Package 'optedr'

July 3, 2026

Title Calculating Optimal and D-Augmented Designs for Single- and Multi-Factor Models

Version 3.0.1

Description Calculates D-, Ds-, A-, I- and L-optimal designs, weighted combinations of these via a Compound criterion, and KL-optimal designs for model discrimination, for non-linear single- and multi-factor models, via an implementation of the cocktail algorithm (Yu, 2011, <doi:10.1007/s11222-010-9183-2>). Multi-factor models use design variables x1, x2, ... with a named-list design space; single-factor models remain backward compatible. Compares designs via their efficiency, augments any design with a controlled efficiency loss, and provides efficient rounding functions to convert approximate designs to exact ones.

License GPL-3

Encoding UTF-8

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

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

RoxygenNote 7.3.1

VignetteBuilder knitr

Suggests testthat (>= 3.0.0), mockery, knitr, rmarkdown, markdown, DT, shinydashboard, shinyalert, plotly, hrbrthemes, shinyjs, orthopolynom, magrittr, tidyverse, nleqslv

Imports ggplot2, purrr, rlang, crayon, cli, shiny, utils

Config/testthat/edition 3

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-07-03 16:00:02 UTC

Contents

add_design . . . 3
add_points . . . 4
augment_design . . . 4
check_inputs . . . 6
combinatorial_round . . . 7
crit . . . 9
crosspoints . . . 10
CWFMult . . . 10
daugment_design . . . 11
dcrit . . . 13
delete_points . . . 13
design_efficiency . . . 14
detect_design_vars . . . 14
dsaugment_design . . . 15
dscrit . . . 16
dsens . . . 17
dssens . . . 17
DsWFMult . . . 18
DWFMult . . . 19
eff . . . 20
efficient_round . . . 20
findmax . . . 21
findmaxval . . . 22
findminval . . . 22
getCross2 . . . 23
getPar . . . 23
getStart . . . 24
get_augment_region . . . 24
get_daugment_region . . . 26
get_dsaugment_region . . . 27
get_laugment_region . . . 28
gradient . . . 30
gradient22 . . . 30
icrit . . . 31
inf_mat . . . 32
integrate_reg_int . . . 32
isens . . . 33
IWFMult . . . 33
KLWFMult . . . 34
laugment_design . . . 36
make_glm_family . . . 37
make_kl_fun . . . 38

opt_des 39
plot.optdes 42
plot_convergence 43
plot_sens 43
print.augment_region 44
print.optdes 44
sens 45
shiny_augment 45
shiny_optimal 46
summary.optdes 46
tr 47
update_design 47
update_design_total 48
update_sequence 48
update_weights 49
update_weightsDS 49
update_weightsI 50
weight_function 50
WFMult 51

Index **53**

`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 columns that represent the first design to add
`design_2`	A dataframe with 'Point' and 'Weight' as columns 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

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(
  criterion,
  init_design,
  alpha,
  model,
  parameters,
  par_values,
  design_space,
  calc_optimal_design,
  par_int = NA,
  matB = NULL,
```

```
  distribution = NA,
  weight_fun = function(x) 1,
  delta_val = NULL,
  new_points = NULL,
  n_lhs = 2000L
)
```

Arguments

criterion	character variable with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality' • 'L-Optimality'
init_design	dataframe with "Point" and "Weight" columns that represents the initial design to augment
alpha	combined weight of the new points
model	formula that represents the model with x as the independent variable
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 limits 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
par_int	optional numeric vector with the index of the parameters of interest for Ds-optimality.
matB	optional matrix of dimensions k x k, for L-optimality.
distribution	character specifying the probability distribution of the response. Can be one of the following: • 'Homoscedasticity' • 'Gamma', which can be used for exponential or normal heteroscedastic with constant relative error • 'Poisson' • 'Logistic' • 'Log-Normal' (work in progress)
weight_fun	optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response
delta_val	optional numeric value for the minimum relative efficiency. If NULL (default), the user is prompted interactively. Providing this value enables non-interactive use.

new_points	optional dataframe with Point and Weight columns specifying the points to add. All points must lie within the candidate region determined by delta_val. If NULL (default), the user is prompted interactively.
n_lhs	integer number of LHS points used to visualise the candidate region for multi-factor models with d > 2 (default 2000). Increase for smoother scatter plots.

Value

A dataframe that represents the augmented design

Examples

```
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
region <- get_augment_region("D-Optimality", 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, delta_val = 0.85)
new_pts <- data.frame(Point = mean(region$region[1:2]), Weight = 1)
augment_design("D-Optimality", 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,
  delta_val = 0.85, new_points = new_pts)
```

---

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,
  weight_fun
)
```

Arguments

criterion	character variable with the chosen optimality criterion.
model	formula describing the model. For single-factor models use x as the design variable; for multi-factor models use x1, x2, etc.
parameters	character vector with the parameter names.
par_values	numeric vector with the nominal parameter values.
design_space	For single-factor models: numeric vector c(min, max). For multi-factor models: named list list(x1 = c(min, max), x2 = c(min, max), ...).
init_design	optional dataframe with the initial design. For single-factor models use columns Point and Weight; for multi-factor models use one column per design variable plus Weight.
join_thresh	optional numeric threshold for merging nearby design points.
delete_thresh	optional numeric minimum weight to keep a support point.
delta	optional numeric in (0, 1), damping parameter of the algorithm.
tol	optional numeric for convergence of the weight loop.
tol2	optional numeric for the outer stop condition.
par_int	optional numeric vector of parameter indices for Ds-optimality.
matB	optional k x k matrix for L-optimality.
reg_int	optional bounds for the I-optimality integration region. Single-factor: c(min, max). Multi-factor: named list matching design_space.
weight_fun	optional variance-structure weight function.

---

combinatorial_round *Combinatorial round*

Description

Given an approximate design and a number of points, computes all the possible combinations of roundings of each point to the nearest integer, keeps the ones that amount to the requested number of points, and returns the one with the best value for the criterion function.

The search is exhaustive and requires 2^k evaluations where k is the number of support points. For designs with more than max_support support points the function requests confirmation (interactive sessions) or stops with an informative error (non-interactive sessions), unless ask = FALSE.

Usage

```
combinatorial_round(
  design,
  n,
  criterion = NULL,
  model = NULL,
  parameters = NULL,
```

```
  par_values = NULL,
  weight_fun = function(x) 1,
  par_int = NULL,
  reg_int = NULL,
  matB = NULL,
  max_support = 15L,
  ask = TRUE
)
```

Arguments

design	either a dataframe with the design to round, or an object of class "optdes". If a dataframe, the criterion, model and parameters must be specified. It must have a Weight column and one or more design-variable columns (Point for single-factor, x1, x2, ... for multi-factor).
n	integer with the desired number of points of the resulting design.
criterion	character variable with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality' • 'L-Optimality'
model	formula describing the model. Must use x (single-factor) or x1, x2, ... (multi-factor) as design variables.
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.
weight_fun	optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response.
par_int	optional numeric vector with the index of the parameters of interest for Ds-optimality.
reg_int	optional numeric vector with the ranges of integration, for I-optimality.
matB	optional matrix of dimensions k x k, for L-optimality.
max_support	integer. Number of support points above which the function triggers the confirmation mechanism. Default is 15 ($2^{15} \approx 32\,000$ combinations).
ask	logical. If TRUE (default) and the design exceeds max_support: in an interactive session the user is prompted; in a non-interactive session an error is raised. Set ask = FALSE to skip confirmation in scripts or pipelines (a message is still emitted).

Value

A data.frame with the rounded design to n number of points, or NULL invisibly if the user declines the confirmation prompt.

Examples

```
aprox_design <- opt_des("D-Optimality", y ~ a * exp(-b / x), c("a", "b"), c(1, 1500), c(212, 422))
combinatorial_round(aprox_design, 27)
```

`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 variable with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality' • 'L-Optimality'
`M`	information matrix for which the criterion value wants to be calculated.
`k`	numeric variable with the 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`	optional matrix of dimensions k x k, for I- and L-optimality.

Value

Numeric value of the optimality criterion for the information matrix.

crosspoints *Calculate crosspoints*

Description

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

Usage

```
crosspoints(val, sens, gridlength, tol, xmin, xmax)
```

Arguments

val	Efficiency value to solve in the curve relationing the space of the design and efficiency of new design
sens	Sensitivity function of the design for the model
gridlength	Number of points in the initial grid used to bracket roots
tol	Tolerance for root refinement passed to uniroot
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

CWFMult *Cocktail Algorithm implementation for Compound Optimality*

Description

Cocktail Algorithm implementation for Compound Optimality

Usage

```
CWFMult(
  init_design,
  grad,
  compound_specs,
  design_space,
  grid.length,
  join_thresh,
  delete_thresh,
  delta_weights,
```

```
  tol,
  tol2,
  max_iter
)
```

Arguments

init_design	optional dataframe with the initial design for the algorithm.
grad	function of partial derivatives of the model.
compound_specs	preprocessed list of per-criterion specifications.
design_space	named list with bounds for each design variable.
grid.length	numeric value for the sensitivity search grid / LHS size.
join_thresh	numeric value for the merge heuristic.
delete_thresh	numeric value for the weight deletion threshold.
delta_weights	numeric value in (0, 1), parameter of the algorithm.
tol	numeric value for convergence of the weight loop.
tol2	numeric value for the outer stop condition.
max_iter	maximum number of outer iterations.

Value

An object of class optdes.

See Also

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

---

daugment_design *D-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

```
daugment_design(
  init_design,
  alpha,
  model,
  parameters,
  par_values,
```

```
  design_space,
  calc_optimal_design,
  weight_fun = function(x) 1,
  delta_val = NULL,
  new_points = NULL
)
```

Arguments

`init_design`	dataframe with "Point" and "Weight" columns that represents the initial design to augment
`alpha`	combined weight of the new points
`model`	formula that represents the model with x as the independent variable
`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 limits 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
`delta_val`	optional numeric value for the minimum relative efficiency. If `NULL` (default), the user is prompted interactively. Providing this value enables non-interactive use.
`new_points`	optional dataframe with `Point` and `Weight` columns specifying the points to add. All points must lie within the candidate region determined by `delta_val`. If `NULL` (default), the user is prompted interactively.

Value

A dataframe that represents the augmented design

See Also

Other augment designs: `dsaugment_design()`, `laugment_design()`

Examples

```
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
region <- get_augment_region("D-Optimality", 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, delta_val = 0.85)
new_pts <- data.frame(Point = mean(region$region[1:2]), Weight = 1)
augment_design("D-Optimality", 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,
  delta_val = 0.85, new_points = new_pts)
```

dcrit *Criterion function for D-Optimality*

Description

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

$$\phi_D = \left(\frac{1}{|M|}\right)^{1/k}$$

Usage

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

Arguments

M	information matrix for which the criterion value wants to be calculated.
k	numeric variable with the 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.

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

Description

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

Usage

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

Arguments

`design`	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`.
`opt_des_obj`	an object given by the function `opt_des`.

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(design, result)
```

detect_design_vars *Detect design variables from a model formula*

Description

Inspects the formula to identify which symbols are design variables (as opposed to parameters or the response). Enforces the naming convention: `"x"` for single-factor models and `"x1"`, `"x2"`, ... for multi-factor models.

Usage

```
detect_design_vars(model, parameters)
```

Arguments

model	A formula describing the model.
parameters	Character vector of parameter names (excluded from design variables).

Value

Character vector of design variable names, sorted in numerical order for multi-factor models (e.g. `c("x1", "x2", "x10")`).

---

dsaugment_design *Ds-Augment Design*

Description

Ds-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

```
dsaugment_design(
  init_design,
  alpha,
  model,
  parameters,
  par_values,
  par_int,
  design_space,
  calc_optimal_design,
  weight_fun = function(x) 1,
  delta_val = NULL,
  new_points = NULL
)
```

Arguments

init_design	dataframe with "Point" and "Weight" columns that represents the initial design to augment
alpha	combined weight of the new points
model	formula that represents the model with x as the independent variable
parameters	character vector with the unknown parameters of the model to estimate
par_values	numeric vector with the initial values of the unknown parameters
par_int	optional numeric vector with the index of the `parameters` of interest for Ds-optimality.
design_space	numeric vector with the limits 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

`delta_val`
: optional numeric value for the minimum relative efficiency. If `NULL` (default), the user is prompted interactively. Providing this value enables non-interactive use.

`new_points`
: optional dataframe with `Point` and `Weight` columns specifying the points to add. All points must lie within the candidate region determined by `delta_val`. If `NULL` (default), the user is prompted interactively.

Value

A dataframe that represents the Ds-augmented design

See Also

Other augment designs: `daugment_design()`, `laugment_design()`

Examples

```
## Not run:
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
augment_design("Ds-Optimality", init_des, 0.25, y ~ 10^(a-b/(c+x)), c("a","b","c"),
  c(8.07131,  1730.63, 233.426), c(1, 100), par_int = c(1), TRUE)

## End(Not run)
```

---

dscrit *Criterion function for Ds-Optimality*

Description

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

$$\phi_{Ds} = \left(\frac{|M_{22}|}{|M|} \right)^{1/s}$$

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

Cocktail Algorithm implementation for Ds-Optimality

Usage

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

Arguments

init_design	optional dataframe with the initial design for the algorithm.
grad	function of partial derivatives of the model.
par_int	numeric vector with the index of the parameters of interest (Ds-Optimality).
design_space	named list with bounds for each design variable.
grid.length	numeric value for the sensitivity search grid / LHS size.
join_thresh	numeric value for the merge heuristic.
delete_thresh	numeric value for the weight deletion threshold.
delta_weights	numeric value in (0, 1), parameter of the algorithm.
tol	numeric value for convergence of the weight loop.
tol2	numeric value for the outer stop condition.
max_iter	maximum number of outer iterations.

Value

An object of class optdes.

See Also

Other cocktail algorithms: CWFMult(), DWFMult(), IWFMult(), KLWFMult(), WFMult()

DWFMult *Cocktail Algorithm implementation for D-Optimality*

Description

Cocktail Algorithm implementation for D-Optimality

Usage

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

Arguments

init_design	optional dataframe with the initial design for the algorithm.
grad	function of partial derivatives of the model.
design_space	named list with bounds for each design variable.
grid.length	numeric value for the sensitivity search grid / LHS size.
join_thresh	numeric value for the merge heuristic.
delete_thresh	numeric value for the weight deletion threshold.
k	number of unknown parameters of the model.
delta_weights	numeric value in (0, 1), parameter of the algorithm.
tol	numeric value for convergence of the weight loop.
tol2	numeric value for the outer stop condition.
max_iter	maximum number of outer iterations.

Value

An object of class optdes.

See Also

Other cocktail algorithms: CWFMult(), DsWFMult(), IWFMult(), KLWFMult(), 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 variable with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality' • 'L-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, seed = NULL)
```

Arguments

design	a dataframe with a Weight column and one or more design-variable columns (Point for single-factor designs, x1, x2, ... for multi-factor designs).
n	an integer that represents the desired number of observations of the exact design
tol	optional parameter for the consideration of an integer in the rounding process
seed	optional integer seed for reproducibility. When the rounded weights sum to less than n, a random tie-breaking step is used; supplying seed makes that step deterministic by calling set.seed(seed) immediately before it. NULL (default) leaves the global RNG state unchanged.

Value

a data.frame with the same columns as design, with Weight replaced by integer observation counts summing to n.

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)

# Reproducible tie-breaking
efficient_round(design_test, 20, seed = 42)
```

---

findmax *Find Maximum*

Description

Searches the location of the maximum of a sensitivity function over the design space. For single-factor models the search uses a regular grid followed by direct selection. For multi-factor models a Latin Hypercube Sample is evaluated and then refined with L-BFGS-B local optimisation from the n_starts best candidate points.

Usage

```
findmax(sens, design_space, grid.length, n_starts = 5L)
```

Arguments

sens	Sensitivity function (scalar for 1D, named numeric vector for multi-factor).
design_space	Numeric vector c(min, max) or named list.
grid.length	Number of grid / LHS points for the initial sweep.
n_starts	Number of local-optimisation restarts (multi-factor only).

Value

The design point (scalar or named numeric vector) at which `sens` is maximised.

`findmaxval` *Find Maximum Value*

Description

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

Usage

```
findmaxval(sens, design_space, grid.length)
```

Arguments

`sens`	A numeric function to evaluate (scalar argument for 1D, named numeric vector for multi-factor).
`design_space`	Numeric vector `c(min, max)` for single-factor models, or a named list `list(x1 = c(min, max), ...)` for multi-factor models.
`grid.length`	Number of grid points (1D) or LHS samples (multi-factor).

Value

The value of the maximum

`findminval` *Find Minimum Value*

Description

Searches the minimum of a function over a grid on the design space.

Usage

```
findminval(sens, design_space, grid.length)
```

Arguments

`sens`	A numeric function to evaluate (scalar for 1D, named numeric vector for multi-factor).
`design_space`	Numeric vector `c(min, max)` for single-factor models, or a named list `list(x1 = c(min, max), ...)` for multi-factor models.
`grid.length`	Number of grid points (1D) or LHS samples (multi-factor).

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

get_augment_region *Get Augment Regions*

Description

Given a model and criterion, calculates the candidate points region. 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.

Usage

```
get_augment_region(
  criterion,
  init_design,
  alpha,
  model,
  parameters,
  par_values,
  design_space,
  calc_optimal_design,
```

```
  par_int = NA,
  matB = NA,
  distribution = NA,
  weight_fun = function(x) 1,
  delta_val = NULL,
  n_lhs = 2000L
)
```

Arguments

criterion	character with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality' • 'L-Optimality'
init_design	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 independent variable
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 limits 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
par_int	optional numeric vector with the index of the parameters of interest for Ds-optimality.
matB	optional matrix of dimensions k x k, for L-optimality.
distribution	character specifying the probability distribution of the response. Can be one of the following: • 'Homoscedasticity' • 'Gamma', which can be used for exponential or normal heteroscedastic with constant relative error • 'Poisson' • 'Logistic' • 'Log-Normal' (work in progress)
weight_fun	optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response
delta_val	optional numeric value for the minimum relative efficiency. If NULL (default), the user is prompted interactively. Providing this value enables non-interactive use.
n_lhs	integer number of LHS points used to visualise and sample the candidate region for multi-factor models with d > 2 (default 2000). Increase for smoother scatter plots.

Value

A vector of the points limiting the candidate points region

Examples

```
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
get_augment_region("D-Optimality", 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,
  delta_val = 0.85)
```

---

get_daugment_region *Get D-augment region*

Description

Given a model, calculates the candidate points region for D-Optimality. 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.

Usage

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

Arguments

init_design	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 independent variable
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 limits 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
delta_val	optional numeric value for the minimum relative efficiency. If NULL (default), the user is prompted interactively. Providing this value enables non-interactive use.

Value

A vector of the points limiting the candidate points region

Examples

```
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
get_augment_region("D-Optimality", 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,
  delta_val = 0.85)
```

---

get_dsaugment_region *Get Ds-augment region*

Description

Given a model, calculates the candidate points region for Ds-Optimality. 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.

Usage

```
get_dsaugment_region(
  init_design,
  alpha,
  model,
  parameters,
  par_values,
  par_int,
  design_space,
  calc_optimal_design,
  weight_fun = function(x) 1,
  delta_val = NULL
)
```

Arguments

init_design	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 independent variable
parameters	character vector with the unknown parameters of the model to estimate
par_values	numeric vector with the initial values of the unknown parameters
par_int	optional numeric vector with the index of the parameters of interest for Ds-optimality.
design_space	numeric vector with the limits 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
delta_val	optional numeric value for the minimum relative efficiency. If NULL (default), the user is prompted interactively. Providing this value enables non-interactive use.

Value

A vector of the points limiting the candidate points region

See Also

Other augment region: get_laugment_region()

Examples

```
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
get_augment_region("D-Optimality", 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,
  delta_val = 0.85)
```

---

get_laugment_region *Get L-augment region*

Description

Given a model, calculates the candidate points region for L-Optimality. 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.

Usage

```
get_laugment_region(
  init_design,
  alpha,
  model,
  parameters,
  par_values,
  design_space,
  calc_optimal_design,
  matB,
  weight_fun = function(x) 1,
  delta_val = NULL
)
```

Arguments

init_design	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 independent variable
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 limits 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
matB	optional matrix of dimensions k x k, for L-optimality.
weight_fun	optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response
delta_val	optional numeric value for the minimum relative efficiency. If NULL (default), the user is prompted interactively. Providing this value enables non-interactive use.

Value

A vector of the points limiting the candidate points region

See Also

Other augment region: get_dsaugment_region()

Examples

```
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
get_augment_region("D-Optimality", 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,
  delta_val = 0.85)
```

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`	formula describing the model, which must contain only `x`, the parameters defined in `char_vars` and the numerical operators.
`char_vars`	character vector of the parameters of the model.
`values`	numeric vector with the nominal values of the parameters in `char_vars`.
`weight_fun`	optional function variable 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`

gradient22 *Gradient function for a subset of variables*

Description

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

Usage

```
gradient22(model, char_vars, values, par_int, weight_fun = function(x) 1)
```

Arguments

model	formula describing the model, which must contain only x, the parameters defined in char_vars and the numerical operators.
char_vars	character vector of the parameters of the model.
values	numeric vector with the nominal values of the parameters in char_vars.
par_int	vector of indexes indicating the subset of variables to omit in the calculation of the gradient.
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 and L-Optimality*

Description

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

$$\phi_I = Tr(M^{-1} \cdot B)$$

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 I-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 Information 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 or lineal matrix for L-Optimality.

Value

The sensitivity function as a matrix of single variable.

IWFMult *Cocktail Algorithm implementation for L-, I- and A-Optimality*

Description

Cocktail Algorithm implementation for L-, I- and A-Optimality

Usage

```
IWFMult(
  init_design,
  grad,
  matB,
  design_space,
  grid.length,
  join_thresh,
  delete_thresh,
  delta_weights,
  tol,
  tol2,
```

```
  criterion,
  max_iter
)
```

Arguments

init_design	optional dataframe with the initial design for the algorithm.
grad	function of partial derivatives of the model.
matB	optional matrix for L-optimality.
design_space	named list with bounds for each design variable.
grid.length	numeric value for the sensitivity search grid / LHS size.
join_thresh	numeric value for the merge heuristic.
delete_thresh	numeric value for the weight deletion threshold.
delta_weights	numeric value in (0, 1), parameter of the algorithm.
tol	numeric value for convergence of the weight loop.
tol2	numeric value for the outer stop condition.
criterion	character variable with the chosen optimality criterion.
max_iter	maximum number of outer iterations.

Value

An object of class optdes.

See Also

Other cocktail algorithms: CWFMult(), DWFMult(), DsWFMult(), KLWFMult(), WFMult()

KLWFMult *Cocktail Algorithm for KL-Optimality*

Description

Cocktail Algorithm for KL-Optimality

Usage

```
KLWFMult(
  init_design,
  kl_fun,
  beta2_init,
  lower,
  upper,
  design_space,
  grid.length,
```

```
  join_thresh,
  delete_thresh,
  delta_weights,
  tol,
  tol2,
  max_iter,
  kl_meta = list(type = "kl_fun")
)
```

Arguments

`init_design`	optional dataframe with the initial design for the algorithm.
`kl_fun`	function(x, beta2); returns the KL divergence at design point `x` given rival parameters `beta2`.
`beta2_init`	numeric vector of initial rival parameter values.
`lower`	lower bounds for rival parameters in the inner optimisation.
`upper`	upper bounds for rival parameters in the inner optimisation.
`design_space`	named list with bounds for each design variable.
`grid.length`	numeric value for the sensitivity search grid / LHS size.
`join_thresh`	numeric value for the merge heuristic.
`delete_thresh`	numeric value for the weight deletion threshold.
`delta_weights`	numeric value in (0, 1), parameter of the algorithm.
`tol`	numeric value for convergence of the weight loop.
`tol2`	numeric value for the outer stop condition.
`max_iter`	maximum number of outer iterations.
`kl_meta`	list with summary metadata (`type`, and optionally `family` and `phi` for the standard path).

Value

An object of class `optdes`.

See Also

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

`laugment_design` *L-Augment Design*

Description

L-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

```
laugment_design(
  init_design,
  alpha,
  model,
  parameters,
  par_values,
  design_space,
  calc_optimal_design,
  matB,
  weight_fun = function(x) 1,
  delta_val = NULL,
  new_points = NULL
)
```

Arguments

`init_design`	dataframe with "Point" and "Weight" columns that represents the initial design to augment
`alpha`	combined weight of the new points
`model`	formula that represents the model with x as the independent variable
`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 limits 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
`matB`	optional matrix of dimensions k x k, for L-optimality.
`weight_fun`	optional one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response
`delta_val`	optional numeric value for the minimum relative efficiency. If `NULL` (default), the user is prompted interactively. Providing this value enables non-interactive use.
`new_points`	optional dataframe with `Point` and `Weight` columns specifying the points to add. All points must lie within the candidate region determined by `delta_val`. If `NULL` (default), the user is prompted interactively.

Value

A dataframe that represents the L-augmented design

See Also

Other augment designs: daugment_design(), dsaugment_design()

Examples

```
## Not run:
init_des <- data.frame("Point" = c(30, 60, 90), "Weight" = c(1/3, 1/3, 1/3))
augment_design("I-Optimality", 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)

## End(Not run)
```

---

make_glm_family *GLM family specification for KL-Optimality*

Description

Returns a GLM family object for use with `criterion = "KL-Optimality"`. The family encodes the cumulant generating function and canonical link needed to compute the Kullback-Leibler divergence between two distributions.

Usage

```
make_glm_family(name)
```

Arguments

name	character; one of "Normal", "Poisson", "Binomial" or "Gamma".

Value

A named list with elements b, b_prime, b_dbl, link, link_inv and name.

Examples

```
make_glm_family("Poisson")
make_glm_family("Normal")
```

make_kl_fun *Build a KL-divergence point function for use with opt_des()*

Description

Returns a `function(x, beta2)` computing the point KL divergence at design point `x` given rival parameters `beta2`. The result is passed to `opt_des` via the `kl_fun` argument, allowing discrimination between models with different family, dispersion, or mean structure without having to derive the formula manually.

Supported family pairs:

- Same family and same `phi`: any of `"Normal"`, `"Poisson"`, `"Binomial"`, `"Gamma"`. Uses the standard exponential-family cumulant formula.
- `"Normal"` vs `"Normal"` with `phi1 != phi2`: $KL = \frac{1}{2}\left[\log(\phi_2/\phi_1) + \phi_1/\phi_2 + (\mu_1 - \mu_2)^2/\phi_2 - 1\right]$.
- `"Gamma"` vs `"Gamma"` with different shape ($k_i = 1/\phi_i$): closed form involving `digamma` and `lgamma`.

For other cross-family pairs provide `kl_fun` directly.

Usage

```
make_kl_fun(
  family1,
  model1,
  params1,
  par_values1,
  family2 = family1,
  model2 = model1,
  params2 = params1,
  phi1 = 1,
  phi2 = phi1
)
```

Arguments

`family1`	character; reference distribution (`"Normal"`, `"Poisson"`, `"Binomial"` or `"Gamma"`).
`model1`	formula; reference model mean function.
`params1`	character vector; parameter names in `model1`.
`par_values1`	numeric vector; nominal values for the reference parameters.
`family2`	character; rival distribution (default: same as `family1`).
`model2`	formula; rival model mean function (default: same as `model1`).
`params2`	character vector; rival parameter names (optimised internally). Default: same as `params1`.
`phi1`	positive numeric; dispersion of the reference (`Normal`: variance; `Gamma`: $1/\text{shape}$).
`phi2`	positive numeric; dispersion of the rival (default: same as `phi1`).

Value

A function `function(x, beta2)` giving the point KL divergence. Works for both 1-D (`x` scalar) and multi-factor designs (`x` named numeric vector).

Examples

```
# Same family (Normal), different model structures
kl_fn <- make_kl_fun(
  "Normal",
  model1 = y ~ Vmax * x / (Km + x), params1 = c("Vmax", "Km"),
  par_values1 = c(2, 1),
  model2 = y ~ a * x, params2 = "a"
)
kl_fn(x = 1, beta2 = 0.5)


# Normal vs Normal with different variance (phi2 = 4)
kl_fn2 <- make_kl_fun(
  "Normal",
  model1 = y ~ a * exp(-b * x), params1 = c("a", "b"),
  par_values1 = c(1, 0.5), phi1 = 1,
  family2 = "Normal",
  model2 = y ~ c * exp(-d * x), params2 = c("c", "d"), phi2 = 4
)
opt_des("KL-Optimality",
        model = y ~ a * exp(-b * x), parameters = c("a", "b"),
        par_values = c(1, 0.5), design_space = c(0, 4),
        kl_fun = kl_fn2, rival_pars = c(1, 1),
        rival_lower = c(0.5, 0.8), rival_upper = c(2, 1.5))
```

`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. Supports single-factor models (design variable named `x`) and multi-factor models (design variables named `x1`, `x2`, ..., detected automatically from the formula).

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,
  max_iter = 21L,
  distribution = NA,
  weight_fun = function(x) 1,
  compound = NULL,
  rival_model = NULL,
  rival_params = NULL,
  rival_pars = NULL,
  family = "Normal",
  phi = 1,
  rival_lower = NULL,
  rival_upper = NULL,
  kl_fun = NULL
)
```

Arguments

criterion	character variable with the chosen optimality criterion.
model	formula describing the model. For single-factor models use x as the design variable; for multi-factor models use x1, x2, etc.
parameters	character vector with the parameter names.
par_values	numeric vector with the nominal parameter values.
design_space	For single-factor models: numeric vector c(min, max). For multi-factor models: named list list(x1 = c(min, max), x2 = c(min, max), ...).
init_design	optional dataframe with the initial design. For single-factor models use columns Point and Weight; for multi-factor models use one column per design variable plus Weight.
join_thresh	optional numeric threshold for merging nearby design points.
delete_thresh	optional numeric minimum weight to keep a support point.
delta	optional numeric in (0, 1), damping parameter of the algorithm.
tol	optional numeric for convergence of the weight loop.
tol2	optional numeric for the outer stop condition.
par_int	optional numeric vector of parameter indices for Ds-optimality.
matB	optional k x k matrix for L-optimality.
reg_int	optional bounds for the I-optimality integration region. Single-factor: c(min, max). Multi-factor: named list matching design_space.
max_iter	optional integer maximum number of outer cocktail iterations.

distribution	character variable specifying the response distribution.
weight_fun	optional variance-structure weight function.
compound	optional list of criterion specifications for criterion = "Compound". Each element must be a named list with at least criterion (character) and weight (positive numeric). Additional fields per sub-criterion: par_int (Ds-Optimality), reg_int (I-Optimality), matB (L-Optimality). Weights are normalised to sum to 1. Example: list(list(criterion="D-Optimality", weight=0.7), list(criterion="I-Optimal weight=0.3, reg_int=c(380,422))).
rival_model	optional formula for the rival model used with criterion = "KL-Optimality". Defaults to model (same structure, rival parameters explored by inner optimisation).
rival_params	optional character vector of rival model parameter names. Defaults to parameters.
rival_pars	optional numeric vector of initial rival parameter values for the inner optimisation. Defaults to par_values.
family	character; GLM family for criterion = "KL-Optimality". One of "Normal", "Poisson", "Binomial", "Gamma". Default "Normal".
phi	positive numeric dispersion parameter for KL-Optimality. Default 1.
rival_lower	optional numeric vector of lower bounds for rival parameters in the inner optimisation. Defaults to -Inf for each parameter.
rival_upper	optional numeric vector of upper bounds for rival parameters in the inner optimisation. Defaults to Inf for each parameter.
kl_fun	optional user-supplied function(x, beta2) giving the point KL divergence at design point x for rival parameters beta2, as built by make_kl_fun. When provided, bypasses the family/rival_model path entirely; rival_pars must still be supplied for the inner optimisation.

Value

a list of class optdes with components optdes, convergence, sens, criterion, crit_value, and atwood.

Examples

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


# Two-factor Michaelis-Menten bisubstrate model
opt_des("D-Optimality",
        y ~ Vmax * x1 * x2 / ((K1 + x1) * (K2 + x2)),
        c("Vmax", "K1", "K2"), c(1, 1, 1),
        list(x1 = c(0.1, 10), x2 = c(0.1, 10)))

# Compound D+I (70% D, 30% I) for Antoine equation
opt_des("Compound",
        y ~ 10^(a - b/(c + x)), c("a","b","c"),
        c(8.07131, 1730.63, 233.426), c(1, 100),
```

```
        compound = list(
          list(criterion = "D-Optimality", weight = 0.7),
          list(criterion = "I-Optimality", weight = 0.3, reg_int = c(60, 100))
        ))

# KL-Optimality: discriminate quadratic from linear mean model (Normal)
opt_des("KL-Optimality",
        model        = y ~ a * x^2,
        parameters   = c("a"),
        par_values   = c(1),
        design_space = c(1, 5),
        rival_model  = y ~ b * x,
        rival_params = c("b"),
        rival_pars   = c(3),
        family       = "Normal",
        phi          = 1)
```

`plot.optdes` *Plot function for optdes*

Description

For single-factor models, overlays the support points on the sensitivity function curve. For two-factor models, shows a heatmap of the sensitivity function with the support points overlaid and the Equivalence Theorem contour highlighted. For models with more than two factors, shows a pairwise scatter matrix with one panel per pair of design variables and point size proportional to weight.

Usage

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

Arguments

`x`	An object of class `optdes`.
`...`	Possible extra arguments (currently unused).

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.augment_region *Print method for augment_region objects*

Description

Print method for augment_region objects

Usage

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

Arguments

x	An object of class augment_region returned by get_augment_region.
...	Unused.

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 variable with the chosen optimality criterion. Can be one of the following: • 'D-Optimality' • 'Ds-Optimality' • 'A-Optimality' • 'I-Optimality' • 'L-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 or lineal matrix for L-Optimality.

Value

The sensitivity function as a matrix of single variable.

shiny_augment *Shiny D-augment*

Description

Launches the demo Shiny application to D-augment several pre-specified models. Requires an interactive R session; if called non-interactively a message with the web URL is emitted instead.

Usage

```
shiny_augment()
```

Examples

```
shiny_augment()
```

`shiny_optimal` *Shiny Optimal*

Description

Launches the demo Shiny application to calculate optimal designs for Antoine's Equation. Requires an interactive R session; if called non-interactively a message with the web URL is emitted instead.

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 parameters of interest 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.

`weight_function` *Weight function per distribution*

Description

Weight function per distribution

Usage

```
weight_function(model, char_vars, values, distribution = "Normal")
```

Arguments

model	formula describing the model to use. Must use x as the design variable.
char_vars	character vector with the parameters of the models, as written in the formula
values	numeric vector with the parameters nominal values, in the same order as given in parameters.
distribution	character variable specifying the probability distribution of the response. Can be one of the following: • 'Normal', for normal homoscedastic (default) • 'Gamma', which can be used for exponential or normal heteroscedastic with constant relative error • 'Poisson' • 'Logistic'

Value

one variable function that represents the square of the structure of variance, in case of heteroscedastic variance of the response.

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,
  design_space,
  grid.length,
  join_thresh,
  delete_thresh,
  k,
  delta_weights,
  tol,
  tol2,
  max_iter,
  compound = NULL,
  kl_spec = NULL
)
```

Arguments

init_design	optional dataframe with the initial design for the algorithm.
grad	function of partial derivatives of the model.
criterion	character variable with the chosen optimality criterion.
par_int	numeric vector with the index of the parameters of interest (Ds-Optimality).
matB	optional matrix for L-optimality.
design_space	named list with bounds for each design variable.
grid.length	numeric value for the sensitivity search grid / LHS size.
join_thresh	numeric value for the merge heuristic.
delete_thresh	numeric value for the weight deletion threshold.
k	number of unknown parameters of the model.
delta_weights	numeric value in (0, 1), parameter of the algorithm.
tol	numeric value for convergence of the weight loop.
tol2	numeric value for the outer stop condition.
max_iter	maximum number of outer iterations.
compound	preprocessed list of compound criterion specifications (internal use).
kl_spec	preprocessed list of KL-Optimality specifications (internal use).

Value

An object of class optdes.

See Also

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

Index

∗ **augment designs**
daugment_design, 11
dsaugment_design, 15
laugment_design, 36
∗ **augment regions**
get_daugment_region, 26
∗ **augment region**
get_dsaugment_region, 27
get_laugment_region, 28
∗ **cocktail algorithms**
CWFMult, 10
DsWFMult, 18
DWFMult, 19
IWFMult, 33
KLWFMult, 34
WFMult, 51

add_design, 3
add_points, 4
augment_design, 4

check_inputs, 6
combinatorial_round, 7
crit, 9
crosspoints, 10
CWFMult, 10, *18, 19*, *34, 35*, *52*

daugment_design, 11, *16*, *37*
dcrit, 13
delete_points, 13
design_efficiency, 14
detect_design_vars, 14
dsaugment_design, *12*, 15, *37*
dscrit, 16
dsens, 17
dssens, 17
DsWFMult, *11*, 18, *19*, *34, 35*, *52*
DWFMult, *11*, *18*, 19, *34, 35*, *52*

eff, 20
efficient_round, 20

findmax, 21
findmaxval, 22
findminval, 22

get_augment_region, 24
get_daugment_region, 26
get_dsaugment_region, 27, *29*
get_laugment_region, *28*, 28
getCross2, 23
getPar, 23
getStart, 24
gradient, 30
gradient22, 30

icrit, 31
inf_mat, 32
integrate_reg_int, 32
isens, 33
IWFMult, *11*, *18, 19*, 33, *35*, *52*

KLWFMult, *11*, *18, 19*, *34*, 34, *52*

laugment_design, *12*, *16*, 36

make_glm_family, 37
make_kl_fun, 38, *41*

opt_des, *38*, 39

plot.optdes, 42
plot_convergence, 43
plot_sens, 43
print.augment_region, 44
print.optdes, 44

sens, 45
shiny_augment, 45
shiny_optimal, 46
summary.optdes, 46

tr, 47

update_design, 47
update_design_total, 48
update_sequence, 48
update_weights, 49
update_weightsDS, 49
update_weightsI, 50

weight_function, 50
WFMult, *11*, *18, 19*, *34, 35*, 51