the bounds of the interest region for I-Optimality.
desired_output	not functional yet: decide which kind of output you want.
weight_fun	optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response

---

crit *Master function for the criterion function*

Description

Depending on the Criterion input, the function returns the output of the corresponding criterion function given the information matrix.

Usage

```
crit(Criterion, M, k = 0, par_int = c(1), matB = NA)
```

Arguments

`Criterion`	Character with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality'
`M`	Information matrix for which the criterion value wants to be calculated.
`k`	Numeric number of parameters of the model. Taken from the number of rows of the matrix if omitted.
`par_int`	Numeric vector with the index of the parameters of interest of the model. Only for "Ds-Optimality".
`matB`	Matrix of the integral of the information matrix over the interest region. Only for "I-Optimality".

Value

Numeric value of the optimality criterion for the information matrix.

`crosspoints` *Calculate crosspoints*

Description

Given the parameters for D-augmenting a design, this function calculates the crosspoints in the sensitivity function that delimit the candidate points region

Usage

```
crosspoints(deff, alpha, k, sens, gridlength, tol, xmin, xmax)
```

Arguments

`deff`	Minimum efficiency chosen by the user
`alpha`	Combined weight of the new points
`k`	Number of unknown parameters of the model
`sens`	Sensitivity function of the design for the model
`gridlength`	Number of points in the grid to find the crosspoints
`tol`	Tolerance that establishes how close two points close to one another are considered the same
`xmin`	Minimum of the space of the design
`xmax`	Maximum of the space of the design

Value

A numeric vector of crosspoints that define the candidate points region

dcrit *Criterion function for D-Optimality*

Description

Calculates the value of the D-Optimality criterion, which follows the expression:

$$\phi_D = \frac{1}{|M|}^{1/k}$$

Usage

```
dcrit(M, k)
```

Arguments

M	Information matrix for which the criterion value wants to be calculated.
k	Numeric number of parameters of the model. Taken from the number of rows of the matrix if omitted.

Value

numeric value of the D-optimality criterion for the information matrix.

delete_points *Remove low weight points*

Description

Removes the points of a design with a weight lower than a threshold, `delta`, and distributes that weights proportionally to the rest of the points.

Usage

```
delete_points(design, delta)
```

Arguments

design	The design from which to remove points as a dataframe with two columns: • `Point` contains the support points of the design. • `Weight` contains the corresponding weights of the `Points`.
delta	The threshold from which the points with such a weight or lower will be removed.

Value

The design without the removed points.

delta_bound *Calculate efficiency bounds*

Description

Given the weight of new points, number of parameters and range of the sensitivity function, calculates the range of possible minimum efficiency for the D-augmented design.

Usage

```
delta_bound(alpha, k, sens_min, sens_max = Inf)
```

Arguments

alpha	Combined weight of the new points to add
k	Number of parameters of the model
sens_min	Minimum value of the sensitivity function
sens_max	Maximum value of the sensitivity function

Value

A numeric vector with two components, minimum and maximum efficiency (over 1)

design_efficiency *Efficiency between optimal design and a user given design*

Description

Takes a optimal design provided from the function opt_des and a user given design and compares their efficiency

Usage

```
design_efficiency(opt_des_obj, design)
```

Arguments

opt_des_obj	An object given by the function opt_des.
design	A dataframe that represents the design. Must have two columns: • Point contains the support points of the design. • Weight contains the corresponding weights of the Points.

Value

The efficiency as a value between 0 and 1

See Also

opt_des

Examples

```
result <- opt_des("D-Optimality", y ~ a * exp(-b / x), c("a", "b"), c(1, 1500), c(212, 422))
design <- data.frame("Point" = c(220, 240, 400), "Weight" = c(1 / 3, 1 / 3, 1 / 3))
design_efficiency(result, design)
```

---

dscrit *Criterion function for Ds-Optimality*

Description

Calculates the value of the Ds-Optimality criterion, which follows the expression:

$$\phi_D = \frac{|M_{22}|}{|M|}^{1/s}$$

Usage

```
dscrit(M, par_int)
```

Arguments

M	Information matrix for which the criterion value wants to be calculated.
par_int	Numeric vector with the index of the parameters of interest of the model.

Value

Numeric value of the Ds-optimality criterion for the information matrix.

---

dsens *Sensitivity function for D-Optimality*

Description

Calculates the sensitivity function from the gradient vector and the Identity Matrix.

Usage

```
dsens(grad, M)
```

Arguments

grad	A function in a single variable that returns the partial derivatives vector of the model.
M	Information Matrix for the sensitivity function.

Value

The sensitivity function as a matrix of single variable.

---

dssens *Sensitivity function for Ds-Optimality*

Description

Calculates the sensitivity function from the gradient vector, the Identity Matrix and the parameters of interest.

Usage

```
dssens(grad, M, par_int)
```

Arguments

grad	A function in a single variable that returns the partial derivatives vector of the model.
M	Information Matrix for the sensitivity function.
par_int	Numeric vector of the indexes of the parameters of interest for Ds-Optimality.

Value

The sensitivity function as a matrix of single variable.

DsWFMult *Cocktail Algorithm implementation for Ds-Optimality*

Description

Function that calculates the Ds-Optimal designs for the interest parameters given by `intPar`. The rest of the parameters can help the convergence of the algorithm.

Usage

```
DsWFMult(
  init_design,
  grad,
  par_int,
  min,
  max,
  grid.length,
  join_thresh,
  delete_thresh,
  delta_weights,
  tol,
  tol2
)
```

Arguments

`init_design`	with the initial design for the algorithm. A dataframe with two columns: • `Point` contains the support points of the design. • `Weight` contains the corresponding weights of the `Points`.
`grad`	function of partial derivatives of the model.
`par_int`	numeric vector with the index of the `parameters` of interest. Only necessary when the `Criterion` chosen is 'Ds-Optimality'.
`min`	numeric value with the inferior bound of the space of the design.
`max`	numeric value with the upper bound of the space of the design.
`grid.length`	numeric value that gives the grid to evaluate the sensitivity function when looking for a maximum.
`join_thresh`	numeric value that states how close, in real units, two points must be in order to be joined together by the join heuristic.
`delete_thresh`	numeric value with the minimum weight, over 1 total, that a point needs to have in order to not be deleted from the design.
`delta_weights`	numeric value in (0, 1), parameter of the algorithm.
`tol`	numeric value for the convergence of the weight optimizing algorithm.
`tol2`	numeric value for the stop condition of the algorithm.

Value

list correspondent to the output of the correspondent algorithm called, dependent on the criterion. A list of two objects:

- optdes: a dataframe with the optimal design in two columns, `Point` and `Weight`.
- sens: a plot with the sensitivity function to check for optimality of the design.

See Also

Other cocktail algorithms: `DWFMult()`, `IWFMult()`, `WFMult()`

DWFMult *Cocktail Algorithm implementation for D-Optimality*

Description

Function that calculates the DsOptimal design. The rest of the parameters can help the convergence of the algorithm.

Usage

```
DWFMult(
  init_design,
  grad,
  min,
  max,
  grid.length,
  join_thresh,
  delete_thresh,
  k,
  delta_weights,
  tol,
  tol2
)
```

Arguments

init_design	with the initial design for the algorithm. A dataframe with two columns: • Point contains the support points of the design. • Weight contains the corresponding weights of the Points.
grad	function of partial derivatives of the model.
min	numeric value with the inferior bound of the space of the design.
max	numeric value with the upper bound of the space of the design.
grid.length	numeric value that gives the grid to evaluate the sensitivity function when looking for a maximum.
join_thresh	numeric value that states how close, in real units, two points must be in order to be joined together by the join heuristic.
delete_thresh	numeric value with the minimum weight, over 1 total, that a point needs to have in order to not be deleted from the design.
k	number of unknown parameters of the model.
delta_weights	numeric value in (0, 1), parameter of the algorithm.
tol	numeric value for the convergence of the weight optimizing algorithm.
tol2	numeric value for the stop condition of the algorithm.

Value

list correspondent to the output of the correspondent algorithm called, dependent on the criterion. A list of two objects:

- optdes: a dataframe with the optimal design in two columns, `Point` and `Weight`.
- sens: a plot with the sensitivity function to check for optimality of the design.

See Also

Other cocktail algorithms: `DsWFMult()`, `IWFMult()`, `WFMult()`

---

eff *Efficiency between two Information Matrices*

Description

Efficiency between two Information Matrices

Usage

```
eff(Criterion, mat1, mat2, k = 0, intPars = c(1), matB = NA)
```

Arguments

`Criterion`	Character with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality'
`mat1`	First information matrix, for the numerator.
`mat2`	Second information matrix, for the denominator.
`k`	Number of parameters of the model. Taken from the number of rows of the matrix if omitted.
`intPars`	Numeric vector with the index of the parameters of interest of the model. Only for "Ds-Optimality".
`matB`	Matrix of the integral of the information matrix over the interest region. Only for "I-Optimality".

Value

Efficiency of first design with respect to the second design, as a decimal number.

efficient_round *Efficient Round*

Description

Takes an approximate design, and a number of points and converts the design to an approximate design. It uses the multiplier (n - l/2) and evens the total number of observations afterwards.

Usage

```
efficient_round(design, n, tol = 1e-05)
```

Arguments

design	a data.frame with columns "Point" and "Weight" that represents a design
n	an integer that represents the desired number of observations of the exact design
tol	optional parameter for the consideration of a integer in the rounding process

Value

a data.frame with columns "Point" and "Weight" representing an exact design with n observations

Examples

```
design_test <- data.frame("Point" = seq(1, 5, length.out = 7),
         "Weight" = c(0.1, 0.0001, 0.2, 0.134, 0.073, 0.2111, 0.2818))

efficient_round(design_test, 20)

exact_design <- efficient_round(design_test, 21)
aprox_design <- exact_design
aprox_design$Weight <- aprox_design$Weight/sum(aprox_design$Weight)
```

findmax *Find Maximum*

Description

Searches the maximum of a function over a grid on a given interval.

Usage

```
findmax(sens, min, max, grid.length)
```

Arguments

sens	A single variable numeric function to evaluate.
min	Minimum value of the search interval.
max	Maximum value of the search interval.
grid.length	Length of the search interval.

Value

The value at which the maximum is obtained

---

findmaxval *Find Maximum Value*

Description

Searches the maximum of a function over a grid on a given interval.

Usage

```
findmaxval(sens, min, max, grid.length)
```

Arguments

sens	A single variable numeric function to evaluate.
min	Minimum value of the search interval.
max	Maximum value of the search interval.
grid.length	Length of the search interval.

Value

The value of the maximum

findminval *Find Minimum Value*

Description

Searches the maximum of a function over a grid on a given interval.

Usage

```
findminval(sens, min, max, grid.length)
```

Arguments

sens	A single variable numeric function to evaluate.
min	Minimum value of the search interval.
max	Maximum value of the search interval.
grid.length	Length of the search interval.

Value

The value of the minimum

getCross2 *Give effective limits to candidate points region*

Description

Given the start of the candidates points region, the parity of the crosspoints and the boundaries of the space of the design returns the effective limits of the candidate points region. Those points, taken in pairs from the first to the last delimit the region.

Usage

```
getCross2(cross, min, max, start, par)
```

Arguments

cross	Vector of crosspoints in the sensitivity function given an efficiency and weight
min	Minimum of the space of the design
max	Maximum of the space of the design
start	Boolean that gives the effective start of the candidate points region
par	Boolean with the parity of the region

Value

Vector of effective limits of the candidate points region. Taken in pairs from the beginning delimit the region.

getPar *Parity of the crosspoints*

Description

Determines if the number of crosspoints is even or odd given the vector of crosspoints

Usage

```
getPar(cross)
```

Arguments

cross	Vector of crosspoints in the sensitivity function given an efficiency and weight

Value

True if the number of crosspoints is even, false otherwise

getStart *Find where the candidate points region starts*

Description

Given the crosspoints and the sensitivity function, this function finds where the candidate points region starts, either on the extreme of the space of the design or the first crosspoints

Usage

```
getStart(cross, min, max, val, sens_opt)
```

Arguments

cross	Vector of crosspoints in the sensitivity function given an efficiency and weight
min	Minimum of the space of the design
max	Maximum of the space of the design
val	Value of the sensitivity function at the crosspoints
sens_opt	Sensitivity function

Value

True if the candidate points region starts on the minimum, False otherwise

gradient *Gradient function*

Description

Calculates the gradient function of a `model` with respect to the parameters, `char_vars`, evaluates it at the provided `values` and returns the result as a function of the variable `x`.

Usage

```
gradient(model, char_vars, values, weight_fun = function(x) 1)
```

Arguments

`model`	A formula describing the model, which must contain only `x`, the parameters defined in `char_vars` and the numerical operators.
`char_vars`	A character vector of the parameters of the model.
`values`	Numeric vector with the nominal values of the parameters in `char_vars`.
`weight_fun`	Optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response

Value

A function depending on `x` that's the gradient of the `model` with respect to `char_vars`

icrit *Criterion function for I-Optimality*

Description

Calculates the value of the Ds-Optimality criterion, which follows the expression:

$$\phi_D = \frac{|M_{22}|}{|M|}^{1/s}$$

Usage

```
icrit(M, matB)
```

Arguments

`M`	Information matrix for which the criterion value wants to be calculated.
`matB`	Matrix of the integral of the information matrix over the interest region. Identity matrix for A-Optimality.

Value

Numeric value of the Ds-optimality criterion for the information matrix.

---

inf_mat *Information Matrix*

Description

Given the gradient vector of a model in a single variable model and a design, calculates the information matrix.

Usage

```
inf_mat(grad, design)
```

Arguments

grad	A function in a single variable that returns the partial derivatives vector of the model.
design	A dataframe that represents the design. Must have two columns: • Point contains the support points of the design. • Weight contains the corresponding weights of the Points.

Value

The information matrix of the design, a $k \times k$ matrix where k is the length of the gradient.

---

integrate_reg_int *Integrate IM*

Description

Integrates the information matrix over the region of interest to calculate matrix B to be used in I-Optimality calculation.

Usage

```
integrate_reg_int(grad, k, reg_int)
```

Arguments

grad	function of partial derivatives of the model.
k	number of unknown parameters of the model.
reg_int	optional numeric vector of two components with the bounds of the interest region for I-Optimality.

Value

The integrated information matrix.

`isens` *Sensitivity function for I-Optimality*

Description

Calculates the sensitivity function from the gradient vector, the Identity Matrix and the integral of the one-point Identity Matrix over the interest region. If instead the identity matrix is used, it can be used for A-Optimality.

Usage

```
isens(grad, M, matB)
```

Arguments

`grad`	A function in a single variable that returns the partial derivatives vector of the model.
`M`	Information Matrix for the sensitivity function.
`matB`	Matrix resulting from the integration of the one-point Information Matrix along the interest region.

Value

The sensitivity function as a matrix of single variable.

`IWFMult` *Cocktail Algorithm implementation for I-Optimality and A-Optimality (with matB = diag(k))*

Description

Function that calculates the I-Optimal designs given the matrix B (should be integral of the information matrix over the interest region), or A-Optimal if given diag(k). The rest of the parameters can help the convergence of the algorithm.

Usage

```
IWFMult(
  init_design,
  grad,
  matB,
  min,
  max,
  grid.length,
  join_thresh,
  delete_thresh,
  delta_weights,
  tol,
  tol2
)
```

Arguments

`init_design`	with the initial design for the algorithm. A dataframe with two columns: • `Point` contains the support points of the design. • `Weight` contains the corresponding weights of the `Points`.
`grad`	function of partial derivatives of the model.
`matB`	optional matrix of dimensions k x k, integral of the information matrix of the model over the interest region.
`min`	numeric value with the inferior bound of the space of the design.
`max`	numeric value with the upper bound of the space of the design.
`grid.length`	numeric value that gives the grid to evaluate the sensitivity function when looking for a maximum.
`join_thresh`	numeric value that states how close, in real units, two points must be in order to be joined together by the join heuristic.
`delete_thresh`	numeric value with the minimum weight, over 1 total, that a point needs to have in order to not be deleted from the design.
`delta_weights`	numeric value in (0, 1), parameter of the algorithm.
`tol`	numeric value for the convergence of the weight optimizing algorithm.
`tol2`	numeric value for the stop condition of the algorithm.

Value

list correspondent to the output of the correspondent algorithm called, dependent on the criterion. A list of two objects:

- optdes: a dataframe with the optimal design in two columns, `Point` and `Weight`.
- sens: a plot with the sensitivity function to check for optimality of the design.

See Also

Other cocktail algorithms: `DWFMult()`, `DsWFMult()`, `WFMult()`

opt_des *Calculates the optimal design for a specified Criterion*

Description

The opt_des function calculates the optimal design for an optimality Criterion and a model input from the user. The parameters allows for the user to customize the parameters for the cocktail algorithm in case the default set don't provide a satisfactory output. Depending on the criterion, additional details are necessary. For 'Ds-Optimality' the par_int parameter is necessary. For 'I-Optimality' either the matB or reg_int must be provided.

Usage

```
opt_des(
  Criterion,
  model,
  parameters,
  par_values = c(1),
  design_space,
  init_design = NULL,
  join_thresh = -1,
  delete_thresh = 0.02,
  delta = 1/2,
  tol = 1e-05,
  tol2 = 1e-05,
  par_int = NULL,
  matB = NULL,
  reg_int = NULL,
  desired_output = c(1, 2),
  weight_fun = function(x) 1
)
```

Arguments

Criterion	character with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality'
model	formula describing the model to calculate the optimal design. Must use x as the variable.
parameters	character vector with the parameters of the models, as written in the formula.
par_values	numeric vector with the parameters nominal values, in the same order as given in parameters.

design_space	numeric vector of length 2, first component with the minimum of the space of the design and second component with the maximum.
init_design	optimal dataframe with the initial design for the algorithm. A dataframe with two columns: • Point contains the support points of the design. • Weight contains the corresponding weights of the Points.
join_thresh	optional numeric value that states how close, in real units, two points must be in order to be joined together by the join heuristic.
delete_thresh	optional numeric value with the minimum weight, over 1 total, that a point needs to have in order to not be deleted from the design.
delta	optional numeric value in (0, 1), parameter of the algorithm.
tol	optional numeric value for the convergence of the weight optimizing algorithm.
tol2	optional numeric value for the stop criterion: difference between maximum of sensitivity function and optimality criterion.
par_int	optional numeric vector with the index of the parameters of interest.
matB	optional matrix of dimensions k x k, integral of the information matrix of the model over the interest region.
reg_int	optional numeric vector of two components with the bounds of the interest region for I-Optimality.
desired_output	not functional yet: decide which kind of output you want.
weight_fun	optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response

Value

a list of two objects:

- optdes: a dataframe with the optimal design in two columns, Point and Weight.
- sens: a plot with the sensitivity function to check for optimality of the design.

Examples

```
opt_des("D-Optimality", y ~ a * exp(-b / x), c("a", "b"), c(1, 1500), c(212, 422))
```

---

plot.optdes *Plot function for optdes*

Description

Plot function for optdes

Usage

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

Arguments

x	An object of class optdes.
...	Possible extra arguments for plotting dataframes

Examples

```
rri <- opt_des(Criterion = "I-Optimality", model = y ~ a * exp(-b / x),
  parameters = c("a", "b"), par_values = c(1, 1500), design_space = c(212, 422),
  reg_int = c(380, 422))
plot(rri)
```

---

plot_convergence *Plot Convergence of the algorithm*

Description

Plots the criterion value on each of the steps of the algorithm, both for optimizing weights and points, against the total step number.

Usage

```
plot_convergence(convergence)
```

Arguments

convergence	A dataframe with two columns: • criteria contains value of the criterion on each step. • step contains number of the step.

Value

A ggplot object with the criteria in the y axis and step in the x axis.

---

plot_sens *Plot sensitivity function*

Description

Plots the sensitivity function and the value of the Equivalence Theorem as an horizontal line, which helps assess the optimality of the design of the given sensitivity function.

Usage

```
plot_sens(min, max, sens_function, criterion_value)
```

Arguments

min	Minimum of the space of the design, used in the limits of the representation.
max	Maximum of the space of the design, used in the limits of the representation.
sens_function	A single variable function, the sensitivity function.
criterion_value	A numeric value representing the other side of the inequality of the Equivalence Theorem.

Value

A ggplot object that represents the sensitivity function

---

print.optdes *Print function for optdes*

Description

Print function for optdes

Usage

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

Arguments

x	An object of class optdes.
...	Possible extra arguments for printing dataframes

Examples

```
rri <- opt_des(Criterion = "I-Optimality", model = y ~ a * exp(-b / x),
  parameters = c("a", "b"), par_values = c(1, 1500), design_space = c(212, 422),
  reg_int = c(380, 422))
print(rri)
```

sens — *Master function to calculate the sensitivity function*

Description

Calculates the sensitivity function given the desired `Criterion`, an information matrix and other necessary values depending on the chosen criterion.

Usage

```
sens(Criterion, grad, M, par_int = c(1), matB = NA)
```

Arguments

`Criterion`	Character with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality'
`grad`	A function in a single variable that returns the partial derivatives vector of the model.
`M`	Information Matrix for the sensitivity function.
`par_int`	Numeric vector of the indexes of the parameters of interest for Ds-Optimality.
`matB`	Matrix resulting from the integration of the one-point Information Matrix along the interest region.

Value

The sensitivity function as a matrix of single variable.

sens_val_to_add — *Calculates sensitivity function value for given delta and efficiency*

Description

Uses the formula to calculate the sensitivity function value that delimits which points can be added to the design guaranteeing the chosen efficiency.

Usage

```
sens_val_to_add(deff, alpha, k)
```

Arguments

deff	Minimum efficiency of the resulting design
alpha	Combined weight of the new points to add
k	Number of parameters of the model

Value

Value of the sensitivity function over. Points with a sensitivity function over that are suitable to be added.

---

shiny_augment *Shiny D-augment*

Description

Launches the demo shiny application to D-augment several preespecified models

Usage

```
shiny_augment()
```

Examples

```
shiny_augment()
```

---

shiny_optimal *Shiny Optimal*

Description

Launches the demo shiny application to calculate optimal designs for Antoine's Equation

Usage

```
shiny_optimal()
```

Examples

```
shiny_optimal()
```

summary.optdes *Summary function for optdes*

Description

Summary function for optdes

Usage

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

Arguments

object	An object of class optdes.
...	Possible extra arguments for the summary

Examples

```
rri <- opt_des(Criterion = "I-Optimality", model = y ~ a * exp(-b / x),
  parameters = c("a", "b"), par_values = c(1, 1500), design_space = c(212, 422),
  reg_int = c(380, 422))
summary(rri)
```

tr *Trace*

Description

Return the mathematical trace of a matrix, the sum of its diagonal elements.

Usage

```
tr(M)
```

Arguments

M	The matrix from which to calculate the trace.

Value

The trace of the matrix.

update_design *Update Design with new point*

Description

Updates a design adding a new point to it. If the added point is closer than `delta` to an existing point of the design, the two points are merged together as their arithmetic average. Then updates the weights to be equal to all points of the design.

Usage

```
update_design(design, xmax, delta, new_weight)
```

Arguments

`design`	Design to update. It's a dataframe with two columns: • `Point` contains the support points of the design. • `Weight` contains the corresponding weights of the `Points`.
`xmax`	The point to add as a numeric value.
`delta`	Threshold which defines how close the new point has to be to any of the existing ones in order to merge with them.
`new_weight`	Number with the weight for the new point.

Value

The updated design.

update_design_total *Merge close points of a design*

Description

Takes a design and merge together all points that are closer between them than a certain threshold `delta`.

Usage

```
update_design_total(design, delta)
```

Arguments

`design`	The design to update. It's a dataframe with two columns: • `Point` contains the support points of the design. • `Weight` contains the corresponding weights of the `Points`.
`delta`	Threshold which defines how close two points have to be to any of the existing ones in order to merge with them.

Value

The updated design.

`update_sequence` *Deletes duplicates points*

Description

Within a vector of points, deletes points that are close enough (less than the tol parameter). Returns the points without the "duplicates"

Usage

```
update_sequence(points, tol)
```

Arguments

`points`	Points to be updated
`tol`	Tolerance for which two points are considered the same

Value

The points without duplicates

`update_weights` *Update weight D-Optimality*

Description

Implementation of the weight update formula for D-Optimality used to optimize the weights of a design, which is to be applied iteratively until no sizable changes happen.

Usage

```
update_weights(design, sens, k, delta)
```

Arguments

`design`	Design to optimize the weights from. It's a dataframe with two columns: • `Point` contains the support points of the design. • `Weight` contains the corresponding weights of the `Points`.
`sens`	Sensibility function for the design and model.
`k`	Number of parameters of the model.
`delta`	A parameter of the algorithm that can be tuned. Must be $0 < delta < 1$.

Value

returns the new weights of the design after one iteration.

`update_weightsDS` *Update weight Ds-Optimality*

Description

Implementation of the weight update formula for Ds-Optimality used to optimize the weights of a design, which is to be applied iteratively until no sizable changes happen.

Usage

```
update_weightsDS(design, sens, s, delta)
```

Arguments

`design`	Design to optimize the weights from. It's a dataframe with two columns: • `Point` contains the support points of the design. • `Weight` contains the corresponding weights of the `Points`.
`sens`	Sensibility function for the design and model.
`s`	Number of interest parameters of the model
`delta`	A parameter of the algorithm that can be tuned. Must be $0 < delta < 1$.

Value

returns the new weights of the design after one iteration.

`update_weightsI` *Update weight I-Optimality*

Description

Implementation of the weight update formula for I-Optimality used to optimize the weights of a design, which is to be applied iteratively until no sizable changes happen. A-Optimality if instead of the integral matrix the identity function is used.

Usage

```
update_weightsI(design, sens, crit, delta)
```

Arguments

design	Design to optimize the weights from. It's a dataframe with two columns: • Point contains the support points of the design. • Weight contains the corresponding weights of the Points.
sens	Sensibility function for the design and model.
crit	Value of the criterion function for I-Optimality.
delta	A parameter of the algorithm that can be tuned. Must be $0 < delta < 1$.

Value

returns the new weights of the design after one iteration.

WFMult *Master function for the cocktail algorithm, that calls the appropriate one given the criterion.*

Description

Depending on the Criterion the cocktail algorithm for the chosen criterion is called, and the necessary parameters for the functions are given from the user input.

Usage

```
WFMult(
  init_design,
  grad,
  Criterion,
  par_int = NA,
  matB = NA,
  min,
  max,
  grid.length,
  join_thresh,
  delete_thresh,
  k,
  delta_weights,
  tol,
  tol2
)
```

Arguments

init_design	with the initial design for the algorithm. A dataframe with two columns: • Point contains the support points of the design. • Weight contains the corresponding weights of the Points.

grad	function of partial derivatives of the model.
Criterion	character with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality'
par_int	numeric vector with the index of the parameters of interest. Only necessary when the Criterion chosen is 'Ds-Optimality'.
matB	optional matrix of dimensions k x k, integral of the information matrix of the model over the interest region.
min	numeric value with the inferior bound of the space of the design.
max	numeric value with the upper bound of the space of the design.
grid.length	numeric value that gives the grid to evaluate the sensitivity function when looking for a maximum.
join_thresh	numeric value that states how close, in real units, two points must be in order to be joined together by the join heuristic.
delete_thresh	numeric value with the minimum weight, over 1 total, that a point needs to have in order to not be deleted from the design.
k	number of unknown parameters of the model.
delta_weights	numeric value in (0, 1), parameter of the algorithm.
tol	numeric value for the convergence of the weight optimizing algorithm.
tol2	numeric value for the stop condition of the algorithm.

Value

list correspondent to the output of the correspondent algorithm called, dependent on the criterion. A list of two objects:

- optdes: a dataframe with the optimal design in two columns, Point and Weight.
- sens: a plot with the sensitivity function to check for optimality of the design.

See Also

Other cocktail algorithms: DWFMult(), DsWFMult(), IWFMult()

Index