Package 'voi'

November 27, 2023

Title Expected Value of Information

Version 1.0.2

Date 2023-11-27

Description Methods to calculate the expected value of information from a decision-analytic model. This includes the expected value of perfect information (EVPI), partial perfect information (EVPPI) and sample information (EVSI), and the expected net benefit of sampling (ENBS). A range of alternative computational methods are provided under the same user interface. See Jackson et al. (2022) <doi:10.1146/annurev-statistics-040120-010730>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

VignetteBuilder knitr

Imports mgcv, earth, mvtnorm, progress, dbarts, posterior, ggplot2, gridExtra, Matrix

Suggests testthat, BCEA, INLA, splancs, sf, knitr, rmarkdown, rjags, truncnorm, scales, dplyr, heemod

RoxygenNote 7.2.3

URL https://chjackson.github.io/voi/

BugReports https://github.com/chjackson/voi/issues

Additional_repositories https://inla.r-inla-download.org/R/stable/

NeedsCompilation no

Author Christopher Jackson [aut, cre],
Anna Heath [aut],
Gianluca Baio [ctb] (Author of code taken from the BCEA package),
Mark Strong [ctb] (Author of code taken from the SAVI package),
Kofi Placid Adragni [ctb] (Author of code taken from the ldr package),
Andrew Raim [ctb] (Author of code taken from the ldr package)

Maintainer Christopher Jackson <chris.jackson@mrc-bsu.cam.ac.uk>

Repository CRAN

Date/Publication 2023-11-27 16:10:02 UTC

R topics documented:

`voi-package` *Methods to calculate the Expected Value of Information*

Description

`evppi` calculates the expected value of partial perfect information from a decision-analytic model. The default, recommended computation methods are based on nonparametric regression. `evpi` is also provided for the expected value of perfect information.

`evsi` calculates the expected value of sample information. Currently this implements the same set of nonparametric regression methods as in `evppi`, and methods based on moment matching and importance sampling. `enbs` can then be used to calculate and optimise the expected net benefit of sampling for a simple study with a fixed upfront cost and per-participant costs.

`evppi` and `evsi` both require a sample of inputs and outputs from a Monte Carlo probabilistic analysis of a decision-analytic model.

Analogous functions `evppivar` and `evsivar` calculate the EVPPI and EVSI for models used for estimation rather than decision-making. The value of information is measured by expected reductions in variance of an uncertain model output of interest.

A pure "brute-force" Monte Carlo method for EVPPI calculation is provided in `evppi_mc`, though this is usually computationally impractical.

The package overview / Get Started vignette gives worked examples of the use of all of these functions.

References

Heath, A., Manolopoulou, I., & Baio, G. (2017). A review of methods for analysis of the expected value of information. Medical Decision Making, 37(7), 747-758.

Heath, A., Kunst, N., Jackson, C., Strong, M., Alarid-Escudero, F., Goldhaber-Fiebert, J. D., Baio, G. Menzies, N.A, Jalal, H. (2020). Calculating the Expected Value of Sample Information in Practice: Considerations from 3 Case Studies. Medical Decision Making, 40(3), 314-326.

Kunst, N., Wilson, E. C., Glynn, D., Alarid-Escudero, F., Baio, G., Brennan, A., Fairley, M., Glynn, D., Goldhaber-Fiebert, J. D., Jackson, C., Jalal, H., Menzies, N. A., Strong, M., Thom, H., Heath, A. (2020). Computing the Expected Value of Sample Information Efficiently: Practical Guidance and Recommendations for Four Model-Based Methods. Value in Health, 3(6), 734-742.

`all_interactions` *Generate a string with all interactions of a certain degree, to be used in a GAM formula*

Description

Generate a string with all interactions of a certain degree, to be used in a GAM formula

Usage

```
all_interactions(x, degree = 2)
```

Arguments

`x`	Character vector of variable names
`degree`	Maximum interaction degree

Value

A string looking like the right hand side of a GAM formula with tensor product interactions.

For example, if `x` is `c("x1","x2","x3")`, then `all_interactions(x, degree=2)` should return

`"te(x1,x2) + te(x1,x3) + te(x1,x3)"`

Examples

```
x <- c("x1","x2","x3")
all_interactions(x, 2)
```

check_regression *Check the fit of a regression model used to estimate EVPPI or EVSI*

Description

Produces diagnostic plots and summaries of regression models used to estimate EVPPI or EVSI, mainly in order to check that the residuals have mean zero.

Usage

```
check_regression(
  x,
  pars = NULL,
  n = NULL,
  comparison = 1,
  outcome = "costs",
  plot = TRUE
)
```

Arguments

x	Output from evppi or evsi. The argument check=TRUE must have been used when calling evppi or evsi, to allow the regression model objects from gam or earth to be preserved. (This is not done by default, since these objects can be large.). attr(x, "models") contains these objects.
pars	Parameter (or parameter group) whose EVPPI calculation is to be checked. This should be in the pars component of the object returned by evppi. Only relevant if x is the result of an evppi calculation. By default, the first calculation shown in x is checked.
n	Sample size whose EVSI calculation is to be checked. This should be in the n component of the object returned by evsi. Only relevant if x is the result of an evsi calculation.
comparison	Only relevant if there are more than two treatments in the decision model. Different regression models are then used for the comparisons of different treatments with the baseline treatment. comparison is an integer identifying which of these models is checked.
outcome	"costs" or "effects". Only relevant if outputs was in cost-effectiveness format when calling evppi or evsi, hence different regressions are used for costs and effects. By default, outcome="costs" is used, so that the regression for costs is checked.
plot	If FALSE, only numerical statistics are returned, and a plot is not made.

Details

For VoI estimation, the key thing we are looking for is that the residuals have mean zero, hence that the mean of the model output is represented well by the regression function of the model input parameters. It should not matter if the variance of the residuals is non-constant, or non-normally distributed.

Models produced with `method="gam"` are summarised using `gam.check`.

Models produced `method="earth"` are summarised using `plot.earth`.

For any regression model, if `fitted()` and `residuals()` methods are defined for those models, then a histogram of the residuals and a scatterplot of residuals against fitted values is produced.

Value

Where possible, an appropriate statistic is returned that allows the regression model to be compared with other regression models implemented using the same `method` but with different assumptions. For `method="gam"`, this is Akaike's information criterion (AIC). For `method="earth"`, this is the generalised cross-validation statistic `gcv`. Currently not implemented for other methods.

Examples

```
pars <- c("p_side_effects_t1", "p_side_effects_t2")
evtest <- evppi(chemo_nb, chemo_pars, pars=pars, check=TRUE)
evtest
check_regression(evtest)

## with no interaction term
evtest2 <- evppi(chemo_nb, chemo_pars, pars=pars,
                 gam_formula="s(p_side_effects_t1)+s(p_side_effects_t2)",
                 check=TRUE)
evtest2
check_regression(evtest2)

## doesn't make much difference to the estimate
## fit is OK in either case
```

---

chemo_cea *Chemotherapy cost-effectiveness model*

Description

An artificial health economic decision model with a typical Markov model structure, used for illustrating Value of Information methods. Functions are provided to generate model parameters and evaluate the model, and samples from probabilistic analysis of the model are provided as built-in datasets.

Usage

```
chemo_cea

chemo_nb

chemo_pars

chemo_cea_501

chemo_constants

chemo_evsi_or

chemo_pars_fn(n)

chemo_model_nb(
  p_side_effects_t1,
  p_side_effects_t2,
  p_hospitalised_total,
  p_died,
  lambda_home,
  lambda_hosp,
  c_home_care,
  c_hospital,
  c_death,
  u_recovery,
  u_home_care,
  u_hospital,
  rate_longterm
)

chemo_model_cea(
  p_side_effects_t1,
  p_side_effects_t2,
  p_hospitalised_total,
  p_died,
  lambda_home,
  lambda_hosp,
  c_home_care,
  c_hospital,
  c_death,
  u_recovery,
  u_home_care,
  u_hospital,
  rate_longterm
)

chemo_model_lor_nb(
```

```
  p_side_effects_t1,
  logor_side_effects,
  p_hospitalised_total,
  p_died,
  lambda_home,
  lambda_hosp,
  c_home_care,
  c_hospital,
  c_death,
  u_recovery,
  u_home_care,
  u_hospital,
  rate_longterm
)

chemo_model_lor_cea(
  p_side_effects_t1,
  logor_side_effects,
  p_hospitalised_total,
  p_died,
  lambda_home,
  lambda_hosp,
  c_home_care,
  c_hospital,
  c_death,
  u_recovery,
  u_home_care,
  u_hospital,
  rate_longterm
)
```

Arguments

n	Number of samples to generate from the uncertainty distribution of the parameters in chemo_pars_fn.
p_side_effects_t1	Probability of side effects under treatment 1
p_side_effects_t2	Probability of side effects under treatment 2
p_hospitalised_total	Probability of hospitalisation in the year after receiving treatment
p_died	Probability of death in the year after receiving treatment
lambda_home	Recovery probability for someone treated at home
lambda_hosp	Recovery probability for someone treated in hospital who does not die
c_home_care	Cost of a yearly period under treatment at home
c_hospital	Cost of hospital treatment

c_death	Cost of death
u_recovery	Utility of a period in the recovery state
u_home_care	Utility of home care state
u_hospital	Utility of hospital state
rate_longterm	Long term mortality rate
logor_side_effects	Log odds ratio of side effects for treatment 2 compared to 1

Format

An object of class list of length 33.

An object of class evsi (inherits from data.frame) with 15030 rows and 3 columns.

Samples of 10000 from probabilistic analysis of this model are made available in the package, in the following data objects:

chemo_pars: Sample from the distributions of the parameters, as a data frame with names as documented above.

chemo_cea: List with components e (sampled effects), c (sampled costs), and k (a set of five equally-spaced willingess-to-pay values from 10000 to 50000 pounds). The effects and costs are data frames with two columns, one for each decision option.

chemo_nb: Data frame with two columns, giving the net monetary benefit for each decision option, at a willingness-to-pay of 20000 pounds.

chemo_cea_501: List with components e (sampled effects), c (sampled costs), and k (a set of 501 willlingess-to-pay values from 10000 to 50000) This is provided to facilitate illustrations of plots of VoI measures against willingness-to-pay.

The following additional data objects are supplied:

chemo_constants includes various constants required by the code.

chemo_evsi_or is the result of an EVSI analysis to estimate the expected value of a two-arm trial, with a binary outcome, to estimate the log odds ratio of side effects. This object is a data frame with three columns, giving the sample size per arm (n), willingness-to-pay (k) and the corresponding EVSI (evsi).

Details

For more details, refer to Heath et al. (forthcoming book...)

Value

Two alternative functions are provided to evaluate the decision model for given parameters.

chemo_model_nb returns a vector with elements giving the net monetary benefit for standard of care and novel treatment, respectively, at a willingness-to-pay of 20,000 pounds per QALY.

chemo_model_cea returns a matrix with:

- two rows, the first for expected costs and the second for expected effects (QALYs) over the fifty year time horizon, and

- two columns, the first for the "standard of care" decision option, and the second for the novel treatment.

`chemo_model_lor_nb` and `chemo_model_lor_cea` are the same model, but parameterised in terms of the probability of side effects for the standard of care `p_side_effects_t1` and the log odds ratio of side effects between treatment groups `logor_side_effects`, rather than in terms of `p_side_effects_t1` and `p_side_effects_t2`

`chemo_pars_fn` generates a sample from the uncertainty distribution of the parameters in the chemotherapy model . This returns a data frame with parameters matching the arguments of `chemo_model_nb`, and the following additional derived parameters:

- `p_side_effects_t2`:
- `p_hospitalised_total`: probability of hospitalisation over the 50 year time horizon
- `p_died`: probability of death over the time horizon, given hospitalisation
- `lambda_home`: conditional probability that a patient recovers given they are not hospitalised
- `lambda_hosp`: conditional probability that a patient in hospital recovers given they do not die

References

Value of Information for Healthcare Decision Making (CRC Press, eds. Heath, Kunst and Jackson: forthcoming)

`enbs` *Expected net benefit of sampling*

Description

Calculates the expected net benefit of sampling for a typical study to inform a health economic evaluation, given estimates of the per-person expected value of sample information, decision population size and study setup and per-participant costs. The optimal sample size for each willingness-to-pay, population size and time horizon is also determined.

Usage

```
enbs(
  evsi,
  costs_setup,
  costs_pp,
  pop,
  time,
  dis = 0.035,
  smooth = FALSE,
  smooth_df = NULL,
  pcut = 0.05
)
```

Arguments

evsi	Data frame giving estimates of the expected value of sample information, as returned by evsi. This may contain multiple estimates, one for each sample size and willingness to pay.
costs_setup	Setup costs of the study. This can either be a constant, or a vector of two elements giving a 95% credible interval (with mean defined by the midpoint), or a vector of three elements assumed to define the mean and 95% credible interval.
costs_pp	Per-participant costs of the study, supplied in the same format as cost_setup.
pop	Size of the population who would be affected by the decision.
time	Time horizon over which discounting will be applied.
dis	Discount rate used when converting per-person to population EVSI.
smooth	If TRUE, then the maximum ENBS is determined after fitting a nonparametric regression to the data frame x, which estimates and smooths the ENBS for every integer sample size in the range of x$n. The regression is done using the default settings of gam from the **mgcv** package. If this is FALSE, then no smoothing or interpolation is done, and the maximum is determined by searching over the values supplied in x.
smooth_df	Basis dimension for the smooth regression. Passed as the k argument to the s() term in gam. Defaults to 6, or the number of unique sample sizes minus 1 if this is lower. Set to a higher number if you think the smoother does not capture the relation of ENBS to sample size accurately enough.
pcut	Cut-off probability which defines a "near-optimal" sample size. The minimum and maximum sample size for which the ENBS is within pcut (by default 5%) of its maximum value will be determined.

Details

pop,time and dis may be supplied as vectors of different lengths. In that case, the ENBS is calculated for all possible combinations of the values in these vectors.

Value

Data frame with components enbs giving the ENBS, and sd giving the corresponding standard deviation. The rows of the data frame correspond to the rows of evsi, and any n and k are inherited from evsi. Additional columns include:

pce: the probability that the study is cost-effective, i.e. that the ENBS is positive, obtained from a normal distribution defined by the estimate and standard deviation.

enbsmax: The maximum ENBS for each willingness-to-pay k.

nmax: The sample size n at which this maximum is achieved.

A second data frame is returned as the "enbsmax" attribute. This has one row per willingness-to-pay (k), giving the optimal ENBS (enbsmax) the optimal sample size (nmax) and an interval estimate for the optimal sample size (nlower to nupper).

If pop, time or dis were supplied as vectors of more than one element, then additional columns will be returned in these data frames to identify the population, time or discount rate for each ENBS calculation. An index ind is also returned to identify the unique combination that each row refers to.

References

Value of Information for Healthcare Decision Making (CRC Press, eds. Heath, Kunst and Jackson: forthcoming)

`enbs_opt` *Determine the optimum sample size in an analysis of the expected net benefit of sampling*

Description

The optimum sample size for a given willingness to pay is determined either by a simple search over the supplied ENBS estimates for different sample sizes, or by a regression and interpolation method.

Usage

```
enbs_opt(x, pcut = 0.05, smooth = FALSE, smooth_df = NULL, keep_preds = FALSE)
```

Arguments

`x`	Data frame containing a set of ENBS estimates for different sample sizes, which will be optimised over. Usually this is for a common willingness-to-pay. The required components are `enbs` and `n`.
`pcut`	Cut-off probability which defines a "near-optimal" sample size. The minimum and maximum sample size for which the ENBS is within `pcut` (by default 5%) of its maximum value will be determined.
`smooth`	If `TRUE`, then the maximum ENBS is determined after fitting a nonparametric regression to the data frame `x`, which estimates and smooths the ENBS for every integer sample size in the range of `x$n`. The regression is done using the default settings of `gam` from the **mgcv** package. If this is `FALSE`, then no smoothing or interpolation is done, and the maximum is determined by searching over the values supplied in `x`.
`smooth_df`	Basis dimension for the smooth regression. Passed as the `k` argument to the `s()` term in `gam`. Defaults to 6, or the number of unique sample sizes minus 1 if this is lower. Set to a higher number if you think the smoother does not capture the relation of ENBS to sample size accurately enough.
`keep_preds`	If `TRUE` and `smooth=TRUE` then the data frame of predictions from the smooth regression model is stored in the `"preds"` attribute of the result.

Value

A data frame with one row, and the following columns:

`ind`: An integer index identifying, e.g. the willingness to pay and other common characteristics of the ENBS estimates (e.g. incident population size, decision time horizon). This is copied from `x$ind`.

enbsmax: the maximum ENBS

nmax: the sample size at which this maximum is achieved

nlower: the lowest sample size for which the ENBS is within

pcut (default 5%) of its maximum value

nupper: the corresponding highest ENBS

---

evpi *Calculate the expected value of perfect information from a decision model*

Description

Calculate the expected value of perfect information from a decision model using standard Monte Carlo simulation

Usage

```
evpi(outputs, nsim = NULL)
```

Arguments

outputs	This could take one of two forms "net benefit" form: a matrix or data frame of samples from the uncertainty distribution of the expected net benefit. The number of rows should equal the number of samples, and the number of columns should equal the number of decision options. "cost-effectiveness analysis" form: a list with the following named components: "c": a matrix or data frame of samples from the distribution of costs. There should be one column for each decision option. "e": a matrix or data frame of samples from the distribution of effects, likewise. "k": a vector of willingness-to-pay values. Objects of class "bcea", as created by the **BCEA** package, are in this "cost-effectiveness analysis" format, therefore they may be supplied as the outputs argument. Users of **heemod** can create an object of this form, given an object produced by run_psa (obj, say), with import_heemod_outputs. If outputs is a matrix or data frame, it is assumed to be of "net benefit" form. Otherwise if it is a list, it is assumed to be of "cost effectiveness analysis" form.
nsim	Number of simulations from the decision model to use for calculating EVPPI. The first nsim rows of the objects in inputs and outputs are used.

Value

The expected value of perfect information, either as a single value, or a data frame indicating the value for each willingness-to-pay.

evppi *Calculate the expected value of partial perfect information from a decision-analytic model*

Description

Calculate the expected value of partial perfect information from a decision-analytic model

Usage

```
evppi(
  outputs,
  inputs,
  pars = NULL,
  method = NULL,
  se = FALSE,
  B = 1000,
  nsim = NULL,
  verbose = FALSE,
  check = FALSE,
  ...
)
```

Arguments

outputs	This could take one of two forms "net benefit" form: a matrix or data frame of samples from the uncertainty distribution of the expected net benefit. The number of rows should equal the number of samples, and the number of columns should equal the number of decision options. "cost-effectiveness analysis" form: a list with the following named components: "c": a matrix or data frame of samples from the distribution of costs. There should be one column for each decision option. "e": a matrix or data frame of samples from the distribution of effects, likewise. "k": a vector of willingness-to-pay values. Objects of class "bcea", as created by the **BCEA** package, are in this "cost-effectiveness analysis" format, therefore they may be supplied as the outputs argument. Users of **heemod** can create an object of this form, given an object produced by run_psa (obj, say), with import_heemod_outputs. If outputs is a matrix or data frame, it is assumed to be of "net benefit" form. Otherwise if it is a list, it is assumed to be of "cost effectiveness analysis" form.
inputs	Matrix or data frame of samples from the uncertainty distribution of the input parameters of the decision model. The number of columns should equal the number of parameters, and the columns should be named. This should have

the same number of rows as there are samples in `outputs`, and each row of the samples in `outputs` should give the model output evaluated at the corresponding parameters.

Users of **heemod** can create an object of this form, given an object produced by `run_psa` (`obj`, say), with `import_heemod_inputs`.

`pars` Either a character vector, or a list of character vectors.

If a character vector is supplied, then a single, joint EVPPI calculation is done with for the parameters named in this vector.

If a list of character vectors is supplied, then multiple EVPPI calculations are performed, one for each list component defined in the above vector form.

`pars` must be specified if `inputs` is a matrix or data frame. This should then correspond to particular columns of `inputs`. If `inputs` is a vector, this is assumed to define the single parameter of interest, and then `pars` is not required.

`method` Character string indicating the calculation method. If one string is supplied, this is used for all calculations. A vector of different strings can be supplied if a different method is desired for different list components of `pars`.

The default methods are based on nonparametric regression:

`"gam"` for a generalized additive model implemented in the `gam` function from the **mgcv** package. This is the default method for calculating the EVPPI of 4 or fewer parameters.

`"gp"` for a Gaussian process regression, as described by Strong et al. (2014) and implemented in the **SAVI** package (`https://github.com/Sheffield-Accelerated-VoI/SAVI`). This is the default method for calculating the EVPPI of more than 4 parameters.

`"inla"` for an INLA/SPDE Gaussian process regression method, from Heath et al. (2016).

`"bart"` for Bayesian additive regression trees, using the **dbarts** package. Particularly suited for joint EVPPI of many parameters.

`"earth"` for a multivariate adaptive regression spline with the **earth** package (Milborrow, 2019).

`"so"` for the method of Strong and Oakley (2013). Only supported for single parameter EVPPI.

`"sal"` for the method of Sadatsafavi et al. (2013). Only supported for single parameter EVPPI.

`se` If this is `TRUE`, calculate a standard error for the EVPPI if possible. Currently only supported for methods `"gam"`, `"earth"` and `method="bart"`. (In the latter method it is more correctly called a posterior standard deviation). These represent uncertainty about the parameters of the fitted regression model, and will naturally be lower when more simulations from the decision model are used to fit it. They do not represent uncertainty about the structure of the regression model,

`B` Number of parameter replicates for calculating the standard error. Only applicable to `method="gam"`. For `method="bart"` the analogous quantity is the number of MCMC samples, which is controlled by the `ndpost` argument to `bart`, which can be passed as an argument to `evppi`.

nsim — Number of simulations from the decision model to use for calculating EVPPI. The first nsim rows of the objects in inputs and outputs are used.

verbose — If TRUE, then messages are printed describing each step of the calculation, if the method supplies these. Can be useful to see the progress of slow calculations.

check — If TRUE, then extra information about the estimation is saved inside the object that this function returns. This currently only applies to the regression-based methods "gam" and "earth" where the fitted regression model objects are saved. This allows use of the check_regression function, which produces some diagnostic checks of the regression models.

... — Other arguments to control specific methods.

For method="gam", the following arguments can be supplied:

- gam_formula: a character string giving the right hand side of the formula supplied to the gam() function. By default, this is a tensor product of all the parameters of interest, e.g. if pars = c("pi","rho"), then gam_formula defaults to t(pi, rho,bs="cr"). The option bs="cr" indicates a cubic spline regression basis, which is more computationally efficient than the default "thin plate" basis. If there are four or more parameters of interest, then the additional argument k=4 is supplied to te(), specifying a four-dimensional basis, which is currently the default in the SAVI package.
  If there are spaces in the variable names in inputs, then these should be converted to underscores before forming an explicit gam_formula.

For method="gp", the following arguments can be supplied:

- gp_hyper_n: number of samples to use to estimate the hyperparameters in the Gaussian process regression method. By default, this is the minimum of the following three quantities: 30 times the number of parameters of interest, 250, and the number of simulations being used for calculating EVPPI.
- maxSample: Maximum sample size to employ for method="gp". Only increase this from the default 5000 if your computer has sufficent memory to invert square matrices with this dimension.

For method="inla", the following arguments can be supplied, as described in detail in Baio, Berardi and Heath (2017):

- int.ord (integer) maximum order of interaction terms to include in the regression predictor, e.g. if int.ord=k then all k-way interactions are used. Currently this applies to both effects and costs.
- cutoff (default 0.3) controls the density of the points inside the mesh in the spatial part of the mode. Acceptable values are typically in the interval (0.1,0.5), with lower values implying more points (and thus better approximation and greatercomputational time).
- convex.inner (default = -0.4) and convex.outer (default = -0.7) control the boundaries for the mesh. These should be negative values and can be decreased (say to -0.7 and -1, respectively) to increase the distance between the points and the outer boundary, which also increases precision and computational time.
- robust. if TRUE then INLA will use a t prior distribution for the coefficients of the linear predictor, rather than the default normal distribution.

- h.value (default=0.00005) controls the accuracy of the INLA grid-search for the estimation of the hyperparameters. Lower values imply a more refined search (and hence better accuracy), at the expense of computational speed.
- plot_inla_mesh (default FALSE) Produce a plot of the mesh.
- max.edge Largest allowed triangle edge length when constructing the mesh, passed to inla.mesh.2d.
- pfc_struc Variance structure to pass to pfc in the **ldr** package for principal fitted components. The default "AIC" selects the one that fits best given two basis terms. Change this to, e.g. "iso", "aniso" or "unstr" if an "Error in eigen..." is obtained.

For any of the nonparametric regression methods:

- ref The reference decision option used to define the incremental net benefit, cost or effects before performing nonparametric regression. Either an integer column number, or the name of the column from outputs.

For method="so":

- n.blocks Number of blocks to split the sample into. Required.

For method="sal":

- n.seps Number of separators (default 1).

Value

A data frame with a column pars, indicating the parameter(s), and a column evppi, giving the corresponding EVPPI.

If outputs is of "cost-effectiveness analysis" form, so that there is one EVPPI per willingness-to-pay value, then a column k identifies the willingness-to-pay.

If standard errors are requested, then the standard errors are returned in the column se.

References

Strong, M., Oakley, J. E., & Brennan, A. (2014). Estimating multiparameter partial expected value of perfect information from a probabilistic sensitivity analysis sample: a nonparametric regression approach. Medical Decision Making, 34(3), 311-326.

Heath, A., Manolopoulou, I., & Baio, G. (2016). Estimating the expected value of partial perfect information in health economic evaluations using integrated nested Laplace approximation. Statistics in Medicine, 35(23), 4264-4280.

Baio, G., Berardi, A., & Heath, A. (2017). Bayesian cost-effectiveness analysis with the R package BCEA. New York: Springer.

Milborrow, S. (2019) earth: Multivariate Adaptive Regression Splines. R package version 5.1.2. Derived from mda:mars by Trevor Hastie and Rob Tibshirani. Uses Alan Miller's Fortran utilities with Thomas Lumley's leaps wrapper. https://CRAN.R-project.org/package=earth.

Strong, M., & Oakley, J. E. (2013). An efficient method for computing single-parameter partial expected value of perfect information. Medical Decision Making, 33(6), 755-766. Chicago

Sadatsafavi, M., Bansback, N., Zafari, Z., Najafzadeh, M., & Marra, C. (2013). Need for speed: an efficient algorithm for calculation of single-parameter expected value of partial perfect information. Value in Health, 16(2), 438-448.

`evppivar` *Calculate the expected value of partial perfect information for an estimation problem*

Description

Calculate the expected value of partial perfect information for an estimation problem. This computes the expected reduction in variance in some quantity of interest with perfect information about a parameter or parameters of interest.

Usage

```
evppivar(
  outputs,
  inputs,
  pars = NULL,
  method = NULL,
  nsim = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`outputs`	a vector of values for the quantity of interest, sampled from the uncertainty distribution of this quantity that is induced by the uncertainty about the parameters. This can also be a data frame with one column. Typically this will come from a Monte Carlo sample, where we first sample from the uncertainty distributions of the parameters, and then compute the quantity of interest as a function of the parameters. It might also be produced by a Markov Chain Monte Carlo sample from the joint distribution of parameters and outputs.
`inputs`	Matrix or data frame of samples from the uncertainty distribution of the input parameters of the decision model. The number of columns should equal the number of parameters, and the columns should be named. This should have the same number of rows as there are samples in `outputs`, and each row of the samples in `outputs` should give the model output evaluated at the corresponding parameters. Users of **heemod** can create an object of this form, given an object produced by `run_psa` (`obj`, say), with `import_heemod_inputs`.
`pars`	Either a character vector, or a list of character vectors. If a character vector is supplied, then a single, joint EVPPI calculation is done with for the parameters named in this vector. If a list of character vectors is supplied, then multiple EVPPI calculations are performed, one for each list component defined in the above vector form. `pars` must be specified if `inputs` is a matrix or data frame. This should then correspond to particular columns of `inputs`. If `inputs` is a vector, this is assumed to define the single parameter of interest, and then `pars` is not required.

`method` Character string indicating the calculation method. If one string is supplied, this is used for all calculations. A vector of different strings can be supplied if a different method is desired for different list components of `pars`.

The default methods are based on nonparametric regression:

`"gam"` for a generalized additive model implemented in the `gam` function from the **mgcv** package. This is the default method for calculating the EVPPI of 4 or fewer parameters.

`"gp"` for a Gaussian process regression, as described by Strong et al. (2014) and implemented in the **SAVI** package (https://github.com/Sheffield-Accelerated-VoI/SAVI). This is the default method for calculating the EVPPI of more than 4 parameters.

`"inla"` for an INLA/SPDE Gaussian process regression method, from Heath et al. (2016).

`"bart"` for Bayesian additive regression trees, using the **dbarts** package. Particularly suited for joint EVPPI of many parameters.

`"earth"` for a multivariate adaptive regression spline with the **earth** package (Milborrow, 2019).

`"so"` for the method of Strong and Oakley (2013). Only supported for single parameter EVPPI.

`"sal"` for the method of Sadatsafavi et al. (2013). Only supported for single parameter EVPPI.

`nsim` Number of simulations from the decision model to use for calculating EVPPI. The first `nsim` rows of the objects in `inputs` and `outputs` are used.

`verbose` If `TRUE`, then messages are printed describing each step of the calculation, if the method supplies these. Can be useful to see the progress of slow calculations.

`...` Other arguments to control specific methods.

For `method="gam"`, the following arguments can be supplied:

- `gam_formula`: a character string giving the right hand side of the formula supplied to the `gam()` function. By default, this is a tensor product of all the parameters of interest, e.g. if `pars = c("pi","rho")`, then `gam_formula` defaults to `t(pi, rho,bs="cr")`. The option `bs="cr"` indicates a cubic spline regression basis, which is more computationally efficient than the default "thin plate" basis. If there are four or more parameters of interest, then the additional argument `k=4` is supplied to `te()`, specifying a four-dimensional basis, which is currently the default in the SAVI package.
  If there are spaces in the variable names in `inputs`, then these should be converted to underscores before forming an explicit `gam_formula`.

For `method="gp"`, the following arguments can be supplied:

- `gp_hyper_n`: number of samples to use to estimate the hyperparameters in the Gaussian process regression method. By default, this is the minimum of the following three quantities: 30 times the number of parameters of interest, 250, and the number of simulations being used for calculating EVPPI.
- `maxSample`: Maximum sample size to employ for `method="gp"`. Only increase this from the default 5000 if your computer has sufficent memory to invert square matrices with this dimension.

For `method="inla"`, the following arguments can be supplied, as described in detail in Baio, Berardi and Heath (2017):

- `int.ord` (integer) maximum order of interaction terms to include in the regression predictor, e.g. if `int.ord=k` then all k-way interactions are used. Currently this applies to both effects and costs.
- `cutoff` (default 0.3) controls the density of the points inside the mesh in the spatial part of the mode. Acceptable values are typically in the interval (0.1,0.5), with lower values implying more points (and thus better approximation and greatercomputational time).
- `convex.inner` (default = -0.4) and `convex.outer` (default = -0.7) control the boundaries for the mesh. These should be negative values and can be decreased (say to -0.7 and -1, respectively) to increase the distance between the points and the outer boundary, which also increases precision and computational time.
- `robust`. if `TRUE` then INLA will use a t prior distribution for the coefficients of the linear predictor, rather than the default normal distribution.
- `h.value` (default=0.00005) controls the accuracy of the INLA grid-search for the estimation of the hyperparameters. Lower values imply a more refined search (and hence better accuracy), at the expense of computational speed.
- `plot_inla_mesh` (default `FALSE`) Produce a plot of the mesh.
- `max.edge` Largest allowed triangle edge length when constructing the mesh, passed to `inla.mesh.2d`.
- `pfc_struc` Variance structure to pass to `pfc` in the **ldr** package for principal fitted components. The default `"AIC"` selects the one that fits best given two basis terms. Change this to, e.g. `"iso"`, `"aniso"` or `"unstr"` if an "Error in eigen..." is obtained.

For any of the nonparametric regression methods:

- `ref` The reference decision option used to define the incremental net benefit, cost or effects before performing nonparametric regression. Either an integer column number, or the name of the column from `outputs`.

For `method="so"`:

- `n.blocks` Number of blocks to split the sample into. Required.

For `method="sal"`:

- `n.seps` Number of separators (default 1).

Value

A data frame with a column `pars`, indicating the parameter(s), and a column `evppi`, giving the corresponding EVPPI.

References

Jackson, C., Presanis, A., Conti, S., & De Angelis, D. (2019). Value of information: Sensitivity analysis and research design in Bayesian evidence synthesis. Journal of the American Statistical Association, 114(528), 1436-1449.

Jackson, C., Johnson, R., de Nazelle, A., Goel, R., de Sa, T. H., Tainio, M., & Woodcock, J. (2021). A guide to value of information methods for prioritising research in health impact modelling. Epidemiologic Methods, 10(1).

Jackson, C. H., Baio, G., Heath, A., Strong, M., Welton, N. J., & Wilson, E. C. (2022). Value of Information analysis in models to inform health policy. Annual Review of Statistics and its Application, 9, 95-118.

`evppi_mc` *Traditional two-level Monte Carlo estimator of EVPPI.*

Description

Traditional two-level Monte Carlo estimator of the expected value of partial perfect information from a decision-analytic model. Only useful in the simplest of examples. For realistically complex examples, the methods implemented in the `evppi` function, based on regression, will usually be much more computationally efficient.

Usage

```
evppi_mc(
  model_fn,
  par_fn,
  pars,
  nouter,
  ninner,
  k = NULL,
  mfargs = NULL,
  verbose = FALSE
)
```

Arguments

`model_fn`	A function to evaluate a decision-analytic model at a given set of parameters. This should have one argument per parameter, and return either: (net benefit format) a vector giving the net benefit for each decision option, or (cost-effectiveness analysis format) a matrix or data frame with two rows, and one column for each decision option. If the rows have names `"e"` and `"c"` then these are assumed to be the effects and costs respectively. Otherwise, the first row is assumed to be the effects, and the second the costs.
`par_fn`	A function to generate a random sample of values for the parameters of `model_fn`. This should return a matrix or a data frame with named columns matching the arguments of `model_fn`. If any required arguments to `model_fn` are not supplied in this return value, then `evppi_mc` looks for them in the list supplied as the `mfargs` argument.

If any required arguments are not found in the results of `par_fn` or `mfargs`, and if `model_fn` defines default values for those arguments, then those default values are used.

The first argument of `par_fn` should be an integer `n` denoting the number of random values to draw for each parameter. The object returned by `par_fn` should then have `n` rows, and one column for each parameter. If one value is drawn, then `par_fn` is also allowed to return a vector, but this should still be named.

The parameters may be correlated. If we wish to compute the EVPPI for a parameter which is correlated with a different parameter `q`, then `par_fn` must have an argument with the name of that parameter. If that argument is set to a fixed value, then `par_fn` should return a sample drawn conditionally on that value. If that argument is not supplied, then `par_fn` must return a sample drawn from the marginal distribution. See the vignette for an example.

`pars` A character vector giving the parameters of interest, for which the EVPPI is required. This should correspond to an explicit argument to `model_fn`.

The parameters of interest are assumed to have uncertainty distributions that are independent of those of the other parameters.

`nouter` Number of outer samples

`ninner` Number of inner samples

`k` Vector of willingness-to-pay values. Only used if `model_fn` is in cost-effectiveness analyis format.

`mfargs` Named list of additional arguments to supply to `model_fn`.

`verbose` Set to `TRUE` to print some additional messages to help with debugging.

Details

See the package overview / Get Started vignette for an example of using this function.

Value

A data frame with a column `pars`, indicating the parameter(s), and a column `evppi`, giving the corresponding EVPPI.

If `outputs` is of "cost-effectiveness analysis" form, so that there is one EVPPI per willingness-to-pay value, then a column `k` identifies the willingness-to-pay.

`evsi` *Calculate the expected value of sample information from a decision-analytic model*

Description

Calculate the expected value of sample information from a decision-analytic model

Usage

```
evsi(
  outputs,
  inputs,
  study = NULL,
  datagen_fn = NULL,
  pars = NULL,
  pars_datagen = NULL,
  n = 100,
  aux_pars = NULL,
  method = NULL,
  likelihood = NULL,
  analysis_fn = NULL,
  analysis_args = NULL,
  model_fn = NULL,
  par_fn = NULL,
  Q = 50,
  npreg_method = "gam",
  nsim = NULL,
  verbose = FALSE,
  check = FALSE,
  ...
)
```

Arguments

outputs	This could take one of two forms "net benefit" form: a matrix or data frame of samples from the uncertainty distribution of the expected net benefit. The number of rows should equal the number of samples, and the number of columns should equal the number of decision options. "cost-effectiveness analysis" form: a list with the following named components: "c": a matrix or data frame of samples from the distribution of costs. There should be one column for each decision option. "e": a matrix or data frame of samples from the distribution of effects, likewise. "k": a vector of willingness-to-pay values. Objects of class "bcea", as created by the **BCEA** package, are in this "cost-effectiveness analysis" format, therefore they may be supplied as the outputs argument. Users of **heemod** can create an object of this form, given an object produced by run_psa (obj, say), with import_heemod_outputs. If outputs is a matrix or data frame, it is assumed to be of "net benefit" form. Otherwise if it is a list, it is assumed to be of "cost effectiveness analysis" form.
inputs	Matrix or data frame of samples from the uncertainty distribution of the input parameters of the decision model. The number of columns should equal the number of parameters, and the columns should be named. This should have the same number of rows as there are samples in outputs, and each row of the

samples in outputs should give the model output evaluated at the corresponding parameters.

Users of **heemod** can create an object of this form, given an object produced by run_psa (obj, say), with import_heemod_inputs.

study

Name of one of the built-in study types supported by this package for EVSI calculation. If this is supplied, then the columns of inputs that correspond to the parameters governing the study data should be identified in pars.

Current built-in studies are

"binary" A study with a binary outcome observed on one sample of individuals. Requires one parameter: the probability of the outcome. The sample size is specifed in the n argument to evsi(), and the binomially-distributed outcome is named X1.

"trial_binary" Two-arm trial with a binary outcome. Requires two parameters: the probability of the outcome in arm 1 and 2 respectively. The sample size is the same in each arm, specifed in the n argument to evsi(), and the binomial outcomes are named X1 and X2 respectively.

"normal_known" A study of a normally-distributed outcome, with a known standard deviation, on one sample of individuals. Likewise the sample size is specified in the n argument to evsi(). The standard deviation defaults to 1, and can be changed by specifying sd as a component of the aux_pars argument, e.g. evsi(..., aux_pars=list(sd=2)).

Either study or datagen_fn should be supplied to evsi().

For the EVSI calculation methods where explicit Bayesian analyses of the simulated data are performed, the prior parameters for these built-in studies are supplied in the analysis_args argument to evsi(). These assume Beta priors for probabilities, and Normal priors for the mean of a normal outcome.

datagen_fn

If the proposed study is not one of the built-in types supported, it can be specified in this argument as an R function to sample predicted data from the study. This function should have the following specification:

1. the function's first argument should be a data frame of parameter simulations, with one row per simulation and one column per parameter. The parameters in this data frame must all be found in inputs, but need not necessarily be in the same order or include all of them.
2. the function should return a data frame.
3. the returned data frame should have number of rows equal to the number of parameter simulations in inputs.
4. if inputs is considered as a sample from the posterior, then datagen_fn(inputs) returns a corresponding sample from the posterior predictive distribution, which includes two sources of uncertainty: (a) uncertainty about the parameters and (b) sampling variation in observed data given fixed parameter values.
5. the function can optionally have more than one argument. If so, these additional arguments should be given default values in the definition of datagen_fn. If there is an argument called n, then it is interpreted as the sample size for the proposed study.

pars — Character vector identifying which parameters are learned from the proposed study. This is required for the moment matching and importance sampling methods, and these should be columns of inputs. This is not required for the nonparametric regression methods.

pars_datagen — Character vector identifying which columns of inputs are the parameters required to generate data from the proposed study. These should be columns of inputs.

If pars_datagen is not supplied, then it is assumed to be the same as pars. Note that these can be different. Even if the study data are generated by a particular parameter, when analysing the data we could choose to ignore the information that the data provides about that parameter.

n — Sample size of future study, or vector of alternative sample sizes. This is understood by the built-in study designs. For studies specified by the user with datagen_fn, if datagen_fn has an argument n, then this is interpreted as the sample size. However if calling evsi for a user-specified design where datagen_fn does not have an n argument, then any n argument supplied to evsi will be ignored.

Currently this shortcut is not supported if more than one quantity is required to describe the sample size, for example, trials with unbalanced arms. In that case, you will have to hard-code the required sample sizes into datagen_fn.

For the nonparametric regression and importance sampling methods, the computation is simply repeated for each sample size supplied here.

The moment matching method uses a regression model to estimate the dependency of the EVSI on the sample size, hence to enable EVSI to be calculated efficiently for any number of sample sizes (Heath et al. 2019).

aux_pars — A list of additional fixed arguments to supply to the function to generate the data, whether that is a built-in study design or user-defined function supplied in datagen_fn. For example, evsi(..., aux_pars = list(sd=2)) defines the fixed standard deviation in the "normal_known" model.

method — Character string indicating the calculation method. Defaults to "gam".

All the nonparametric regression methods supported for evppi, that is "gam","gp","earth","inla", can also be used for EVSI calculation by regressing on a summary statistic of the predicted data (Strong et al 2015).

"is" for importance sampling (Menzies 2016)

"mm" for moment matching (Heath et al 2018)

Note that the "is" and "mm" methods are used in conjunction with nonparametric regression, and the gam_formula argument can be supplied to evsi to specify this regression - see evppi for documentation of this argument.

likelihood — Likelihood function, required (and only required) for the importance sampling method when a study design other than one of the built-in ones is used. This should have two arguments, named as follows:

Y: a one-row data frame of predicted data. Columns are defined by different outcomes in the data, with names matching the names of the data frame returned by datagen_fn.

inputs. a data frame of simulated parameter values. Columns should correspond to different variables in inputs. The column names should all be found

in the names of inputs, though they do not have to be in the same order, or include everything in inputs. The number or rows should be the same as the number of rows in inputs.

The function should return a vector whose length matches the number of rows of the parameters data frame given as the second argument. Each element of the vector gives the likelihood of the corresponding set of parameters, given the data in the first argument. An example is given in the vignette.

The likelihood can optionally have a n argument, which is interpreted as the sample size of the study. If the n argument to evsi is used then this is passed to the likelihood function. Conversely any n argument to evsi will be ignored by a likelihood function that does not have its own n argument.

Note the definition of the likelihood should agree with the definition of datagen_fn to define a consistent sampling distribution for the data. No automatic check is performed for this.

analysis_fn Function which fits a Bayesian model to the generated data. Required for method="mm" if a study design other than one of the built-in ones is used. This should be a function that takes the following arguments:

data: A data frame with names matching the output of datagen_fn

args: A list with constants required in the Bayesian analysis, e.g. prior parameters, or options for the analysis, e.g. number of MCMC simulations. The component of this list called n is assumed to contain the sample size of the study.

pars Names of the parameters whose posterior is being sampled.

The function should return a data frame with names matching pars, containing a sample from the posterior distribution of the parameters given data supplied through data.

analysis_fn is required to have all three of these arguments, but you do not need to use any elements of args or pars in the body of analysis_fn. Instead, sample sizes, prior parameters, MCMC options and parameter names can alternatively be hard-coded inside analysis_fn. Passing these through the function arguments (via the analysis_args argument to evsi) is only necessary if we want to use the same analysis_fn to do EVSI calculations with different sample sizes or other settings.

analysis_args List of arguments required for the Bayesian analysis of the predicted data, e.g. definitions of the prior and options to control sampling. Only used in method="mm". This is required if the study design is one of the built-in ones specified in study. If a custom design is specifed through analysis_fn, then any constants needed in analysis_fn can either be supplied in analysis_args, or hard-coded in analysis_fn itself.

For the built-in designs, the lists should have the following named components. An optional component niter in each case defines the posterior sample size (default 1000).

study="binary": a and b: Beta shape parameters

study="trial_binary": a1 and b1: Beta shape parameters for the prior for the first arm, a2 and b2: Beta shape parameters for the prior for the second arm.

study="normal_known": prior_mean, prior_sd (mean and standard deviation deviation of the Normal prior) and sampling_sd (SD of an individual-level

	normal observation, so that the sampling SD of the mean outcome over the study is `sampling_sd/sqrt(n)`.
`model_fn`	Function which evaluates the decision-analytic model, given parameter values. Required for `method="mm"`. See `evppi_mc` for full documentation of the required specification of this function.
`par_fn`	Function to simulate values from the uncertainty distributions of parameters needed by the decision-analytic model. Should take one argument and return a data frame with one row for each simulated value, and one column for each parameter. See `evppi_mc` for full specification.
`Q`	Number of quantiles to use in `method="mm"`.
`npreg_method`	Method to use to calculate the EVPPI, for those methods that require it. This is passed to `evppi` as the `method` argument.
`nsim`	Number of simulations from the model to use for calculating EVPPI. The first `nsim` rows of the objects in `inputs` and `outputs` are used.
`verbose`	If `TRUE`, then messages are printed describing each step of the calculation, if the method supplies these. Can be useful to see the progress of slow calculations.
`check`	If `TRUE`, then extra information about the estimation is saved inside the object that this function returns. This currently only applies to the regression-based methods `"gam"` and `"earth"` where the fitted regression model objects are saved. This allows use of the `check_regression` function, which produces some diagnostic checks of the regression models.
`...`	Other arguments understood by specific methods, e.g. `gam_formula` and other controlling options (see `evppi`) can be passed to the nonparametric regression used inside the moment matching method.

Details

See the package overview / Get Started vignette for some examples of using this function.

Value

A data frame with a column `pars`, indicating the parameter(s), and a column `evsi`, giving the corresponding EVPPI. If the EVSI for multiple sample sizes was requested, then the sample size is returned in the column `n`, and if `outputs` is of "cost-effectiveness analysis" form, so that there is one EVPPI per willingness-to-pay value, then a column `k` identifies the willingness-to-pay.

References

Strong, M., Oakley, J. E., Brennan, A., & Breeze, P. (2015). Estimating the expected value of sample information using the probabilistic sensitivity analysis sample: a fast, nonparametric regression-based method. Medical Decision Making, 35(5), 570-583.

Menzies, N. A. (2016). An efficient estimator for the expected value of sample information. Medical Decision Making, 36(3), 308-320.

Heath, A., Manolopoulou, I., & Baio, G. (2018). Efficient Monte Carlo estimation of the expected value of sample information using moment matching. Medical Decision Making, 38(2), 163-173.

Heath, A., Manolopoulou, I., & Baio, G. (2019). Estimating the expected value of sample information across different sample sizes using moment matching and nonlinear regression. Medical Decision Making, 39(4), 347-359.

`evsivar` *Calculate the expected value of sample information for an estimation problem*

Description

Calculate the expected value of sample information for an estimation problem. This computes the expected reduction in variance in some quantity of interest from a study of a certain design that informs the parameters of interest.

Usage

```
evsivar(
  outputs,
  inputs,
  study = NULL,
  datagen_fn = NULL,
  pars = NULL,
  n = 100,
  aux_pars = NULL,
  method = NULL,
  nsim = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`outputs`	a vector of values for the quantity of interest, sampled from the uncertainty distribution of this quantity that is induced by the uncertainty about the parameters.
`inputs`	Matrix or data frame of samples from the uncertainty distribution of the input parameters of the decision model. The number of columns should equal the number of parameters, and the columns should be named. This should have the same number of rows as there are samples in `outputs`, and each row of the samples in `outputs` should give the model output evaluated at the corresponding parameters. Users of **heemod** can create an object of this form, given an object produced by `run_psa` (`obj`, say), with `import_heemod_inputs`.
`study`	Name of one of the built-in study types supported by this package for EVSI calculation. If this is supplied, then the columns of `inputs` that correspond to the parameters governing the study data should be identified in `pars`. Current built-in studies are

"binary" A study with a binary outcome observed on one sample of individuals. Requires one parameter: the probability of the outcome. The sample size is specifed in the n argument to evsi(), and the binomially-distributed outcome is named X1.

"trial_binary" Two-arm trial with a binary outcome. Requires two parameters: the probability of the outcome in arm 1 and 2 respectively. The sample size is the same in each arm, specifed in the n argument to evsi(), and the binomial outcomes are named X1 and X2 respectively.

"normal_known" A study of a normally-distributed outcome, with a known standard deviation, on one sample of individuals. Likewise the sample size is specified in the n argument to evsi(). The standard deviation defaults to 1, and can be changed by specifying sd as a component of the aux_pars argument, e.g. evsi(..., aux_pars=list(sd=2)).

Either study or datagen_fn should be supplied to evsi().

For the EVSI calculation methods where explicit Bayesian analyses of the simulated data are performed, the prior parameters for these built-in studies are supplied in the analysis_args argument to evsi(). These assume Beta priors for probabilities, and Normal priors for the mean of a normal outcome.

datagen_fn If the proposed study is not one of the built-in types supported, it can be specified in this argument as an R function to sample predicted data from the study. This function should have the following specification:

1. the function's first argument should be a data frame of parameter simulations, with one row per simulation and one column per parameter. The parameters in this data frame must all be found in inputs, but need not necessarily be in the same order or include all of them.
2. the function should return a data frame.
3. the returned data frame should have number of rows equal to the number of parameter simulations in inputs.
4. if inputs is considered as a sample from the posterior, then datagen_fn(inputs) returns a corresponding sample from the posterior predictive distribution, which includes two sources of uncertainty: (a) uncertainty about the parameters and (b) sampling variation in observed data given fixed parameter values.
5. the function can optionally have more than one argument. If so, these additional arguments should be given default values in the definition of datagen_fn. If there is an argument called n, then it is interpreted as the sample size for the proposed study.

pars Character vector identifying which parameters are learned from the proposed study. This is required for the moment matching and importance sampling methods, and these should be columns of inputs. This is not required for the non-parametric regression methods.

n Sample size of future study, or vector of alternative sample sizes. This is understood by the built-in study designs. For studies specified by the user with datagen_fn, if datagen_fn has an argument n, then this is interpreted as the sample size. However if calling evsi for a user-specified design where datagen_fn does not have an n argument, then any n argument supplied to evsi will be ignored.

Currently this shortcut is not supported if more than one quantity is required to describe the sample size, for example, trials with unbalanced arms. In that case, you will have to hard-code the required sample sizes into `datagen_fn`.

For the nonparametric regression and importance sampling methods, the computation is simply repeated for each sample size supplied here.

The moment matching method uses a regression model to estimate the dependency of the EVSI on the sample size, hence to enable EVSI to be calculated efficiently for any number of sample sizes (Heath et al. 2019).

`aux_pars` A list of additional fixed arguments to supply to the function to generate the data, whether that is a built-in study design or user-defined function supplied in `datagen_fn`. For example, `evsi(..., aux_pars = list(sd=2))` defines the fixed standard deviation in the `"normal_known"` model.

`method` See `evsi`, only nonparametric regression methods are currently supported in `evsivar`.

`nsim` Number of simulations from the model to use for calculating EVPPI. The first `nsim` rows of the objects in `inputs` and `outputs` are used.

`verbose` If `TRUE`, then messages are printed describing each step of the calculation, if the method supplies these. Can be useful to see the progress of slow calculations.

`...` Other arguments understood by specific methods, e.g. `gam_formula` and other controlling options (see `evppi`) can be passed to the nonparametric regression used inside the moment matching method.

Value

A data frame with a column `pars`, indicating the parameter(s), and a column `evsi`, giving the corresponding EVSI. If there are EVSI estimates for multiple sample sizes, the sample size is returned in the column `n`.

References

Jackson, C., Presanis, A., Conti, S., & De Angelis, D. (2019). Value of information: Sensitivity analysis and research design in Bayesian evidence synthesis. Journal of the American Statistical Association, 114(528), 1436-1449.

`import_heemod` *Import results of probabilistic analysis from heemod*

Description

heemod is a package for constructing common forms of health economic decision models. The outputs from probabilistic analysis of these models can be imported using these functions, to allow Value of Information measures to be calculated for them using the **voi** package.

Usage

```
import_heemod_outputs(obj, k = NULL)

import_heemod_inputs(obj)
```

Arguments

obj	Object returned by the run_psa function in **heemod**, containing samples from probabilistic analysis of a decision model.
k	Vector of willingness-to-pay values. The default is inherited from the bcea function from the **BCEA** package.

Value

`import_heemod_outputs` produces a list of model outputs in "cost-effectiveness analysis" format, that can be supplied as the `outputs` argument to `evppi` and similar functions in the **voi** package. Both the **heemod** and **BCEA** packages need to be installed to use this.

`import_heemod_inputs` produces a data frame with samples of parameter values under uncertainty, that can be supplied as the `inputs` argument to `evppi` and similar functions in **voi**.

---

plot.evppi *Plot EVPPI estimates*

Description

Plot EVPPI estimates as simple dot or curve plots.

Usage

```
## S3 method for class 'evppi'
plot(x, type = NULL, order = FALSE, top = NULL, ...)
```

Arguments

x	Object returned from evppi.
type	"dots" for a dot plot of the EVPPI by parameter. If x includes multiple willingness-to-pay values for the same parameter, these are shown as multiple dots. "curves" for a plot of EVPPI against willingness-to-pay, with different parameters distinguished as different curves. This is only applicable if there are multiple willingness-to-pay values included in x.
order	For dot plots, order the plot with highest EVPPI values at the top.
top	A positive integer. If specified, for example as top=5 then only five parameters are included in the plot, those with the top five maximum EVPPI values by parameter.
...	Other arguments (currently unused).

Details

These plotting functions are intended for quick interactive exploration of EVPPI results, so they deliberately have limited options for customising them.

For publication quality graphics, it is advised to use `ggplot2` by hand on the data returned by `evppi`. Examine the code for `plot_evppi_dots` and `plot_evppi_curves` to see how these plots might be constructed.

Value

A `ggplot2` object.

`pop_voi` *Population expected value of information*

Description

Convert per-person expected value of information to the population expected value of information, given a discount rate over some time horizon.

Usage

```
pop_voi(voi, pop, time, dis = 0.035)
```

Arguments

`voi`	Vector of estimates of any per-person value of information measure, e.g. the `evsi` column of the data frame returned by `evsi` or the correspondingly-named columns of the data frames returned by `evppi` or `evpi`.
`pop`	Size of the population who would be affected by the decision.
`time`	Time horizon over which discounting will be applied.
`dis`	Discount rate used when converting per-person to population EVSI.

Details

Calculated as `voi*pop/dis*(1 - exp(-dis*time))`, or `voi*pop` if the discount rate is zero. This is a continuous-time variant of the typical discrete-time discounting formula.

Any arguments may be supplied as vectors, in which case, all arguments are replicated to the length of the longest argument.

Value

A vector of population VoI estimates.

Index