

Package ‘ICAOD’

January 22, 2018

Title Optimal Designs for Nonlinear Models

Version 0.9.6

Description Finds optimal designs for nonlinear models using a metaheuristic algorithm called imperialist competitive algorithm ICA. See, for details, Masoudi et al. (2017) <doi:10.1016/j.csda.2016.06.014>.

Depends R (>= 3.1.3)

License GPL (>= 2)

LazyData true

LinkingTo Rcpp, RcppEigen

Imports Rcpp, nloptr, stats, graphics, grDevices, cubature, sn, mnormt, methods

Suggests rgl, lattice, knitr, rmarkdown

BugReports <https://github.com/ehsan66/ICAOD/issues>

URL <https://github.com/ehsan66/ICAOD>

RoxygenNote 6.0.1

NeedsCompilation yes

Author Ehsan Masoudi [aut, cre],
Heinz Holling [aut],
Weng Kee Wong [aut]

Maintainer Ehsan Masoudi <ehsan.masoudi@wwu.de>

Repository CRAN

Date/Publication 2018-01-21 23:02:52 UTC

R topics documented:

bayes	2
bayescomp	11
crt.bayes.control	14
crt.minimax.control	15
FIM_2par_exp_censor1	17

FIM_2par_exp_censor2	17
FIM_3par_exp_censor1	18
FIM_3par_exp_censor2	19
FIM_exp_2par	19
FIM_logistic	20
FIM_logistic_2pred	21
FIM_logistic_4par	21
FIM_loglin	22
FIM_mixed_inhibition	23
FIM_power_logistic	24
ICA.control	24
ICAOD	26
iterate	28
iterate.bayes	28
iterate.minimax	29
locally	29
minimax	33
multiple	41
normal	44
plot.bayes	45
plot.minimax	46
print.bayes	48
print.minimax	48
print.sensbayes	49
print.sensminimax	49
robust	50
sens.bayes.control	53
sens.minimax.control	54
sensbayes	57
sensbayescomp	60
senslocally	62
sensminimax	65
sensmultiple	72
sensrobust	75
skewnormal	77
student	78
uniform	79

Index **81**

bayes *Bayesian D-Optimal Designs*

Description

Finds (pseudo) Bayesian D-optimal designs for nonlinear models. It should be used when the user assumes a (truncated) prior distribution for the unknown model parameters. If the prior is discrete, please use [robust](#).

Usage

```
bayes(formula, predvars, parvars, family = gaussian(), prior, lx, ux, iter, k,
      fimfunc = NULL, ICA.control = list(), crt.bayes.control = list(),
      sens.bayes.control = list(), initial = NULL, npar = NULL,
      plot_3d = c("lattice", "rgl"))
```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.
prior	An object of class <code>cprior</code> . User can also use one of the functions uniform , normal , skewnormal or student to create the prior. See 'Details' of bayes .
lx	Vector of lower bounds for the predictors. Should be in the same order as <code>predvars</code> .
ux	Vector of upper bounds for the predictors. Should be in the same order as <code>predvars</code> .
iter	Maximum number of iterations.
k	Number of design points. Must be at least equal to the number of model parameters to avoid singularity of the FIM.
fimfunc	A function. Returns the FIM as a matrix. Required when <code>formula</code> is missing. See 'Details' of minimax .
ICA.control	ICA control parameters. For details, see ICA.control .
crt.bayes.control	Control parameters to approximate the integral in Bayesian criterion at a given design over the parameter space. For details, see crt.bayes.control .
sens.bayes.control	Control parameters to verify the general equivalence theorem. For details, see sens.bayes.control .
initial	A matrix of the initial designs that will be used as initial solutions (countries). Every row is a design, i.e. a concatenation of <code>x</code> and <code>w</code> . Will be coerced to a matrix if necessary. See 'Details' of minimax .
npar	Number of model parameters. Used when <code>fimfunc</code> is given instead of <code>formula</code> to specify the number of model parameters. If not specified truly, the sensitivity (derivative) plot may be shifted below the y-axis. When <code>NULL</code> , it will be set to <code>length(parvars)</code> or <code>prior\$npar</code> when <code>missing(formula)</code> .
plot_3d	Which package should be used to plot the sensitivity (derivative) function for two-dimensional design space. Defaults to "lattice".

Details

Let Ξ be the space of all approximate designs with k design points (support points) at x_1, x_2, \dots, x_k from design space χ with corresponding weights w_1, \dots, w_k . Let $M(\xi, \theta)$ be the Fisher information matrix (FIM) of a k -point design ξ and $\pi(\theta)$ is a user-given prior distribution for the vector of unknown parameters θ . A Bayesian D-optimal design ξ^* minimizes over Ξ

$$\int_{\theta \in \Theta} -\log |M(\xi, \theta)| \pi(\theta) d\theta.$$

An object of class `cprior` is a list with the following components:

- `fn`: Prior distribution as an R function with argument `param` that is the vector of the unknown parameters. See below.
- `npar`: Number of unknown parameters and is equal to the length of `param`.
- `lower`: Argument lower. It has the same length as `param`.
- `upper`: Argument lower. It has the same length as `param`.

A `cprior` object will be passed to the argument `prior` of the function `bayes`. The argument `param` in `fn` has the same order as the argument `parvars` when the model is specified by a formula. Otherwise, it is same as the argument `param` in the function `fimfunc`.

The user can apply the available prior (object `cprior`) creators that are `uniform`, `normal`, `skewnormal` and `student` to create a `cprior` object.

Use `plot` function to verify the general equivalence theorem for the output design.

To increase the speed of the algorithm, change the tuning parameters `tol` and `maxEval` via the argument `crt.bayes.control`. In this case, the user should find a trade-off between accuracy and speed for his/her example.

If some of the parameters are fixed in a model, they should be set to their values via the argument `paravars`. In this case, you must provide the number of parameters via argument `npar` for verifying the general equivalence theorem. See 'Examples', Section 'Weibull', 'Richards' and 'Exponential' model.

Value

an object of class `bayes` that is a list including three sub-lists:

`arg` A list of design and algorithm parameters.

`evol` A list of length equal to the number of iterations that stores the information about the best design (design with least criterion value) of each iteration as follows: `evol[[iter]]` contains:

<code>iter</code>	Iteration number.
<code>x</code>	Design points.
<code>w</code>	Design weights.
<code>min_cost</code>	Cost (Bayesian criterion value) of the best imperialist.
<code>mean_cost</code>	Mean of costs of all imperialists.
<code>sens</code>	An object of class 'sensbayes'. See below.

empires A list of all empires of the last iteration.

alg A list with following information:

nfeval	Number of function evaluations. See below.
nlocal	Number of successful local search.
nrevol	Number of successful revolutions.
nimprove	Number of successful movements toward the imperialists in assimilation step.
convergence	Stopped by 'maxiter' or 'equivalence'?

sens contains information about design verification by the general equivalence theorem. See sensbayes for more Details. It is only available every `ICA.control$checkfreq` iterations and the last iteration if `ICA.control$checkfreq >= 0`. Otherwise, NULL.

nfeval does not count the function evaluations from checking the general equivalence theorem.

See Also

[sensbayes](#)

Examples

```
#####
# Two parameter logistic model: uniform prior
#####
# set the uniform prior
uni <- uniform(lower = c(-3, .1), upper = c(3, 2))
# set the logistic model with formula
res1 <- bayes(formula = ~1/(1 + exp(-b *(x - a))),
              predvars = "x", parvars = c("a", "b"),
              family = binomial(), lx = -3, ux = 3,
              k = 5, iter = 1, prior = uni,
              ICA.control = list(rseed = 1366))

## Not run:
res1 <- iterate(res1, 500)
plot(res1)

## End(Not run)

# You can also use your Fisher information matrix (FIM) if you think it is faster!
## Not run:
bayes(fimfunc = FIM_logistic, lx = -3, ux = 3, k = 5, iter = 500,
      prior = uni, ICA.control = list(rseed = 1366))

## End(Not run)

#####
# Two parameter logistic model: normal prior #1
#####
# defining the normal prior #1
norm1 <- normal(mu = c(0, 1),
```

```

sigma = matrix(c(1, -0.17, -0.17, .5), nrow = 2),
lower = c(-3, .1), upper = c(3, 2))

## Not run:
# initializing
res2 <- bayes(formula = ~1/(1 + exp(-b *(x - a))), predvars = "x", parvars = c("a", "b"),
family = binomial(), lx = -3, ux = 3, k = 4, iter = 1, prior = norm1,
ICA.control = list(rseed = 1366))
res2 <- iterate(res2, 500)
plot(res2)

## End(Not run)

#####
# Two parameter logistic model: normal prior #2
#####
# defining the normal prior #1
norm2 <- normal(mu = c(0, 1),
sigma = matrix(c(1, 0, 0, .5), nrow = 2),
lower = c(-3, .1), upper = c(3, 2))

## Not run:
# initializing
res3 <- bayes(formula = ~1/(1 + exp(-b *(x - a))), predvars = "x", parvars = c("a", "b"),
family = binomial(), lx = -3, ux = 3, k = 4, iter = 1, prior = norm2,
ICA.control = list(rseed = 1366))

res3 <- iterate(res3, 700)
plot(res3, sens.bayes.control = list(cubature = list(maxEval = 3000, tol = 1e-4),
optplist = list(maxeval = 3000)))

## End(Not run)

#####
# Two parameter logistic model: skewed normal prior #1
#####
skew1 <- skewnormal(xi = c(0, 1),
Omega = matrix(c(1, -0.17, -0.17, .5), nrow = 2),
alpha = c(1, 0), lower = c(-3, .1), upper = c(3, 2))

## Not run:
res4 <- bayes(formula = ~1/(1 + exp(-b *(x - a))), predvars = "x", parvars = c("a", "b"),
family = binomial(), lx = -3, ux = 3, k = 4, iter = 700, prior = skew1,
ICA.control = list(rseed = 1366, ncount = 60))
plot(res4, sens.bayes.control = list(cubature = list(maxEval = 3000, tol = 1e-4),
optplist = list(maxeval = 3000)))

## End(Not run)

#####
# Two parameter logistic model: skewed normal prior #2
#####
skew2 <- skewnormal(xi = c(0, 1),
Omega = matrix(c(1, -0.17, -0.17, .5), nrow = 2),

```

```

alpha = c(-1, 0), lower = c(-3, .1), upper = c(3, 2))
## Not run:
res5 <- bayes(formula = ~1/(1 + exp(-b *(x - a))), predvars = "x", parvars = c("a", "b"),
family = binomial(), lx = -3, ux = 3, k = 4, iter = 700, prior = skew2,
ICA.control = list(rseed = 1366, ncount = 60))
plot(res5, sens.bayes.control = list(cubature = list(maxEval = 3000, tol = 1e-4),
optslis = list(maxeval = 3000)))

## End(Not run)

#####
# Two parameter logistic model: t student prior
#####
# set the prior
stud <- student(mean = c(0, 1), S = matrix(c(1, -0.17, -0.17, .5), nrow = 2),
df = 3, lower = c(-3, .1), upper = c(3, 2))
## Not run:
res6 <- bayes(formula = ~1/(1 + exp(-b *(x - a))), predvars = "x", parvars = c("a", "b"),
family = binomial(), lx = -3, ux = 3, k = 5, iter = 500, prior = stud,
ICA.control = list(ncount = 50, rseed = 1366))
plot(res6)

## End(Not run)
# not bad, but to find a very accurate designs we increase
# the ncount to 200 and repeat the optimization
## Not run:
res6 <- bayes(formula = ~1/(1 + exp(-b *(x - a))),
predvars = "x", parvars = c("a", "b"),
family = binomial(), lx = -3, ux = 3, k = 5, iter = 1000, prior = stud,
ICA.control = list(ncount = 200, rseed = 1366))
plot(res6)

## End(Not run)

#####
# 4-parameter sigmoid Emax model: uniform prior
#####
lb <- c(4, 11, 100, 5)
ub <- c(8, 15, 130, 9)
## Not run:
res7 <- bayes(formula = ~ theta1 + (theta2 - theta1)*(x^theta4)/(x^theta4 + theta3^theta4),
predvars = c("x"), parvars = c("theta1", "theta2", "theta3", "theta4"),
lx = .001, ux = 500, k = 5, iter = 200, prior = uniform(lb, ub),
ICA.control = list(rseed = 1366, ncount = 60))
plot(res7, sens.bayes.control = list(cubature = list(maxEval = 500, tol = 1e-3),
optslis = list(maxeval = 500)))

## End(Not run)

#####
# 2-parameter Cox Proportional-Hazards Model for type one censored data
#####

```

```

# The Fisher information matrix is available here with name FIM_2par_exp_censor1
# However, we should reparameterize the function to match the standard of the argument 'fimfunc'
myfim <- function(x, w, param)
  FIM_2par_exp_censor1(x = x, w = w, param = param, tcensor = 30)
## Not run:
res8 <- bayes(fimfunc = myfim, lx = 0, ux = 1, k = 4,
             iter = 1, prior = uniform(c(-11, -11), c(11, 11)),
             ICA.control = list(rseed = 1366))

res8 <- iterate(res8, 200)
plot(res8, sens.bayes.control = list(cubature = list(maxEval = 500, tol = 1e-3),
                                     optslist = list(maxeval = 500)))

## End(Not run)

#####
# 2-parameter Cox Proportional-Hazards Model for random censored data
#####
# The Fisher information matrix is available here with name FIM_2par_exp_censor2
# However, we should reparameterize the function to match the standard of the argument 'fimfunc'
myfim <- function(x, w, param)
  FIM_2par_exp_censor2(x = x, w = w, param = param, tcensor = 30)
## Not run:
res9 <- bayes(fimfunc = myfim, lx = 0, ux = 1, k = 2,
             iter = 200, prior = uniform(c(-11, -11), c(11, 11)),
             ICA.control = list(rseed = 1366))
plot(res9, sens.bayes.control = list(cubature = list(maxEval = 100, tol = 1e-3),
                                     optslist = list(maxeval = 100)))

## End(Not run)

#####
# Weibull model: Uniform prior
#####
# see Dette, H., & Pepelyshev, A. (2008).
# Efficient experimental designs for sigmoidal growth models.
# Journal of statistical planning and inference, 138(1), 2-17.

## See how we fixed a some parameters in Bayesian designs
## Not run:
res10 <- bayes(formula = ~a - b * exp(-lambda * t ^h),
              predvars = c("t"),
              parvars = c("a=1", "b=1", "lambda", "h=1"),
              lx = .00001, ux = 20,
              prior = uniform(.5, 2.5), k = 5, iter = 400,
              ICA.control = list(rseed = 1366))
plot(res10)

## End(Not run)

#####
# Weibull model: Normal prior

```



```
#####
norm3 <- normal(mu = 1, sigma = .1, lower = .5, upper = 2.5)
res11 <- bayes(formula = ~a - b * exp(-lambda * t ^h),
               predvars = c("t"),
               parvars = c("a=1", "b=1", "lambda", "h=1"),
               lx = .00001, ux = 20, prior = norm3, k = 4, iter = 1,
               ICA.control = list(rseed = 1366))

## Not run:
res11 <- iterate(res11, 400)
plot(res11)

## End(Not run)

#####
# Richards model: Normal prior
#####
norm4 <- normal(mu = c(1, 1), sigma = matrix(c(.2, 0.1, 0.1, .4), 2, 2),
               lower = c(.4, .4), upper = c(1.6, 1.6))

## Not run:
res12 <- bayes(formula = ~a/(1 + b * exp(-lambda*t))^h,
               predvars = c("t"),
               parvars = c("a=1", "b", "lambda", "h=1"),
               lx = .00001, ux = 10,
               prior = norm4,
               k = 5, iter = 400,
               ICA.control = list(rseed = 1366))
plot(res12, sens.bayes.control = list(cubature = list(maxEval = 1000, tol = 1e-3),
                                     optslis = list(maxeval = 1000)))

## End(Not run)

#####
# Exponential model: Uniform prior
#####
res13 <- bayes(formula = ~a + exp(-b*x), predvars = "x",
               parvars = c("a = 1", "b"),
               lx = 0.0001, ux = 1,
               prior = uniform(lower = 1, upper = 20),
               iter = 1, k = 3,
               ICA.control= list(rseed = 100))

## Not run:
res13 <- iterate(res13, 300)
plot(res13)

## End(Not run)

#####
# Power logistic model
#####
# See, Duarte, B. P., & Wong, W. K. (2014).
# A Semidefinite Programming based approach for finding
# Bayesian optimal designs for nonlinear models
```

```

uni1 <- uniform(lower = c(-.3, 6, .5), upper = c(.3, 8, 1))
## Not run:
res14 <- bayes(formula = ~1/(1 + exp(-b *(x - a)))^s, predvars = "x",
               parvars = c("a", "b", "s"),
               lx = -1, ux = 1, prior = uni1, k = 5, iter = 1)
res14 <- iterate(res14, 300)
plot(res14)

## End(Not run)

#####
# A two-variable generalized linear model with a gamma distributed response
#####
lb <- c(.5, 0, 0, 0, 0, 0)
ub <- c(2, 1, 1, 1, 1, 1)
myformula1 <- ~beta0+beta1*x1+beta2*x2+beta3*x1^2+beta4*x2^2+beta5*x1*x2
## Not run:
res15 <- bayes(formula = myformula1,
               predvars = c("x1", "x2"), parvars = paste("beta", 0:5, sep = ""),
               family = Gamma(),
               lx = rep(0, 2), ux = rep(1, 2),
               prior = uniform(lower = lb, upper = ub),
               k = 7, iter = 1, ICA.control = list(rseed = 1366))
res14 <- iterate(res14, 500)
plot(res14, sens.bayes.control = list(cubature = list(maxEval = 5000, tol = 1e-4),
                                     optslist = list(maxeval = 3000)))

## End(Not run)

#####
# Three parameter logistic model
#####

sigma1 <- matrix(-0.1, nrow = 3, ncol = 3)
diag(sigma1) <- c(.5, .4, .1)
norm5 <- normal(mu = c(0, 1, .2), sigma = sigma1,
               lower = c(-3, .1, 0), upper = c(3, 2, .7))
## Not run:
res16 <- bayes(formula = ~ c + (1-c)/(1 + exp(-b *(x - a))), predvars = "x",
               parvars = c("a", "b", "c"),
               family = binomial(), lx = -3, ux = 3,
               k = 4, iter = 500, prior = norm5,
               ICA.control = list(rseed = 1366, ncount = 50),
               crt.bayes.control = list(cubature = list(maxEval = 2500, tol = 1e-4)))
plot(res16, sens.bayes.control = list(cubature = list(maxEval = 3000, tol = 1e-4),
                                     optslist = list(maxeval = 3000)))

# took 925 second on my system

## End(Not run)

```

Description

Finds compound Bayesian DP-optimal designs that meets the dual goal of the parameter estimation and increasing the probability of a particular outcome in a binary response model. A compound Bayesian DP-optimal design maximizes the product of the efficiencies of a design ξ with respect to D- and average P-optimality, weighted by a pre-defined mixing constant $0 \leq \alpha \leq 1$.

Usage

```
bayescomp(formula, predvars, parvars, family = binomial(), prior, alpha, prob,
  lx, ux, iter, k, fimfunc = NULL, ICA.control = list(),
  crt.bayes.control = list(), sens.bayes.control = list(), initial = NULL,
  npar = NULL, plot_3d = c("lattice", "rgl"))
```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.
prior	An object of class <code>cprior</code> . User can also use one of the functions uniform , normal , skewnormal or student to create the prior. See 'Details' of bayes .
alpha	A value between 0 and 1. Compound or combined DP-criterion is the product of the efficiencies of a design with respect to D- and average P- optimality, weighted by alpha.
prob	Either formula or a function. when function, its argument is <code>x</code> and <code>param</code> same as the arguments in <code>fimfunc</code> . <code>prob</code> as a function takes the design points and vector of parameters and returns the probability of success at each design points. See 'Examples'.
lx	Vector of lower bounds for the predictors. Should be in the same order as <code>predvars</code> .
ux	Vector of upper bounds for the predictors. Should be in the same order as <code>predvars</code> .
iter	Maximum number of iterations.

<code>k</code>	Number of design points. Must be at least equal to the number of model parameters to avoid singularity of the FIM.
<code>fimfunc</code>	A function. Returns the FIM as a matrix. Required when formula is missing. See 'Details' of minimax .
<code>ICA.control</code>	ICA control parameters. For details, see ICA.control .
<code>crt.bayes.control</code>	Control parameters to approximate the integral in Bayesian criterion at a given design over the parameter space. For details, see crt.bayes.control .
<code>sens.bayes.control</code>	Control parameters to verify the general equivalence theorem. For details, see sens.bayes.control .
<code>initial</code>	A matrix of the initial designs that will be used as initial solutions (countries). Every row is a design, i.e. a concatenation of <code>x</code> and <code>w</code> . Will be coerced to a matrix if necessary. See 'Details' of minimax .
<code>npar</code>	Number of model parameters. Used when <code>fimfunc</code> is given instead of formula to specify the number of model parameters. If not specified truly, the sensitivity (derivative) plot may be shifted below the y-axis. When NULL, it will be set to <code>length(parvars)</code> or <code>prior\$npar</code> when <code>missing(formula)</code> .
<code>plot_3d</code>	Which package should be used to plot the sensitivity (derivative) function for two-dimensional design space. Defaults to "lattice".

Details

Let Ξ be the space of all approximate designs with k design points (support points) at x_1, x_2, \dots, x_k from design space χ with corresponding weights w_1, \dots, w_k . Let $M(\xi, \theta)$ be the Fisher information matrix (FIM) of a k -point design ξ , $\pi(\theta)$ is a user-given prior distribution for the vector of unknown parameters θ and $p(x_i, \theta)$ is the i th probability of success given by x_i in a binary response model. A Bayesian compound DP-optimal criterion maximizes over Ξ

$$\int_{\theta \in \Theta} \frac{\alpha}{q} \log |M(\xi, \theta)| + (1 - \alpha) \log \left(\sum_{i=1}^k w_i p(x_i, \theta) \right) \pi(\theta) d\theta.$$

Use [plot](#) function to verify the general equivalence theorem for the output design.

To increase the speed of the algorithm, change the tuning parameters `tol` and `maxEval` via the argument `crt.bayes.control`.

Value

an object of class `bayes` that is a list including three sub-lists:

`arg` A list of design and algorithm parameters.

`evol` A list of length equal to the number of iterations that stores the information about the best design (design with least criterion value) of each iteration as follows: `evol[[iter]]` contains:

<code>iter</code>	Iteration number.
<code>x</code>	Design points.

w	Design weights.
min_cost	Cost (Bayesian criterion value) of the best imperialist.
mean_cost	Mean of costs of all imperialists.
sens	An object of class 'sensbayes'. See below.

empires A list of all empires of the last iteration.

alg A list with following information:

nfeval	Number of function evaluations. See below.
nlocal	Number of successful local search.
nrevol	Number of successful revolutions.
nimprove	Number of successful movements toward the imperialists in assimilation step.
convergence	Stopped by 'maxiter' or 'equivalence'?

sens contains information about design verification by the general equivalence theorem. See sensbayes for more Details. It is only available every ICA.control\$checkfreq iterations and the last iteration if ICA.control\$checkfreq >= 0. Otherwise, NULL.

nfeval does not count the function evaluations from checking the general equivalence theorem.

References

McGree, J. M., Eccleston, J. A., and Duffull, S. B. (2008). Compound optimal design criteria for nonlinear models. *Journal of Biopharmaceutical Statistics*, 18(4), 646-661.

See Also

[sensbayescomp](#)

Examples

```
#####
# DP-optimal design for a logitic model with two predictors: with formula
#####
p <- c(1, -2, 1, -1)
myprior <- uniform(p -1.5, p + 1.5)
myformula1 <- ~exp(b0+b1*x1+b2*x2+b3*x1*x2)/(1+exp(b0+b1*x1+b2*x2+b3*x1*x2))
res1 <- bayescomp(formula = myformula1,
                  predvars = c("x1", "x2"),
                  parvars = c("b0", "b1", "b2", "b3"),
                  family = binomial(),
                  lx = c(-1, -1), ux = c(1, 1),
                  prior = myprior, iter = 1, k = 7,
                  prob = ~1-1/(1+exp(b0 + b1 * x1 + b2 * x2 + b3 * x1 * x2)),
                  alpha = .5, ICA.control = list(rseed = 1366))

## Not run:
res1 <- iterate(res1, 1000)
```

```

plot(res1, sens.bayes.control = list(cubature = list(tol = 1e-3, maxEval = 1000)))

## End(Not run)

#####
# DP-optimal design for a logitic model with two predictors: with fimfunc
#####
# The function of the Fisher information matrix for this model is 'FIM_logistic_2pred'
# We should reparameterize it to match the standard of the argument 'fimfunc'
myfim <- function(x, w, param){
  npoint <- length(x)/2
  x1 <- x[1:npoint]
  x2 <- x[(npoint+1):(npoint*2)]
  FIM_logistic_2pred(x1 = x1,x2 = x2, w = w, param = param)
}

## The following function is equivalent to the function created
# by the formula:  $\sim 1-1/(1+\exp(b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_1 * x_2))$ 
# It returns probability of success given x and param
# x = c(x1, x2) and param = c()

myprob <- function(x, param){
  npoint <- length(x)/2
  x1 <- x[1:npoint]
  x2 <- x[(npoint+1):(npoint*2)]
  b0 <- param[1]
  b1 <- param[2]
  b2 <- param[3]
  b3 <- param[4]
  out <- 1-1/(1+exp(b0 + b1 * x1 + b2 * x2 + b3 * x1 * x2))
  return(out)
}

## Not run:
res2 <- bayescomp(fimfunc = myfim,
  lx = c(-1, -1), ux = c(1, 1),
  prior = myprior, iter = 1000, k = 7,
  prob = myprob, alpha = .5,
  ICA.control = list(rseed = 1366))
plot(res2, sens.bayes.control = list(cubature = list(maxEval = 1000, tol = 1e-4)))

## End(Not run)

```

Description

The function `crt.bayes.control` returns a list of `hcubature` control parameters for approximating the integrals in Bayesian criteria. The key tuning parameters here are `tol` and `maxEval`. Their value

affect the algorithm speed and the accuracy of the results. The user should find a trade-off between accuracy and speed for his/her example.

Usage

```
crt.bayes.control(cubature = list(tol = 1e-05, maxEval = 50000, absError = 0))
```

Arguments

cubature A list that will be passed to the arguments of the function [hcubature](#). See 'Details'.

Details

cubature is a list that its components will be passed to the function [hcubature](#). Its components are:

tol The maximum tolerance. Defaults to $1e-5$.

maxEval The maximum number of function evaluations needed. Note that the actual number of function evaluations performed is only approximately guaranteed not to exceed this number. Defaults to 5000 .

absError The maximum absolute error tolerated. Defaults to 0 .

One can specify a maximum number of function evaluations. Otherwise, the integration stops when the estimated error is less than the absolute error requested, or when the estimated error is less than **tol** times the integral, in absolute value, or the maximum number of iterations is reached, whichever is earlier.

Value

A list of control parameters for [hcubature](#).

Examples

```
crt.bayes.control()
crt.bayes.control(cubature = list(tol = 1e-4))
```

crt.minimax.control *Control Parameters for Optimizing Minimax Criteria Over Parameter Space*

Description

The function `crt.minimax.control` returns a list of [nloptr](#) control parameters for optimizing the minimax criterion over the parameter space.

The key tuning parameter here is `maxeval`. Its value should be increased when either the dimension or the size of the parameter space becomes larger to avoid pre-mature convergence in the inner optimization problem over the parameter space. The user should find a trade-off between accuracy and speed for his/her example.

Usage

```
crt.minimax.control(x0 = NULL, optslist = list(stopval = -Inf, algorithm =
  "NLOPT_GN_DIRECT_L", xtol_rel = 1e-05, ftol_rel = 1e-08, maxeval = 1000), ...)
```

Arguments

`x0` Vector of starting values for the optimization problem (must be from the parameter space).

`optslist` A list. It will be passed to the argument `opts` of the function `nloptr`. See 'Details'.

`...` Further arguments will be passed to `nl.opts` from package `nloptr`.

Details

Argument `optslist` will be passed to the argument `opts` of the function `nloptr`:

`stopval` Stop minimization when an objective value \leq `stopval` is found. Setting `stopval` to `-Inf` disables this stopping criterion (default).

`algorithm` Defaults to `NLOPT_GN_DIRECT_L`. `DIRECT-L` is a deterministic-search algorithm based on systematic division of the search domain into smaller and smaller hyperrectangles.

`xtol_rel` Stop when an optimization step (or an estimate of the optimum) changes every parameter by less than `xtol_rel` multiplied by the absolute value of the parameter. Criterion is disabled if `xtol_rel` is non-positive. Defaults to `1e-5`.

`ftol_rel` Stop when an optimization step (or an estimate of the optimum) changes the objective function value by less than `ftol_rel` multiplied by the absolute value of the function value. Criterion is disabled if `ftol_rel` is non-positive. Defaults to `1e-8`.

`maxeval` Stop when the number of function evaluations exceeds `maxeval`. Criterion is disabled if `maxeval` is non-positive. Defaults to `1000`. See below.

A full description of all options is shown by `nloptr.print.options()` in package `nloptr`.

Value

A list of control parameters for the function `nloptr`.

Examples

```
crt.minimax.control(optslist = list(maxeval = 2000))
```

FIM_2par_exp_censor1 *Fisher Information Matrix for a 2-Parameter Cox Proportional-Hazards Model for Type One Censored Data*

Description

It provides the cpp function for the FIM introduced in Eq. (3.1) of Schmidt and Schwabe (2015) for type one censored data.

Usage

```
FIM_2par_exp_censor1(x, w, param, tcensor)
```

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(\beta_0, \beta_1)$.
tcensor	The experiment is terminated at the fixed time point tcensor.

Value

Fisher information matrix.

References

Schmidt, D., & Schwabe, R. (2015). On optimal designs for censored data. *Metrika*, 78(3), 237-257.

FIM_2par_exp_censor2 *Fisher Information Matrix for a 2-Parameter Cox Proportional-Hazards Model for Random Censored Data*

Description

It provides the cpp function for the FIM introduced in Eq. (3.1) of Schmidt and Schwabe (2015) for random censored data (type two censored data).

Usage

```
FIM_2par_exp_censor2(x, w, param, tcensor)
```

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(\beta_0, \beta_1)$.
tcensor	The experiment is terminated at the fixed time point tcensor.

Value

Fisher information matrix.

References

Schmidt, D., & Schwabe, R. (2015). On optimal designs for censored data. *Metrika*, 78(3), 237-257.

FIM_3par_exp_censor1 *Fisher Information Matrix for a 3-Parameter Cox Proportional-Hazards Model for Type One Censored Data*

Description

It provides the cpp function for the FIM introduced in Page 247 of Schmidt and Schwabe (2015) for type one censored data.

Usage

```
FIM_3par_exp_censor1(x, w, param, tcensor)
```

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(\beta_0, \beta_1, \beta_2)$.
tcensor	The experiment is terminated at the fixed time point tcensor.

Value

Fisher information matrix.

References

Schmidt, D., & Schwabe, R. (2015). On optimal designs for censored data. *Metrika*, 78(3), 237-257.

FIM_3par_exp_censor2 *Fisher Information Matrix for a 3-Parameter Cox Proportional-Hazards Model for Random Censored Data*

Description

It provides the cpp function for the FIM introduced in Page 247 of Schmidt and Schwabe (2015) for random censored data (type two censored data).

Usage

```
FIM_3par_exp_censor2(x, w, param, tcensor)
```

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $(\beta_0, \beta_1, \beta_2)$.
tcensor	The experiment is terminated at the fixed time point tcensor.

Value

Fisher information matrix.

References

Schmidt, D., & Schwabe, R. (2015). On optimal designs for censored data. *Metrika*, 78(3), 237-257.

FIM_exp_2par *Fisher Information Matrix for the 2-Parameter Exponential Model*

Description

It provides the cpp function for FIM for the model $\sim a + \exp(-b \cdot x)$.

Usage

```
FIM_exp_2par(x, w, param)
```

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(a, b)$.

Details

The FIM does not depend on the value of a .

Value

Fisher information matrix.

References

Dette, H., & Neugebauer, H. M. (1997). Bayesian D-optimal designs for exponential regression models. *Journal of Statistical Planning and Inference*, 60(2), 331-349.

Examples

```
FIM_exp_2par(x = c(1, 2), w = c(.5, .5), param = c(3, 4))
```

FIM_logistic

Fisher Information Matrix for the 2-Parameter Logistic (2PL) Model

Description

It provides the cpp function for FIM for the model $\sim 1/(1 + \exp(-b * (x - a)))$. In item response theory (IRT), a is the item difficulty parameter, b is the item discrimination parameter and x is the person ability parameter.

Usage

```
FIM_logistic(x, w, param)
```

Arguments

<code>x</code>	Vector of design points.
<code>w</code>	Vector of design weight. Its length must be equal to the length of <code>x</code> and $\text{sum}(w) = 1$.
<code>param</code>	Vector of values for the model parameters $c(a, b)$.

Details

It can be shown that minimax and standardized D-optimal designs for the 2PL model is symmetric around point $a_M = (a^L + a^U)/2$ where a^L and a^U are the lower bound and upper bound for parameter a , respectively. In [ICA.control](#), arguments `sym` and `sym_point` can be used to specify a_M and find accurate symmetric optimal designs.

Value

Fisher information matrix.

Examples

```
FIM_logistic(x = c(1, 2), w = c(.5, .5), param = c(2, 1))
```

FIM_logistic_2pred *Fisher Information Matrix for the Logistic Model with Two Predictors*

Description

It provides the cpp function for FIM for the following model:
 $\sim \exp(b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_1 * x_2) / (1 + \exp(b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_1 * x_2))$.

Usage

FIM_logistic_2pred(x1, x2, w, param)

Arguments

x1	Vector of design points (for first predictor).
x2	Vector of design points (for second predictor).
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(b_0, b_1, b_2, b_3)$.

Value

Fisher information matrix.

FIM_logistic_4par *Fisher Information Matrix for the 4-Parameter Logistic Model*

Description

It provides the cpp function for the FIM for the model $\sim \text{theta}_1 / (1 + \exp(\text{theta}_2 * x + \text{theta}_3)) + \text{theta}_4$.
 This model is another re-parameterization of the 4-parameter Hill model. For more details, see Eq. (1) and (2) in Hyun and Wong (2015).

Usage

FIM_logistic_4par(x, w, param)

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(\text{theta}_1, \text{theta}_2, \text{theta}_3, \text{theta}_4)$.

Details

The fisher information matrix does not depend on theta4.

Value

Fisher information matrix.

References

Hyun, S. W., & Wong, W. K. (2015). Multiple-Objective Optimal Designs for Studying the Dose Response Function and Interesting Dose Levels. *The international journal of biostatistics*, 11(2), 253-271.

See Also

[multiple](#)

Examples

```
FIM_logistic_4par(x = c(-6.9, -4.6, -3.9, 6.7 ),
                 w = c(0.489, 0.40, 0.061, 0.050),
                 param = c(1.563, 1.790, 8.442, 0.137))
```

FIM_loglin

Fisher Information Matrix for the Mixed Inhibition Model

Description

It provides the cpp function for the FIM for the model $\sim \text{theta0} + \text{theta1} * \log(x + \text{theta2})$.

Usage

```
FIM_loglin(x, w, param)
```

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(\text{theta0}, \text{theta1}, \text{theta2})$.

Details

The FIM of this model does not depend on the parameter theta0.

Value

Fisher information matrix.

References

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

FIM_mixed_inhibition @title Fisher Information Matrix for the Mixed Inhibition Model.

Description

It provides the cpp function for FIM for the model $\sim V \cdot S / (K_m \cdot (1 + I/K_{ic}) + S \cdot (1 + I/K_{iu}))$

Usage

```
FIM_mixed_inhibition(S, I, w, param)
```

Arguments

S	Vector of S component of design points. S is the substrate concentration.
I	Vector of I component of design points. I is the inhibitor concentration.
w	Vector of design weight. Its length must be equal to the length of S and I, besides $\text{sum}(w) = 1$.
param	Vector of values for the model parameters $c(V, K_m, K_{ic}, K_{iu})$.

Details

The optimal design does not depend on parameter V .

Value

Fisher information matrix of design.

References

Bogacka, B., Patan, M., Johnson, P. J., Youdim, K., & Atkinson, A. C. (2011). Optimum design of experiments for enzyme inhibition kinetic models. *Journal of biopharmaceutical statistics*, 21(3), 555-572.

Examples

```
FIM_mixed_inhibition(S = c(30, 3.86, 30, 4.60),
                    I = c(0, 0, 5.11, 4.16), w = rep(.25, 4),
                    param = c(1.5, 5.2, 3.4, 5.6))
```

FIM_power_logistic *Fisher Information Matrix for the Power Logistic Model*

Description

It provides the cpp function for FIM for the model $\sim 1/(1 + \exp(-b * (x - a)))^s$, but when s is fixed (a two by two matrix).

Usage

```
FIM_power_logistic(x, w, param, s)
```

Arguments

x	Vector of design points.
w	Vector of design weight. Its length must be equal to the length of x and $\text{sum}(w) = 1$.
param	Vector of values for the model parameters c(a, b).
s	parameter s.

Value

Fisher information matrix.

Note

This matrix is a two by two matrix and not equal to the Fisher information matrix for the power logistic model when the derivative is taken with respect to all the three parameters. This matrix is only given to be used in some illustrative examples.

ICA.control *ICA Control Optimization Parameters*

Description

The function ICA.control returns a list of ICA control parameters.

Usage

```
ICA.control(ncount = 40, nimp = ncount/10, assim_coeff = 4,
  revol_rate = 0.3, damp = 0.99, uniting_threshold = 0.02,
  equal_weight = FALSE, sym = FALSE, sym_point = NULL,
  stop_rule = c("maxiter", "equivalence"), stoptol = 0.99, checkfreq = 0,
  plot_cost = TRUE, plot_sens = TRUE, plot_3d = c("lattice", "rgl"),
  trace = TRUE, rseed = NULL)
```


Arguments

<code>ncount</code>	Number of countries. Defaults to 40.
<code>nimp</code>	Number of imperialists. Defaults to 10 percent of <code>ncount</code> .
<code>assim_coeff</code>	Assimilation coefficient. Defaults to 4.
<code>revol_rate</code>	Revolution rate. Defaults to 0.3.
<code>damp</code>	Damp ratio for revolution rate. <code>revol_rate</code> is decreased in every iteration by <code>damp</code> . Defaults to 0.99.
<code>uniting_threshold</code>	If the distance between two imperialists is less than the product of the uniting threshold by the largest distance in the search space, ICA unites the empires. Defaults to 0.02.
<code>equal_weight</code>	Should the weights of design points assumed to be equal? Defaults to FALSE. If TRUE, it reduces the dimension of the search space and produces a design that gives equal weight to all of its support points.
<code>sym</code>	Should the design points be symmetric around <code>sym_point</code> ? Defaults to FALSE. When TRUE, <code>sym_point</code> must be given.
<code>sym_point</code>	If <code>sym = TRUE</code> , the design points will be symmetric around <code>sym_point</code> . See 'Details'.
<code>stop_rule</code>	Either 'maxiter' or 'equivalence'. Denotes the type of stopping rule. See 'Details'. Defaults to 'maxiter'.
<code>stoptol</code>	If <code>stop_rule = 'equivalence'</code> , algorithm stops when ELB is larger than <code>stoptol</code> . Defaults to 0.99.
<code>checkfreq</code>	The algorithm checks the general equivalence theorem. It plots the sensitivity or derivative function and calculates the efficiency lower bound ELB, if <code>plot_sens == TRUE</code> , every <code>checkfreq</code> iterations. When <code>checkfreq = 0</code> , no check will be made. When <code>checkfreq = Inf</code> , only the output design will be verified. Defaults to 0.
<code>plot_cost</code>	Plot the iteration (evolution) of algorithm? Defaults to TRUE.
<code>plot_sens</code>	Plot the sensitivity (derivative) function at every <code>checkfreq</code> . Defaults to TRUE.
<code>plot_3d</code>	Character. Which package should be used to plot the sensitivity plot for models with two explanatory variables?
<code>trace</code>	Print the information in every iteration? Defaults to TRUE.
<code>rseed</code>	Random seed. Defaults to NULL.

Details

If `stop_rule = 'maxiter'`, the algorithm iterates until maximum number of iterations.

If `stop_rule = 'equivalence'`, the algorithm stops when either ELB is greater than `stoptol` or reaches `maxiter`. In this case, you must specify the check frequency by `checkfreq`. Note that checking equivalence theorem is a very time consuming process, especially for Bayesian and minimax problems. We advise using this option only for locally, multiple objective and robust optimal designs.

Assume the 2PL model of the form $P(Y = 1) = \frac{1}{1 + \exp(-b(x-a))}$ and let the parameters a and b belong to $[a_L, a_U]$ and $[b_L, b_U]$, respectively. It can be shown that the optimal design for this model is symmetric about $a_M = \frac{a_L + a_U}{2}$. For this model, to find accurate symmetric designs, one can set `sym = TRUE` and provide the value of the a_M via `sym_point`. In this case, the output design will be symmetric around the point `sym_point`. The length of `sym_point` must be equal to the number of model predictors, here is 1.

Value

A list of ICA control parameters.

Examples

```
ICA.control(ncount = 100)
```

ICAOD

ICAOD: Finding Optimal Designs for Nonlinear Models

Description

Different functions are available to find optimal designs for nonlinear models. The user should choose either of them based on her/his strategy to deal with the unknown model parameters: :

- **locally**: finds locally D-optimal designs. A vector of initial estimates or guess is available for the vector of model parameters.
- **robust**: finds robust optimal designs or optimal designs in average. Some (weighted) vector of initial estimates are available for the vector of unknown model parameters (discrete prior).
- **bayes**: finds Bayesian D-optimal designs. A continuous prior is available for the vector of unknown model parameters.
- **minimax**: finds minimax and standardized maximin D-optimal designs. Each of the unknown model parameters belong to an user-specified interval. The purpose is to find a design that protects the user against the worst scenario over the parameter space. Standardized designs should be used when locally optimal design of the model of interest has analytical solution.

There are also some functions to find optimal designs for special problems or models:

- **multiple**: finds locally multiple objective optimal designs for the 4-parameter Hill model. It uses the same strategy as the function `locally` to deal with the unknown model parameters.
- **bayescomp**: finds a design that that meets the dual goal of the parameter estimation and increasing the probability of a particular outcome in a binary response model. It uses the same strategy as the function `bayes` to deal with the unknown mode parameters.

Details

The optimization of the outer problem (over the design space) is done using a metaheuristic algorithm called imperialist competitive algorithm (ICA). The user can always adjust its tuning parameters through the function `ICA.control`. Based on our experience, the most important parameter here is the number of countries, equivalent to the number of particles in the PSO algorithm and that can be regulated via the argument `ncount`.

Depending on the strategy dealing with the unknown model parameters, each type of optimal design problem may have an inner problem. Given a known design (support points and weights), the inner problems is:

- **locally**: equivalent to a simple function (criterion) evaluation at the vector of initial estimates of the vector of model parameters.
- **robust**: equivalent to a weighted some of function (criterion) evaluations at the vectors of initial estimates of the model parameters.
- **bayes**: equivalent to an integral approximation over the multiple prior distribution of the unknown model parameters (by function `hcubature`). The integration tuning parameters can be adjusted by the function `crt.bayes.control`. The most important ones are `maxEval` and `tol`.
- **bayes**: equivalent to an optimization problem over the parameter space that is the cartesian product of some intervals for the model unknown parameters (by function `nloptr`). The optimization tuning parameters can be regulated by the function `crt.minimax.control`. The most important tuning parameter is `maxeval`. The convergence of the algorithm for minimax and standardized maximin designs requires solving global multi-optima optimization problems over the parameter space. Therefore, the parameter `maxeval` plays a very important role to avoid getting trapped in the local optima of the inner problem.

The functions `locally` and `robust` are very easy to be applied and they are usually fast. The speed of the functions `bayes` and `minimax` considerably depends on the value of the tuning parameters.

Because the output design in this package is always an **approximate** or **continuous** design, the general equivalence theorem may be used to assess the proximity of the design to the true-optimal design based on derivative plot and efficiency lower bound.

For each type of design, the user can use the following functions to verify the optimality of her/his output design:

- `senslocally`
- `sensrobust`
- `sensbayes`
- `sensminimax`
- `sensmultiple`
- `sensbayescomp`

References

Masoudi E, Holling H, Wong W.K. (2017). Application of Imperialist Competitive Algorithm to Find Minimax and Standardized Maximin Optimal Designs. *Computational Statistics and Data Analysis*, 113, 330-345. <doi:10.1016/j.csda.2016.06.014>

iterate *Updating an Object of Class 'bayes' or 'minimax'*

Description

Runs the ICA optimization algorithm on an object of class 'bayes' or 'minimax' for more number of iterations and updates the results.

Usage

```
iterate(object, iter)
```

Arguments

object	An object of class 'minimax' or 'Bayesian'.
iter	Number of iterations.

Value

An (updated) object of class 'bayes' or 'minimax'.

See Also

[iterate.minimax](#) and [iterate.bayes](#).

iterate.bayes *Updating an Object of Class bayes*

Description

Runs the ICA optimization algorithm on an object of class bayes for more number of iterations and updates the results.

Usage

```
## S3 method for class 'bayes'  
iterate(object, iter)
```

Arguments

object	An object of class bayes.
iter	Number of iterations.

See Also

[bayes](#)

iterate.minimax	<i>Updating an Object of Class minimax</i>
-----------------	--

Description

Runs the ICA optimization algorithm on an object of class `minimax` for more number of iterations and updates the results.

Usage

```
## S3 method for class 'minimax'  
iterate(object, iter)
```

Arguments

<code>object</code>	An object of class <code>minimax</code> .
<code>iter</code>	Number of iterations.

See Also

[minimax](#)

locally	<i>Locally D-Optimal Designs</i>
---------	----------------------------------

Description

Finds locally D-optimal designs for nonlinear models. It should be used when initial estimates for the unknown parameters are available from, say, a pilot study or previous research. Locally optimal designs may not be efficient when the initial estimates are far away from the true values of the parameters.

Usage

```
locally(formula, predvars, parvars, family = gaussian(), lx, ux, iter, k,  
        inipars, fimfunc = NULL, ICA.control = list(),  
        sens.minimax.control = list(), initial = NULL, npar = length(inipars),  
        plot_3d = c("lattice", "rgl"))
```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.
lx	Vector of lower bounds for the predictors. Should be in the same order as <code>predvars</code> .
ux	Vector of upper bounds for the predictors. Should be in the same order as <code>predvars</code> .
iter	Maximum number of iterations.
k	Number of design points. Must be at least equal to the number of model parameters to avoid singularity of the FIM.
inipars	Vector. Initial values for the unknown parameters. Must match <code>parvars</code> or argument <code>param</code> of the function provided in <code>fimfunc</code> .
fimfunc	A function. Returns the FIM as a matrix. Required when <code>formula</code> is missing. See 'Details' of minimax .
ICA.control	ICA control parameters. For details, see ICA.control .
sens.minimax.control	Control parameters to verify the general equivalence theorem. For details, see the function sens.minimax.control .
initial	A matrix of the initial designs that will be used as initial solutions (countries). Every row is a design, i.e. a concatenation of <code>x</code> and <code>w</code> . Will be coerced to a matrix if necessary. See 'Details' of minimax .
npar	Number of model parameters. Used when <code>fimfunc</code> is given instead of <code>formula</code> to specify the number of model parameters. If not given, the sensitivity plot may be shifted below the y-axis. When NULL, it will be set here to <code>length(inipars)</code> .
plot_3d	Which package should be used to plot the sensitivity (derivative) function for two-dimensional design space. Defaults to "lattice".

Details

Let $M(\xi, \theta_0)$ be the Fisher information matrix (FIM) of a k -point design ξ and θ_0 be the initial user-specified estimates for the unknown parameters.

A locally D-optimal design ξ^* minimizes over Ξ

$$-\log |M(\xi, \theta_0)|.$$

One can adjust the tuning parameters in [ICA.control](#) to set a stopping rule based on the general equivalence theorem. See "Examples".

Value

an object of class `minimax` that is a list including three sub-lists:

`arg` A list of design and algorithm parameters.

`evol` A list of length equal to the number of iterations that stores the information about the best design (design with least criterion value) of each iteration. `evol[[iter]]` contains:

<code>iter</code>	Iteration number.
<code>x</code>	Design points.
<code>w</code>	Design weights.
<code>min_cost</code>	Cost (criterion value) of the best imperialist in iteration <code>iter</code> .
<code>mean_cost</code>	Mean of costs of all imperialists in iteration <code>iter</code> .
<code>sens</code>	An object of class <code>'sensminimax'</code> . See below.
<code>param</code>	Vector of parameters.

`empires` A list of all empires of the last iteration.

`alg` A list with following information:

<code>nfeval</code>	Number of function evaluations. See below.
<code>nlocal</code>	Number of successful local search.
<code>nrevol</code>	Number of successful revolutions.
<code>nimprove</code>	Number of successful movements toward the imperialists in assimilation step.
<code>convergence</code>	Stopped by <code>'maxiter'</code> or <code>'equivalence'</code> ?

`sens` contains information about the design verification by the general equivalence theorem. See `sensminimax` for more details. It is only available every `ICA.control$checkfreq` iterations and the last iteration if `ICA.control$checkfreq >= 0`. Otherwise, `NULL`.

`param` is a vector of parameters that is the global minimum of the minimax criterion or the global maximum of the standardized maximin criterion over the parameter space, given `x`, `w`.

`nfeval` does not count the function evaluations from checking the general equivalence theorem.

References

Masoudi E, Holling H, Wong W.K. (2017). Application of Imperialist Competitive Algorithm to Find Minimax and Standardized Maximin Optimal Designs. *Computational Statistics and Data Analysis*, 113, 330-345.

See Also

[senslocally](#)

Examples

```
#####
# Exponential growth model
```

```
#####
# See how we set stopping rule by adjusting 'stop_rule', 'checkfreq' and 'stoptol'
# It calls the 'senslocally' function every checkfreq = 50 iterations to
# calculate the ELB. if ELB is greater than stoptol = .95, then the algorithm stops.

# initializing by one iteration
res1 <- locally(formula = ~a + exp(-b*x), predvars = "x", parvars = c("a", "b"),
               lx = 0, ux = 1, inipars = c(1, 10),
               iter = 1, k = 2,
               ICA.control= ICA.control(rseed = 100,
                                       stop_rule = "equivalence",
                                       checkfreq = 20, stoptol = .95))

## Not run:
# update the algorithm
res1 <- iterate(res1, 150)
#stops at iteration 21 because ELB is greater than .95

## End(Not run)

plot(res1)

#####
## two parameter logistic model
#####
res2 <- locally(formula = ~1/(1 + exp(-b *(x - a))),
               predvars = "x", parvars = c("a", "b"),
               family = binomial(), lx = -3, ux = 3,
               inipars = c(1, 3), iter = 1, k = 2,
               ICA.control= list(rseed = 100, stop_rule = "equivalence",
                                checkfreq = 50, stoptol = .95))

## Not run:
res2 <- iterate(res2, 100)
# stops at iteration 51

## End(Not run)

#####
# A model with two predictors
#####
# mixed inhibition model
## Not run:
res3 <- locally(formula = ~ V*S/(Km * (1 + I/Kic)+ S * (1 + I/Kiu)),
               predvars = c("S", "I"),
               parvars = c("V", "Km", "Kic", "Kiu"),
               family = gaussian(),
               lx = c(0, 0), ux = c(30, 60),
               k = 4,
               iter = 300,
```



```

inipars = c(1.5, 5.2, 3.4, 5.6),
ICA.control= list(rseed = 100, stop_rule = "equivalence",
                  checkfreq = 50, stoptol = .95))

# stops at iteration 100

## End(Not run)

```

minimax

Minimax and Standardized Maximin D-Optimal Designs

Description

Finds minimax and standardized maximin D-optimal designs for nonlinear models. It should be used when the user assumes each of the unknown parameters belongs to an uncertainty interval (or parameter interval) and the purpose is to **protect the experiment from the worst scenario** over the space of the Cartesian product of the given uncertainty intervals, denoted by Θ . Sometimes, Θ is called 'region of uncertainty', that is the parameter space.

If you are looking for a design that is **in average** robust with respect to the parameter space, then you should apply Bayesian optimal designs (continuous parameter space) or robust designs (discrete parameter space). See, functions [bayes](#) and [robust](#).

In general, although standardized criteria have some good theoretical features, we recommend applying them **only** when the user can provide a closed-form for locally D-optimal designs for the model of interest by the argument `localdes`.

Usage

```

minimax(formula, predvars, parvars, family = gaussian(), lx, ux, lp, up, iter,
        k, n.grid = 0, fimfunc = NULL, ICA.control = list(),
        sens.minimax.control = list(), crt.minimax.control = list(),
        standardized = FALSE, initial = NULL, localdes = NULL,
        npar = length(lp), plot_3d = c("lattice", "rgl"))

```

Arguments

<code>formula</code>	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
<code>predvars</code>	A vector of characters. Denotes the predictors in the formula .
<code>parvars</code>	A vector of characters. Denotes the unknown parameters in the formula .
<code>family</code>	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.

lx	Vector of lower bounds for the predictors. Should be in the same order as predvars.
ux	Vector of upper bounds for the predictors. Should be in the same order as predvars.
lp	Vector of lower bounds for the model parameters. Should be in the same order as parvars or param in the argument fimfunc.
up	Vector of upper bounds for the model parameters. Should be in the same order as parvars or param in the argument fimfunc. When a parameter is known (has a fixed value), set its associated lower and upper bounds in lp and up equal to the known value.
iter	Maximum number of iterations.
k	Number of design points. Must be at least equal to the number of model parameters to avoid singularity of the FIM.
n.grid	The total number of grid points from the parameter space is $n.grid^p$. Used to map the parameter space to a discrete grid. Disabled when $n.grid = 0$ (default). When set, optimal design protects the experimenter against the worst scenario over the grid points. See 'Details' of minimax .
fimfunc	A function. Returns the FIM as a matrix. Required when formula is missing. See 'Details' of minimax .
ICA.control	ICA control parameters. For details, see ICA.control .
sens.minimax.control	Control parameters to verify the general equivalence theorem. For details, see the function sens.minimax.control .
crt.minimax.control	Control parameters to optimize the minimax or standardized maximin criterion at a given design over a continuous parameter space (when $n.grid = 0$). For details, see the function crt.minimax.control .
standardized	Maximin standardized design? When <code>standardized = TRUE</code> , the argument <code>localdes</code> must be given. Defaults to <code>FALSE</code> . See 'Details' of minimax .
initial	A matrix of the initial designs that will be used as initial solutions (countries). Every row is a design, i.e. a concatenation of x and w. Will be coerced to a matrix if necessary. See 'Details' of minimax .
localdes	A function that takes the parameter values as inputs and returns the design points and weights of the locally optimal design. Required when <code>standardized = "TRUE"</code> . See 'Details' of minimax .
npar	Number of model parameters. Used when <code>fimfunc</code> is given instead of <code>formula</code> to specify the number of model parameters. If not specified truly, the sensitivity (derivative) plot may be shifted below the y-axis. When <code>NULL</code> , it will be set here to <code>length(lp)</code> .
plot_3d	Which package should be used to plot the sensitivity (derivative) function for two-dimensional design space. Defaults to <code>"lattice"</code> .

Details

Let Ξ be the space of all approximate designs with k design points (support points) at x_1, x_2, \dots, x_k from design space χ with corresponding weights w_1, \dots, w_k . Let $M(\xi, \theta)$ be the Fisher information matrix (FIM) of a k -point design ξ and θ be the vector of unknown parameters.

A minimax D-optimal design ξ^* minimizes over Ξ

$$\max_{\theta \in \Theta} -\log |M(\xi, \theta)|.$$

A standardized maximin D-optimal design ξ^* maximizes over Ξ

$$\inf_{\theta \in \Theta} \left[\left(\frac{|M(\xi, \theta)|}{|M(\xi_\theta, \theta)|} \right)^{\frac{1}{p}} \right],$$

where p is the number of model parameters and ξ_θ is the locally D-optimal design with respect to θ .

A minimax criterion (cost function or objective function) is evaluated at each design (decision variables) by maximizing the criterion over the parameter space. We call the optimization problem over the parameter space as *inner optimization problem*. Here, two separate strategies may be applied to solve the inner problem at a given design (design points and weights):

1. **Continuous inner problem:** optimizes the criterion over a continuous parameter space. We use function `nloptr` for this purpose. The tuning parameters can be regulated via the argument `crt.minimax.control`. The most important control parameter is `maxeval` that requires especial attention. The output design here protects the experiment from the worst scenario over the continuous parameter space.
2. **Discrete inner problem:** maps the parameter space to some grid points and optimizes the criterion over a discrete parameter space (the inner optimization problem becomes a discrete space). In this case, the number of grid points can be regulated via `n.grid`. This strategy is quite efficient (and fast) when the maxima most likely attain the vertices of the continuous parameter space at any given design. The output design here protects the experiment from the worst scenario over the grid points.

The formula is used to create the Fisher information matrix (FIM) for a nonlinear model when the distribution of the response variable belongs to the natural exponential family. User can also apply the argument `fimfunc` to provide the FIM her/himself, directly. In this case, argument `fimfunc` takes a function that has three arguments as follows:

1. `x` a vector of design points. For design points with more than one dimension, it is a concatenation of the design points, but **dimension-wise**. For example, let the model has three predictors (I, S, Z) . Then, (three-dimensional) design points of a two-point optimal design are $\{\text{point1} = (I_1, S_1, Z_1), \text{point2} = (I_2, S_2, Z_2)\}$. Then, the argument `x` is equivalent to `x = c(I1, I2, S1, S2, Z1, Z2)`.
2. `w` a vector that includes the design weights associated with `x`.
3. `param` a vector of parameter values associated with `lp` and `up`.

The output must be the Fisher information matrix with number of rows equal to `length(param)`. See 'Examples'.

Minimax optimal designs can have very different criterion values depending on the nominal set of parameter values. Accordingly, it is desirable to standardize the criterion and control for the potentially widely varying magnitude of the criterion (Dette, 1997). Evaluating a standardized maximin criterion requires knowing locally optimal designs. We strongly advise setting `standardized = 'TRUE'`, only when analytical solutions for the locally D-optimal designs is available. When `standardized = 'TRUE'`, the user must provide the closed-form of the locally optimal design (support points x and weights w) as a function of the model parameters.

`localdes` takes a function with the following arguments.

- If `formula` is given (`!missing(formula)`):
 - The parameter names given by `parvars` in the same order.
- FIM is given by the argument `fimfunc` (`missing(formula)`):
 - `param`: A vector of the parameters same as the argument `param` in `fimfunc`.

The function must return a list with components x and w (they match the same arguments in the function `fimfunc`). See 'Examples'.

The standardized D-criterion is equal to the D-efficiency and it must be between 0 and 1, theoretically. However, in practice, it may take values larger than one because the function given via `localdes` does not return the true (accurate) locally optimal designs for some requested parameter values from Θ . In this case, the function `minimax` stops and returns a set of values for the model parameters. The user here must investigate the accuracy of the locally D-optimal designs by checking the output of the function set via the argument `localdes` given this set of parameter values.

Each row of `initial` is one design, i.e. a concatenation of values for design (support) points and the associated design weights. Let x_0 and w_0 be the vector of initial values with exactly the same length and order as x and w (the arguments of `fimfunc`). As an example, the first row of the matrix `initial` is equal to `initial[1,] = c(x0, w0)`. For models with more than one predictors, x_0 is a concatenation of the initial values for design points, but **dimension-wise**. See the details of the argument `fimfunc`, above.

To verify the optimality of the output design by the general equivalence theorem, the user can either plot the results or set `checkfreq` in `ICA.control` to `Inf`. In either case, the function `sensminimax` is called for verification. Note that the function `sensminimax` always verifies the optimality of a design assuming a continuous parameter space. See 'Examples'.

Value

an object of class `minimax` that is a list including three sub-lists:

`arg` A list of design and algorithm parameters.

`evol` A list of length equal to the number of iterations that stores the information about the best design (design with least criterion value) of each iteration. `evol[[iter]]` contains:

<code>iter</code>	Iteration number.
<code>x</code>	Design points.
<code>w</code>	Design weights.
<code>min_cost</code>	Cost (criterion value) of the best imperialist in iteration <code>iter</code> .
<code>mean_cost</code>	Mean of costs of all imperialists in iteration <code>iter</code> .
<code>sens</code>	An object of class 'sensminimax'. See below.
<code>param</code>	Vector of parameters.

empires A list of all empires of the last iteration.

alg A list with following information:

nfeval	Number of function evaluations. See below.
nlocal	Number of successful local search.
nrevol	Number of successful revolutions.
nimprove	Number of successful movements toward the imperialists in assimilation step.
convergence	Stopped by 'maxiter' or 'equivalence'?

sens contains information about the design verification by the general equivalence theorem. See sensminimax for more details. It is only available every `ICA.control$checkfreq` iterations and the last iteration if `ICA.control$checkfreq >= 0`. Otherwise, NULL.

param is a vector of parameters that is the global minimum of the minimax criterion or the global maximum of the standardized maximin criterion over the parameter space, given x , w .

nfeval does not count the function evaluations from checking the general equivalence theorem.

Note

For larger parameter space or model with more number of unknown parameters, it is always important to increase the value of `ncount` in `ICA.control` and `optslist$maxeval` in `crt.minimax.control` to produce very accurate designs.

References

Masoudi E, Holling H, Wong W.K. (2017). Application of Imperialist Competitive Algorithm to Find Minimax and Standardized Maximin Optimal Designs. Computational Statistics and Data Analysis, 113, 330-345.

Dette, H. (1997). Designing experiments with respect to 'standardized' optimality criteria. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 59(1), 97-110.

See Also

[sensminimax](#)

Examples

```
#####
# Two-parameter exponential growth model
#####
res1 <- minimax (formula = ~a + exp(-b*x), predvars = "x", parvars = c("a", "b"),
               lx = 0, ux = 1, lp = c(1, 1), up = c(1, 10),
               iter = 1, k = 4, ICA.control= ICA.control(rseed = 100))

## Not run:
res1 <- iterate(res1, 150)
# iterating the algorithm up to 150 more iterations

## End(Not run)
```

```

res1 # print method
plot(res1) # verifying the general equivalence theorem

#####
# Two-parameter logistic model.
#####
# A little bit tickling with the tuning parameters
# reducing the value of maxeval to 200 to increase the speed
cont1 <- crt.minimax.control(optslist = list(maxeval = 500))
cont2 <- ICA.control(rseed = 100, checkfreq = Inf, ncount = 60)

## Not run:
res2 <- minimax (formula = ~1/(1 + exp(-b *(x - a))), predvars = "x",
                 parvars = c("a", "b"),
                 family = binomial(), lx = -3, ux = 3,
                 lp = c(0, 1), up = c(1, 2.5), iter = 200, k = 3,
                 ICA.control= cont2, crt.minimax.control = cont1)

print(res2)
plot(res2)

## End(Not run)

#####
# An example of a model with two predictors
#####
# Mixed inhibition model
lower <- c(1, 4, 2, 4)
upper <- c(1, 5, 3, 5)
cont <- crt.minimax.control(optslist = list(maxeval = 100)) # to be faster
## Not run:
res3 <- minimax(formula = ~ V*S/(Km * (1 + I/Kic)+ S * (1 + I/Kiu)),
                predvars = c("S", "I"),
                parvars = c("V", "Km", "Kic", "Kiu"),
                lx = c(0, 0), ux = c(30, 60), k = 4,
                iter = 1, lp = lower, up = upper,
                ICA.control= list(rseed = 100),
                crt.minimax.control = cont)

res3 <- iterate(res3, 100)
print(res3)
plot(res3) # sensitivity plot

## End(Not run)

# Now consider grid points instead of assuming continuous parameter space
# set n.grid to 5
## Not run:
res4 <- minimax(formula = ~ V*S/(Km * (1 + I/Kic)+ S * (1 + I/Kiu)),
                predvars = c("S", "I"),
                parvars = c("V", "Km", "Kic", "Kiu"),
                lx = c(0, 0), ux = c(30, 60),

```

```

k = 4, iter = 130, n.grid = 5, lp = lower, up = upper,
ICA.control= list(rseed = 100, checkfreq = Inf),
crt.minimax.control = cont)

print(res4)
plot(res4) # sensitivity plot

## End(Not run)

#####
# Standardized maximin D-optimal designs
#####
# Now assume the purpose is finding STANDARDIZED designs
# We know from the literature that the locally D-optimal design (LDOD)
# for this model has analytical solution.
# The following function takes the parameter as input and returns
# the design points and weights of LDOD.
# x and w are exactly similar to the arguments of 'fimfunc'.
# x is a vector and returns the design points 'dimension-wise'.
# see explanation of the arguments of 'fimfunc' in 'Details'.

LDOD <- function(V, Km, Kic, Kiu){
  #first dimension is for S and the second one is for I.
  S_min <- 0
  S_max <- 30
  I_min <- 0
  I_max <- 60
  s2 <- max(S_min, S_max*Km*Kiu*(Kic+I_min)/
            (S_max*Kic*I_min+S_max*Kic*Kiu+2*Km*Kiu*I_min+2*Km*Kiu*Kic))
  i3 <- min((2*S_max*Kic*I_min + S_max*Kic*Kiu+2*Km*Kiu*I_min+Km*Kiu*Kic)/
            (Km*Kiu+S_max*Kic), I_max)
  i4 <- min(I_min + (sqrt((Kic+I_min)*(Km*Kic*Kiu+Km*Kiu*I_min+
            S_max*Kic*Kiu+S_max*Kic*I_min)/
            (Km*Kiu+S_max*Kic))), I_max )
  s4 <- max(-Km*Kiu*(Kic+2*I_min-i4)/(Kic*(Kiu+2*I_min-i4)), S_min)
  x <- c(S_max, s2, S_max, s4, I_min, I_min, i3, i4)
  return(list(x = x, w =rep(1/4, 4)))
}
formalArgs(LDOD)
## Not run:
minimax(formula = ~ V*S/(Km * (1 + I/Kic)+ S * (1 + I/Kiu)),
predvars = c("S", "I"),
parvars = c("V", "Km", "Kic", "Kiu"),
lx = c(0, 0), ux = c(30, 60),
k = 4, iter = 300,
lp = lower, up = upper,
ICA.control= list(rseed = 100, checkfreq = Inf),
crt.minimax.control = cont,
standardized = TRUE,
localdes = LDOD)

## End(Not run)

```

```
#####
# Not necessary!
# The rest of the examples here are only for professional uses.
#####
# Imagine you have written your own FIM, say in Rcpp that is faster than
# the FIM created by the formula interface above.

#####
# An example of a model with two predictors
#####
# For example, th cpp FIM function for the mixed inhibition model is named:
formalArgs(FIM_mixed_inhibition)

# We should reparamterize the arguments to match the standard of the
# argument 'fimfunc' (see 'Details').
myfim <- function(x, w, param){
  npoint <- length(x)/2
  S <- x[1:npoint]
  I <- x[(npoint+1):(npoint*2)]
  out <- FIM_mixed_inhibition(S = S, I = I, w = w, param = param)
  return(out)
}
formalArgs(myfim)

# Finds minimax optimal design, exactly as before, but NOT using the
# formula interface.
## Not run:
res5 <- minimax(fimfunc = myfim,
                lx = c(0, 0), ux = c(30, 60), k = 4,
                iter = 100, lp = lower, up = upper,
                ICA.control= list(rseed = 100),
                crt.minimax.control = cont)

print(res5)
plot(res5) # sensitivity plot

## End(Not run)
#####
# Standardized maximin D-optimal designs
#####
# To match the argument 'localdes' when no formula inteface is used,
# we should reparameterize LDOD.
# The input must be 'param' same as the argument of 'fimfunc'
LDOD2 <- function(param)
  LDOD(V = param[1], Km = param[2], Kic = param[3], Kiu = param[4])

# compare these two:
formalArgs(LDOD)
formalArgs(LDOD2)
## Not run:
res6 <- minimax(fimfunc = myfim,
                lx = c(0, 0), ux = c(30, 60), k = 4,
                iter = 300, lp = lower, up = upper,
```



```

ICA.control= list(rseed = 100, checkfreq = Inf),
crt.minimax.control = cont,
standardized = TRUE,
localdes = LDOD2)

res6
plot(res6)

## End(Not run)

```

multiple	<i>Locally Multiple Objective Optimal Designs for the 4-Parameter Hill Model</i>
----------	--

Description

The 4-parameter Hill model is of the form

$$f(D) = c + \frac{(d - c)\left(\frac{D}{a}\right)^b}{1 + \left(\frac{D}{a}\right)^b} + \epsilon,$$

where $\epsilon \sim N(0, \sigma^2)$, D is the dose level and the predictor, a is the ED50, d is the upper limit of response, c is the lower limit of response and b denotes the Hill constant that control the flexibility in the slope of the response curve.

Sometimes, the Hill model is re-parameterized and written as

$$f(x) = \frac{\theta_1}{1 + \exp(\theta_2 x + \theta_3)} + \theta_4,$$

where $\theta_1 = d - c$, $\theta_2 = -b$, $\theta_3 = b \log(a)$, $\theta_4 = c$, $\theta_1 > 0$, $\theta_2 \neq 0$, and $-\infty < ED50 < \infty$, where $x = \log(D) \in [-M, M]$ for some sufficiently large value of M . The new form is sometimes referred to as **4-parameter logistic model**.

The function `multiple` finds locally multiple-objective optimal designs for estimating the model parameters, the ED50, and the MED, simultaneously. For more details, see Hyun and Wong (2015).

Usage

```

multiple(minDose, maxDose, iter, k, inipars, Hill_par = FALSE, delta, lambda,
  fimfunc = NULL, ICA.control = list(), sens.minimax.control = list(),
  initial = NULL, tol = sqrt(.Machine$double.xmin))

```

Arguments

minDose	Minimum dose D . For the 4-parameter logistic model, i.e. when <code>Hill_par = FALSE</code> , it is the minimum of $\log(D)$.
maxDose	Maximum dose D . For the 4-parameter logistic model, i.e. when <code>Hill_par = FALSE</code> , it is the maximum of $\log(D)$.

<code>iter</code>	Maximum number of iterations.
<code>k</code>	Number of design points. Must be at least equal to the number of model parameters to avoid singularity of the FIM.
<code>inipars</code>	A vector of initial estimates for the vector of parameters (a, b, c, d) . For the 4-parameter logistic model, i.e. when <code>Hill_par = FALSE</code> , it is a vector of initial estimates for $(\theta_1, \theta_2, \theta_3, \theta_4)$.
<code>Hill_par</code>	Hill model parameterization? Defaults to TRUE.
<code>delta</code>	Predetermined meaningful value of the minimum effective dose MED. When $\delta < 0$, then $\theta_2 > 0$ or when $\delta > 0$, then $\theta_2 < 0$.
<code>lambda</code>	A vector of relative importance of each of the three criteria, i.e. $\lambda = (\lambda_1, \lambda_2, \lambda_3)$. Here $0 < \lambda_i < 1$ and $s \sum \lambda_i = 1$.
<code>fimfunc</code>	A function. Returns the FIM as a matrix. Required when formula is missing. See 'Details' of <code>minimax</code> .
<code>ICA.control</code>	ICA control parameters. For details, see <code>ICA.control</code> .
<code>sens.minimax.control</code>	Control parameters to verify the general equivalence theorem. For details, see the function <code>sens.minimax.control</code> .
<code>initial</code>	A matrix of the initial designs that will be used as initial solutions (countries). Every row is a design, i.e. a concatenation of <code>x</code> and <code>w</code> . Will be coerced to a matrix if necessary. See 'Details' of <code>minimax</code> .
<code>tol</code>	Tolerance for finding the general inverse of the Fisher information matrix. Defaults to <code>.Machine\$double.xmin</code> .

Details

When $\lambda_1 > 0$, then the number of support points `k` must at least be four to avoid singularity of the Fisher information matrix.

One can adjust the tuning parameters in `ICA.control` to set a stopping rule based on the general equivalence theorem. See 'Examples'.

Value

an object of class `minimax` that is a list including three sub-lists:

`arg` A list of design and algorithm parameters.

`evol` A list of length equal to the number of iterations that stores the information about the best design (design with least criterion value) of each iteration. `evol[[iter]]` contains:

<code>iter</code>	Iteration number.
<code>x</code>	Design points.
<code>w</code>	Design weights.
<code>min_cost</code>	Cost (criterion value) of the best imperialist in iteration <code>iter</code> .
<code>mean_cost</code>	Mean of costs of all imperialists in iteration <code>iter</code> .
<code>sens</code>	An object of class 'sensminimax'. See below.
<code>param</code>	Vector of parameters.

empires A list of all empires of the last iteration.

alg A list with following information:

nfeval	Number of function evaluations. See below.
nlocal	Number of successful local search.
nrevol	Number of successful revolutions.
nimprove	Number of successful movements toward the imperialists in assimilation step.
convergence	Stopped by 'maxiter' or 'equivalence'?

sens contains information about the design verification by the general equivalence theorem. See sensminimax for more details. It is only available every `ICA.control$checkfreq` iterations and the last iteration if `ICA.control$checkfreq >= 0`. Otherwise, NULL.

param is a vector of parameters that is the global minimum of the minimax criterion or the global maximum of the standardized maximin criterion over the parameter space, given x, w .

nfeval does not count the function evaluations from checking the general equivalence theorem.

Note

DO NOT use this function for finding c-optimal designs for estimating 'MED' or 'ED50' (single objective optimal designs) because the results may be not be stable. The reason is that for the c-optimal criterion the generalized inverse of the Fisher information matrix is not stable and depends on the tolerance value (tol).

References

Hyun, S. W., and Wong, W. K. (2015). Multiple-Objective Optimal Designs for Studying the Dose Response Function and Interesting Dose Levels. The international journal of biostatistics, 11(2), 253-271.

See Also

[sensmultiple](#)

Examples

```
# All th examples here are available in Hyun and Wong (2015)

#####
# 4-parameter logistic model
# Example 1, Table 3
#####
lam <- c(0.05, 0.05, .90)
# Initial estimates are derived from Table 1
# See how the stopping rules are set via 'stop_rul', 'checkfreq' and 'stoptol'
Theta1 <- c(1.563, 1.790, 8.442, 0.137)
res1 <- multiple(minDose = log(.001), maxDose = log(1000),
                inipars = Theta1, k = 4, lambda = lam, delta = -1,
                iter = 1,
```

```

ICA.control = list(rseed = 1366, ncount = 100,
                  stop_rule = "equivalence",
                  checkfreq = 100, stoptol = .95))

## Not run:
res1 <- iterate(res1, 1000)
# stops at iteration 101

## End(Not run)

#####
# 4-parameter Hill model
#####
## initial estimates for the parameters of Hill model:
a <- 0.008949 # ED50
b <- -1.79 # Hill constant
c <- 0.137 # lower limit
d <- 1.7 # upper limit
# D belongs to c(.001, 1000) ## dose in mg
## the vector of Hill parameters are now c(a, b, c, d)
## Not run:
res2 <- multiple(minDose = .001, maxDose = 1000,
                inipars = c(a, b, c, d),
                Hill_par = TRUE, k = 4, lambda = lam,
                delta = -1, iter = 1000,
                ICA.control = list(rseed = 1366, ncount = 100,
                                  stop_rule = "equivalence",
                                  checkfreq = 100, stoptol = .95))

# stops at iteration 100

## End(Not run)

```

normal

Multivariate Normal Prior Distribution for Model Parameters

Description

Creates a multivariate normal prior distribution for the unknown parameters as an object of class cprior.

Usage

```
normal(mu, sigma, lower, upper)
```

Arguments

mu	A vector of length of parameters, representing the mean value.
sigma	A symmetric positive-definite matrix representing the variance-covariance matrix of the distribution.
lower	A vector of lower bounds for the unknown parameters.
upper	A vector of upper bounds for the unknown parameters.

Value

An object of class `cprior` that is a list with components:

- `fn`: prior distribution as an R function with argument `param` that is the vector of the unknown parameters. See below.
- `npar`: Number of unknown parameters and is equal to the length of `param`.
- `lower`: Argument `lower`. It has the same length as `param`.
- `upper`: Argument `lower`. It has the same length as `param`.

The list will be passed to the argument `prior` of the function `bayes`. The order of the argument `param` in `fn` has the same order as the argument `parvars` when the model is specified by a formula. Otherwise, it is the same as the argument `param` in the function `fimfunc`.

See Also

[bayes](#) [sensbayes](#)

Examples

```
normal(mu = c(0, 1), sigma = matrix(c(1, -0.17, -0.17, .5), nrow = 2),
      lower = c(-3, .1), upper = c(3, 2))
```

plot.bayes

Plotting bayes Objects

Description

This function plots the evolution of the algorithm till iteration number `iter` iteration and re-checks the general equivalence theorem by plotting the sensitivity function and calculating the ELB.

Usage

```
## S3 method for class 'bayes'
plot(x, iter = NULL, sensitivity = TRUE,
     calculate_criterion = FALSE, sens.bayes.control = list(),
     crt.bayes.control = list(), silent = FALSE, plot_3d = c("lattice",
     "rgl"), evolution = FALSE, ...)
```

Arguments

<code>x</code>	An object of class <code>bayes</code> .
<code>iter</code>	Iteration number. if NULL, will be set to last iteration.
<code>sensitivity</code>	Logical. If TRUE, the general equivalence theorem is used to check the optimality if the best design in iteration number <code>iter</code> and the sensitivity plot will be plotted.
<code>calculate_criterion</code>	Re-calculate the criterion value? It only assumes a continuous parameter space for the minimax and standardized maximin designs. Defaults to FALSE. See 'Details'.
<code>sens.bayes.control</code>	Control parameters to verify general equivalence theorem. For details, see sens.bayes.control .
<code>crt.bayes.control</code>	Control parameters to approximate the integration in the Bayesian criterion at a given design. For details, see crt.bayes.control .
<code>silent</code>	Do not print anything? Defaults to FALSE.
<code>plot_3d</code>	Which package should be used to plot the sensitivity function for two-dimensional design space. Defaults to <code>plot_3d = "lattice"</code> . Only applicable when <code>sensitivity = TRUE</code> .
<code>evolution</code>	Plot Evolution? Defaults to FALSE.
<code>...</code>	Argument with no further use.

Details

The criterion value can also be re-calculated for the output designs using new set of tuning parameters in the function [crt.bayes.control](#). This is useful for Bayesian optimal designs to assess the robustness of the criterion value with respect to different values of the tuning parameters. To put it simple, for these designs, the user can re-calculate the criterion value (approximate the integration given an output design in a Bayesian problem) with different values for `maxEval` and `tol` in [crt.bayes.control](#) to be sure that the function `hcubature` approximates the integrals to an acceptable accuracy using the default values (or a new user-given values, in case it has been reset) of `maxEval` and `tol`.

See Also

[bayes](#), [bayescomp](#)

plot.minimax

Plotting minimax Objects

Description

This function plots the evolution of the algorithm till iteration number `iter` iteration and re-checks the general equivalence theorem by plotting the sensitivity function and calculating the ELB.

Usage

```
## S3 method for class 'minimax'
plot(x, iter = NULL, sensitivity = TRUE,
     calculate_criterion = FALSE, sens.minimax.control = list(),
     crt.minimax.control = list(), silent = FALSE, plot_3d = c("lattice",
     "rgl"), evolution = FALSE, ...)
```

Arguments

x	An object of class <code>minimax</code> .
iter	Iteration number. if <code>NULL</code> , will be set to last iteration.
sensitivity	Logical. If <code>TRUE</code> , the general equivalence theorem is used to check the optimality if the best design in iteration number <code>iter</code> and the sensitivity plot will be plotted.
calculate_criterion	Re-calculate the criterion value? It only assumes a continuous parameter space for the minimax and standardized maximin designs. Defaults to <code>FALSE</code> . See 'Details'.
sens.minimax.control	Control parameters to verify general equivalence theorem. For details, see sens.minimax.control .
crt.minimax.control	Control parameters to optimize the minimax or standardized maximin criterion at a given design over a continuous parameter space. For details, see crt.minimax.control .
silent	Do not print anything? Defaults to <code>FALSE</code> .
plot_3d	Which package should be used to plot the sensitivity function for two-dimensional design space. Defaults to <code>plot_3d = "lattice"</code> . Only applicable when <code>sensitivity = TRUE</code> .
evolution	Plot Evolution? Defaults to <code>FALSE</code> .
...	Argument with no further use.

Details

The criterion value can also be re-calculated for the output designs using new set of tuning parameters in the function [crt.minimax.control](#). This is useful for minimax and standardized maximin optimal designs to assess the robustness of the criterion value with respect to different values of the tuning parameters. To put it simple, for these designs, the user can re-calculate the criterion value (finds the global maximum over the parameter space given an output design in a minimax problem) with larger values for `maxeval` in [crt.minimax.control](#) to be sure that the function `nloptr` finds global optima of the inner optimization problem (over the parameter space) using the default value (or a new user-given value, in case it has been reset) of `maxeval`. If increasing the value of `maxeval` gives different criterion values, then the results can not be trusted and should be repeated with a higher value for `maxeval`.

See Also

[minimax](#), [locally](#), [robust](#)

`print.bayes` *Printing bayes Objects*

Description

Print method for an object of class bayes.

Usage

```
## S3 method for class 'bayes'  
print(x, iter = NULL, ...)
```

Arguments

<code>x</code>	an object of class bayes.
<code>iter</code>	Iteration number. if NULL, will be set to last iteration.
<code>...</code>	Argument with no further use.

See Also

[bayes](#)

`print.minimax` *Printing minimax Objects*

Description

Print method for an object of class minimax.

Usage

```
## S3 method for class 'minimax'  
print(x, iter = NULL, ...)
```

Arguments

<code>x</code>	An object of class minimax.
<code>iter</code>	Iteration number. if NULL, will be set equal to last iteration.
<code>...</code>	Argument with no further use.

See Also

[minimax](#), [locally](#), [robust](#)

print.sensbayes *Printing sensbayes Objects*

Description

Print method for an object of class sensbayes.

Usage

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

Arguments

x An object of class sensbayes.
... Argument with no further use.

See Also

[sensbayes](#), [sensbayescomp](#)

print.sensminimax *Printing sensminimax Objects*

Description

Print method for an object of class sensminimax.

Usage

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

Arguments

x An object of class sensminimax.
... Argument with no further use.

See Also

[sensminimax](#), [senslocally](#), [sensrobust](#)

robust

*Robust D-Optimal Designs***Description**

Finds Robust designs or optimal designs in average for nonlinear models. It is useful when a set of initial estimates for the unknown model parameters along with a discrete probability measure on that set is available.

Usage

```
robust(formula, predvars, parvars, family = gaussian(), lx, ux, iter, k, prob,
       parset, fimfunc = NULL, ICA.control = list(),
       sens.minimax.control = list(), initial = NULL, npar = dim(parset)[2],
       plot_3d = c("lattice", "rgl"))
```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.
lx	Vector of lower bounds for the predictors. Should be in the same order as <code>predvars</code> .
ux	Vector of upper bounds for the predictors. Should be in the same order as <code>predvars</code> .
iter	Maximum number of iterations.
k	Number of design points. Must be at least equal to the number of model parameters to avoid singularity of the FIM.
prob	A vector of the probability measure π associated with each row of <code>parset</code> .
parset	A matrix to provide the set of initial estimates for the model parameters, i.e. support of π . Every row is one vector of initial estimates for the model parameters (<code>nrow(parset) == length(prob)</code>). See 'Details'.
fimfunc	A function. Returns the FIM as a matrix. Required when <code>formula</code> is missing. See 'Details' of minimax .
ICA.control	ICA control parameters. For details, see ICA.control .

<code>sens.minimax.control</code>	Control parameters to verify the general equivalence theorem. For details, see the function <code>sens.minimax.control</code> .
<code>initial</code>	A matrix of the initial designs that will be used as initial solutions (countries). Every row is a design, i.e. a concatenation of <code>x</code> and <code>w</code> . Will be coerced to a matrix if necessary. See 'Details' of <code>minimax</code> .
<code>npar</code>	Number of model parameters. Used when <code>fimfunc</code> is given instead of <code>formula</code> to specify the number of model parameters. If not given, the sensitivity plot may be shifted below the <code>y</code> -axis. When <code>NULL</code> , it will be set to <code>dim(parset)[2]</code> .
<code>plot_3d</code>	Which package should be used to plot the sensitivity (derivative) function for two-dimensional design space. Defaults to "lattice".

Details

Let Θ be a set of initial estimates for the unknown parameters. A robust criterion is evaluated at the elements of Θ weighted by a probability measure π as follows:

$$B(\xi, \pi) = \int_{\Theta} |M(\xi, \theta)| \pi(\theta) d\theta.$$

A robust design ξ^* maximizes $B(\xi, \pi)$ over the space of all designs.

When the model is given via `formula`, columns of `parset` must match the parameters introduced in `parvars`. Otherwise, when the model is introduced via `fimfunc`, columns of `parset` must match the argument `param` in `fimfunc`.

To verify the optimality of the output design by the general equivalence theorem, the user can either plot the results or set `checkfreq` in `ICA.control` to `Inf`. In either case, the function `sensrobust` is called for verification. One can also adjust the tuning parameters in `ICA.control` to set a stopping rule based on the general equivalence theorem. See 'Examples'.

Value

an object of class `minimax` that is a list including three sub-lists:

`arg` A list of design and algorithm parameters.

`evol` A list of length equal to the number of iterations that stores the information about the best design (design with least criterion value) of each iteration. `evol[[iter]]` contains:

<code>iter</code>	Iteration number.
<code>x</code>	Design points.
<code>w</code>	Design weights.
<code>min_cost</code>	Cost (criterion value) of the best imperialist in iteration <code>iter</code> .
<code>mean_cost</code>	Mean of costs of all imperialists in iteration <code>iter</code> .
<code>sens</code>	An object of class 'sensminimax'. See below.
<code>param</code>	Vector of parameters.

`empires` A list of all empires of the last iteration.

`alg` A list with following information:

nfeval	Number of function evaluations. See below.
nlocal	Number of successful local search.
nrevol	Number of successful revolutions.
nimprove	Number of successful movements toward the imperialists in assimilation step.
convergence	Stopped by 'maxiter' or 'equivalence'?

sens contains information about the design verification by the general equivalence theorem. See sensminimax for more details. It is only available every ICA.control\$checkfreq iterations and the last iteration if ICA.control\$checkfreq >= 0. Otherwise, NULL.

param is a vector of parameters that is the global minimum of the minimax criterion or the global maximum of the standardized maximin criterion over the parameter space, given x, w.

nfeval does not count the function evaluations from checking the general equivalence theorem.

Note

When a continuous prior distribution for the unknown model parameters is available, use [bayes](#). When only one initial estimates of the unknown model parameters is available (Θ has only one element), use [locally](#).

See Also

[bayes](#) [sensrobust](#)

Examples

```
# Finding a robust design for the two-parameter logistic model
# See how we set a stopping rule.
# The ELB is computed every checkfreq = 30 iterations
# The optimization stops when the ELB is larger than stoptol = .95
res <- robust(formula = ~1/(1 + exp(-b *(x - a))),
              predvars = c("x"), parvars = c("a", "b"),
              family = binomial(),
              lx = -5, ux = 5, prob = rep(1/4, 4),
              parset = matrix(c(0.5, 1.5, 0.5, 1.5, 4.0, 4.0, 5.0, 5.0), 4, 2),
              iter = 1, k = 3,
              ICA.control = list(stop_rule = "equivalence",
                                stoptol = .95, checkfreq = 30))

## Not run:
res <- iterate(res, 100)
# stops at iteration 51

## End(Not run)
```

sens.bayes.control *Control Parameters for Verifying General Equivalence Theorem for Bayesian Designs*

Description

The function `sens.bayes.control` returns a list of `hcubature` control parameters for approximating the integrals in the sensitivity (derivative) function of Bayesian criteria and also `nloptr` control parameters to find maximum of the sensitivity (derivative) function over the design space and calculate the efficiency lower bound (ELB).

Usage

```
sens.bayes.control(cubature = list(tol = 1e-06, maxEval = 1e+05, absError =
  0), x0 = NULL, optslst = list(stopval = -Inf, algorithm =
  "NLOPT_GN_DIRECT_L", xtol_rel = 1e-08, ftol_rel = 1e-10, maxeval = 2000), ...)
```

Arguments

<code>cubature</code>	A list that will be passed to the arguments of the <code>hcubature</code> function. See 'Details'.
<code>x0</code>	Vector of starting values for maximizing the sensitivity (derivative) function over the design space x . It will be passed to the optimization function <code>nloptr</code> .
<code>optslst</code>	A list will be passed to <code>opts</code> argument of the function <code>nloptr</code> to find the maximum of the sensitivity function over the design space. See 'Details'.
<code>...</code>	Further arguments will be passed to <code>nl.opts</code> from package <code>nloptr</code> .

Details

`cubature` is a list that its components will be passed to the function `hcubature`. Its components are:

`tol` The maximum tolerance. Defaults to $1e-6$.

`maxEval` The maximum number of function evaluations needed. Note that the actual number of function evaluations performed is only approximately guaranteed not to exceed this number. Defaults to 100000 .

`absError` The maximum absolute error tolerated. Defaults to 0 .

ELB is a measure of proximity of a design to the optimal design without knowing the latter. Given a design, let ϵ be the global maximum of the sensitivity (derivative) function with respect the vector of the model predictors x over the design space. ELB is given by

$$ELB = p/(p + \epsilon),$$

where p is the number of model parameters. Obviously, calculating ELB requires finding ϵ and therefore, a maximization problem to be solved. The function `nloptr` is used here to solve this maximization problem. The arguments `x0` and `optslst` will be passed to this function as follows:

Argument `x0` provides the user initial values for this maximization problem and will be passed to the argument with the same name of the function `nloptr`.

Argument `optslist` will be passed to the argument `opts` of the function `nloptr`. `optslist` is a list and the most important components are listed as follows:

`stopval` Stop minimization when an objective value \leq `stopval` is found. Setting `stopval` to `-Inf` disables this stopping criterion (default).

`algorithm` Defaults to `NLOPT_GN_DIRECT_L`. `DIRECT-L` is a deterministic-search algorithm based on systematic division of the search domain into smaller and smaller hyperrectangles.

`xtol_rel` Stop when an optimization step (or an estimate of the optimum) changes every parameter by less than `xtol_rel` multiplied by the absolute value of the parameter. Criterion is disabled if `xtol_rel` is non-positive. Defaults to `1e-8`.

`ftol_rel` Stop when an optimization step (or an estimate of the optimum) changes the objective function value by less than `ftol_rel` multiplied by the absolute value of the function value. Criterion is disabled if `ftol_rel` is non-positive. Defaults to `1e-10`.

`maxeval` Stop when the number of function evaluations exceeds `maxeval`. Criterion is disabled if `maxeval` is non-positive. Defaults to `6000`. See below.

A full description of all options is shown by the function `nloptr.print.options()` in package `nloptr`.

Value

A list of control parameters for verifying the general equivalence theorem with respect to the Bayesian optimality criteria.

Note

When the value of `ELB` is larger than 1, it means the maximum found by the optimization function set by `algorithm` is not global. In this case, please increase the value of the parameter `maxeval` to find the global maximum of the sensitivity (derivative) function and avoid false `ELB`.

Examples

```
sens.bayes.control()
sens.bayes.control(cubature = list(maxEval = 50000))
sens.bayes.control(optslist = list(maxeval = 3000))
```

`sens.minimax.control` *Control Parameters for Verifying General Equivalence Theorem*

Description

It returns a list of control parameters required for verifying the general equivalence theorem that is plotting the sensitivity (derivative) function of the optimality criterion at a given approximate (continuous) design and also calculating the efficiency lower bound (`ELB`).

Usage

```
sens.minimax.control(answering.set = list(n_seg = 6, merge_tol = 0.005),
  x0 = NULL, optslst = list(stopval = -Inf, algorithm =
  "NLOPT_GN_DIRECT_L", xtol_rel = 1e-08, ftol_rel = 1e-10, maxeval = 6000), ...)
```

Arguments

answering.set	A list of control parameters to find the answering set in minimax and standardized maximin optimal design problems. See 'Details'.
x0	Vector of starting values for maximizing the sensitivity (derivative) function over the design space x . It will be passed to the optimization function <code>nloptr</code> .
optslst	A list. It will be passed to the argument <code>opts</code> of the function <code>nloptr</code> to find the maximum of the sensitivity function over the design space. See 'Details'.
...	Further arguments will be passed to <code>nl.opts</code> from package <code>nloptr</code> .

Details

Given a design, answering set is a subset of all local optima of the optimality criterion over the parameter space (only in minimax and standardized maximin problems). Answering set (and its associated measure) is the basis of the creation of the sensitivity (derivative) functions in minimax and standardized maximin problems. That means, calculating the ELB or plotting the sensitivity (derivative) function requires a known answering set. Unfortunately, there is no theoretical rule on how to choose the number of elements of the answering set and they would have to be found by trial and error. Given a design, we find the answering set in two steps:

- Step 1: Find all local maxima of the minimax criterion (or local minima of the standardized maximin criterion) over the parameter space. Here, we divide the parameter space into $(n_seg + 1)^p$ segments (p is the number of unknown model parameters) and use the endpoints as initial values for local searches directed by function `optim` ("L-BFGS-B" method).
- Step 2: Pick the ones nearest to the global minimum (or maximum in standardized maximin problems) subject to a merging tolerance `merge_tol` (default 0.005).

Obviously, the answering set is a subset of all local maxima (or local minima in standardized maximin problems), given a design. Therefore, it is very important to be able to find all the local maxima (or the local minima) to create the true answering set with no missing elements. Otherwise, even when the design is optimal, the sensitivity (derivative) plot may not verify the optimality of the design.

Please increase the value of `n_seg` for models with large number of unknown parameters or large parameter space.

We advise not changing the default value of the parameter `merge_tol` as it has been tested for many examples. Finding all local optimal of a function is not an easy task and is the main reason that checking general equivalence theorem (even plotting) in minimax and standardized maximin problems is very time-consuming.

ELB is a measure of proximity of a design to the optimal design without knowing the latter. Given a design, let ϵ be the global maximum of the sensitivity (derivative) function with respect the vector of the model predictors x over the design space. ELB is given by

$$ELB = p/(p + \epsilon),$$

where p is the number of model parameters. Obviously, calculating ELB requires finding ϵ and therefore, a maximization problem to be solved. The function `nloptr` is used here to solve this maximization problem. The arguments `x0` and `optslist` will be passed to this function as follows:

Argument `x0` provides the user initial values for this maximization problem and will be passed to the argument with the same name of the function `nloptr`.

Argument `optslist` will be passed to the argument `opts` of the function `nloptr`. `optslist` is a list and the most important components are listed as follows:

`stopval` Stop minimization when an objective value \leq `stopval` is found. Setting `stopval` to `-Inf` disables this stopping criterion (default).

`algorithm` Defaults to `NLOPT_GN_DIRECT_L`. `DIRECT-L` is a deterministic-search algorithm based on systematic division of the search domain into smaller and smaller hyperrectangles.

`xtol_rel` Stop when an optimization step (or an estimate of the optimum) changes every parameter by less than `xtol_rel` multiplied by the absolute value of the parameter. Criterion is disabled if `xtol_rel` is non-positive. Defaults to `1e-8`.

`ftol_rel` Stop when an optimization step (or an estimate of the optimum) changes the objective function value by less than `ftol_rel` multiplied by the absolute value of the function value. Criterion is disabled if `ftol_rel` is non-positive. Defaults to `1e-10`.

`maxeval` Stop when the number of function evaluations exceeds `maxeval`. Criterion is disabled if `maxeval` is non-positive. Defaults to `6000`. See below.

A full description of all options is shown by the function `nloptr.print.options()` in package `nloptr`.

Value

A list of control parameters for verifying the general equivalence theorem.

Note

When the value of ELB is larger than 1, it means the maximum found by the optimization function set by `algorithm` is not global. In this case, please increase the value of the parameter `maxeval` to find the global maximum of the derivative function over the design space and avoid false ELB.

Examples

```
sens.minimax.control()
sens.minimax.control(answering.set = list(n_seg = 4))
sens.minimax.control(answering.set = list(n_seg = 4), optslist = list(maxeval = 1000))
# faster checking process
sens.minimax.control(answering.set = list(n_seg = 4), optslist = list(maxeval = 2000))
```


Description

This function plot the sensitivity (derivative) function given an approximate (continuous) design and calculate the efficiency lower bound (ELB) for Bayesian D-optimal designs. Let \boldsymbol{x} belongs to χ that denotes the design space. Based on the general equivalence theorem, generally, a design ξ^* is optimal if and only if the value of its sensitivity (derivative) function be non-positive for all \boldsymbol{x} in χ and it only reaches zero when \boldsymbol{x} belong to the support of ξ^* (be equal to one of the design point). Therefore, the user can look at the sensitivity plot and the ELB and decide whether the design is optimal or close enough to the true optimal design (ELB tells us that without knowing the latter).

Usage

```
sensbayes(formula, predvars, parvars, family = gaussian(), x, w, lx, ux,
  fimfunc = NULL, prior = list(), sens.bayes.control = list(),
  crt.bayes.control = list(), plot_3d = c("lattice", "rgl"),
  plot_sens = TRUE, npar = NULL, calculate_criterion = TRUE,
  silent = FALSE)
```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.
x	Vector of design (support) points. See 'Details' of sensminimax .
w	Vector of corresponding design weights for x.
lx	Vector of lower bounds for the predictors. Should be in the same order as predvars.
ux	Vector of upper bounds for the predictors. Should be in the same order as predvars.
fimfunc	A function. Returns the FIM as a matrix. Required when formula is missing. See 'Details' of minimax .
prior	An object of class <code>cprior</code> . User can also use one of the functions uniform , normal , skewnormal or student to create the prior. See 'Details' of bayes .

<code>sens.bayes.control</code>	Control parameters to verify the general equivalence theorem. For details, see sens.bayes.control .
<code>crt.bayes.control</code>	Control parameters to approximate the integral in Bayesian criterion at a given design over the parameter space. For details, see crt.bayes.control .
<code>plot_3d</code>	Which package should be used to plot the sensitivity (derivative) function for two-dimensional design space. Defaults to "lattice".
<code>plot_sens</code>	Plot the sensitivity (derivative) function? Defaults to TRUE.
<code>npar</code>	Number of model parameters. Used when <code>fimfunc</code> is given instead of formula to specify the number of model parameters. If not specified truly, the sensitivity (derivative) plot may be shifted below the y-axis. When NULL, it will be set to <code>length(parvars)</code> or <code>prior\$npar</code> when <code>missing(formula)</code> .
<code>calculate_criterion</code>	Evaluate the D-criterion? See 'Details' of sensminimax .
<code>silent</code>	Do not print anything? Defaults to FALSE.

Details

Let Ξ be the space of all approximate designs with k design points (support points) at x_1, x_2, \dots, x_k from design space χ with corresponding weights w_1, \dots, w_k . Let $M(\xi, \theta)$ be the Fisher information matrix (FIM) of a k -point design ξ and $\pi(\theta)$ is a user-given prior distribution for the vector of unknown parameters θ . A design ξ^* is Bayesian D-optimal among all designs on χ if and only if the following inequality holds for all $\mathbf{x} \in \chi$

$$c(\mathbf{x}, \xi^*) = \int_{\theta \in \Theta} \text{tr} M^{-1}(\xi^*, \theta) I(\mathbf{x}, \theta) - p\pi(\theta) d\theta \leq 0,$$

with equality at all support points of ξ^* . Here, p is the number of model parameters. $c(\mathbf{x}, \xi^*)$ is called **sensitivity** or **derivative** function.

Sometimes, the CPU time can be considerably reduced by choosing less conservative values for the tuning parameters `tol` and `maxEval` in the function [sens.bayes.control](#). The user should find a trade-off between accuracy and speed for his/her problem. See 'Examples'.

Note

Having accurate plots for the sensitivity (derivative) function and calculating ELB to a high precision is the primary goal here, although the process may take too long (even hours) due to requesting very accurate integral approximations.

Examples

```
#####
# Checking the Bayesian D-optimality of a design for the 2Pl model
#####
skew2 <- skewnormal(xi = c(0, 1), Omega = matrix(c(1, -0.17, -0.17, .5), nrow = 2),
               alpha = c(-1, 0), lower = c(-3, .1), upper = c(3, 2))

## Not run:
sensbayes(formula = ~1/(1 + exp(-b *(x - a))),
```

```

    predvars = "x", parvars = c("a", "b"),
    family = binomial(),
    x= c(-2.50914, -1.16780, -0.36904, 1.29227),
    w =c(0.35767, 0.11032, 0.15621, 0.37580),
    lx = -3, ux = 3,
    prior = skew2)
# took 29 seconds on my system!

## End(Not run)

# It took very long.
# We re-adjust the tuning parameters in sens.bayes.control to be faster
# See how we drastically reduce the maxEval and increase the tolerance
## Not run:
sensbayes(formula = ~1/(1 + exp(-b *(x - a))),
    predvars = "x", parvars = c("a", "b"),
    family = binomial(),
    x= c(-2.50914, -1.16780, -0.36904, 1.29227),
    w =c(0.35767, 0.11032, 0.15621, 0.37580),
    lx = -3, ux = 3,prior = skew2,
    sens.bayes.control = list(cubature = list(tol = 1e-4, maxEval = 300)))
# took 5 Seconds on my system!

## End(Not run)

# Compare it with the following:
sensbayes(formula = ~1/(1 + exp(-b *(x - a))),
    predvars = "x", parvars = c("a", "b"),
    family = binomial(),
    x= c(-2.50914, -1.16780, -0.36904, 1.29227),
    w =c(0.35767, 0.11032, 0.15621, 0.37580),
    lx = -3, ux = 3,prior = skew2,
    sens.bayes.control = list(cubature = list(tol = 1e-4, maxEval = 200)))
# Look at the plot!
# took 3 seconds on my system

#####
# Checking the Bayesian D-optimality of a design for the 4-parameter sigmoid emax model
#####
lb <- c(4, 11, 100, 5)
ub <- c(9, 17, 140, 10)
## Not run:
sensbayes(formula = ~ theta1 + (theta2 - theta1)*(x^theta4)/(x^theta4 + theta3^theta4),
    predvars = c("x"), parvars = c("theta1", "theta2", "theta3", "theta4"),
    x = c(0.78990, 95.66297, 118.42964,147.55809, 500),
    w = c(0.23426, 0.17071, 0.17684, 0.1827, 0.23549),
    lx = .001, ux = 500, prior = uniform(lb, ub))
# took 200 seconds on my system

## End(Not run)

```

```

# Re-adjust the tuning parameters to have it faster
## Not run:
sensbayes(formula = ~ theta1 + (theta2 - theta1)*(x^theta4)/(x^theta4 + theta3^theta4),
  predvars = c("x"), parvars = c("theta1", "theta2", "theta3", "theta4"),
  x = c(0.78990, 95.66297, 118.42964, 147.55809, 500),
  w = c(0.23426, 0.17071, 0.17684, 0.1827, 0.23549),
  lx = .001, ux = 500, prior = uniform(lb, ub),
  sens.bayes.control = list(cubature = list(tol = 1e-3, maxEval = 300)))
# took 4 seconds on my system. See how much it makes difference

## End(Not run)

```

sensbayescomp

Verifying Optimality of Bayesian Compound DP-optimal Designs

Description

This function plot the sensitivity (derivative) function given an approximate (continuous) design and calculate the efficiency lower bound (ELB) for Bayesian D-optimal designs. Let \boldsymbol{x} belongs to χ that denotes the design space. Based on the general equivalence theorem, generally, a design ξ^* is optimal if and only if the value of its sensitivity (derivative) function be non-positive for all \boldsymbol{x} in χ and it only reaches zero when \boldsymbol{x} belong to the support of ξ^* (be equal to one of the design point). Therefore, the user can look at the sensitivity plot and the ELB and decide whether the design is optimal or close enough to the true optimal design (ELB tells us that without knowing the latter).

Usage

```

sensbayescomp(formula, predvars, parvars, family = gaussian(), x, w, lx, ux,
  fimfunc = NULL, prior = list(), prob, alpha,
  sens.bayes.control = list(), crt.bayes.control = list(),
  plot_3d = c("lattice", "rgl"), plot_sens = TRUE, npar = NULL,
  calculate_criterion = TRUE, silent = FALSE)

```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.

x	Vector of design (support) points. See 'Details' of sensminimax .
w	Vector of corresponding design weights for x.
lx	Vector of lower bounds for the predictors. Should be in the same order as predvars.
ux	Vector of upper bounds for the predictors. Should be in the same order as predvars.
fimfunc	A function. Returns the FIM as a matrix. Required when formula is missing. See 'Details' of minimax .
prior	An object of class cprior. User can also use one of the functions uniform , normal , skewnormal or student to create the prior. See 'Details' of bayes .
prob	Either formula or a function. when function, its argument is x and param same as the arguments in fimfunc. prob as a function takes the design points and vector of parameters and returns the probability of success at each design points. See 'Examples'.
alpha	A value between 0 and 1. Compound or combined DP-criterion is the product of the efficiencies of a design with respect to D- and average P- optimality, weighted by alpha.
sens.bayes.control	Control parameters to verify the general equivalence theorem. For details, see sens.bayes.control .
crt.bayes.control	Control parameters to approximate the integral in Bayesian criterion at a given design over the parameter space. For details, see crt.bayes.control .
plot_3d	Which package should be used to plot the sensitivity (derivative) function for two-dimensional design space. Defaults to "lattice".
plot_sens	Plot the sensitivity (derivative) function? Defaults to TRUE.
npar	Number of model parameters. Used when fimfunc is given instead of formula to specify the number of model parameters. If not specified truly, the sensitivity (derivative) plot may be shifted below the y-axis. When NULL, it will be set to length(parvars) or prior\$npar when missing(formula).
calculate_criterion	Evaluate the D-criterion? See 'Details' of sensminimax .
silent	Do not print anything? Defaults to FALSE.

Details

Sometimes, the CPU time can be considerably reduced by choosing less conservative values for the tuning parameters `tol` and `maxEval` in the function [sens.bayes.control](#). The user should find a trade-off between accuracy and speed for his/her problem.

Note

Having accurate plots for the sensitivity (derivative) function and calculating ELB to a high precision is the primary goal here, although the process may take too long (even hours) due to requesting very accurate integral approximations.

See Also[bayescomp](#)**Examples**

```
#####
# Verifying the DP-optimality of a design
# The logistic model with two predictors
#####

# The design points and corresponding weights are as follows:
# Point1    Point2    Point3    Point4    Point5    Point6    Point7
# 0.07410  -0.31953   -1.00000    1.00000   -1.00000    1.00000    0.30193
# -1.00000  1.00000   -1.00000    1.00000   -0.08251   -1.00000    1.00000
# Weight1   Weight2   Weight3   Weight4   Weight5   Weight6   Weight7
# 0.020     0.275     0.224     0.131     0.092     0.156     0.103

# It should be given to the function as two separate vectors:
x1 <- c(0.07409639, -0.3195265, -1, 1, -1, 1, 0.3019317, -1, 1, -1, -0.08251169, -1, 1)
w1 <- c(0.01992863, 0.2745394, 0.2236575, 0.1312331, 0.09161503, 0.1561454, 0.1028811)

p <- c(1, -2, 1, -1)

## Not run:
sensbayescomp(formula = ~exp(b0+b1*x1+b2*x2+b3*x1*x2)/(1+exp(b0+b1*x1+b2*x2+b3*x1*x2)),
  predvars = c("x1", "x2"),
  parvars = c("b0", "b1", "b2", "b3"),
  family = binomial(),
  x = x1, w = w1,
  lx = c(-1, -1), ux = c(1, 1),
  prior = uniform(p - 1.5, p + 1.5),
  prob = ~1-1/(1+exp(b0 + b1 * x1 + b2 * x2 + b3 * x1 * x2)),
  alpha = .5, plot_3d = "rgl",
  sens.bayes.control = list(cubature = list(tol = 1e-3, maxEval = 1000)))

## End(Not run)
```

Description

It plots the sensitivity (derivative) function of the locally D-optimal criterion at a given approximate (continuous) design and also calculates its efficiency lower bound (ELB) with respect to this

optimality criterion. The user can confirm or reject the optimality of a design by looking at the sensitivity plot. ELB shows the proximity of a design to the true optimal design without knowing the latter. See, for more details, Masouadi et al. (2017).

Usage

```
senslocally(formula, predvars, parvars, family = gaussian(), x, w, lx, ux,
  inipars, fimfunc = NULL, sens.minimax.control = list(),
  calculate_criterion = TRUE, plot_3d = c("lattice", "rgl"),
  plot_sens = TRUE, npar = length(inipars), silent = FALSE)
```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model gaussian() is applied.
x	Vector of design (support) points. See 'Details' of sensminimax .
w	Vector of corresponding design weights for x.
lx	Vector of lower bounds for the predictors. Should be in the same order as predvars.
ux	Vector of upper bounds for the predictors. Should be in the same order as predvars.
inipars	Vector. Initial values for the unknown parameters. Must match parvars or argument param of the function provided in fimfunc.
fimfunc	A function. Returns the FIM as a matrix. Required when formula is missing. See 'Details' of minimax .
sens.minimax.control	Control parameters to verify the general equivalence theorem. For details, see the function sens.minimax.control .
calculate_criterion	Evaluate the D-criterion? See 'Details' of sensminimax .
plot_3d	Which package should be used to plot the sensitivity (derivative) function for models with two predictors. Either "rgl" or "lattice" (default).
plot_sens	Plot the sensitivity (derivative) function? Defaults to TRUE.
npar	Number of model parameters. Used when fimfunc is given instead of formula to specify the number of model parameters. If not given, the sensitivity plot may be shifted below the y-axis. When NULL, it will be set here to length(inipars).
silent	Do not print anything? Defaults to FALSE.

Details

Let θ_0 denotes the vector of initial estimates for the unknown parameters. A design ξ^* is locally D-optimal among all designs on χ if and only if the following inequality holds for all $x \in \chi$

$$c(x, \xi^*, \theta_0) = \text{tr}M^{-1}(\xi^*, \theta_0)I(x, \theta_0) - p \leq 0,$$

with equality at all support points of ξ^* . Here, p is the number of model parameters. $c(x, \xi^*, \theta_0)$ is called **sensitivity** or **derivative** function.

ELB is a measure of proximity of a design to the optimal design without knowing the latter. Given a design, let ϵ be the global maximum of the sensitivity (derivative) function over $x \in \chi$. ELB is given by

$$ELB = p/(p + \epsilon),$$

where p is the number of model parameters. Obviously, calculating ELB requires finding ϵ and another optimization problem to be solved. The tuning parameters of this optimization can be regulated via the argument [sens.minimax.control](#). See, for more details, Masoudi et al. (2017).

Value

an object of class `sensminimax` that is a list contains:

`type` Argument type that is required for print methods.

`optima` A matrix that stores all the local optima over the parameter space given the best design of the iteration. The cost (criterion) values are stored in a column named `Criterion_Value`. The last column (`Answering_Set`) shows if the optimum, based on a merging tolerance `merge_tol`, belongs to the answering set (1) or not (0). Only applicable for minimax or standardized maximin designs.

`mu` Probability measure on the answering set. Corresponds to the rows of `optima` when the associated row in column `Answering_Set` is equal to 1. Only applicable for minimax or standardized maximin designs.

`max_deriv` Global maximum of the sensitivity (derivative) function (ϵ in 'Details').

`ELB` D-efficiency lower bound. Can not be larger than 1. See 'Note' in [sensminimax](#) or 'Details' in [sens.minimax.control](#).

`merge_tol` Merging tolerance to create the answering set from the set of all local optima. See 'Details' in [sens.minimax.control](#). Only applicable for minimax or standardized maximin designs.

`crtval` Criterion value. Compare with column `Crtriterion_Value` in `optima` for minimax and standardized maximin designs.

`time` Required CPU time (rough approximation).

Note

Theoretically, ELB can not be larger than 1. But if so, it may have one of the following reasons:

- `max_deriv` is not a GLOBAL maximum. Please increase the value of the parameter `maxeval` in [sens.minimax.control](#) to find the global maximum.
- The sensitivity function is shifted below the y-axis because the number of model parameters has not been specified correctly (less value given). Please specify the correct number of model parameters via argument `npar`.

References

Masoudi E, Holling H, Wong W.K. (2017). Application of Imperialist Competitive Algorithm to Find Minimax and Standardized Maximin Optimal Designs. Computational Statistics and Data Analysis, 113, 330-345.

Examples

```
#####
# Exponential growth model
#####
# Verifying optimailty of a locally D-optimal design
senslocally(formula = ~a + exp(-b*x),
             predvars = "x", parvars = c("a", "b"),
             x = c(.1, 1), w = c(.5, .5),
             lx = 0, ux = 1, inipars = c(1, 10))

#####
# A model with two predictors
#####
x0 <- c(30, 3.861406, 30, 4.600633, 0, 0, 5.111376, 4.168798)
w0 <- rep(.25, 4)
senslocally(formula = ~ V*S/(Km * (1 + I/Kic)+ S * (1 + I/Kiu)),
             predvars = c("S", "I"),
             parvars = c("V", "Km", "Kic", "Kiu"),
             x = x0, w = w0,
             lx = c(0, 0), ux = c(30, 60),
             inipars = c(1.5, 5.2, 3.4, 5.6))

## Not run:
# using package rgl for 3d plot:
res<- senslocally(formula = ~ V*S/(Km * (1 + I/Kic)+ S * (1 + I/Kiu)),
                  predvars = c("S", "I"),
                  parvars = c("V", "Km", "Kic", "Kiu"),
                  x = x0, w = w0,
                  lx = c(0, 0), ux = c(30, 60),
                  inipars = c(1.5, 5.2, 3.4, 5.6),
                  plot_3d = "rgl")

## End(Not run)
```

sensminimax

Verifying Optimality of Minimax and Standardized maximin D-optimal Designs

Description

It plots the sensitivity (derivative) function of the minimax or standardized maximin D-optimal criterion at a given approximate (continuous) design and also calculates its efficiency lower bound

(ELB) with respect to the optimality criterion. The user can confirm or reject the optimality of a design by looking at the sensitivity plot. ELB shows the proximity of a design to the true optimal design without knowing the latter. See, for more details, Masoudi et al. (2017).

Usage

```
sensminimax(formula, predvars, parvars, family = gaussian(), x, w, lx, ux, lp,
  up, fimfunc = NULL, standardized = FALSE, localdes = NULL,
  sens.minimax.control = list(), calculate_criterion = TRUE,
  crt.minimax.control = list(), plot_3d = c("lattice", "rgl"),
  plot_sens = TRUE, npar = length(lp), silent = FALSE)
```

Arguments

formula	A nonlinear model <code>formula</code> . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a <code>formula</code> if necessary.
predvars	A vector of characters. Denotes the predictors in the <code>formula</code> .
parvars	A vector of characters. Denotes the unknown parameters in the <code>formula</code> .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see <code>family</code> . By default, a linear gaussian model <code>gaussian()</code> is applied.
x	Vector of design (support) points. See 'Details' of <code>sensminimax</code> .
w	Vector of corresponding design weights for x.
lx	Vector of lower bounds for the predictors. Should be in the same order as <code>predvars</code> .
ux	Vector of upper bounds for the predictors. Should be in the same order as <code>predvars</code> .
lp	Vector of lower bounds for the model parameters. Should be in the same order as <code>parvars</code> or <code>param</code> in the argument <code>fimfunc</code> .
up	Vector of upper bounds for the model parameters. Should be in the same order as <code>parvars</code> or <code>param</code> in the argument <code>fimfunc</code> . When a parameter is known (has a fixed value), set its associated lower and upper bounds in <code>lp</code> and <code>up</code> equal to the known value.
fimfunc	A function. Returns the FIM as a matrix. Required when <code>formula</code> is missing. See 'Details' of <code>minimax</code> .
standardized	Maximin standardized design? When <code>standardized = TRUE</code> , the argument <code>localdes</code> must be given. Defaults to <code>FALSE</code> . See 'Details' of <code>minimax</code> .
localdes	A function that takes the parameter values as inputs and returns the design points and weights of the locally optimal design. Required when <code>standardized = "TRUE"</code> . See 'Details' of <code>minimax</code> .

sens.minimax.control	Control parameters to verify the general equivalence theorem. For details, see the function sens.minimax.control .
calculate_criterion	Evaluate the D-criterion? See 'Details' of sensminimax .
crt.minimax.control	Control parameters to optimize the minimax or standardized maximin criterion at a given design over a continuous parameter space. For details, see the function crt.minimax.control . Only applicable when calculate_criterion = TRUE.
plot_3d	Which package should be used to plot the sensitivity (derivative) function for models with two predictors. Either "rgl" or "lattice" (default).
plot_sens	Plot the sensitivity (derivative) function? Defaults to TRUE.
npar	Number of model parameters. Used when fimfunc is given instead of formula to specify the number of model parameters. If not specified truly, the sensitivity (derivative) plot may be shifted below the y-axis. When NULL, it will be set here to length(lp).
silent	Do not print anything? Defaults to FALSE.

Details

Let the unknown parameters belong to Θ , that is a Cartesian product of (user-given) intervals for the unknown model parameters. A design ξ^* is minimax D-optimal among all designs on χ if and only if there exists a probability measure μ^* on

$$A(\xi^*) = \left\{ \nu \in \Theta \mid -\log|M(\xi^*, \nu)| = \max_{\theta \in \Theta} -\log|M(\xi^*, \theta)| \right\},$$

such that the following inequality holds for all $\mathbf{x} \in \chi$

$$c(\mathbf{x}, \mu^*, \xi^*) = \int_{A(\xi^*)} \text{tr} M^{-1}(\xi^*, \nu) I(\mathbf{x}, \nu) \mu^* d(\nu) - p \leq 0,$$

with equality at all support points of ξ^* . Here, p is the number of model parameters. $c(\mathbf{x}, \mu^*, \xi^*)$ is called **sensitivity** or **derivative** function. The set $A(\xi^*)$ is sometimes called **answering set** of ξ^* and the measure μ^* is a subgradient of the non-differentiable criterion evaluated at $M(\xi^*, \nu)$.

For standardized maximin D-optimal designs, the answering set $N(\xi^*)$ is

$$N(\xi^*) = \left\{ \nu \in \Theta \mid \text{eff}_D(\xi^*, \nu) = \min_{\theta \in \Theta} \text{eff}_D(\xi^*, \theta) \right\}.$$

where $\text{eff}_D(\xi, \theta) = \left(\frac{|M(\xi, \theta)|}{|M(\xi_\theta, \theta)|} \right)^{\frac{1}{p}}$ and ξ_θ is the locally D-optimal design with respect to θ .

See 'Details' of [sens.minimax.control](#) on how to calculate the answering set $N(\xi^*)$ and $A(\xi^*)$.

The argument \mathbf{x} is the vector of design points. For design points with more than one dimension, it is a concatenation of the design points, but **dimension-wise**. For example, let the model has three predictors (I, S, Z) . Then, (three-dimensional) design points of a two-point optimal design are $\{\text{point1} = (I_1, S_1, Z_1), \text{point2} = (I_2, S_2, Z_2)\}$. Then, the argument \mathbf{x} is equivalent to $\mathbf{x} = c(I1, I2, S1, S2, Z1, Z2)$.

ELB is a measure of proximity of a design to the optimal design without knowing the latter. Given a design, let ϵ be the global maximum of the sensitivity (derivative) function with respect x where $x \in \chi$. ELB is given by

$$ELB = p/(p + \epsilon),$$

where p is the number of model parameters. Obviously, calculating ELB requires finding ϵ and another optimization problem to be solved. The tuning parameters of this optimization can be regulated via the argument `sens.minimax.control`. See, for more details, Masoudi et al. (2017).

The criterion value for the minimax D-optimal design is (global maximum over Θ)

$$\max_{\theta \in \Theta} -\log |M(\xi, \theta)|;$$

for standardized maximin D-optimal design is (global minimum over Θ)

$$\inf_{\theta \in \Theta} \left[\left(\frac{|M(\xi, \theta)|}{|M(\xi_{\theta}, \theta)|} \right)^{\frac{1}{p}} \right].$$

For locally optimal designs, the criterion value is obtained simply by evaluating the criterion at the design (design points and weights) and the initial values for the parameters.

This function confirms the optimality assuming only a continuous parameter space Θ .

Value

an object of class `sensminimax` that is a list contains:

`type` Argument type that is required for print methods.

`optima` A matrix that stores all the local optima over the parameter space given the best design of the iteration. The cost (criterion) values are stored in a column named `Criterion_Value`. The last column (`Answering_Set`) shows if the optimum, based on a merging tolerance `merge_tol`, belongs to the answering set (1) or not (0). Only applicable for minimax or standardized maximin designs.

`mu` Probability measure on the answering set. Corresponds to the rows of `optima` when the associated row in column `Answering_Set` is equal to 1. Only applicable for minimax or standardized maximin designs.

`max_deriv` Global maximum of the sensitivity (derivative) function (ϵ in 'Details').

`ELB` D-efficiency lower bound. Can not be larger than 1. See 'Note' in `sensminimax` or 'Details' in `sens.minimax.control`.

`merge_tol` Merging tolerance to create the answering set from the set of all local optima. See 'Details' in `sens.minimax.control`. Only applicable for minimax or standardized maximin designs.

`crtval` Criterion value. Compare with column `Crtiterion_Value` in `optima` for minimax and standardized maximin designs.

`time` Required CPU time (rough approximation).

Note

Theoretically, ELB can not be larger than 1. But if so, it may have one of the following reasons:

- `max_deriv` is not a GLOBAL maximum. Please increase the value of the parameter `maxeval` in `sens.minimax.control` to find the global maximum.
- The sensitivity function is shifted below the y-axis because the number of model parameters has not been specified correctly (less value given). Please specify the correct number of model parameters via argument `npar`.

Please increase the value of the parameter `n_seg` in `sens.minimax.control` for models with large number of parameters or large parameter space.

References

Masoudi E, Holling H, Wong W.K. (2017). Application of Imperialist Competitive Algorithm to Find Minimax and Standardized Maximin Optimal Designs. *Computational Statistics and Data Analysis*, 113, 330-345.

Examples

```
#####
# Power logistic model
#####
# verifying the minimax D-optimality of a design with points x0 and weights w0
x0 <- c(-4.5515, 0.2130, 2.8075)
w0 <- c(0.4100, 0.3723, 0.2177)
# Power logistic model when s = .2
sensminimax(formula = ~ (1/(1 + exp(-b * (x-a))))^s,
             predvars = "x",
             parvars = c("a", "b"),
             family = binomial(),
             x = x0, w = w0,
             lx = -5, ux = 5,
             lp = c(0, 1), up = c(3, 1.5))

#####
# A model with two predictors
#####
# Verifying the minimax D-optimality of a design for a model with two predictors
# The model is the mixed inhibition model.
# X0 is the vector of four design points that are:
# (3.4614, 0) (4.2801, 3.1426) (30, 0) (30, 4.0373)
x0 <- c(3.4614, 4.2801, 30, 30, 0, 3.1426, 0, 4.0373)
w0 <- rep(1/4, 4)
sensminimax(formula = ~ V*S/(Km * (1 + I/Kic)+ S * (1 + I/Kiu)),
             predvars = c("S", "I"),
             parvars = c("V", "Km", "Kic", "Kiu"),
             family = "gaussian",
             x = x0, w = w0,
             lx = c(0, 0), ux = c(30, 60),
```

```

lp = c(1, 4, 2, 4), up = c(1, 5, 3, 5))

#####
# Standardized maximin D-optimal designs
#####
# Verifying the standardized maximin D-optimality of a design for
# the loglinear model
# First we should define the function for 'localdes' argument
# The function LDOD takes the parameters and returns the points and
# weights of the locally D-optimal design
LDOD <- function(theta0, theta1, theta2){
  ## param is the vector of theta = (theta0, theta1, theta2)
  lx <- 0 # lower bound of the design space
  ux <- 150 # upper bound of the design space
  param <- c()
  param[1] <- theta0
  param[2] <- theta1
  param[3] <- theta2
  xstar <- (ux+param[3]) * (lx + param[3]) *
    (log(ux + param[3]) - log(lx + param[3]))/(ux - lx) - param[3]
  return(list(x = c(lx, xstar, ux) , w = rep(1/3, 3)))
}
x0 <- c(0, 4.2494, 17.0324, 149.9090)
w0 <- c(0.3204, 0.1207, 0.2293, 0.3296)
## Not run:
sensminimax(formula = ~theta0 + theta1* log(x + theta2),
             predvars = c("x"),
             parvars = c("theta0", "theta1", "theta2"),
             x = x0, w = w0,
             lx = 0, ux = 150,
             lp = c(2, 2, 1), up = c(2, 2, 15),
             localdes = LDOD,
             standardized = TRUE,
             sens.minimax.control = list(answering.set = list(n_seg = 10)))

## End(Not run)
#####
# Not necessary!
# The rest of the examples here are only for professional uses.
#####
# Imagine you have written your own FIM, say in Rcpp that is faster than
# the FIM created by the formula interface here.

#####
# Power logistic model
#####
# For example, th cpp FIM function for the power logistic model is named:
FIM_power_logistic
args(FIM_power_logistic)
# The arguments do not match the standard of the argument 'fimfunc'
# in 'sensminimax'
# So we reparameterize it:
myfim1 <- function(x, w, param)

```

```

    FIM_power_logistic(x = x, w = w, param =param, s = .2)

args(myfim1)
## Not run:
# Verify minimax D-optimality of a design
sensminimax(fimfunc = myfim1,
            x = c(-4.5515, 0.2130, 2.8075),
            w = c(0.4100, 0.3723, 0.2177),
            lx = -5, ux = 5,
            lp = c(0, 1), up = c(3, 1.5))

## End(Not run)
#####
# A model with two predictors
#####
# An example of a model with two-predictors: mixed inhibition model
# Fisher information matrix:
FIM_mixed_inhibition
args(FIM_mixed_inhibition)

# We should first reparameterize the FIM to match the standard of the
# argument 'fimfunc'
myfim2 <- function(x, w, param){
  npoint <- length(x)/2
  S <- x[1:npoint]
  I <- x[(npoint+1):(npoint*2)]
  out <- FIM_mixed_inhibition(S = S, I = I, w = w, param = param)
  return(out)
}
args(myfim2)
## Not run:
# Verifying minimax D-optimality of a design
sensminimax(fimfunc = myfim2,
            x = c(3.4614, 4.2801, 30, 30, 0, 3.1426, 0, 4.0373),
            w = rep(1/4, 4),
            lx = c(0, 0), ux = c(30, 60),
            lp = c(1, 4, 2, 4), up = c(1, 5, 3, 5))

## End(Not run)

#####
# Standardized maximin D-optimal designs
#####
# An example of a user-written FIM function:
help(FIM_loglin)
# An example of verifying standardized maximin D-optimality for a design
# Look how we re-define the function LDOD above
LDOD2 <- function(param){
  ## param is the vector of theta = (theta0, theta1, theta2)
  lx <- 0 # lower bound of the design space
  ux <- 150 # upper bound of the design space
  xstar <- (ux + param[3]) * (lx + param[3]) *
    (log(ux + param[3]) - log(lx + param[3]))/(ux - lx) - param[3]

```

```

    return(list(x = c(lx, xstar, ux) , w = rep(1/3, 3)))
}

args(LDOD2)

sensminimax(fimfunc = FIM_loglin,
            x = x0,
            w = w0,
            lx = 0, ux = 150,
            lp = c(2, 2, 1), up = c(2, 2, 15),
            localdes = LDOD2,
            standardized = TRUE)

```

sensmultiple	<i>Verifying Optimality of Multiple Objective Designs for The 4-Parameter Hill Model</i>
--------------	--

Description

This function uses general equivalence theorem to verify optimality of a multiple objective optimal design found for either the 4-Parameter Hill model or its another re-parameterization, the 4-parameter logistic model. See, for details, Hyun and Wong (2015).

Usage

```

sensmultiple(dose, w, minDose, maxDose, inipars, lambda, delta,
            Hill_par = FALSE, sens.minimax.control = list(),
            calculate_criterion = TRUE, plot_sens = TRUE,
            tol = sqrt(.Machine$double.xmin), silent = FALSE)

```

Arguments

dose	A vector of design points. It is a vector of either dose values or logarithm of dose values, when <code>Hill_par = TRUE</code> .
w	A vector of design weights, i.e. associated weights for dose.
minDose	Minimum dose D . For the 4-parameter logistic model, i.e. when <code>Hill_par = FALSE</code> , it is the minimum of $\log(D)$.
maxDose	Maximum dose D . For the 4-parameter logistic model, i.e. when <code>Hill_par = FALSE</code> , it is the maximum of $\log(D)$.
inipars	A vector of initial estimates for the vector of parameters (a, b, c, d) . For the 4-parameter logistic model, i.e. when <code>Hill_par = FALSE</code> , it is a vector of initial estimates for $(\theta_1, \theta_2, \theta_3, \theta_4)$.
lambda	A vector of relative importance of each of the three criteria, i.e. $\lambda = (\lambda_1, \lambda_2, \lambda_3)$. Here $0 < \lambda_i < 1$ and $\sum \lambda_i = 1$.

delta	Predetermined meaningful value of the minimum effective dose MED. When $\delta < 0$, then $\theta_2 > 0$ or when $\delta > 0$, then $\theta_2 < 0$.
Hill_par	Hill model parameterization? Defaults to TRUE.
sens.minimax.control	Control parameters to verify the general equivalence theorem. For details, see the function sens.minimax.control .
calculate_criterion	Calculate the criterion? Defaults to TRUE.
plot_sens	Plot the sensitivity (derivative) function? Defaults to TRUE.
tol	Tolerance for finding the general inverse of the Fisher information matrix. Defaults to <code>.Machine\$double.xmin</code> .
silent	Do not print anything? Defaults to FALSE.

Details

ELB is a measure of proximity of a design to the optimal design without knowing the latter. Given a design, let ϵ be the global maximum of the sensitivity (derivative) function over $x \in \chi$. ELB is given by

$$ELB = p/(p + \epsilon),$$

where p is the number of model parameters. Obviously, calculating ELB requires finding ϵ and another optimization problem to be solved. The tuning parameters of this optimization can be regulated via the argument [sens.minimax.control](#). See, for more details, Masoudi et al. (2017).

Value

an object of class `sensminimax` that is a list contains:

`type` Argument type that is required for print methods.

`optima` A matrix that stores all the local optima over the parameter space given the best design of the iteration. The cost (criterion) values are stored in a column named `Criterion_Value`. The last column (`Answering_Set`) shows if the optimum, based on a merging tolerance `merge_tol`, belongs to the answering set (1) or not (0). Only applicable for minimax or standardized maximin designs.

`mu` Probability measure on the answering set. Corresponds to the rows of `optima` when the associated row in column `Answering_Set` is equal to 1. Only applicable for minimax or standardized maximin designs.

`max_deriv` Global maximum of the sensitivity (derivative) function (ϵ in 'Details').

`ELB` D-efficiency lower bound. Can not be larger than 1. See 'Note' in [sensminimax](#) or 'Details' in [sens.minimax.control](#).

`merge_tol` Merging tolerance to create the answering set from the set of all local optima. See 'Details' in [sens.minimax.control](#). Only applicable for minimax or standardized maximin designs.

`crtval` Criterion value. Compare with column `Crtiterion_Value` in `optima` for minimax and standardized maximin designs.

`time` Required CPU time (rough approximation).

Note

DO NOT use this function to verify c-optimal designs for estimating 'MED' or 'ED50' (verifying single objective optimal designs) because the results may be unstable. The reason is that for the c-optimal criterion the generalized inverse of the Fisher information matrix is not stable and depends on the tolerance value (tol).

Theoretically, ELB can not be larger than 1. But if so, it may have one of the following reasons:

- max_deriv is not a GLOBAL maximum. Please increase the value of the parameter maxeval in `sens.minimax.control` to find the global maximum.
- The sensitivity function is shifted below the y-axis because the number of model parameters has not been specified correctly (less value given). Please specify the correct number of model parameters via argument npar.

References

Hyun, S. W., and Wong, W. K. (2015). Multiple-Objective Optimal Designs for Studying the Dose Response Function and Interesting Dose Levels. *The international journal of biostatistics*, 11(2), 253-271.

See Also

[multiple](#)

Examples

```
#####
# Verifying optimality of a design for the 4-parameter Hill model
#####

## initial estimates for the parameters of the Hill model
a <- 0.008949 # ED50
b <- -1.79 # Hill constant
c <- 0.137 # lower limit
d <- 1.7 # upper limit
# D belongs to c(.001, 1000) ## dose in mg
## Hill parameters are c(a, b, c, d)
# dose, minDose and maxDose vector in mg scale

sensmultiple (dose = c(0.001, 0.009426562, 0.01973041, 999.9974),
              w = c(0.4806477, 0.40815, 0.06114173, 0.05006055),
              minDose = .001, maxDose = 1000,
              Hill_par = TRUE,
              inipars = c(a, b, c, d),
              lambda = c(0.05, 0.05, .90),
              delta = -1)
```

Description

It plots the sensitivity (derivative) function of the robust criterion at a given approximate (continuous) design and also calculates its efficiency lower bound (ELB) with respect to this optimality criterion. The user can confirm or reject the optimality of a design by looking at the sensitivity plot. ELB shows the proximity of a design to the true optimal design without knowing the latter.

Usage

```
sensrobust(formula, predvars, parvars, family = gaussian(), x, w, lx, ux,
  prob, parset, fimfunc = NULL, sens.minimax.control = list(),
  calculate_criterion = TRUE, plot_3d = c("lattice", "rgl"),
  plot_sens = TRUE, npar = dim(parset)[2], silent = FALSE)
```

Arguments

formula	A nonlinear model formula . A symbolic description of the model consists of predictors and the unknown model parameters. Will be coerced to a formula if necessary.
predvars	A vector of characters. Denotes the predictors in the formula .
parvars	A vector of characters. Denotes the unknown parameters in the formula .
family	A description of the response distribution and link function to be used in the model. This can be a family function, a call to a family function or a character string naming the family. Every family function has a link argument allowing to specify the link function to be applied on the response variable. If not specified, default links are used. For details see family . By default, a linear gaussian model <code>gaussian()</code> is applied.
x	Vector of design (support) points. See 'Details' of sensminimax .
w	Vector of corresponding design weights for x.
lx	Vector of lower bounds for the predictors. Should be in the same order as predvars.
ux	Vector of upper bounds for the predictors. Should be in the same order as predvars.
prob	A vector of the probability measure π associated with each row of parset.
parset	A matrix to provide the set of initial estimates for the model parameters, i.e. support of π . Every row is one vector of initial estimates for the model parameters (<code>nrow(parset) == length(prob)</code>). See 'Details'.
fimfunc	A function. Returns the FIM as a matrix. Required when formula is missing. See 'Details' of minimax .

<code>sens.minimax.control</code>	Control parameters to verify the general equivalence theorem. For details, see the function <code>sens.minimax.control</code> .
<code>calculate_criterion</code>	Evaluate the D-criterion? See 'Details' of <code>sensminimax</code> .
<code>plot_3d</code>	Which package should be used to plot the sensitivity (derivative) function for models with two predictors. Either "rgl" or "lattice" (default).
<code>plot_sens</code>	Plot the sensitivity (derivative) function? Defaults to TRUE.
<code>npar</code>	Number of model parameters. Used when <code>fimfunc</code> is given instead of formula to specify the number of model parameters. If not given, the sensitivity plot may be shifted below the y-axis. When NULL, it will be set here to <code>length(inipars)</code> .
<code>silent</code>	Do not print anything? Defaults to FALSE.

Details

Let Θ be the set initial estimates for the model parameters and π be a probability measure having support in Θ . A design ξ^* is robust with respect to π if the following inequality holds for all $\mathbf{x} \in \chi$:

$$c(\mathbf{x}, \pi, \xi^*) = \int_{\pi} \text{tr} M^{-1}(\xi^*, \theta) I(\mathbf{x}, \theta) \pi(\theta) d(\theta) - p \leq 0,$$

with equality at all support points of ξ^* . Here, p is the number of model parameters.

ELB is a measure of proximity of a design to the optimal design without knowing the latter. Given a design, let ϵ be the global maximum of the sensitivity (derivative) function over $x \in \chi$. ELB is given by

$$ELB = p/(p + \epsilon),$$

where p is the number of model parameters. Obviously, calculating ELB requires finding ϵ and another optimization problem to be solved. The tuning parameters of this optimization can be regulated via the argument `sens.minimax.control`.

Value

an object of class `sensminimax` that is a list contains:

`type` Argument type that is required for print methods.

`optima` A matrix that stores all the local optima over the parameter space given the best design of the iteration. The cost (criterion) values are stored in a column named `Criterion_Value`. The last column (`Answering_Set`) shows if the optimum, based on a merging tolerance `merge_tol`, belongs to the answering set (1) or not (0). Only applicable for minimax or standardized maximin designs.

`mu` Probability measure on the answering set. Corresponds to the rows of `optima` when the associated row in column `Answering_Set` is equal to 1. Only applicable for minimax or standardized maximin designs.

`max_deriv` Global maximum of the sensitivity (derivative) function (ϵ in 'Details').

`ELB` D-efficiency lower bound. Can not be larger than 1. See 'Note' in `sensminimax` or 'Details' in `sens.minimax.control`.

`merge_tol` Merging tolerance to create the answering set from the set of all local optima. See 'Details' in [sens.minimax.control](#). Only applicable for minimax or standardized maximin designs.

`crtval` Criterion value. Compare with column `Crtiterion_Value` in `optima` for minimax and standardized maximin designs.

`time` Required CPU time (rough approximation).

Note

Theoretically, ELB can not be larger than 1. But if so, it may have one of the following reasons:

- `max_deriv` is not a GLOBAL maximum. Please increase the value of the parameter `maxeval` in [sens.minimax.control](#) to find the global maximum.
- The sensitivity function is shifted below the y-axis because the number of model parameters has not been specified correctly (less value given). Please specify the correct number of model parameters via argument `npar`.

See Also

[bayes](#) [sensbayes](#) [robust](#)

Examples

```
# Verifying a robust design for the two-parameter logistic model
sensrobust(formula = ~1/(1 + exp(-b *(x - a))),
  predvars = c("x"),
  parvars = c("a", "b"),
  family = binomial(),
  prob = rep(1/4, 4),
  parset = matrix(c(0.5, 1.5, 0.5, 1.5, 4.0, 4.0, 5.0, 5.0), 4, 2),
  x = c(0.260, 1, 1.739), w = c(0.275, 0.449, 0.275),
  lx = -5, ux = 5)
```

skewnormal

Multivariate Skewed Normal Prior Distribution for Model Parameters

Description

Creates a multivariate skewed normal prior distribution for the unknown parameters as an object of class `cprior`.

Usage

```
skewnormal(xi, Omega, alpha, lower, upper)
```

Arguments

<code>xi</code>	A numeric vector of length $d = \text{length}(\text{alpha})$ representing the location parameter of the distribution. See 'Background' in dmsn .
<code>Omega</code>	A symmetric positive-definite matrix of dimension (d, d) . See 'Background' in dmsn .
<code>alpha</code>	A numeric vector which regulates the slant of the density. See 'Background' in dmsn .
<code>lower</code>	A vector of lower bounds for the unknown parameters.
<code>upper</code>	A vector of upper bounds for the unknown parameters.

Value

An object of class `cprior` that is a list with components:

- `fn`: prior distribution as an R function with argument `param` that is the vector of the unknown parameters. See below.
- `npar`: Number of unknown parameters and is equal to the length of `param`.
- `lower`: Argument `lower`. It has the same length as `param`.
- `upper`: Argument `lower`. It has the same length as `param`.

The list will be passed to the argument `prior` of the function [bayes](#). The order of the argument `param` in `fn` has the same order as the argument `parvars` when the model is specified by a formula. Otherwise, it is the same as the argument `param` in the function `fimfunc`.

See Also

[bayes sensbayes](#)

Examples

```
skewnormal(xi = c(0, 1),
  Omega = matrix(c(1, -0.17, -0.17, .5), nrow = 2),
  alpha = c(1, 0), lower = c(-3, .1), upper = c(3, 2))
```

student

Multivariate Student's t Prior Distribution for Model Parameters

Description

Creates the prior distribution for the parameters as an object of class `cprior`.

Usage

```
student(mean, S, df, lower, upper)
```

Arguments

mean	A vector of length $d = \text{ncol}(S)$, representing the location parameter (equal to the mean vector when $df > 1$). See 'Arguments' in dmt .
S	A symmetric positive-definite matrix representing the scale matrix of the distribution, such that $S * df / (df - 2)$ is the variance-covariance matrix when $df > 2$. See 'Arguments' in dmt .
df	Degrees of freedom; it must be a positive integer. See 'Arguments' in dmt .
lower	A vector of lower bounds for the unknown parameters.
upper	A vector of upper bounds for the unknown parameters.

Value

An object of class `cprior` that is a list with components:

- `fn`: prior distribution as an R function with argument `param` that is the vector of the unknown parameters. See below.
- `npar`: Number of unknown parameters and is equal to the length of `param`.
- `lower`: Argument `lower`. It has the same length as `param`.
- `upper`: Argument `upper`. It has the same length as `param`.

The list will be passed to the argument `prior` of the function [bayes](#). The order of the argument `param` in `fn` has the same order as the argument `parvars` when the model is specified by a formula. Otherwise, it is the same as the argument `param` in the function `fimfunc`.

See Also

[bayes](#) [sensbayes](#)

Examples

```
skewnormal(xi = c(0, 1),
  Omega = matrix(c(1, -0.17, -0.17, .5), nrow = 2),
  alpha = c(1, 0), lower = c(-3, .1), upper = c(3, 2))
```

uniform

Multivariate Uniform Prior Distribution for Model Parameters

Description

Creates independent uniform prior distributions for the unknown model parameters as an object of class `cprior`.

Usage

```
uniform(lower, upper)
```

Arguments

lower	A vector of lower bounds for the unknown parameters.
upper	A vector of upper bounds for the unknown parameters.

Value

An object of class `cprior` that is a list with components:

- `fn`: prior distribution as an R function with argument `param` that is the vector of the unknown parameters. See below.
- `npar`: Number of unknown parameters and is equal to the length of `param`.
- `lower`: Argument `lower`. It has the same length as `param`.
- `upper`: Argument `lower`. It has the same length as `param`.

The list will be passed to the argument `prior` of the function `bayes`. The order of the argument `param` in `fn` has the same order as the argument `parvars` when the model is specified by a formula. Otherwise, it is the same as the argument `param` in the function `fimfunc`.

Note

The order of the argument `param` in `fn` has the same order as the argument `parvars` when the model is specified by a formula. Otherwise, it is the same as the argument `param` in the function `fimfunc`.

See Also

[bayes](#) [sensbayes](#)

Examples

```
uniform(lower = c(-3, .1), upper = c(3, 2))
```


Index

bayes, *2, 3, 4, 11, 26–28, 33, 45, 46, 48, 52, 57, 61, 77–80*
bayescomp, *11, 26, 46, 62*

crt.bayes.control, *3, 12, 14, 46, 58, 61*
crt.minimax.control, *15, 34, 35, 47, 67*

dmsn, *78*
dmt, *79*

family, *3, 11, 30, 33, 50, 57, 60, 63, 66, 75*
FIM_2par_exp_censor1, *17*
FIM_2par_exp_censor2, *17*
FIM_3par_exp_censor1, *18*
FIM_3par_exp_censor2, *19*
FIM_exp_2par, *19*
FIM_logistic, *20*
FIM_logistic_2pred, *21*
FIM_logistic_4par, *21*
FIM_loglin, *22*
FIM_mixed_inhibition, *23*
FIM_power_logistic, *24*
formula, *3, 11, 30, 33, 50, 57, 60, 63, 66, 75*

hcubature, *14, 15, 27, 53*

ICA.control, *3, 12, 20, 24, 27, 30, 34, 36, 42, 50, 51*
ICAOD, *26*
ICAOD-package (ICAOD), *26*
iterate, *28*
iterate.bayes, *28, 28*
iterate.minimax, *28, 29*

locally, *26, 27, 29, 47, 48, 52*

minimax, *3, 12, 26, 27, 29, 30, 33, 34, 42, 47, 48, 50, 51, 57, 61, 63, 66, 75*
multiple, *22, 26, 41, 74*

nl.opts, *16, 53, 55*

nloptr, *15, 16, 27, 35, 53–56*
normal, *3, 4, 11, 44, 57, 61*

plot, *4, 12*
plot.bayes, *45*
plot.minimax, *46*
print.bayes, *48*
print.minimax, *48*
print.sensbayes, *49*
print.sensminimax, *49*

robust, *2, 26, 27, 33, 47, 48, 50, 77*

sens.bayes.control, *3, 12, 46, 53, 58, 61*
sens.minimax.control, *30, 34, 42, 47, 51, 54, 63, 64, 67–69, 73, 74, 76, 77*
sensbayes, *5, 27, 45, 49, 57, 77–80*
sensbayescomp, *13, 27, 49, 60*
senslocally, *27, 31, 49, 62*
sensminimax, *27, 36, 37, 49, 57, 58, 61, 63, 64, 65, 66–68, 73, 75, 76*
sensmultiple, *27, 43, 72*
sensrobust, *27, 49, 51, 52, 75*
skewnormal, *3, 4, 11, 57, 61, 77*
student, *3, 4, 11, 57, 61, 78*

uniform, *3, 4, 11, 57, 61, 79*