Package ‘dampack’

September 30, 2024

Type Package

Title Decision-Analytic Modeling Package

Version 1.0.2.1000

Description A suite of functions for analyzing and visualizing the health economic outputs of mathematical models.
This package was developed with funding from the National Institutes of Allergy and Infectious Diseases of the
National Institutes of Health under award no. R01AI138783. The content of this package is solely the
responsibility of the authors and does not necessarily represent the official views of the National Institutes
of Health. The theoretical underpinnings of 'dampack''s functionality are detailed in Hunink et al. (2014)
<doi:10.1017/CBO9781139506779>.

License GPL-3

URL https://github.com/DARTH-git/dampack

BugReports https://github.com/DARTH-git/dampack/issues

Encoding UTF-8

Depends R (>= 3.5), ggplot2 (>= 3.3.0)

Imports ellipse, dplyr, scales, stringr, mgcv, truncnorm, triangle, ggrepel, tidyr, rlang

Suggests testthat, lintr, knitr, rmarkdown, kableExtra

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Fernando Alarid-Escudero [aut]
(<https://orcid.org/0000-0001-5076-1172>),
Greg Knowlton [aut],
Caleb W. Easterly [aut] (<https://orcid.org/0000-0001-7853-377X>),
David Garibay [ctb, cre],
Eva Enns [aut] (<https://orcid.org/0000-0003-0693-7358>)

Maintainer David Garibay <dgari039@uottawa.ca>

Repository CRAN

Date/Publication 2024-09-30 17:00:06 UTC

Contents

`beta_params` *Calculate alpha and beta parameters of beta distribution.*

Description

Function to calculate the alpha and beta parameters of the beta distribution based on the method of moments using the mean μ and standard deviation σ of the random variable of interest.

Usage

```
beta_params(mean, sigma)
```

Arguments

`mean`	mean of the random variable.
`sigma`	standard deviation of the random variable (i.e., standard error).

Value

a list containing the following:

alpha The method-of-moments estimate for the alpha parameter of the beta distribution

beta The method-of-moments estimate for the beta parameter of the beta distribution

Details

Based on methods of moments. If μ is the mean and σ is the standard deviation of the random variable, then

$$\alpha = (\frac{1-\mu}{\sigma^2} - \frac{1}{\mu})\mu^2$$

and

$$\beta = \alpha(\frac{1}{\mu} - 1)$$

calculate_icers *Calculate incremental cost-effectiveness ratios (ICERs)*

Description

This function takes in strategies and their associated cost and effect, assigns them one of three statuses (non-dominated, extended dominated, or dominated), and calculates the incremental cost-effectiveness ratios for the non-dominated strategies

The cost-effectiveness frontier can be visualized with `plot`, which calls `plot.icers`.

An efficent way to get from a probabilistic sensitivity analysis to an ICER table is by using `summary` on the PSA object and then using its columns as inputs to `calculate_icers`.

Usage

```
calculate_icers(cost, effect, strategies)
```

Arguments

`cost`	vector of cost for each strategy
`effect`	vector of effect for each strategy
`strategies`	string vector of strategy names With the default (NULL), there is no reference strategy, and the strategies are ranked in ascending order of cost.

Value

A data frame and `icers` object of strategies and their associated status, incremental cost, incremental effect, and ICER.

See Also

`plot.icers`

Examples

```
## Base Case
# if you have a base case analysis, can use calculate_icers on that
data(hund_strat)
hund_icers <- calculate_icers(hund_strat$Cost,
                              hund_strat$QALYs,
                              hund_strat$Strategy)

plot(hund_icers)
# we have so many strategies that we may just want to plot the frontier
plot(hund_icers, plot_frontier_only = TRUE)
# see ?plot.icers for more options

## Using a PSA object
```

```
data(psa_cdiff)

# summary() gives mean cost and effect for each strategy
sum_cdiff <- summary(psa_cdiff)

# calculate icers
icers <- calculate_icers(sum_cdiff$meanCost,
                         sum_cdiff$meanEffect,
                         sum_cdiff$Strategy)
icers

# visualize
plot(icers)

# by default, only the frontier is labeled
# if using a small number of strategies, you can label all the points
# note that longer strategy names will get truncated
plot(icers, label = "all")
```

`calculate_icers_psa` *Calculate incremental cost-effectiveness ratios from a* `psa` *object.*

Description

The mean costs and QALYs for each strategy in a PSA are used to conduct an incremental cost-effectiveness analysis. `calculate_icers` should be used if costs and QALYs for each strategy need to be specified manually, whereas `calculate_icers_psa` can be used if mean costs and mean QALYs from the PSA are assumed to represent a base case scenario for calculation of ICERS.

Optionally, the `uncertainty` argument can be used to provide the 2.5th and 97.5th quantiles for each strategy's cost and QALY outcomes based on the variation present in the PSA. Because the dominated vs. non-dominated status and the ordering of strategies in the ICER table are liable to change across different samples of the PSA, confidence intervals are not provided for the incremental costs and QALYs along the cost-effectiveness acceptability frontier. `link{plot.psa}` does not show the confidence intervals in the resulting plot even if present in the ICER table.

Usage

```
calculate_icers_psa(psa, uncertainty = FALSE)
```

Arguments

`psa` psa object from `link{make_psa_object}`

`uncertainty` whether or not 95 in the resulting ICER table. Defaults to `FALSE`.

Value

A data frame and `icers` object of strategies and their associated status, cost, effect, incremental cost, incremental effect, and ICER. If `uncertainty` is set to `TRUE`, four additional columns are provided for the 2.5th and 97.5th quantiles for each strategy's cost and effect.

See Also

plot.icers

calculate_icers

calc_evpi *Expected Value of Perfect Information (EVPI)*

Description

calc_evpi is used to compute the expected value of perfect information (EVPI) from a probabilistic sensitivity analysis (PSA) dataset.

Usage

```
calc_evpi(psa, wtp, pop = 1)
```

Arguments

psa	psa object from make_psa_obj
wtp	numeric vector with willingness-to-pay (WTP) thresholds
pop	scalar that corresponds to the total population

Value

A data frame and evpi object with the EVPI at each WTP threshold.

Details

evpi calculates the value of eliminating all the uncertainty of a cost-effectiveness analysis at each WTP threshold.

See Also

plot.evpi, make_psa_obj

Examples

```
# load psa object provided with package
data("example_psa_obj")

# define wtp threshold vector (can also use a single wtp)
wtp <- seq(1e4, 1e5, by = 1e4)
evpi <- calc_evpi(example_psa_obj, wtp)
plot(evpi) # see ?plot.evpi for options

# can use plot options (# see ?plot.evpi for details)
plot(evpi, effect_units = "QALE")
```

```
# or can use ggplot layers
plot(evpi) + ggtitle("Expected Value of Perfect Information")
```

calc_evppi *Estimation of the Expected Value of Partial Perfect Information (EVPPI) using a linear regression metamodel approach*

Description

evppi is used to estimate the Expected Value of Partial Perfect Information (EVPPI) using a linear regression metamodel approach from a probabilistic sensitivity analysis (PSA) dataset.

Usage

```
calc_evppi(
  psa,
  wtp,
  params = NULL,
  outcome = c("nmb", "nhb"),
  type = c("gam", "poly"),
  poly.order = 2,
  k = -1,
  pop = 1,
  progress = TRUE
)
```

Arguments

psa	object of class psa, produced by make_psa_obj
wtp	willingness-to-pay threshold
params	A vector of parameter names to be analyzed in terms of EVPPI.
outcome	either net monetary benefit ("nmb") or net health benefit ("nhb")
type	either generalized additive models ("gam") or polynomial models ("poly")
poly.order	order of the polynomial, if type == "poly"
k	basis dimension, if type == "gam"
pop	scalar that corresponds to the total population
progress	TRUE or FALSE for whether or not function progress should be displayed in console.

Details

The expected value of partial pefect information (EVPPI) is the expected value of perfect information from a subset of parameters of interest, θ_I, of a cost-effectiveness analysis (CEA) of D different strategies with parameters $\theta = \{\theta_I, \theta_C\}$, where θ_C is the set of complimenatry parameters of the CEA. The function `calc_evppi` computes the EVPPI of θ_I from a matrix of net monetary benefits B of the CEA. Each column of B corresponds to the net benefit B_d of strategy d. The function `calc_evppi` computes the EVPPI using a linear regression metamodel approach following these steps:

1. Determine the optimal strategy d^* from the expected net benefits $\bar{B}$

$$d^* = argmax_d\{\bar{B}\}$$

2. Compute the opportunity loss for each d strategy, L_d

$$L_d = B_d - B_{d^*}$$

3. Estimate a linear metamodel for the opportunity loss of each d strategy, L_d, by regressing them on the spline basis functions of θ_I, $f(\theta_I)$

$$L_d = \beta_0 + f(\theta_I) + \epsilon,$$

where ϵ is the residual term that captures the complementary parameters θ_C and the difference between the original simulation model and the metamodel.

4. Compute the EVPPI of θ_I using the estimated losses for each d strategy, $\hat{L}_d$ from the linear regression metamodel and applying the following equation:

$$EVPPI_{\theta_I} = \frac{1}{K}\sum_{i=1}^{K}\max_d(\hat{L}_d)$$

The spline model in step 3 is fitted using the 'mgcv' package.

Value

A list containing 1) a data.frame with WTP thresholds and corresponding EVPPIs for the selected parameters and 2) a list of metamodels used to estimate EVPPI for each strategy at each willingness to pay threshold.

References

1. Jalal H, Alarid-Escudero F. A General Gaussian Approximation Approach for Value of Information Analysis. Med Decis Making. 2018;38(2):174-188.

2. Strong M, Oakley JE, Brennan A. Estimating Multiparameter Partial Expected Value of Perfect Information from a Probabilistic Sensitivity Analysis Sample: A Nonparametric Regression Approach. Med Decis Making. 2014;34(3):311–26.

calc_evsi *Calculate Expected Value of Sample Information (EVSI)*

Description

Calculate Expected Value of Sample Information (EVSI)

Usage

```
calc_evsi(
  psa,
  wtp,
  params = NULL,
  outcome = c("nhb", "nmb"),
  k = -1,
  n = 100,
  n0 = 10,
  n_by_param = FALSE,
  pop = 1,
  progress = TRUE
)
```

Arguments

psa	object of class psa, produced by make_psa_obj
wtp	willingness-to-pay threshold
params	A vector of parameter names to be analyzed in terms of EVPPI.
outcome	either net monetary benefit ("nmb") or net health benefit ("nhb")
k	basis dimension, if type == "gam"
n	additional sample size
n0	initial sample size
n_by_param	if TRUE, each parameter in the metamodel can have a unique initial and additional sample size. n and n0 must be numerical vectors of equal length to params, with each value corresponding to the initial and additional sample sizes for each parameter in the metamodel. By default, n_by_param = FALSE, and each value of n and n0 is shared by each parameter in the model. When n_by_param = FALSE, n0 must be a single numeric value, and n must be a numerical vector of additional sample sizes for which EVSI is calculated from the metamodel.
pop	scalar that corresponds to the total population
progress	TRUE or FALSE for whether or not function progress should be displayed in console.

Value

A list containing 1) a data.frame with WTP thresholds, new prospective sample sizes (if `n_by_param == FALSE`), and corresponding EVSIs for the selected parameters and 2) a list of metamodels used to estimate EVSI for each strategy at each willingness to pay threshold.

`calc_exp_loss` *Calculate the expected loss at a range of willingness-to-pay thresholds*

Description

The expected loss is the quantification of the foregone benefits when choosing a suboptimal strategy given current evidence.

Usage

```
calc_exp_loss(psa, wtp)
```

Arguments

`psa`	object of class `psa`, produced by function `make_psa_obj`
`wtp`	vector of willingness to pay thresholds

Details

Visualize the expected loss at a variety of WTP thresholds using `plot.exp_loss`.

Value

object with classes `exp_loss` and `data.frame`

References

1. Alarid-Escudero F, Enns EA, Kuntz KM, Michaud TL, Jalal H. "Time Traveling Is Just Too Dangerous" But Some Methods Are Worth Revisiting: The Advantages of Expected Loss Curves Over Cost-Effectiveness Acceptability Curves and Frontier. Value Health. 2019;22(5):611-618.
2. Eckermann S, Briggs A, Willan AR. Health technology assessment in the cost- disutility plane. Med Decis Making. 2008;28(2):172–181.

See Also

`plot.exp_loss`, `make_psa_obj`

Examples

```
data("example_psa_obj")
wtp <- seq(1e4, 1e5, by = 1e4)
exp_loss <- calc_exp_loss(example_psa_obj, wtp)

# can use head(), summary(), print(), etc.
head(exp_loss)

# plot an expected loss curve (ELC)
plot(exp_loss)

# the y axis is on a log scale by default
plot(exp_loss, log_y = FALSE)
```

---

ceac *Cost-Effectiveness Acceptability Curve (CEAC)*

Description

ceac is used to compute and plot the cost-effectiveness acceptability curves (CEAC) from a probabilistic sensitivity analysis (PSA) dataset.

Usage

```
ceac(wtp, psa)
```

Arguments

wtp	numeric vector with willingness-to-pay (WTP) thresholds
psa	psa object from make_psa_obj

Details

ceac computes the probability of each of the strategies being cost-effective at each wtp threshold. The returned object has classes ceac and data.frame, and has its own plot method (plot.ceac).

Value

An object of class ceac that can be visualized with plot. The ceac object is a data.frame that shows the proportion of PSA samples for which each strategy at each WTP threshold is cost-effective. The final column indicates whether or not the strategy at a particular WTP is on the cost-efficient frontier.

See Also

plot.ceac, summary.ceac

Examples

```
# psa input provided with package
data("example_psa")
example_psa_obj <- make_psa_obj(example_psa$cost, example_psa$effectiveness,
                    example_psa$parameters, example_psa$strategies)

# define wtp threshold vector (can also use a single wtp)
wtp <- seq(1e4, 1e5, by = 1e4)
ceac_obj <- ceac(wtp, example_psa_obj)
plot(ceac_obj) # see ?plot.ceac for options

# this is most useful when there are many strategies
# warnings are printed to describe strategies that
# have been filtered out
plot(ceac_obj, min_prob = 0.5)

# standard ggplot layers can be used
plot(ceac_obj) +
    labs(title = "CEAC", y = "Pr(Cost-effective) at WTP")

# the ceac object is also a data frame
head(ceac_obj)

# summary() tells us the regions of cost-effectiveness for each strategy.
# Note that the range_max column is an open parenthesis, meaning that the
# interval over which that strategy is cost-effective goes up to but does not include
# the value in the range_max column.
summary(ceac_obj)
```

`create_dsa_oneway` *Create one-way deterministic sensitivity analysis object*

Description

The object returned by this function can be passed to owsa to do a one-way sensitivity analysis on each parameter of interest.

Usage

```
create_dsa_oneway(
  parameters,
  effectiveness = NULL,
  strategies,
  cost = NULL,
  currency = "$",
  other_outcome = NULL
)
```

Arguments

`parameters` parameter values associated with costs, effectiveness, or other outcomes. The table must have two columns, with each parameter name in the first column and the associated parameter value in the second column:

parameter	value
param1 name	param1 val1
...	...
param2 name	param2 val1
...	...

`strategies` vector with the names of the strategies. Due to requirements in certain uses of this vector, this function uses `make.names` to modify strategy names as necessary. It is strongly suggested that you follow the rules in the `make.names` help page, to avoid unexpected errors.

`cost`, `effectiveness`, `other_outcome`
data frames containing data for costs, effectiveness or another outcome (user-defined), respectively. Each simulation should be a row of the data frame, and each strategy should be a column. Naming the columns of the data frames is not necessary, as they will be renamed with the `strategies` vector.

`currency` symbol for the currency being used (ex. "$", "£")

Value

a class `dsa_oneway` object that can be passed to the `owsa` function to visualize the one-way sensitivity analyses contained in the object.

`create_dsa_twoway` *Create one-way deterministic sensitivity analysis object*

Description

The object returned by this function can be passed to `owsa` to do a one-way sensitivity analysis on each parameter of interest.

Usage

```
create_dsa_twoway(
  parameters,
  effectiveness = NULL,
  strategies,
  cost = NULL,
  currency = "$",
  other_outcome = NULL
)
```

Arguments

`parameters` parameter values associated with effectiveness and outcomes. The table must have two columns, one for each parameter. The parameter names must be the column names.

param1 name	param2 name
param1 val1	param2 val1
param1 val2	param2 val2
...	...

`strategies` vector with the names of the strategies. Due to requirements in certain uses of this vector, this function uses `make.names` to modify strategy names as necessary. It is strongly suggested that you follow the rules in the `make.names` help page, to avoid unexpected errors.

`cost`, `effectiveness`, `other_outcome`
data frames containing data for costs, effectiveness or another outcome (user-defined), respectively. Each simulation should be a row of the data frame, and each strategy should be a column. Naming the columns of the data frames is not necessary, as they will be renamed with the `strategies` vector.

`currency` symbol for the currency being used (ex. "$", "£")

Value

a class `dsa_twoway` object that can be passed to the `twsa` function to visualize the two-way sensitivity analysis contained in the object.

`dirichlet_params` *Calculate alpha parameters of Dirichlet distribution.*

Description

Function to calculate the α parameters of the Dirichlet distribution based on the method of moments (MoM) using the mean μ and standard deviation σ of the random variables of interest.

Usage

```
dirichlet_params(p.mean, sigma)
```

Arguments

`p.mean` Vector of means of the random variables.

`sigma` Vector of standard deviation of the random variables (i.e., standard error).

Value

numeric vector of method-of-moment estimates for the alpha parameters of the dirichlet distribution

Details

Based on methods of moments. If μ is a vector of means and σ is a vector of standard deviations of the random variables, then the second moment X_2 is defined by $\sigma^2 + \mu^2$. Using the mean and the second moment, the J alpha parameters are computed as follows

$$\alpha_i = \frac{(\mu_1 - X_{2_1})\mu_i}{X_{2_1} - \mu_1^2}$$

for $i = 1, \ldots, J-1$, and

$$\alpha_J = \frac{(\mu_1 - X_{2_1})(1 - \sum_{i=1}^{J-1} \mu_i)}{X_{2_1} - \mu_1^2}$$

References

1. Fielitz BD, Myers BL. Estimation of parameters in the beta distribution. Dec Sci. 1975;6(1):1–13.
2. Narayanan A. A note on parameter estimation in the multivariate beta distribution. Comput Math with Appl. 1992;24(10):11–7.

Examples

```
p.mean <- c(0.5, 0.15, 0.35)
p.se   <- c(0.035, 0.025, 0.034)
dirichlet_params(p.mean, p.se)
```

`example_psa` *Sample PSA data for testing*

Description

A dataset containing a number of PSA samples

Usage

```
example_psa
```

Format

An object of class `list` of length 5.

example_psa_obj *Sample PSA data for testing*

Description

A psa object created from the data in example_psa

Usage

```
example_psa_obj
```

Format

An object of class `psa` of length 8.

gamma_params *Calculate shape and scale (or rate) parameters of a gamma distribution.*

Description

Function to calculate the shape, α, and scale, θ, (or rate, β) parameters of a gamma distribution based on the method of moments (MoM) using the mean μ and standard deviation σ of the random variable of interest.

Usage

```
gamma_params(mu, sigma, scale = TRUE)
```

Arguments

`mu`	scalar with the mean of the random variable.
`sigma`	scalar with the standard deviation of the random variable.
`scale`	logical variable indicating scale parameterization of the gamma distribution (Default is TRUE). If FALSE, rate parameterization is retrieved

Value

A list contianing the following:

shape Shape parameter of gamma distribution

scale Scale parameter of gamma distribution (If scale=TRUE)

rate Rate parameter of gamma distribution (If scale=FALSE)

Details

Based on method of moments. If μ is the mean and σ is the standard deviation of the random variable, then the the shape, α, scale, θ, and rate, β, parameters are computed as follows

$$\alpha = \frac{\mu^2}{\sigma^2},$$

$$\theta = \frac{\sigma^2}{\mu}$$

and

$$\beta = \frac{\mu}{\sigma^2}$$

References

- Gamma distribution. (2018, February 7). In Wikipedia, The Free Encyclopedia. Retrieved 17:23, February 11, 2018, from https://en.wikipedia.org/w/index.php?title=Gamma_distribution&oldid=824541785

Examples

```
mu    <- 2
sigma <- 1
# Scale specification
gamma_params(mu, sigma)
# Rate specification
gamma_params(mu, sigma, scale = FALSE)
```

gen_psa_samp *Generate PSA Sample*

Description

gen_psa_samp generates a data.frame of sampled parameter values from user-specified distributions to be used in a probabilistic sensitivity analysis (PSA)

Usage

```
gen_psa_samp(
  params = NULL,
 dists = c("normal", "log-normal", "truncated-normal", "beta", "gamma", "dirichlet",
    "bootstrap", "constant", "triangle"),
 parameterization_types = c("mean, sd", "a, b", "shape, scale", "value, mean_prop, sd",
    "value, n", "value, alpha", "mean, sd, ll, ul", "val", "meanlog, sdlog",
    "ll, ul, mode"),
  dists_params = NULL,
  nsamp = 100
)
```

Arguments

`params` string vector with the names of parameters to be generated by `gen_psa_samp` and used by a user-defined function in `run_psa` to calculate outcomes.

`dists` string vector with the distributions from which `params` will be drawn.

`parameterization_types`
string vector with parameterization types for each `dists`

`dists_params` list of input parameters required to by specific `dists` and `parameterization_types` to fully describe distribution and generate parameter samples.

`nsamp` number of sets of parameter values to be generated

Details

Length of vectors `params`, `dists`, `parameterization_types`, and list `dists_params` must all be the same. The nth element of `dists`, `parameterization_types`, and `dists_params` all define the distribution that will be used to draw samples of the corresponding nth element of the `params` vector.

For a given element of `params`:

- If `dists` == `"normal"`, `parameterization_types` can only be `"mean, sd"`, and the corresponding element of list `dists_params` must be the the vector `c(mean, sd)`
- If `dists` == `"log-normal"`, `parameterization_types` can be either `"mean, sd"` or `"meanlog, sdlog"`, and the corresponding element of list `dists_params` must be either the the vector `c(mean, sd)` or `c(meanlog, sdlog)`. Use `"mean, sd"` if you have sample mean and sample standard deviation of an empirical sample of the random variable, and use `"meanlog, sdlog"` if you want to directly specify the parameters of the log-normal distribution as specified by `rlnorm`
- If `dists` == `"truncated-normal"`, `parameterization_types` can only be `"mean, sd, ll, ul"`, and `dists_params` must be the vector `c(mean, sd, ll, ul)`, where `ll` is the lower limit of the distribution and `ul` is the upper limit of the distribution. If either the lower limit or the upper limit does not exist, simply specify `NA` in the corresponding position of the dists_params vector.
- If `dists` == `"beta"`, `parameterization_types` can be `"mean, sd"` or `"a, b"` and the corresponding element of list `dists_params` must be the the vector `c(mean, sd)` or `c(a, b)`, respectively.
- If `dists` == `"gamma"`, `parameterization_types` can be `"mean, sd"` or `"shape, scale"` and the corresponding element of list `dists_params` must be the the vector `c(mean, sd)` or `c(shape, scale)`, respectively.
- If `dists` == `"dirichlet"`, `parameterization_types` can be `"value, mean_prop, sd"`, `"value, n"`, or `"value, alpha"`.
    - If `parameterization_types` == `"value, mean_prop, sd"`, then the corresponding element of list `dists_params` must be a data.frame where the first column is a string vector of the the different multinomial outcomes. These multinomial outcomes will become column names in the data.frame returned by `gen_psa_samp`, and therefore the strings in this column should correspond to variable names used in `FUN` for `run_psa`. The second and third columns of this `dists_params` should be numerical vectors containing the sample means and sample standard errors for each of the multinomial outcomes.

  - If parameterization_types == "value, n", then dists_params must be a data.frame with the first column being a string vector of the multinomial outcomes, and the second column being a vector of the observed number of each multinomial outcome in a sample.
  - If parameterization_types == "value, alpha", then dists_params must be a data.frame with the first column being a string vector of the multinomial outcomes, and the second column must be a numerical vector of the alpha parameter values for each multinomial outcome in the dirichlet distribution.
- If dists == "bootstrap", parameterization_types can only be "value, weight", and dists_params must be a data.frame with the first column being a numerical vector containing all of the bootstrap sample values, and the second column being an integer vector designating the sampling weights of each bootstrap sample value. For example, the number of rows in the dists_params data.frame is the number of individuals in the population to be sampled from (with replacement) or the number of values an empirical distribution (e.g. a histogram). If each individual value in the sample is unique and should be weighted equally, set each weight to 1. If the sample distribution resembles a histogram, the weights should be equal to the number of observations for each unique value in the empirical distribution.
- If dists == "constant", parameterization_types can only be "val", and dists_params must be a single numerical value.

Value

A dataframe with samples of parameters for a probabilistic sensitivity analysis (PSA)

See Also

run_psa

Examples

```
#define parameter names
params <- c("normal_param", "lognorm_param", "truncnorm_param", "beta_param",
            "gamma_param", "dirichlet_param", "bootstrap_param")

#indicate parent distribution types for each parameter
dists <- c("normal", "log-normal", "truncated-normal", "beta", "gamma", "dirichlet", "bootstrap")

#indicate which type of parameterization is used for each parent distribution
parameterization_types <- c("mean, sd", "mean, sd", "mean, sd, ll, ul", "mean, sd", "mean, sd",
                           "value, mean_prop, sd", "value, weight")

#provide distribution parameters that fully define each parent distribution, and
#ensure that these distribution parameters match the form expected by each combination of dists
#and parameterization_types
dists_params <- list(c(1, 2), c(1, 3), c(1, 0.1, NA, 1), c(.5, .2), c(100, 1),
                     data.frame(value = c("level1", "level2", "level3"),
                                mean_prop = c(.1, .4, .5), sd = c(.05, .01, .1)),
                     data.frame(value = c(1, 2, 4, 6, 7, 8),
                                weight = c(1, 1, 1, 1, 1, 4)))

#generate 100 samples of parameter values to be used in a probabilistic sensitivity analysis
```

```
gen_psa_samp(params = params,
             dists = dists,
             parameterization_types = parameterization_types,
             dists_params = dists_params,
             nsamp = 100)
```

hund_strat *Sample deterministic data for testing*

Description

A dataset containing 100 strategies

Usage

```
hund_strat
```

Format

An object of class `data.frame` with 100 rows and 3 columns.

lnorm_params *Calculate location and scale parameters of a log-normal distribution.*

Description

Function to calculate the location, μ, and scale, σ, parameteres of a log-normal distribution based on the method of moments (MoM) using the mean m and variance v of the non-logarithmized random variable of interest.

Usage

```
lnorm_params(m = 1, v = 1)
```

Arguments

`m`	Scalar with the mean of the random variable.
`v`	Scalar with the variance of the random variable. (i.e., squared standar error).

Value

A list containing the following:

mu Location parameter of log-normal distribution

sigma Scale parameter of log-normal distribution

Details

Based on method of moments. If m is the mean and v is the variance of the random variable, then the the location, μ, and scale, σ, parameteres are computed as follows

$$\mu = \ln{(\frac{m}{\sqrt{(1 + \frac{v}{m^2})}})}$$

and

$$\sigma = \sqrt{\ln{(1 + \frac{v}{m^2})}}$$

References

1. Ginos BF. Parameter Estimation for the Lognormal Distribution. Brigham Young University; 2009.
2. Log-normal distribution. (2017, April 20). In Wikipedia, The Free Encyclopedia. Retrieved 16:47, April 23, 2017, from https://en.wikipedia.org/w/index.php?title=Log-normal_distribution&oldid=776357974

Examples

```
m <- 3
v <- 0.01
lnorm_params(m, v)
# True values: 100, 30, 70
```

---

make_psa_obj *Create a PSA object*

Description

Creates an object to hold probabilistic sensitivity analysis data, while checking the data for validity. The object can then be used for many standard cost-effectiveness analyses (see Details below).

Usage

```
make_psa_obj(
  cost,
  effectiveness,
  parameters = NULL,
  strategies = NULL,
  currency = "$",
  other_outcome = NULL
)
```

Arguments

cost For the data.frame, each simulation should be a row and each strategy should be a column. Naming the columns of the data frames is not necessary, as they will be renamed with the strategies vector.

effectiveness For the data.frame, each simulation should be a row and each strategy should be a column. Naming the columns of the data frames is not necessary, as they will be renamed with the strategies vector.

parameters Data frame with values for each simulation (rows) and parameter (columns). The column names should be the parameter names.

strategies vector with the names of the strategies. Due to requirements in certain uses of this vector, this function uses make.names to modify strategy names as necessary. It is strongly suggested that you follow the rules in the make.names help page, to avoid unexpected errors.

currency symbol for the currency being used (ex. "$", "£")

other_outcome data.frame containing values for another user-defined outcome. Each simulation should be a row of the data frame, and each strategy should be a column. Naming the columns of the data frames is not necessary, as they will be renamed with the strategies vector.

Details

The PSA object forms the backbone of one part of the dampack package.

A scatterplot of the cost-effectiveness plane may be shown by running plot on the output of make_psa_obj.

Using this object, you may calculate:

- Cost-effectiveness acceptability curves (ceac)
- Expected value of perfect information (calc_evpi)
- Expected loss (calc_exp_loss)
- One-way sensitivity analysis (owsa)
- Two-way sensitivity analysis (twsa)
- Metamodels (metamodel)

In addition, the PSA may be converted to a base-case analysis by using summary on the PSA object. The output of summary can be used in calculate_icers.

Value

An object of class psa

See Also

summary.psa, plot.psa

Examples

```
# psa input provided with package
data("example_psa")
psa <- make_psa_obj(example_psa$cost, example_psa$effectiveness,
                    example_psa$parameters, example_psa$strategies)

# custom print and summary methods
print(psa)
summary(psa)

# custom plot method; see ?plot.psa for options
plot(psa)
```

---

metamodel *Linear regression metamodeling*

Description

This function estimates a linear regression metamodel for a given decision-analytic model by using the results of a probabilistic sensitivity analysis (PSA)

Usage

```
metamodel(
  analysis = c("oneway", "twoway", "multiway"),
  psa,
  params = NULL,
  strategies = NULL,
 outcome = c("eff", "cost", "nhb", "nmb", "nhb_loss", "nmb_loss", "nhb_loss_voi",
    "nmb_loss_voi"),
  wtp = NULL,
  type = c("linear", "gam", "poly"),
  poly.order = 2,
  k = -1
)
```

Arguments

`analysis`	either "oneway" or "twoway"
`psa`	psa object
`params`	string vector with the name(s) of the parameter of interest. Defaults to all.
`strategies`	vector of strategies to consider. The default (NULL) is that all strategies are considered.

`outcome` either effectiveness ("eff"), cost ("cost"), net health benefit ("nhb"), net monetary benefit ("nmb"), or the opportunity loss in terms of NHB or NMB ("nhb_loss" and "nmb_loss", respectively). "nmb_loss_voi" and "nhb_loss_voi" are only used in internal function calls of metamodel within other VOI functions.

`wtp` if outcome is NHB or NMB (or the associated loss), must provide the willingness-to-pay threshold

`type` type of metamodel

`poly.order` order of polynomial for the linear regression metamodel. Default: 2

`k` the dimension of the basis used to represent the smooth term. The default depends on the number of variables that the smooth is a function of. `k` should not be less than the dimension of the null space of the penalty for the term (see `null.space.dimension`), but will be reset if it is. See `choose.k` for further information.

Details

The most important option is `analysis`, which can be either `"oneway"` or `twoway`. If `analysis == "oneway"`, a separate metamodel is created for each combination of the parameters in `params` and strategies in `strategies` (by default, this is all strategies and parameters).

If `analysis == "twoway"`, `params` must be a vector of two parameters, and a metamodel is created with these two parameters for each strategy in `strategies`.

Value

A metamodel object, which contains a list of metamodels and other relevant information.

See Also

`predict.metamodel`, `make_psa_obj`, `owsa`, `twsa`

`number_ticks` *Number of ticks for* `ggplot2` *plots*

Description

Function for determining number of ticks on axis of `ggplot2` plots.

Usage

```
number_ticks(n)
```

Arguments

`n` integer giving the desired number of ticks on axis of `ggplot2` plots. Non-integer values are rounded down.

Value

a vector of axis-label breaks

Details

Based on function `pretty`.

`owsa` *One-way sensitivity analysis*

Description

When used on a PSA object, this function uses a polynomial regression metamodel to predict the average outcome of a decision-analytic model as a function of a single input parameter. When used on a DSA object, this function uses the DSA results directly to show how the selected outcome varies as a function of the input parameter of interest. In the DSA context, this function is called internally by `run_owsa_det` and should not be called by the user. In the PSA context, the user must use this function to produce an `owsa` object.

Usage

```
owsa(
  sa_obj,
  params = NULL,
  ranges = NULL,
  nsamp = 100,
  outcome = c("eff", "cost", "nhb", "nmb", "nhb_loss", "nmb_loss"),
  wtp = NULL,
  strategies = NULL,
  poly.order = 2
)
```

Arguments

`sa_obj` sensitivity analysis object; either a probabilistic sensitivity analysis (`make_psa_obj`) or a deterministic sensitivity analysis object (`run_owsa_det`)

`params` string vector with the name(s) of the parameter of interest. Defaults to all.

`ranges` a named list of the form c("param" = c(0, 1), ...) that gives the ranges for the parameter of interest. If NULL, parameter values from the middle 95 from this range is determined by `nsamp`.

`nsamp` number of samples to take from the ranges

`outcome` either effectiveness ("eff"), cost ("cost"), net health benefit ("nhb"), net monetary benefit ("nmb"), or the opportunity loss in terms of NHB or NMB ("nhb_loss" and "nmb_loss", respectively). "nmb_loss_voi" and "nhb_loss_voi" are only used in internal function calls of metamodel within other VOI functions.

`wtp` if outcome is NHB or NMB (or the associated loss), must provide the willingness-to-pay threshold

`strategies` vector of strategies to consider. The default (NULL) is that all strategies are considered.

`poly.order` order of polynomial for the linear regression metamodel. Default: 2

Value

An object of class `data.frame` and `owsa` with the results of the sensitivity analysis. Can be visualized with `plot.owsa`, `owsa_tornado`, and `owsa_opt_strat`

`owsa_opt_strat` *plot the optimal strategy as the parameter values change*

Description

plot the optimal strategy as the parameter values change

Usage

```
owsa_opt_strat(
  owsa,
  params = NULL,
  maximize = TRUE,
  return = c(”plot”, ”data”),
  plot_const = TRUE,
  col = c(”full”, ”bw”),
  greystart = 0.2,
  greyend = 0.8,
  txtsize = 12,
  facet_ncol = 1,
  facet_nrow = NULL,
  facet_lab_txtsize = NULL,
  n_x_ticks = 10
)
```

Arguments

`owsa` An owsa object

`params` vector of parameters to plot

`maximize` whether to maximize (TRUE) or minimize the outcome

`return` either return a ggplot object `plot` or a data frame with ranges of parameters for which each strategy is optimal.

`plot_const` whether to plot parameters that don’t lead to changes in optimal strategy as they vary.

`col` either none, full color, or black and white

`greystart` between 0 and 1. used in greyscale only. smaller numbers are lighter

`greyend` between 0 and 1, greater than greystart.

`txtsize` base text size

`facet_ncol` Number of columns in plot facet.

`facet_nrow` number of rows in plot facet.

`facet_lab_txtsize`
text size for plot facet labels

`n_x_ticks` number of x-axis ticks

Value

If `return == "plot"`, a `ggplot2` optimal strategy plot derived from the `owsa` object, or if `return == "data"`, a `data.frame` containing all data contained in the plot. The plot allows us to see how the strategy that maximizes the expectation of the outcome of interest changes as a function of each parameter of interest.

`owsa_tornado` *Tornado plot of a one-way sensitivity analysis*

Description

Tornado plot of a one-way sensitivity analysis

Usage

```
owsa_tornado(
  owsa,
  return = c("plot", "data"),
  txtsize = 12,
  min_rel_diff = 0,
  col = c("full", "bw"),
  n_y_ticks = 8,
  ylim = NULL,
  ybreaks = NULL
)
```

Arguments

`owsa` an owsa object

`return` either return a ggplot object `plot` or a data frame with ranges of parameters for which each strategy is optimal.

`txtsize` base text size

`min_rel_diff`	this function only plots parameters that lead to a relative change in the outcome greater than or equal to `min_rel_diff`, which must be between 0 and 1. The default (0) is that no strategies are filtered.
`col`	either none, full color, or black and white
`n_y_ticks`	number of y-axis ticks
`ylim`	vector of y-axis limits, or NULL, which sets limits automatically
`ybreaks`	vector of y-axis breaks. will override `n_y_ticks` if provided.

Value

If `return == "plot"`, a `ggplot2` tornado plot derived from the `owsa` object, or if `return == "data"`, a `data.frame` containing all data contained in the plot. A tornado plot is a visual aid used to identify which parameters are driving most of the variation in a specified model outcome.

`plot.evpi` *Plot of Expected Value of Perfect Information (EVPI)*

Description

Plots the `evpi` object created by `calc_evpi`.

Usage

```
## S3 method for class 'evpi'
plot(
  x,
  txtsize = 12,
  currency = "$",
  effect_units = "QALY",
  n_y_ticks = 8,
  n_x_ticks = 20,
  xbreaks = NULL,
  ybreaks = NULL,
  xlim = c(0, NA),
  ylim = NULL,
  ...
)
```

Arguments

`x`	object of class `evpi`, produced by function `calc_evpi`
`txtsize`	base text size
`currency`	string with currency used in the cost-effectiveness analysis (CEA). Default: $, but it could be any currency symbol or word (e.g., £, €, peso)
`effect_units`	units of effectiveness. Default: QALY

`n_y_ticks` number of y-axis ticks

`n_x_ticks` number of x-axis ticks

`xbreaks` vector of x-axis breaks. will override `n_x_ticks` if provided.

`ybreaks` vector of y-axis breaks. will override `n_y_ticks` if provided.

`xlim` vector of x-axis limits, or NULL, which sets limits automatically

`ylim` vector of y-axis limits, or NULL, which sets limits automatically

`...` further arguments to plot. This is not used by `dampack` but required for generic consistency.

Value

A `ggplot2` plot with the EVPI

See Also

`calc_evpi`

`plot.evppi` *Plot of Expected Value of Partial Perfect Information (EVPPI)*

Description

Plots the `evppi` object created by `calc_evppi`.

Usage

```
## S3 method for class 'evppi'
plot(
  x,
  txtsize = 12,
  currency = "$",
  effect_units = "QALY",
  n_y_ticks = 8,
  n_x_ticks = 20,
  xbreaks = NULL,
  ybreaks = NULL,
  xlim = c(0, NA),
  ylim = NULL,
  ...
)
```

Arguments

`x` object of class `evppi`, produced by function `calc_evppi`

`txtsize` base text size

`currency` string with currency used in the cost-effectiveness analysis (CEA). Default: $, but it could be any currency symbol or word (e.g., £, €, peso)

`effect_units` units of effectiveness. Default: QALY

`n_y_ticks` number of y-axis ticks

`n_x_ticks` number of x-axis ticks

`xbreaks` vector of x-axis breaks. will override `n_x_ticks` if provided.

`ybreaks` vector of y-axis breaks. will override `n_y_ticks` if provided.

`xlim` vector of x-axis limits, or NULL, which sets limits automatically

`ylim` vector of y-axis limits, or NULL, which sets limits automatically

`...` further arguments to plot. This is not used by `dampack` but required for generic consistency.

Value

A `ggplot2` plot with the EVPPI

See Also

`calc_evppi`

`plot.evsi` *Plot of Expected Value of Sample Information (EVSI)*

Description

Plots the `evsi` object created by `calc_evsi`. EVSI is either plotted as a function of additional sample size for each willingness-to-pay threshold provided, or as a function of each willingness-to-pay threshold, depending upon the usage of `calc_evsi` used to create the `evsi` object.

Usage

```
## S3 method for class 'evsi'
plot(
  x,
  txtsize = 12,
  currency = "$",
  effect_units = "QALY",
  n_y_ticks = 8,
  n_x_ticks = 20,
  xbreaks = NULL,
  ybreaks = NULL,
  xlim = c(0, NA),
  ylim = NULL,
  col = c("full", "bw"),
  ...
)
```

Arguments

`x`	object of class `evsi`, produced by function `calc_evsi`
`txtsize`	base text size
`currency`	string with currency used in the cost-effectiveness analysis (CEA). Default: $, but it could be any currency symbol or word (e.g., £, €, peso)
`effect_units`	units of effectiveness. Default: QALY
`n_y_ticks`	number of y-axis ticks
`n_x_ticks`	number of x-axis ticks
`xbreaks`	vector of x-axis breaks. will override `n_x_ticks` if provided.
`ybreaks`	vector of y-axis breaks. will override `n_y_ticks` if provided.
`xlim`	vector of x-axis limits, or NULL, which sets limits automatically
`ylim`	vector of y-axis limits, or NULL, which sets limits automatically
`col`	either none, full color, or black and white
`...`	further arguments to plot. This is not used by `dampack` but required for generic consistency.

Value

A `ggplot2` plot with the EVSI

See Also

`calc_evsi`

`plot.exp_loss` *Plot of Expected Loss Curves (ELC)*

Description

Plot of Expected Loss Curves (ELC)

Usage

```
## S3 method for class 'exp_loss'
plot(
  x,
  log_y = TRUE,
  frontier = TRUE,
  points = TRUE,
  lsize = 1,
  txtsize = 12,
  currency = "$",
  effect_units = "QALY",
  n_y_ticks = 8,
  n_x_ticks = 20,
  xbreaks = NULL,
  ybreaks = NULL,
  xlim = c(0, NA),
  ylim = NULL,
  col = c("full", "bw"),
  ...
)
```

Arguments

`x`	object of class `exp_loss`, produced by function `calc_exp_loss`
`log_y`	take the base 10 log of the y axis
`frontier`	indicate the frontier (also the expected value of perfect information). To only plot the EVPI see `calc_evpi`.
`points`	whether to plot points on the curve (TRUE) or not (FALSE)
`lsize`	line size. defaults to 1.
`txtsize`	base text size
`currency`	string with currency used in the cost-effectiveness analysis (CEA). Default: $, but it could be any currency symbol or word (e.g., £, €, peso)
`effect_units`	units of effectiveness. Default: QALY
`n_y_ticks`	number of y-axis ticks
`n_x_ticks`	number of x-axis ticks
`xbreaks`	vector of x-axis breaks. will override `n_x_ticks` if provided.
`ybreaks`	vector of y-axis breaks. will override `n_y_ticks` if provided.
`xlim`	vector of x-axis limits, or NULL, which sets limits automatically
`ylim`	vector of y-axis limits, or NULL, which sets limits automatically
`col`	either none, full color, or black and white
`...`	further arguments to plot. This is not used by `dampack` but required for generic consistency.

Value

A ggplot2 object with the expected loss

---

plot.icers *Plot of ICERs*

Description

Plots the cost-effectiveness plane for a ICER object, calculated with calculate_icers

Usage

```
## S3 method for class 'icers'
plot(
  x,
  txtsize = 12,
  currency = "$",
  effect_units = "QALYs",
  label = c("frontier", "all", "none"),
  label_max_char = NULL,
  plot_frontier_only = FALSE,
  alpha = 1,
  n_x_ticks = 6,
  n_y_ticks = 6,
  xbreaks = NULL,
  ybreaks = NULL,
  xlim = NULL,
  ylim = NULL,
  xexpand = expansion(0.1),
  yexpand = expansion(0.1),
  max.iter = 20000,
  ...
)
```

Arguments

x	Object of class icers.
txtsize	base text size
currency	string. with currency used in the cost-effectiveness analysis (CEA).
effect_units	string. unit of effectiveness
label	whether to label strategies on the efficient frontier, all strategies, or none. defaults to frontier.
label_max_char	max number of characters to label the strategies - if not NULL (the default) longer strategies are truncated to save space.

`plot_frontier_only`
only plot the efficient frontier

`alpha` opacity of points

`n_x_ticks` number of x-axis ticks

`n_y_ticks` number of y-axis ticks

`xbreaks` vector of x-axis breaks. will override `n_x_ticks` if provided.

`ybreaks` vector of y-axis breaks. will override `n_y_ticks` if provided.

`xlim` vector of x-axis limits, or NULL, which sets limits automatically

`ylim` vector of y-axis limits, or NULL, which sets limits automatically

`xexpand` Padding around data. See `scale_continuous` for details.

`yexpand` Padding around data. See `scale_continuous` for details. The default behavior in ggplot2 is `expansion(0.05)`. See `expansion` for how to modify this.

`max.iter` Maximum number of iterations to try to resolve overlaps. Defaults to 10000.

`...` further arguments to plot. This is not used by `dampack` but required for generic consistency.

Value

a ggplot2 object which can be modified by adding additional geoms

plot.owsa *Plot a sensitivity analysis*

Description

Plot a sensitivity analysis

Usage

```
## S3 method for class 'owsa'
plot(
  x,
  txtsize = 12,
  col = c("full", "bw"),
  facet_scales = c("free_x", "free_y", "free", "fixed"),
  facet_nrow = NULL,
  facet_ncol = NULL,
  size = 1,
  n_x_ticks = 6,
  n_y_ticks = 6,
  basecase = NULL,
  ...
)
```

Arguments

x — an owsa object

txtsize — base text size in the plot

col — either full-color ("full") or black and white ("bw")

facet_scales — whether the x or y axes should be fixed. See facet_grid in the ggplo2 package for more details.

facet_nrow — number of rows in plot facet.

facet_ncol — number of columns in plot facet. The default (NULL) is passed to facet_wrap, which determines the number of rows and columns automatically.

size — either point size (ptype = "point") and/or line size (ptype = "line")

n_x_ticks — number of x-axis ticks

n_y_ticks — number of y-axis ticks

basecase — named list of specific values for each parameter to highlight on the returned plot. Each list element must have the same name as the corresponding parameter in the owsa object.

... — further arguments to plot. This is not used by dampack but required for generic consistency.

Value

A ggplot2 plot of the owsa object.

plot.psa *Plot the psa object*

Description

Plot the psa object

Usage

```
## S3 method for class 'psa'
plot(
  x,
  center = TRUE,
  ellipse = TRUE,
  alpha = 0.2,
  txtsize = 12,
  col = c("full", "bw"),
  n_x_ticks = 6,
  n_y_ticks = 6,
  xbreaks = NULL,
  ybreaks = NULL,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`x` the psa object

`center` plot the mean cost and effectiveness for each strategy. defaults to TRUE

`ellipse` plot an ellipse around each strategy. defaults to TRUE

`alpha` opacity of the scatterplot points. 0 is completely transparent, 1 is completely opaque

`txtsize` base text size

`col` either none, full color, or black and white

`n_x_ticks` number of x-axis ticks

`n_y_ticks` number of y-axis ticks

`xbreaks` vector of x-axis breaks. will override `n_x_ticks` if provided.

`ybreaks` vector of y-axis breaks. will override `n_y_ticks` if provided.

`xlim` vector of x-axis limits, or NULL, which sets limits automatically

`ylim` vector of y-axis limits, or NULL, which sets limits automatically

`...` further arguments to plot. This is not used by `dampack` but required for generic consistency.

Value

A `ggplot2` plot of the PSA, showing the distribution of each PSA sample and strategy on the cost-effectiveness plane.

`plot.twsa` *Two-way sensitivity analysis plot*

Description

Two-way sensitivity analysis plot

Usage

```
## S3 method for class 'twsa'
plot(
  x,
  maximize = TRUE,
  col = c("full", "bw"),
  n_x_ticks = 6,
  n_y_ticks = 6,
  txtsize = 12,
  basecase = NULL,
  ...
)
```

Arguments

x	a twsa object
maximize	If TRUE, plot of strategy with maximum expected outcome (default); if FALSE, plot of strategy with minimum expected outcome
col	either none, full color, or black and white
n_x_ticks	number of x-axis ticks
n_y_ticks	number of y-axis ticks
txtsize	base text size
basecase	named list of specific combination of param1 and param2 values to highlight on the returned plot. Each list element must have the same name as the corresponding parameter in the owsa object.
...	further arguments to plot. This is not used by dampack but required for generic consistency.

Value

A ggplot2 plot of the two-way sensitivity analysis.

---

predict.metamodel *Predict from a one-way or two-way metamodel*

Description

Predict from a one-way or two-way metamodel

Usage

```
## S3 method for class 'metamodel'
predict(object, ranges = NULL, nsamp = 100, ...)
```

Arguments

object	object with class "metamodel"
ranges	a named list of the form c("param" = c(0, 1), ...) that gives the ranges for the parameter of interest. If NULL, parameter values from the middle 95 from this range is determined by nsamp.
nsamp	number of samples from ranges
...	further arguments to predict (not used)

Value

a data.frame containing the outcome values predicted by the metamodel for each strategy and each combination of parameter values defined by ranges.

print.metamodel *Print metamodel*

Description

Print metamodel

Usage

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

Arguments

x	metamodel to print
...	further arguments to print

Value

None (invisible NULL)

print.sa *print a psa object*

Description

print a psa object

Usage

```
## S3 method for class 'sa'
print(x, all_strat = FALSE, ...)
```

Arguments

x	the psa object
all_strat	whether or not to print the full list of strategies. defaults to FALSE, which truncates the strategy list to 5
...	further arguments to print (not used)

Value

None (invisible NULL).

`psa_cdiff` *Sample PSA dataset*

Description

Sample PSA dataset

Usage

```
psa_cdiff
```

Format

An object of class `psa` (inherits from `sa`) of length 6.

`rdirichlet` *Random number generation for the Dirichlet distribution with parameter vector alpha.*

Description

Random number generation for the Dirichlet distribution with parameter vector alpha.

Usage

```
rdirichlet(n, alpha)
```

Arguments

`n`	number of observations
`alpha`	vector of parameters defining Dirichlet distribution @importFrom stats rgamma @return A vector random values sampled from a dirichlet distribution @export

run_owsa_det *Run deterministic one-way sensitivity analysis (OWSA)*

Description

This function runs a deterministic one-way sensitivity analysis (OWSA) on a given function that produces outcomes.

Usage

```
run_owsa_det(
  params_range,
  params_basecase,
  nsamp = 100,
  FUN,
  outcomes = NULL,
  strategies = NULL,
  progress = TRUE,
  ...
)
```

Arguments

`params_range`	data.frame with 3 columns in the following order: "pars", "min", and "max". The number of samples from this range is determined by `nsamp`. "pars" are the parameters of interest and must be a subset of the parameters from `params_basecase`.
`params_basecase`	a named list of basecase values for input parameters needed by `FUN`, the user-defined function.
`nsamp`	number of sets of parameter values to be generated. If `NULL`, 100 parameter values are used
`FUN`	function that takes the basecase in `params_basecase` and `...` to produce the `outcome` of interest. The `FUN` must return a dataframe where the first column are the strategy names and the rest of the columns must be outcomes.
`outcomes`	string vector with the outcomes of interest from `FUN` produced by `nsamp`
`strategies`	vector of strategy names. The default `NULL` will use strategy names in `FUN`
`progress`	`TRUE` or `FALSE` for whether or not function progress should be displayed in console.
`...`	Additional arguments to user-defined `FUN`

Value

A list containing dataframes with the results of the sensitivity analyses. The list will contain a dataframe for each outcome specified. List elements can be visualized with `plot.owsa`, `owsa_opt_strat` and `owsa_tornado` from `dampack`

Details

- `params_range`
  - "pars" are the names of the input parameters of interest. These are the parameters that will be varied in the deterministic sensitivity analysis. variables in "pars" column must be a subset of variables in `params_basecase`
  - "min" and "max" are the mininum and maximum values of the parameters of interest.

`run_psa` *Calculate outcomes for a PSA using a user-defined function.*

Description

`run_psa` calculates outcomes using a user-defined function and creates PSA objects corresponding to the specified outcomes.

Usage

```
run_psa(
  psa_samp,
  params_basecase = NULL,
  FUN,
  outcomes = NULL,
  strategies = NULL,
  currency = "$",
  progress = TRUE,
  ...
)
```

Arguments

`psa_samp` A data frame with samples of parameters for a probabilistic sensitivity analysis (PSA)

`params_basecase` a named list of base case values for input parameters needed by `FUN`, the user-defined function.

`FUN` Function that takes the parameter values in `psa_samp` and `...` to produce the `outcome` of interest. The `FUN` must return a data frame where the first column are the strategy names and the rest of the columns must be outcomes.

`outcomes` String vector with the outcomes of interest from `FUN`.

`strategies` vector of strategy names. The default `NULL` will use strategy names in `FUN`

`currency` symbol for the currency being used (ex. "$", "£")

`progress` `TRUE` or `FALSE` for whether or not function progress should be displayed in console.

`...` Additional arguments to user-defined `FUN`

Value

A list containing PSA objects for each outcome in `outcomes`.

See Also

`run_psa`, `make_psa_obj`, `gen_psa_samp`,

`run_twsa_det` *Run deterministic two-way sensitivity analysis (TWSA)*

Description

This function runs a deterministic two-way sensitivity analysis (TWSA) on a given function that produces outcomes.

Usage

```
run_twsa_det(
  params_range,
  params_basecase,
  nsamp = 40,
  FUN,
  outcomes = NULL,
  strategies = NULL,
  progress = TRUE,
  ...
)
```

Arguments

`params_range` data.frame with 2 rows and 3 columns in the following order: "pars", "min", and "max". The number of samples from this range is determined by `nsamp`. "pars" are the 2 parameters of interest, which must be a subset of the parameters from `params_basecase`.

`params_basecase` a named list of base case values for input parameters needed by `FUN`, the user-defined function.

`nsamp` number of parameter values. If `NULL`, 40 parameter values are used

`FUN` Function that takes the base case in `params_all` and `...` to produce the `outcome` of interest. The `FUN` must return a dataframe where the first column are the strategy names and the rest of the columns must be outcomes.

`outcomes` String vector with the outcomes of interest from `FUN` produced by `nsamp`

`strategies` vector of strategy names. The default (NULL) will use strategy names in FUN

`progress` `TRUE` or `FALSE` for whether or not function progress should be displayed in console.

`...` Additional arguments to user-defined `FUN`

Value

A list containing dataframes with the results of the sensitivity analyses. The list will contain a dataframe for each outcome specified.

Details

- `params_range`
  - "pars" are the names of the two input parameters of interest. The two variables in "pars" column must be a subset of variables in `params_basecase`
  - "min" and "max" are the mininum and maximum values of the parameters of interest.

`summary.metamodel` *Summary of metamodel*

Description

Summary of metamodel

Usage

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

Arguments

`object`	metamodel to summarize
`...`	further arguments to summary

Value

a `data.frame` containing the r-squared for each strategy and parameter's metamodel.

`summary.psa` *summarize a psa object across all simulations*

Description

summarize a psa object across all simulations

Usage

```
## S3 method for class 'psa'
summary(object, calc_sds = FALSE, ...)
```

Arguments

`object` the psa object

`calc_sds` whether or not to calculate the standard deviations. Defaults to FALSE

`...` further arguments to summary (not used)

Value

a `data.frame` containing the mean cost and effectiveness for each strategy and, if requested, the standard deviations of the cost and effectiveness for each strategy.

twsa *Two-way sensitivity analysis using linear regression metamodeling*

Description

This function displays a two-way sensitivity analysis (TWSA) graph by estimating a linear regression metamodel of a PSA for a given decision-analytic model

Usage

```
twsa(
  sa_obj,
  param1 = NULL,
  param2 = NULL,
  ranges = NULL,
  nsamp = 100,
  outcome = c("eff", "cost", "nhb", "nmb", "nhb_loss", "nmb_loss"),
  wtp = NULL,
  strategies = NULL,
  poly.order = 2
)
```

Arguments

`sa_obj` sensitivity analysis object; either a probabilistic sensitivity analysis (`make_psa_obj`) or a deterministic sensitivity analysis object (`run_owsa_det`)

`param1` String with the name of the first parameter of interest

`param2` String with the name of the second parameter of interest

`ranges` a named list of the form c("param" = c(0, 1), ...) that gives the ranges for the parameter of interest. If NULL, parameter values from the middle 95 from this range is determined by `nsamp`.

`nsamp` number of samples from ranges

`outcome` either effectiveness ("eff"), cost ("cost"), net health benefit ("nhb"), net monetary benefit ("nmb"), or the opportunity loss in terms of NHB or NMB ("nhb_loss" and "nmb_loss", respectively). "nmb_loss_voi" and "nhb_loss_voi" are only used in internal function calls of metamodel within other VOI functions.

`wtp` if outcome is NHB or NMB (or the associated loss), must provide the willingness-to-pay threshold

`strategies` vector of strategies to consider. The default (NULL) is that all strategies are considered.

`poly.order` order of polynomial for the linear regression metamodel. Default: 2

Value

twsa A `ggplot2` object with the TWSA graph of `param1` and `param2` on the outcome of interest.

Index