Package 'Infusion'

April 4, 2018

Type Package

Title Inference Using Simulation

Description Implements functions for simulation-based inference. In particular, implements functions to perform likelihood inference from data summaries whose distributions are simulated (Rousset et al. 2017 <doi:10.1111/1755-0998.12627>).

Encoding UTF-8

Version 1.2.0

Date 2018-04-04

Imports spaMM (>= 2.3.0), proxy, blackbox (>= 1.0.14), mvtnorm, methods, numDeriv, viridis, pbapply

Suggests testthat

Depends R (>= 3.3.0)

Maintainer François Rousset <francois.rousset@umontpellier.fr>

License CeCILL-2

ByteCompile true

URL https://www.R-project.org,
http://kimura.univ-montp2.fr/~rousset/Infusion.htm

NeedsCompilation no

Author François Rousset [aut, cre, cph]

Repository CRAN

Date/Publication 2018-04-04 21:45:45 UTC

R topics documented:

`add_simulation` *Create or augment a list of simulated distributions of summary statistics*

Description

`add_simulation` creates or augments a list of simulated distributions of summary statistics, and formats the simulation results appropriately for further use. The user does not have to think about this return format. Instead, s-he only has to think about the very simple return format of the function given as its `Simulate` argument. Alternatively, if the simulation function cannot be called directly by the R code, simulated distributions can be added easily using the `newsimuls` argument, again using a simple format (see `onedistrib` in the Examples). `add_reftable` is a wrapper for `add_simulation`, enforcing `nRealizations=1`.

Usage

```
add_simulation(simulations=NULL, Simulate, par.grid=NULL,
               nRealizations = NULL,
               newsimuls = NULL, verbose = interactive())
add_reftable(...)
```

Arguments

`simulations`	A list of simulations
`Simulate`	The name (as a character string) of the *R* function used to generate empirical distributions of summary statistics (notably, an for external simulation program, an R function wrapping the call to the external program must be provided). The function must have a single vector as argument, matching each row of `par.grid`, and must return a vector of summary statistics with named vector members.
`par.grid`	A data frame of which each line matches the single vector argument of `Simulate`.

nRealizations	The number of simulated samples of summary statistics, for each empirical distribution (each row of par.grid). If the argument is NULL, the value is obtained by Infusion.getOption. If the argument is not NULL, Infusion.options(nRealizations) is set, but restored, on exit from add_simulation, to its initial value.
newsimuls	If the function used to generate empirical distributions cannot be called by R, then newsimuls can be used to provide these distributions. This argument should be list of matrices, each with a par attribute (see Examples). Rows of each matrix stand for simulation replicates and rows for the different summary statistics. This is the same format as the return value of add_simulation, except that the latter adds further attributes.
verbose	Whether to print some information or not
...	Arguments passed to add_simulation. Any of the above aguments is valid, except nRealizations.

Value

If only one realization is computed for each (vector-valued) parameter, a data.frame is returned. Otherwise, the return value is an objet of class EDFlist, which is a list of matrices-with-attribute. Each matrix contains a simulated distribution of summary statistics for given parameters, and the "par" attribute is a vector of parameters. If Simulate is used, this must give the full parameters required for a call of the simulation function; otherwise it must at least include all variable parameters in this **or later** simulations to be appended to the simulation list.

Examples

```
# example of building a list of simulations from scratch:
myrnorm <- function(mu,s2,sample.size) {
  s <- rnorm(n=sample.size,mean=mu,sd=sqrt(s2))
  return(c(mean=mean(s),var=var(s)))
}
set.seed(123)
onedistrib <- t(replicate(100,myrnorm(1,1,10))) # toy example of simulated distribution
attr(onedistrib,"par") <- c(mu=1,sigma=1,sample.size=10) ## important!
add_simulation(NULL, Simulate="myrnorm",
                     newsimuls=list("example"=onedistrib))

## see main documentation page for the package for other typical usage
```

---

confint.SLik *Compute confidence intervals by (profile) summary likelihood*

Description

This takes an SLik object (as produced by MSL) and deduces confidence bounds for each parameter, using a (profile, if relevant) likelihood ratio method.

Usage

```
## S3 method for class 'SLik'
confint(object, parm,
                        level=0.95, verbose=interactive(),
                        fixed=NULL,which=c(TRUE,TRUE),...)
```

Arguments

object	an SLik or SLikp object
parm	The parameter which confidence bounds are to be computed
level	The desired coverage of the interval
verbose	Whether to print some information or not
fixed	When this is NULL the computed interval is a profile confidence interval over all parameters excluding parm. fixed allows one to set fixed values to some of these parameters.
which	A pair of booleans, controlling whether to compute respectively the lower and the upper CI bounds.
...	further arguments passed to or from other methods (currently not used).

Value

A list with sublists for each parameter, each sublist containing of three vectors: the bounds of the one-dimensional confidence interval; the "full" (only parameters variable in the SLik object are considered) parameter point for the lower bound, and the full parameter point for the upper bound

Examples

```
## see main documentation page for the package
```

---

densv *Saved computations of inferred log-likelihoods*

Description

These are saved results from toy examples used in other documentation page for the package. It gives estimates by simulation of log-likelihoods of the (mu,s2) parameters of a Gaussian distribution for a given sample of size 20 with mean 4.1416238 and (bias-corrected) variance 0.9460778. densv is based on the sample mean and sample variance as summary statistics, and densb on more contrived summary statistics.

Usage

```
data("densv")
data("densb")
```

Format

Data frames (with additional attributes) with observations on the following 5 variables.

`mu` a numeric vector; mean parameter of simulated Gaussian samples

`s2` a numeric vector; variance parameter of simulated Gaussian samples

`sample.size` a numeric vector; size of simulated Gaussian samples

`logL` a numeric vector; log probability density of a given statistic vector inferred from simulated values for the given parameters

`isValid` a boolean vector. See `infer_logLs` for its meaning.

Both data frames are return objects of a call to `infer_logLs`, and as such they includes attributes providing information about the parameter names and statistics names (not detailed here).

See Also

See step (3) of the workflow in the Example on the main `Infusion` documentation page, showing how `densv` was produced, and the Example in `project` showing how `densb` was produced.

---

handling_NAs *Discrete probability masses and NA/NaN/Inf in distributions of summary statistics.*

Description

This explains the use of the `boundaries` attribute of observed statistics withto handle (1) values of the summary statistics that can occur with some probability mass; (2) special values (NA/NaN/Inf) in distributions of summary statistics. This further explains why `Infusion` handles special values by removing affected distributions unless the `boundaries` attribute is used.

Details

Special values may be encountered in an analysis. For example, trying to estimate a regression coefficient when the predictor variable is constant may return a NaN. Since functions such as `refine` automatically add simulated distributions, this problem must be automatically handled by the user's simulation function or by the package functions, rather than by user's tinkering with the Infusion procedures.

The user must consider what s-he would do if actual data also included NA/NaN/Inf values. If (1) such data would not be used in the statistical analysis, then the simulation procedure must reflect that, otherwise the analysis will be biased. Alternatively (2) if one considers that special values are informative about parameters (in the above example of a regression coefficient, if a constant

predictor variable says something about the parameters), then NA/NaN/Inf must be replaced by a numerical value which is flagged to be distinctly handled.

Thus, in case (1) it may be necessary to simulate alternative data until no NaN's are obtained and the target size of the simulated distribution is reached. One solution is for the user to write a simulation function that calls itself recursively until a valid summary statistic is produced. Care is then needed to avoid infinite recursion (which might well indicate unlikely parameter values).

In case (2), it is necessary to assign some (fixed) dummy numerical value to the summary statistics, and to flag this value using the `boundaries` attribute of the observed summary statistics. The simulation function should return statistic `foo=-1` (say) instead of `foo=NaN`, and one should then set `attr(<observed>,"boundaries") <- c(foo=-1)`.

Without such active decisions by the user, the inference method has no way to determine whether case (1) or (2) holds, and must thus ignore all empirical distributions including NA/NaN/inf. These empirical distributions are thus ignored by the inference functions.

The boundary attribute is also useful to handle all values of the summary statistics that can occur with some probability mass. For example if the estimate `est_p` of a probability takes values 0 or 1 with positive probability, one should set `attr(<observed>,"boundaries") <- c(p_est=0,p_est=1)`.

`infer_logLs` *Infer log Likelihoods using simulated distributions of summary statistics*

Description

For each simulated distribution of summary statistics, `infer_logLs` infers a probability density function, and the density of the observed values of the summary statistics is deduced. By default, inference of each density is performed by `infer_logL_by_Rmixmod`, which fits a distribution of summary statistics using procedures from the `Rmixmod` package.

Usage

```
infer_logLs(object, stat.obs,
            logLname = Infusion.getOption("logLname"),
            verbose = list(most=interactive(),
                           final=FALSE),
            method = Infusion.getOption("infer_logL_method"),
            nb_cores = NULL, packages = NULL,
            ...)
infer_tailp(object, refDensity, stat.obs,
                tailNames=Infusion.getOption("tailNames"),
                verbose=interactive(), method=NULL,...)
infer_logL_by_GLMM(EDF,stat.obs,logLname,verbose)
infer_logL_by_Rmixmod(EDF,stat.obs,logLname,verbose)
infer_logL_by_mclust(EDF,stat.obs,logLname,verbose)
infer_logL_by_Hlscv.diag(EDF,stat.obs,logLname,verbose)
```

Arguments

object	A list of simulated distributions (the return object of add_simulation)
EDF	An empirical distribution, with a required par attribute (an element of the object list).
stat.obs	Named numeric vector of observed values of summary statistics.
logLname	The name to be given to the log Likelihood in the return object, or the root of the latter name in case of conflict with other names in this object.
tailNames	Names of "positives" and "negatives" in the binomial response for the inference of tail probabilities.
refDensity	An object representing a reference density (such as an HLfit fit object or other objects with a similar predict method) which, together with the density inferred from each empirical density, defines a likelihood ratio used to define a rejection region.
verbose	A list as shown by the default, or simply a vector of booleans, indicating respectively whether to display (1) some information about progress; (2) a final summary of the results after all elements of simuls have been processed. If a count of 'outlier'(s) is reported, this typically means that stat.obs is not within the envelope of a simulated distribution (or whatever other meaning the user attaches to an FALSE isValid code: see Details)
method	A function for density estimation. See Description for the default behaviour and Details for the constraints on input and output of the function.
nb_cores	Number of cores for parallel computation. The default is spaMM.getOption("nb_cores"), and 1 if the latter is NULL. nb_cores=1 which prevents the use of parallelisation procedures.
packages	For parallel evaluation: Names of additional libraries to be loaded on the cores, necessary for evaluation of a user-defined 'method'.
...	further arguments passed to or from other methods (currently not used).

Details

By default, density estimation is based on Rmixmod methods. Other available methods are not routinely used and not all of Infusion features may work with them. The function Rmixmod::mixmodCluster is called, with arguments nbCluster=Infusion.getOption("nbCluster") and mixmodGaussianModel=Infusion.getOpt
If Infusion.getOption("nbCluster") specifies a sequence of values, then several clusterings are computed and AIC is used to select among them.

infer_logL_by_GLMM, infer_logL_by_Rmixmod, infer_logL_by_mclust, and infer_logL_by_Hlscv.diag are examples of the method that may be provided for density estimation. Other methods may be provided with the same arguments. Their return value must include the element logL, an estimate of the log-density of stat.obs, and the element isValid with values FALSE/TRUE (or 0/1). The standard format for the return value is unlist(c(attr(EDF,"par"),logL,isValid=isValid)).

isValid is primarily intended to indicate whether the log likelihood of stat.obs inferred by a given density estimation method was suitable input for inference of the likelihood surface. isValid has two effects: to distinguish points for which isValid is FALSE in the plot produced by plot.SLik; and more critically, to control the sampling of new parameter points within refine so that points for which isValid is FALSE are less likely to be sampled.

Invalid values may for example indicate a likelihood estimated as zero (since log(0) is not suitable input), or (for density estimation methods which may infer erroneously large values when extrapolating), whether `stat.obs` is within the convex hull of the EDF. In user-defined `methods`, invalid inferred logL should be replaced by some alternative low estimate, as all methods included in the package do.

The source code of `infer_logL_by_Hlscv.diag` illustrates how to test whether `stat.obs` is within the convex hull of the EDF, using functions `resetCHull` and `isPointInCHull` (exported from the `blackbox` package).

`infer_logL_by_Rmixmod` calls `Rmixmod::mixmodCluster` `infer_logL_by_mclust` calls `mclust::densityMclust`, `infer_logL_by_Hlscv.diag` calls `ks::kde`, and `infer_logL_by_GLMM` fits a binned distribution of summary statistics using a Poisson GLMM with autocorrelated random effects, where the binning is based on a tesselation of a volume containing the whole simulated distribution. Limited experimentations so far suggest that the mixture models methods are fast and appropriate (`Rmixmod`, being a bit faster, is the default method); that the kernel smoothing method is more erratic and moreover requires additional input from the user, hence is not really applicable, for distributions in dimension d= 4 or above; and that the GLMM method is a very good density estimator for d=2 but will challenge one's patience for d=3 and further challenge the computer's memory for d=4.

Value

For `infer_logLs`, a data frame containing parameter values and their log likelihoods, and additional information such as attributes providing information about the parameter names and statistics names (not detailed here). These attributes are essential for further inferences.

See Details for the required value of the `methods` called by `infer_logLs`.

See Also

See step (3) of the workflow in the Example on the main `Infusion` documentation page.

`infer_SLik_joint` *Infer as (summary) likelihood surface from a simulation table*

Description

This infers the likelihood surface from a simulation table where each simulated data set is drawn for a distinct (vector-valued) parameter, as is usual for reference tables in ABC. A parameter density is inferred, as well as a joint density of parameters and summary statistics, and the likelihood surface is inferred from these two densities. This is not yet extensively tested, nor the code has been optimized.

Usage

```
infer_SLik_joint(data, stat.obs, logLname = Infusion.getOption("logLname"),
                 Simulate = attr(data, "Simulate"),
                 nbCluster= Infusion.getOption("nbCluster"),
                 using = Infusion.getOption("using"),
                 verbose = list(most = interactive(), final = FALSE))
```

Arguments

data	A data frame, whose rows contain a vector of parameters and one realization of the summary statistics for these parameters.
stat.obs	Named numeric vector of observed values of summary statistics.
logLname	The name to be given to the log Likelihood in the return object, or the root of the latter name in case of conflict with other names in this object.
Simulate	Either NULL or the name of the simulation function if it can be called from the R session.
nbCluster	nbCluster argument of Rmixmod::mixmodCluster
using	Either "Rmixmod" or "mclust" to select the clustering methods used.
verbose	A list as shown by the default, or simply a vector of booleans, indicating respectively whether to display (1) some information about progress; (2) a final summary of the results after all elements of simuls have been processed.

Value

An object of class SLik_j, which is a list including two Rmixmod::mixmodCluster objects, and additional members not documented here.

Examples

```
if (Infusion.getOption("example_maxtime")>50) {
  myrnorm <- function(mu,s2,sample.size) {
    s <- rnorm(n=sample.size,mean=mu,sd=sqrt(s2))
    return(c(mean=mean(s),var=var(s)))
  } # simulate means and variances of normal samples of size 'sample.size'
  set.seed(123)
  # pseudo-sample with stands for the actual data to be analyzed:
  ssize <- 40
  Sobs <- myrnorm(mu=4,s2=1,sample.size=ssize)
  # Uniform sampling in parameter space:
  npoints <- 600
  parsp <- data.frame(mu=runif(npoints,min=2.8,max=5.2),
                      s2=runif(npoints,min=0.4,max=2.4),sample.size=ssize)
  # Build simulation table:
  simuls <- add_reftable(Simulate="myrnorm",par.grid=parsp)
  # Infer surface:
  densv <- infer_SLik_joint(simuls,stat.obs=Sobs)
  # Usual workflow using inferred suface:
  slik_j <- MSL(densv) ## find the maximum of the log-likelihood surface
  slik_j <- refine(slik_j,maxit=5)
  plot(slik_j)
  # etc:
  profile(slik_j,c(mu=4)) ## profile summary logL for given parameter value
  confint(slik_j,"mu") ## compute 1D confidence interval for given parameter
  plot1Dprof(slik_j,pars="s2",gridSteps=40) ## 1D profile
}
```

`infer_surface` *Infer a (summary) likelihood or tail probability surface from inferred likelihoods*

Description

The `logLs` method uses a standard smoothing method (prediction under linear mixed models, a.k.a. Kriging) to infer a likelihood surface, using as input likelihood values themselves inferred with some error for different parameter values. The `tailp` method use a similar approach for smoothing binomial response data, using the algorithms implemented in the spaMM package for fitting GLMMs with autocorrelated random effects.

Usage

```
## S3 method for class 'logLs'
infer_surface(object, method="REML",verbose=interactive(),allFix=NULL,...)
## S3 method for class 'tailp'
infer_surface(object, method="PQL",verbose=interactive(),allFix,...)
```

Arguments

`object`	A data frame with attributes, containing independent prediction of logL or of LR tail probabilities for different parameter points, as produced by `infer_logLs` or `infer_tailp`.
`method`	methods used to estimate the smoothing parameters. If `method="GCV"`, a generalized cross-validation procedure is used (for `logLs` method only). Other methods are as described in the `HLfit` documentation.
`verbose`	Whether to display some information about progress or not.
`allFix`	Fixed values in the estimation of smoothing parameters. For development purposes, not for routine use. For `infer_surface.logLs`, this should typically include values of all parameters fitted by `spaMM::corrHLfit` (ρ, ν, ϕ, λ, and `$etaFix`=β).
`...`	further arguments passed to or from other methods (currently not used).

Value

An object of class `SLik` or `SLikp`, which is a list including an `HLfit` object as returned by `corrHLfit`, and additional members not documented here.

Examples

```
## see main documentation page for the package
```

`Infusion` *Inference using simulation*

Description

Implements a collection of methods to perform inferences based on simulation of realizations of the model considered. In particular it implements "summary likelihood", an approach that effectively evaluates and uses the likelihood of simulated summary statistics.

Details

The methods implemented in `Infusion` by default assume that the summary statistics have densities. Special values of some statistic, having discrete probability mass, can be handled using the `boundaries` attribute of the observed summary statistics (see `handling_NAs` for a further use of this attribute).

Examples

```
## The following example illustrates the workflow.
## However, most steps run longer than requested by the CRAN checks,
## So by default they will not run.
##
## (1) The user must provide the function for simulation of summary statistics
myrnorm <- function(mu,s2,sample.size) {
 s <- rnorm(n=sample.size,mean=mu,sd=sqrt(s2))
 return(c(mean=mean(s),var=var(s)))
} # simulate means and variances of normal samples of size 'sample.size'
#
## pseudo-sample:
set.seed(123)
Sobs <- myrnorm(mu=4,s2=1,sample.size=40) ## stands for the actual data to be analyzed
#
## (2) Generate, and simulate distributions for,
##        an irregular grid of parameter values, with some replicates
if (Infusion.getOption("example_maxtime")>45) {
  parsp <- init_grid(lower=c(mu=2.8,s2=0.2,sample.size=40),
                     upper=c(mu=5.2,s2=3,sample.size=40))
  simuls <- add_simulation(NULL,Simulate="myrnorm",par.grid=parsp)

  ## (3) infer logL(pars,stat.obs) for each simulated 'pars'
  # Relatively slow, hence saved as data 'densv'
  densv <- infer_logLs(simuls,stat.obs=Sobs)
} else {
  data(densv)
  .Random.seed <- saved_seed
}
#
## (4) infer a log-likelihood surface and its maximum;
##       plot and extract various information.
if (Infusion.getOption("example_maxtime")>17) {
```

```
 slik <- infer_surface(densv)
 slik <- MSL(slik) ## find the maximum of the log-likelihood surface
 plot(slik)
 profile(slik,c(mu=4)) ## profile summary logL for given parameter value
 confint(slik,"mu") ## compute confidence interval for given parameter
 plot1Dprof(slik,pars="s2",gridSteps=40) ## 1D profile
}
#
## (5) ## refine iteratively
if (Infusion.getOption("example_maxtime")>68) {
 slik <- refine(slik)
}
```

`init_grid` *Define starting points in parameter space.*

Description

This function is exported from the blackbox package. It samples the space of estimated parameters. Also handles other fixed arguments that need to be passed to the function simulating the summary statistics (sample size is likely to be one such argument). The current sampling strategy is crude but achieves three desirable effects: It tries to sample the space uniformly, avoiding large gaps; it is not exactly a regular grid; and it includes replicates of some parameter points, required for good smoothing of the likelihood surface.

Usage

```
init_grid(lower=c(par=0), upper=c(par=1), steps=NULL,
          nUnique=NULL, nRepl=min(10L,nUnique),
          jitterFac=0.5
         )
```

Arguments

`lower`	A vector of lower bounds for the parameters, as well as fixed arguments to be passed to the function simulating the summary statistics. Elements must be named. Fixed parameters character strings.
`upper`	A vector of upper bounds for the parameters, as well as fixed parameters. Elements must be named and match those of `lower`.
`steps`	Number of steps of the grid, in each dimension of estimated parameters. If NULL, a default value is defined from the other arguments. If a single value is given, it is applied to all dimensions. Otherwise, this must have the same length as `lower` and `upper` and named in the same way as the variable parameters in these arguments.
`nUnique`	Number of distinct values of parameter vectors in output. Default is an heuristic guess for good start from not too many points, computed as `floor(50^((v/3)^(1/3)))` where `v` is the number of variable parameters.

nRepl	Number of replicates of distinct values of parameter vectors in output.
jitterFac	Controls the amount of jitter of the points around regular grid nodes. The default value 0.5 means that a mode can move by up to half a grid step (independently in each dimension), so that two adjacent nodes moved toward each other can (almost) meet each other.

Value

A data frame. Each row defines a list of arguments of vector of the function simulating the summary statistics.

Examples

```
set.seed(123)
init_grid()
init_grid(lower=c(mu=2.8,s2=0.5,sample.size=20),
          upper=c(mu=5.2,s2=4.5,sample.size=20),
          steps=c(mu=7,s2=9),nUnique=63)
```

MSL *Maximum likelihood from an inferred likelihood surface*

Description

This computes the maximum of an object of class SLik representing an inferred (summary) likelihood surface

Usage

```
MSL(object, CIs = TRUE, level = 0.95, verbose = interactive(),
    eval_RMSEs = inherits(object,"SLik"), ...)
```

Arguments

object	an object of class SLik as produced by infer_surface.logLs
CIs	If TRUE, construct one-dimensional confidence intervals for all parameters.
level	Intended coverage probability of the confidence intervals.
verbose	Whether to display some information about progress and results.
eval_RMSEs	Logical: whether to evaluate prediction uncertainty for likelihoods/ likelihood ratios/ parameters. By default TRUE for SLik objects, and best kept so, as it is necessary for the automated iterative method. May be FALSE for other classes of objects.
...	Further arguments passed from or to other methods.

Details

RMSEs are computed using approximate formulas for prediction (co-)variances in linear mixed midels (see Details in predict). par_RMSEs are computed from RMSEs and from the numerical gradient of profile log-likelihood at each CI bound.

Value

The object is returned invisibly, with added members

MSL , itself with members MSLE and maxlogL that match the par and value returned by an optim call.

RMSEs root mean square errors of the log-likelihood at its inferred maximum and of the log-likelihood ratios at the CI bounds.

par_RMSEs root mean square errors of the CI bounds

Examples

```
## see main documentation page for the package
```

---

multi_binning *Multivariate histogram*

Description

Constructs a multivariate histogram of the points. Optionally, first tests whether a given value is within the convex hull of input points and constructs the histogram only if this test is TRUE. This function is available for development purposes but is not required otherwise . It is sparsely documented and subject to changes without notice.

Usage

```
multi_binning(m, subsize=trunc(nrow(m)^(Infusion.getOption("binningExponent"))),
              expand=5/100, focal=NULL)
```

Arguments

m	A matrix representing points in d-dimensional space, where d is the number of columns
subsize	A control parameter for an undocumented algorithm
expand	A control parameter for an undocumented algorithm
focal	Value to be tested for inclusion within the convex hull. Its elements must have names.

Details

The algorithm may be detailed later.

Value

Either NULL (if the optional test returned FALSE), or an histogram represented as a data frame each row of which represents an histogram cell by its barycenter (a point in d-dimensional space), its "binFactor" (the volume of the cell times the total number of observations) and its "count" (the number of observations within the cell). The returned data frame has the following attributes: `attr(.,"stats")` are the column names of the d-dimensional points; `attr(.,"count")` is the column name of the count, and `attr(.,"binFactor")` is the column name of the binFactor.

`options` *Infusion options settings*

Description

Allow the user to set and examine a variety of *options* which affect operations of the Infusion package. However, typically these should not be modified, and if they are, not more than once in a data analysis.

Usage

```
Infusion.options(...)

Infusion.getOption(x)
```

Arguments

`x`	a character string holding an option name.
`...`	A named value or a list of named values. The following values, with their defaults, are used in `Infusion`:

`projTrainingSize = 200`**:** default value of `trainingsize` argument of `project.character`.

`projKnotNbr = 300`**:** default value of `knotnbr` argument of `project.character`.

`logLname = "logL"`**:** default value of `logLname` argument of `infer_logLs`. The name given to the inferred log likelihoods in all analyses.

`LRthreshold= - qchisq(0.999,df=1)/2`**:** A value used internally by `sample_volume` to sample points in the upper region of the likelihood surface, as defined by the given likelihood ratio threshold.

`precision = 0.1`**:** default value of `precision` argument of `refine`. Targets RMSE of log L and log LR estimates.

`nRealizations=1000`**:** default value of `nRealizations` argument of `add_simulation`. Number of realizations for each empirical distribution.

`mixmodGaussianModel="Gaussian_pk_Lk_Dk_A_Dk"`**:** default models used in clustering by `Rmixmod`. Run `Rmixmod::mixmodGaussianModel()` for a list of possible models, and see the statistical documentation (Mixmod Team 2016) for explanations about them.

`nbCluster = quote(seq(ceiling(nrow(data)^0.3)))`: default value of `nbCluster` used in clustering by `Rmixmod`

`example_maxtime=2.5`: Used in the documentation to control whether the longer examples should be run. The approximate running time of given examples (or some very rough approximation for it) on one author's laptop is compared to this value.

and possibly other undocumented values for development purposes.

Details

The default `nbCluster` value is the upper value of the range recommended in the `mixmod` statistical documentation (Mixmod Team, 2016). If clustering by the given number(s) of clusters fails, decreasing values are tryied until success.

Value

For `Infusion.getOption`, the current value set for option `x`, or `NULL` if the option is unset.

For `Infusion.options()`, a list of all set options. For `Infusion.options(name)`, a list of length one containing the set value, or `NULL` if it is unset. For uses setting one or more options, a list with the previous values of the options changed (returned invisibly).

References

Mixmod Team (2016). Mixmod Statistical Documentation. Université de Franche-Comté, Besançon, France. Version: February 10, 2016 retrieved from http://www.mixmod.org.

Examples

```
  Infusion.options()
  Infusion.getOption("LRthreshold")
  ## Not run:
  Infusion.options(LRthreshold=- qchisq(0.99,df=1)/2)

## End(Not run)
```

`plot.SLik` *Plot SLik or SLikp objects*

Description

Mostly conceived for exposition purposes, for the two-parameters case. The black-filled points are those for which the observed summary statistic was outside of the convex hull of the simulated empirical distribution. The crosses mark the estimated ML point and the confidence intervals points, that is, the outmost points on the contour defined by the profile likelihood threshold for the profile confidence intervals. There is a pair of CI points for each interval. The smaller black dots mark points added in the latest iteration, if `refine` was used.

Usage

```
## S3 method for class 'SLik'
plot(x, y, filled = FALSE, decorations = NULL,
                    color.palette = NULL, plot.axes = NULL,
                    plot.title = NULL, ...)
## S3 method for class 'SLik_j'
plot(x, y, filled = nrow(x$logLs)>5000L, decorations = NULL,
                      color.palette = NULL, plot.axes = NULL,
                      plot.title = NULL, ...)
```

Arguments

x	An object of class SLik or SLikp
y	Not used, but included for consistency with the plot generic.
filled	whether to plot a mapMM or a filled.mapMM.
decorations	Graphic directives added to the default decorations value in calls of mapMM or a filled.mapMM (see the source code of plot.SLik for the latter default values).
color.palette	Either NULL or a function that can replace the default color function used by plot.SLik. Thefunction must have a single argument, giving the number of color levels.
plot.title	statements which replace the default titles to the main plot (see Details).
plot.axes	statements which replace the default axes on the main plot (see Details).
...	further arguments passed to or from other methods (currently can be used to pass a few arguments such as map.asp in all cases, or variances to filled.mapMM).

Details

Different graphic functions are called depending on the number of estimated parameters. For two parameters, mapMM or filled.mapMM are called. For more than two parameters, spaMM.filled.contour is called. See the documentation of these functions for the appropriate format of the plot.title and plot.axes arguments.

Value

Returns the plotted object invisibly.

Examples

```
## Not run:
## Using 'slik' object from the example in help("Infusion-package")
  plot(slik, filled=TRUE,
       plot.title=quote(title("Summary-likelihood-ratio surface",
                              xlab=expression(mu),
                              ylab=expression(sigma^2))))

## End(Not run)
```

plot1Dprof *Plot likelihood profiles*

Description

These functions plot 1D and 2D profiles from an SLik object

Usage

```
plot1Dprof(object, pars=object$colTypes$fittedPars, type="logLR",
           gridSteps=21, xlabs=list(), ylab, scales=NULL,
           plotpar=list(pch=20))
plot2Dprof(object, pars=object$colTypes$fittedPars, type="logLR",
           gridSteps=17, xylabs=list(), main, scales=NULL,
           plotpar=list(pch=20), margefrac = 0)
```

Arguments

object	An SLik object
pars	The parameters for which profiles will be computed. For 2D plots, all pairs of parameters in pars are considered
type	logLR or LR or logL for the (log) likelihood ratio or the log likelihood
gridSteps	The number of values (in each dimension for 2D plots) which likelihood should be computed. For 1D plots, gridSteps=0 will call curve rather than a plot of points
xlabs	A *list* of alternative axis labels. The names of the list elements should be elements of pars (see Examples)
xylabs	Same as xlabs but affecting both axes in 2D plots
ylab	Same as ylab argument of plot. Default depends on type argument.
main	Same as main argument of plot. Default depends on type argument.
scales	A named character vector, which controls ticks and tick labels on axes, so that these can be expressed as (say) the exponential of the parameter inferred in the SLik object. For example if the likelihood of logPop = log(population size) was thus inferred, scales=c(logPop="log") will give population size values on the axis (but will retain a log scale for this parameter).
plotpar	arguments for par() such as font sizes, etc.
margefrac	For development purposes, not documented.

Value

No return value.

Examples

```
if (Infusion.getOption("example_maxtime")>40) {
 data(densv)
 slik <- infer_surface(densv) ## infer a log-likelihood surface
 slik <- MSL(slik) ## find the maximum of the log-likelihood surface
 plot1Dprof(slik,pars="s2",gridSteps=40,xlabs=list(s2=expression(paste(sigma^2))))
}
```

`profile.SLik` *Compute profile summary likelihood*

Description

Predicts the profile likelihood for a given parameter value (or vector of such values) using predictions from an `SLik` object (as produced by `MSL`).

Usage

```
## S3 method for class 'SLik'
profile(fitted, value, fixed=NULL, return.optim=FALSE, ...)
## S3 method for class 'SLik_j'
profile(fitted, ...)
```

Arguments

`fitted`	an `SLik` object.
`value`	The parameter value (as a vector of named values) for which the profile is to be computed
`fixed`	When this is `NULL` the computed interval is a profile confidence interval over all parameters excluding `value`. `fixed` allows one to set fixed values to some of these parameters.
`return.optim`	If this is TRUE, and if maximization of likelihood given `value` and `fixed` is indeed required, then the full result of the `optim` call is returned.
`...`	For `SLik_j` method, arguments passed to `SLik` method. For `SLik_j` method, currently not used.

Value

The predicted summary profile log-likelihood; or possibly the result of an `optim` call if `return.optim` is TRUE.

Examples

```
## see main documentation page for the package
```

`project.character` *Learn a projection method for statistics and applies it*

Description

`project` is a generic function with two methods. If the first argument is a parameter name, `project.character` defines a projection function from several statistics to an output statistic predicting this parameter. `project.default` produces a vector of projected statistics using such a projection. `project` is particulary useful to reduce a large number of summary statistics to a vector of projected summary statistics, with as many elements as parameters to infer. This dimension reduction can substantially speed up subsequent computations. The concept implemented in `project` is to fit a parameter to the various statistics available, using machine-learning or mixed-model prediction methods. All such methods can be seen as nonlinear projection to a one-dimensional space. `project.character` is an interface that allows different projection methods to be used, provided they return an object of a class that has a defined `predict` method with a `newdata` argument (as expected, see `predict`).

Usage

```
project(x,...)

## S3 method for building the projection
## S3 method for class 'character'
project(x, stats, data,
             trainingsize= if (method=="REML")
               {Infusion.getOption("projTrainingSize")} else {NULL},
             knotnbr= if (method %in% c("REML","GCV"))  {
                        Infusion.getOption("projKnotNbr")
                      } else {floor(1000*log2(length(stats)+1))},
             method="REML",methodArgs=list(),verbose=TRUE,...)

## S3 method for applying the projection
## Default S3 method:
project(x, projectors,...)
```

Arguments

`x`	The name of the parameter to be predicted, or a vector/matrix/list of matrices of summary statistics.
`stats`	Statistics from which the predictor is to be predicted
`data`	A list of simulated empirical distributions, as produced by `add_simulation`, or a data frame with all required variables.
`trainingsize`	For REML only: size of random sample of realizations from the `data` from which the smoothing parameters are estimated.
`knotnbr`	Size of random sample of realizations from the `data` from which the predictor is built given the smoothing parameters.

method	character string: "REML", "GCV", or the name of a suitable projection function. The latter may be defined in another package, e.g. "randomForest", or predefined by Infusion (function "nnetwrap"), or defined by the user. See Details for predefined functions and for defining new ones.
methodArgs	A list of arguments for the projection method. One may not need to provide arguments in the following cases, where project kindly (tries to) assign values to the required arguments if they are absent from methodArgs: If "REML" or "GCV" methods are used (in which case methodArgs is completely ignored); or if the projection method uses formula and data arguments (in particular if the formula is of the form response ~ var1 + var2 + ...; otherwise the formula should be provided through methodArgs). This works for example for methods based on nnet; or if the projection method uses x and y arguments. This works for example for randomForest (though not with the generic function method="randomForest", but only with the internal function method="randomForest:::randomForest.default").
projectors	A list with elements of the form <name>=<project result>, where the <name> must differ from any name of x. <project result> may indeed be the return object of a project call.
verbose	Whether to print some information or not. In particular, TRUE, true-vs.-predicted diagnostic plots will be drawn if any of the following methods have been used: "REML", "GCV", or a call to caret::train.
...	further arguments passed to or from other methods (currently not used).

Details

Prediction can be based on a linear mixed model (LMM) with autocorrelated random effects, internally calling the corrHLfit function with formula <parameter> ~ 1+ Matern(1|<stat1>+...+<statn>). This approach allows in principle to produce arbitrarily complex predictors (given sufficient input) and avoids overfitting in the same way as restricted likelihood methods avoids overfitting in LMM. REML methods are then used by default to estimate the smoothing parameters. However, faster methods may be required, and method "neuralNet" interfaces a neural network approach.

The data may involve hundreds of thousands of realizations of the summary statistic, and REML fitting is already slow for much smaller data sets, which is why faster alternative methods may be worth considering, and why random subset(s) of the data may be considered at various steps. The default size of these subsets aim to ensure that the computations can be performed in reasonable time.

For REML, the trainingsize and knotnbr arguments determine respectively the size of the subset used to estimate the smoothing parameters and the size of the subset defining the predictor given the smooothing parameters.

If method="GCV", a generalized cross-validation procedure (Golub et al. 1979) is used to estimate the smoothing parameters. This is faster but still slow, so a random subset of size knotnbr is still used to estimate the smoothing parameters and generate the predictor.

Alternatively, various machine-learning methods can be used (see e.g. Hastie et al., 2009, for an introduction). A random subset of size knotnbr is again used, with a larger default value bearing

the assumption that these methods are faster. `method="neuralNet"` interfaces a neural network method. It calls the `train` function from the `caret` package.

In principle, any object suitable for prediction could be used as one of the `projectors`. That is, if predictions of a parameter can be performed using an object `MyProjector` of class `MyProjectorClass`, `MyProjector` could be used in place of a `project` result if `predict.MyProjectorClass(object,newdata,...)` is defined. However, if the learning method that generated the projector used a formula-data syntax, then its `predict` method is likely to request names for its `newdata`, that need to be provided through `attr(MyProjector,"stats")` (these names cannot be assumed to be in the `newdata` when `predict` is called through `optim`).

Value

`project.character` returns an object of class returned by the `method` (methods `"REML"` and `"GCV"` will call `corrHLfit` which return an object of class `spaMM`) `project.default` returns an object of the same class and structure as the input `x`, containing the projected statistics inferred from the input summary statistics.

References

Golub, G. H., Heath, M. and Wahba, G. (1979) Generalized Cross-Validation as a method for choosing a good ridge parameter. Technometrics 21: 215-223.

T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York, 2nd edition, 2009.

Examples

```
##########
if (Infusion.getOption("example_maxtime")>250) {
## Transform normal random deviates rnorm(,mu,sd)
## so that the mean of transformed sample is not sufficient for mu,
## and that variance of transformed sample is not sufficient for sd,
blurred <- function(mu,s2,sample.size) {
  s <- rnorm(n=sample.size,mean=mu,sd=sqrt(s2))
  s <- exp(s/4)
  return(c(mean=mean(s),var=var(s)))
}

set.seed(123)
dSobs <- blurred(mu=4,s2=1,sample.size=20) ## stands for the actual data to be analyzed

## Sampling design as in canonical example
parsp <- init_grid(lower=c(mu=2.8,s2=0.4,sample.size=20),
                       upper=c(mu=5.2,s2=2.4,sample.size=20))
# simulate distributions
dsimuls <- add_simulation(,Simulate="blurred", par.grid=parsp)

## Use projection to construct better summary statistics for each each parameter
mufit <- project("mu",stats=c("mean","var"),data=dsimuls)
s2fit <- project("s2",stats=c("mean","var"),data=dsimuls)

## plots
```

```
mapMM(mufit,map.asp=1,
  plot.title=title(main="prediction of normal mean",xlab="exp mean",ylab="exp var"))
mapMM(s2fit,map.asp=1,
  plot.title=title(main="prediction of normal var",xlab="exp mean",ylab="exp var"))

## apply projections on simulated statistics
corrSobs <- project(dSobs,projectors=list("MEAN"=mufit,"VAR"=s2fit))
corrSimuls <- project(dsimuls,projectors=list("MEAN"=mufit,"VAR"=s2fit))

## Analyze 'projected' data as any data (cf canonical example)
densb <- infer_logLs(corrSimuls,stat.obs=corrSobs)
} else data(densb)
#########
if (Infusion.getOption("example_maxtime")>10) {
slik <- infer_surface(densb) ## infer a log-likelihood surface
slik <- MSL(slik) ## find the maximum of the log-likelihood surface
}
if (Infusion.getOption("example_maxtime")>500) {
slik <- refine(slik,10) ## refine iteratively
}
```

`refine` *Refine estimates iteratively.*

Description

This is a generic function with currently methods for `SLik` and `SLikp` objects (as produced by `MSL`). Depending on the value of its `newsimuls` argument, and on whether the function used to generate empirical distributions can be called by R, it (1) defines new parameters points and/or (2) infers their summary likelihood or tail probabilities for each parameter point independently, adds the inferred values results as input for refined inference of likelihood or P-value response surface, and provides new point estimates and confidence intervals.

Usage

```
## S3 method for class 'SLik'
refine(object, method=NULL, ...)


## Default S3 method:
refine(object, surfaceData, maxit = 1, n = NULL,
            useEI = list(max=TRUE,profileCI=TRUE,rawCI=FALSE),
            newsimuls = NULL, useCI = TRUE, level = 0.95,
            verbose = list(most=interactive(),movie=FALSE),
            precision = Infusion.getOption("precision"),
            method,...)
```

Arguments

object	an SLik object
surfaceData	A data.frame with attributes, usually taken from the object and thus **not** specified by user, usable as input for infer_surface.
maxit	Maximum number of iterative refinements (see also precision argument)
n	A number of parameter points (excluding replicates and confidence interval points), which likelihood should be computed (see n argument of sample_volume)
useEI	Cf this argument in rparam
newsimuls	For the SLik_j method, a matrix or data frame, with the same parameters and summary statistics as the data of the original infer_SLik_joint call. For other methods, a list of simulation of distributions of summary statistics, in the same format as for link{add_simulation}. If no such list is provided (i.e., if newsimuls remains NULL), the attr(object$logLs,"Simulate") function is used (it is inherited from the Simulate argument of add_simulation through the initial sequence of calls of functions add_simulation, infer_logLs or infer_tailp, and infer_surface). If no such function is available, then this function returns parameters for which new distribution should be provided by the user.
useCI	whether to include parameter points near the inferred confidence interval points in the set of points which likelihood should be computed
level	Intended coverage of confidence intervals
verbose	A list as shown by the default, or simply a vector of booleans, indicating respectively whether to display (1) some information about progress and results; (2) a plot of a the likelihood surface, if a one- or two-parameters surface is inferred. verbose$movie controls whether such plots are produced for all iterations. verbose$most controls other verbosity, including whether a final plot is produced.
precision	Requested local precision of surface estimation, in terms of prediction variance of both the maximum summary log-likelihood and the likelihood ratio at any CI bound available. Iterations will stop when either maxit is reached, or this precision is reached for these MSEs. A given precision on the CI bounds themselves might seem more interesting, but is not well specified by a single precision parameter if the parameters are on widely different scales.
method	(A vector of) suggested method(s) for estimation of smoothing parameters (see method argument of infer_surface). The ith element of the vector is used in the ith iteration, if available; otherwise the last element is used. This argument is not always heeded, in that REML may be used if the suggested method is GCV but it appears to perform poorly. The default for SLikp, SLik_j, and SLikp objects are "REML", "mixmodCluster", and "PQL", respectively.
...	further arguments passed to or from other methods. refine passes these arguments to the plot method suitable for the object.

Value

An updated SLik or SLik_j object.

Examples

```
## see main documentation page for the package
```

---

rparam *Sample the parameter space*

Description

These functions take an SLik object (as produced by MSL) and samples its parameter space in (hopefully) clever ways, not yet well documented. rparam calls sample_volume to define points targeting the likelihood maximum and the bounds of confidence intervals, with n for these different targets dependent on the mean square error of prediction of likelihood at the maximum and at CI bounds.

Usage

```
rparam(object, n= 1, useEI = list(max=TRUE,profileCI=TRUE,rawCI=FALSE),
       useCI = TRUE, verbose = interactive(), tryn=30*n,
       level = 0.95, CIweight=Infusion.getOption("CIweight"))

sample_volume(object, n = 6, useEI, vertices=NULL,
              dlr = NULL, verbose = interactive(),
              fixed = NULL, tryn= 30*n)
```

Arguments

object	an SLik object
n	The number of parameter points to be produced
useEI	List of booleans, each determining whether to use an "expected improvement" (EI) criterion (e.g. Bingham et al., 2014) to select candidate parameter points to better ascertain a particular focal point. The elements max, profileCI and rawCI determine this for three types of focal points, respectively the MSL estimate, profile CI bounds, and full-dimensional bounds. When EI is used, n points with best EI are selected among tryn points randomlygenerated in some neighborhood of the focal point.
vertices	Points are sampled within a convex hull defined by vertices. By default, these vertices are taken from objectfitdata.
useCI	Whether to define points targeting the bounds of confidence intervals for the parameters. An expected improvement criterion is also used here.
level	If useCI is TRUE but confidence intervals are not available from the object, such intervals are computed with coverage level.
dlr	A (log)likelihood ratio threshold used to select points in the upper region of the likelihood surface. Default value is given by Infusion.getOption("LRthreshold")
verbose	Whether to display some information about selection of points, or not

fixed	A list or named vector, of which each element is of the form <parameter name>=<value>, defining a one-dimensional constraint in parameter space. Points will be sampled in the intersection of the volume defined by the object and of such constraint(s).
tryn	See useEI argument.
CIweight	For development purposes, not documented.

Value

a data frame of parameter points. Only parameters variable in the SLik object are considered.

References

D. Bingham, P. Ranjan, and W.J. Welch (2014) Design of Computer Experiments for Optimization, Estimation of Function Contours, and Related Objectives, pp. 109-124 in Statistics in Action: A Canadian Outlook (J.F. Lawless, ed.). Chapman and Hall/CRC.

Examples

```
if (Infusion.getOption("example_maxtime")>10) {
 data(densv)
 summliksurf <- infer_surface(densv) ## infer a log-likelihood surface
 sample_volume(summliksurf)
}
```

Index