Package 'secrdesign'

June 4, 2014

Type Package

Title Sampling Design for Spatially Explicit Capture--Recapture

Version 2.1.3

Depends R (>= 3.0.0), secr(>= 2.8.2)

Imports parallel, abind

Date 2014-06-04

Author Murray Efford

Maintainer Murray Efford <murray.efford@otago.ac.nz>

Description Tools for designing spatially explicit capture--
recapture studies of animal populations (primarily a simulation manager for package secr)

License GPL (>=2)

URL http://www.otago.ac.nz/density

R topics documented:

`secrdesign-package` *Spatially Explicit Capture–Recapture Study Design*

Description

Tools to assist the design of spatially explicit capture–recapture studies of animal populations.

Details

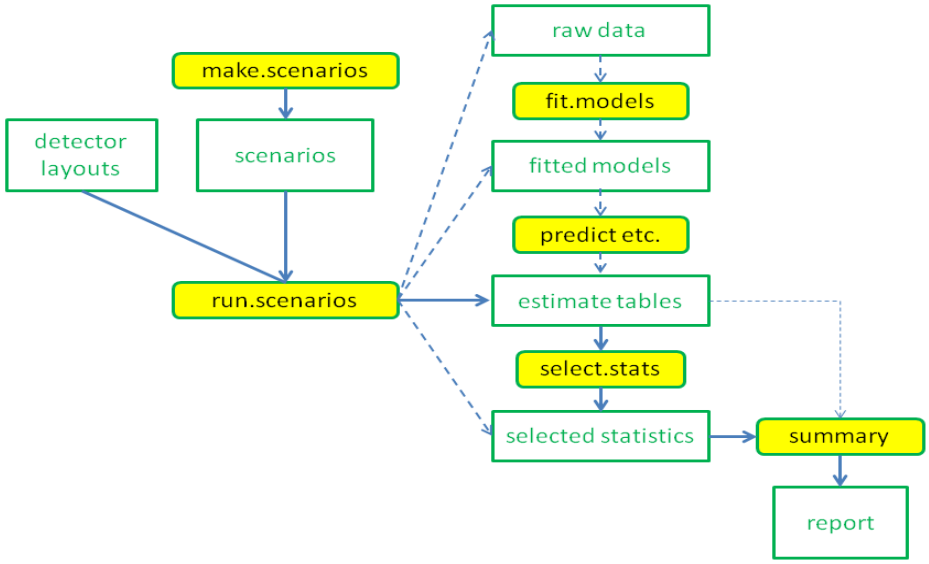
Package:	secr
Type:	Package
Version:	2.1.3
Date:	2014-06-04
License:	GNU General Public License Version 2 or later

The primary use of **secrdesign** is to predict by Monte Carlo simulation the precision or bias of density estimates from different detector layouts, given pilot values for density and the detection parameters g0 and sigma.

The important functions in **secrdesign** are:

`make.scenarios`	generate dataframe of parameter values etc.
`run.scenarios`	perform simulations, with or without model fitting
`fit.models`	fit SECR model(s) to rawdata output from `run.scenarios`
`predict.fittedmodels`	infer 'real' parameter estimates from fitted models
`select.stats`	collect output for a particular parameter
`summary.selectedstatistics`	numerical summary of results
`plot.selectedstatistics`	histogram or CI plot for each scenario

Documentation is provided in a vignette `../doc/secrdesign-vignette.pdf`; an Appendix has code for various examples that should help get you started. The help pages are also available as `../doc/secrdesign-manual.pdf`. The solid blue line in the following figure shows the usual pattern of use.



`run.scenarios` produces an object of class c(x, 'secrdesign', 'list') where x depends on the arguments 'fit' and 'extractfn'. The options for x (the primary class) are

Class	Output from `extractfn()` for each replicate –
`fittedmodels`	complete (or trimmed) fitted model from `secr.fit`

estimatetables	table of estimated parameters
selectedstatistics	vector of numerical values

When output from each replicate is saved from run.scenarios as a full or trimmed fitted model, further processing with predict.fittedmodels or related functions (coef.fittedmodels, derived.SL) will be needed before summarization. The default (when fit = TRUE) is to bypass this step by saving output from run.scenarios in 'estimatetables' form.

Author(s)

Murray Efford <murray.efford@otago.ac.nz>

See Also

make.grid, make.scenarios, run.scenarios, select.stats, predict.fittedmodels, summary.selectedstatist
plot.selectedstatistics, sim.popn, sim.capthist secr.fit

---

make.array *Re-cast Simulated Statistical Output as Array*

Description

This function is used internally by summary.secrdesign, and may occasionally be of general use.

Usage

```
make.array(object)
```

Arguments

object	secrdesign object containing numerical values for a particular parameter (i.e. output from select.stats inheriting from 'selectedstatistics')

Details

make.array converts a particular simulated numerical output into an array with one dimension for each varying input.

Value

A numeric array with dimensions corresponding to the varying inputs.

See Also

run.scenarios

Examples

```
## collect raw counts
scen1 <- make.scenarios(D = c(5,10), sigma = 25, g0 = 0.2)
traps1 <- make.grid()
tmp1 <- run.scenarios(nrepl = 50, trapset = traps1, scenarios = scen1,
    fit = FALSE)
make.array(tmp1)
```

`make.scenarios` *Construct Scenario Data Frame*

Description

This function prepares a dataframe in which each row specifies a simulation scenario. The dataframe is used as input to `run.scenarios`.

Usage

```
make.scenarios(trapsindex = 1, noccasions = 3,  nrepeats = 1, D, g0, sigma, lambda0,
detectfn = 0, recapfactor = 1, popindex = 1, detindex = 1, fitindex = 1,
crosstraps = TRUE)
```

Arguments

`trapsindex`	integer vector determining the traps object to use
`noccasions`	integer vector for the number of sampling occasions
`nrepeats`	integer vector of multipliers for D (see Details)
`D`	numeric vector of values for the density parameter (animals / hectare)
`g0`	numeric vector of values for the g0 parameter
`sigma`	numeric vector of values for the sigma parameter (m)
`lambda0`	numeric vector of values for the lambda0 parameter
`detectfn`	vector of valid detection function codes (numeric or character)
`recapfactor`	numeric vector of values for recapfactor (`sim.capthist`)
`popindex`	integer vector determining which population model is used
`detindex`	integer vector determining which detection options are used
`fitindex`	integer vector determining which model is fitted
`crosstraps`	logical; if TRUE the output includes all combinations of `trapsindex`, `noccasions` and `nrepeats`

Details

The index in `trapsindex` is used in `run.scenarios` to select particular detector arrays from the list of arrays provided as an argument to that function.

The function generates all combinations of the given parameter values using `expand.grid`. By default, it also generates all combinations of the parameters with `trapsindex` and the number of sampling occasions. If `crosstraps` is FALSE then `trapsindex`, `noccasions`, and `nrepeats` are merely used to fillin these columns in the output dataframe.

The argument `lambda0` replaces `g0` for the hazard detection functions 14–18 (`detectfn`).

Designs may use multiple detector arrays with the same internal geometry (e.g., number and spacing of traps). The number of such arrays is varied with the `nrepeats` argument. For example, you may compare designs with many small arrays or a few large ones. In practice, `run.scenarios` simulates a single layout is simulated with density D * nrepeats. This shortcut is not appropriate when animals compete for traps (detector = 'single').

`fitindex` allows a choice of different models when the argument `fit.args` of `run.scenarios` is a compound list.

Value

Dataframe with one row per scenario and the columns

`scenario`	a number identifying the scenario
`trapsindex`	
`noccasions`	
`nrepeats`	
`D`	
`g0`	or lambda0
`sigma`	
`detectfn`	see `detectfn`; always numeric
`recapfactor`	
`popindex`	
`detindex`	
`fitindex`	

See Also

`run.scenarios`, `sim.capthist`

Examples

```
make.scenarios(trapsindex = 1, nrepeats = 1, D = c(5,10), sigma = 25,
g0 = 0.2)
```

`predict.fittedmodels` *Extract Estimates From Fitted Models*

Description

If simulations have been saved from `run.scenarios` as fitted secr models it is necessary to use one of these functions to extract estimates for later summarization.

Usage

```
## S3 method for class 'fittedmodels'
predict(object, ...)

## S3 method for class 'fittedmodels'
coef(object, ...)

derived.SL(object, ...)

regionN.SL(object, ...)
```

Arguments

`object`	fitted model simulation output from `run.scenarios`
`...`	other arguments passed to predict, coef, derived or region.N

Details

These functions are used when output from `run.scenarios` has been saved as fitted models. `derived.SL` and `regionN.SL` require a full fit (including the design object) whereas a trimmed model is sufficient for `predict` and `coef`.

`derived.SL` is used to compute the Horvitz-Thompson-like estimate of density when `secr.fit` has been used with `CL = TRUE`; it is roughly equivalent to `predict`.

`regionN.SL` predicts the realised number (R.N) or expected number (E.N) in a masked area. When detector layouts and/or `sigma` vary, the masked area will also vary (arbitrarily, depending on the buffer argument 'xsigma') unless a mask is provided by the user; this may be done either in `run.scenarios` or in `regionN.SL`.

Value

An object with class ('estimatetables', 'secrdesign', 'list') with appropriate outputtype ('predicted', 'coef', 'derived', 'regionN'; sSee also `run.scenarios`).

See Also

`run.scenarios`

Examples

```
## using nrepl = 2 just for checking
scen1 <- make.scenarios(D = c(5,10), sigma = 25, g0 = 0.2)
traps1 <- make.grid()  ## default 6 x 6 grid of multi-catch traps
tmp1 <- run.scenarios(nrepl = 2, trapset = traps1, scenarios = scen1,
    fit = TRUE, extractfn = trim)
tmp2 <- predict(tmp1)
tmp3 <- select.stats(tmp2, 'D', c('estimate','RB','RSE'))
summary(tmp3)
```

`run.scenarios` *Simulate Sampling Designs*

Description

This function performs simulations to predict the precision of abundance estimates from simple 1-session SECR designs. Scenarios are specified via an input dataframe that will usually be constructed with `make.scenarios`. Each scenario comprises an index to a detector layout, the number of sampling occasions, and specified density (D) and detection parameters (usually g_0 and σ).

Detector layouts are provided in a separate list `trapset`. This may comprise an actual field design input with `read.traps` or 'traps' objects constructed with `make.grid` etc., as in the Examples. Even a single layout must be presented as a component of a list (e.g., `list(make.grid())`).

If `ncores` > 1 then each scenario will be run in a separate worker process using `parLapply` from **parallel** (see also Parallel). Setting `ncores` greater than the number of scenarios causes an error.

Alternative approaches are offered for predicting precision. Both start by generating a pseudorandom dataset under the design using the parameter values for a particular scenario. The first estimates the parameter values and their standard errors from each dataset by maximizing the full likelihood, as usual in `secr.fit`. The second takes the short cut of computing variances and SE from the Hessian estimated numerically at the known expected values of the parameters, without maximizing the likelihood. Set `method = "none"` for this shortcut.

Usage

```
run.scenarios(nrepl, scenarios, trapset, masksset, xsigma = 4, nx = 32,
    pop.args, det.args, fit = FALSE, fit.args, extractfn = NULL, ncores = 1,
    seed = 123, ...)

fit.models(rawdata, fit = FALSE, fit.args, extractfn = NULL,
    ncores = 1, ...)
```

Arguments

`nrepl`	integer number of replicate simulations
`scenarios`	dataframe of simulation scenarios
`trapset`	secr traps object or a list of traps objects
`masksset`	secr mask object or a list of mask objects (optional)
`xsigma`	numeric buffer width as multiple of sigma (alternative to maskset)
`nx`	integer number of cells in mask in x direction (alternative to maskset)

pop.args	list of named arguments to sim.popn (optional)
det.args	list of named arguments to sim.capthist (optional)
fit	logical; if TRUE a model is fitted with secr.fit, otherwise data are generated but no model is fitted
fit.args	list of named arguments to secr.fit (optional)
extractfn	function to extract a vector of statistics from secr model
ncores	integer number of cores for parallel processing
seed	integer pseudorandom number seed
...	other arguments passed to extractfn
rawdata	'rawdata' object from previous call to run.scenarios

Details

Designs are constructed from the trap layouts in trapset, the numbers of grids in ngrid, and the numbers of sampling occasions (secondary sessions) in noccasions. These are *not* crossed: the number of designs is the maximum length of any of these arguments. Any of these arguments whose length is less than the maximum will be replicated to match.

pop.args is used to customize the simulated population distribution. It will usually comprise a single list, but may be a list of lists (one per popindex value in scenarios).

det.args may be used to customize some aspects of the detection modelling in sim.capthist, but not traps, popn, detectpar, detectfn, and noccasions, which are controlled directly by the scenarios. It will usually comprise a single list, but may be a list of lists (one per detindex value in scenarios).

fit.args is used to customize the fitted model; it will usually comprise a single list. If you are interested in precision alone, use fit.args=list(method = 'none') to obtain variance estimates from the hessian evaluated at the parameter estimates. This is much faster than a complete model fit, and usually accurate enough.

If no extractfn is supplied then a default is used - see Examples. Replacement functions should follow this pattern i.e. test for whether the single argument is an secr object, and if not supply a named vector of NA values of the correct length.

The L'Ecuyer pseudorandom generator is used with a separate random number stream for each core (see clusterSetRNGStream).

A summary method is provided (see summary.secrdesign). It is usually necessary to process the simulation results further with predict.fittedmodels and/or select.stats before summarization.

Value

An object of class (x, 'secrdesign', 'list'), where x is one of 'fittedmodels', 'estimatetables', 'selectedstatistics' or 'rawdata', with components

call	function call
version	character string including the software version number
starttime	character string for date and time of run
proctime	processor time for simulations, in seconds
scenarios	dataframe as input
trapset	list of trap layouts as input

maskset	list of habitat masks (input or generated)
xsigma	from input
nx	from input
pop.args	from input
det.args	from input
fit	from input
fit.args	from input
extractfn	function used to extract statistics from each simulation
seed	from input
nrepl	from input
output	list with one component per scenario
outputtype	character code - see vignette

If `fit = FALSE` and `extractfn = identity` the result is of class ('rawdata', 'secrdesign', 'list'). This may be used as input to `fit.models`, which interprets each model specification in `fit.args` as a new 'sub-scenario' of each input scenario (i.e. all models are fitted to every dataset). The output possibilities are the same as for `run.scenarios`.

Note

100 ha = 1 km^2

Note

For `ncores > 1` it pays to keep an eye on the processes from the Performance page of Windows Task Manager (<ctrl><alt><del>), or 'top' in linux OS. If you interrupt `run.scenarios` (<Esc> from Windows) you may occasionally find some processes do not terminate and have to be manually terminated from the Task Manager - they appear as Rscript.exe on the Processes page.

Author(s)

Murray Efford

See Also

`predict.fittedmodels`, `select.stats`, `summary.secrdesign`, `summary.selectedstatistics`, `sim.popn`, `sim.capthist`, `secr.fit`

Examples

```
## Simple example: generate and summarise trapping data
## at two densities and for two levels of sampling frequency
scen1 <- make.scenarios(D = c(5,10), sigma = 25, g0 = 0.2, noccasions =
    c(5,10))
traps1 <- make.grid()   ## default 6 x 6 trap grid
tmp1 <- run.scenarios(nrepl = 20, trapset = traps1, scenarios = scen1,
    fit = FALSE)
summary(tmp1)

## Not run:
```

```
############################
## 2-phase example
## first make and save rawdata
scen1 <- make.scenarios(D = c(5,10), sigma = 25, g0 = 0.2)
traps1 <- make.grid()   ## default 6 x 6 trap grid
tmp1 <- run.scenarios(nrepl = 20, trapset = traps1, scenarios = scen1,
    fit = FALSE, extractfn = identity)

## review rawdata
summary(tmp1)

## then fit and summarise models
tmp2 <- fit.models(tmp1, fit.args = list(list(model = g0~1),
    list(model = g0~T)), fit = TRUE, ncores = 4)
summary(tmp2)
############################

## Construct a list of detector arrays
## Each is a set of 5 parallel lines with variable between-line spacing;
## the argument that we want to vary (spacey) follows nx, ny and spacex
## in the argument list of make.grid().

spacey <- seq(2000,5000,500)
names(spacey) <- paste('line', spacey, sep = '.')
trapset <- lapply(spacey, make.grid, nx = 101, ny = 5, spacex = 1000,
    detector = 'proximity')

## Make corresponding set of masks with constant spacing (1 km)
maskset <- lapply(trapset, make.mask, buffer = 8000, spacing = 1000,
    type = 'trapbuffer')

## Generate scenarios
scen <- make.scenarios (trapsindex = 1:length(spacey), nrepeats = 8,
    noccasions = 2, D = 0.0002, g0 = c(0.05, 0.1), sigma = 1600, cross = TRUE)

## RSE without fitting model
sim <- run.scenarios (50, scenarios = scen, trapset = trapset, maskset = maskset,
    ncores = 8, fit = TRUE, fit.args = list(method = 'none'), seed = 123)

## Extract statistics for predicted density
sim <- select.stats(sim, parameter = 'D')

## Plot to compare line spacing
summ <- summary (sim, type='array',  fields = c('mean','lcl','ucl'))$summary
plot(0,0,type='n', xlim=c(1.500,5.500), ylim = c(0,0.36), yaxs = 'i',
    xaxs = 'i', xlab = 'Line spacing  km', ylab = 'RSE (D)')
xv <- seq(2,5,0.5)
points(xv, summ$mean[,1,'RSE'], type='b', pch=1)
points(xv, summ$mean[,2,'RSE'], type='b', pch=16)
segments(xv, summ$lcl[,1,'RSE'], xv, summ$ucl[,1,'RSE'])
segments(xv, summ$lcl[,2,'RSE'], xv, summ$ucl[,2,'RSE'])
legend(4,0.345, pch=c(1,16), title = 'Baseline detection',
    legend = c('g0 = 0.05', 'g0 = 0.1'))

## End(Not run)
```

`select.stats` *Select Statistics to Summarize*

Description

When the results of each simulation with `run.scenarios` are saved as a dataframe (e.g. from predict()) it is necessary to select estimates of just one parameter for numerical summarization. This does the job. `find.param` is a helper function to quickly display the parameters available for summarisation.

Usage

```
select.stats(object, parameter = "D", statistics)
find.param(object)
find.stats(object)
```

Arguments

`object`	'estimatetables' object from `run.scenarios`
`parameter`	character name of parameter to extract
`statistics`	character vector of statistic names

Details

`select.stats` is used to select a particular vector of numeric values for summarization. The 'parameter' argument indexes a row in the data.frame for one replicate (i.e., one 'real' parameter). Each 'statistic' is either a column in that data.frame or a statistic derived from a column.

If `statistics` is not specified, the default is to use all numeric columns in the input (i.e., c('estimate', 'SE.estimate', 'lcl', 'ucl') for predict and c('beta', 'SE.beta', 'lcl', 'ucl') for coef).

`statistics` may include any of 'estimate', 'SE.estimate', 'lcl', 'ucl', 'true', 'RB', 'RSE', 'COV' and 'ERR' (for outputtype 'coef' use 'beta' and 'SE.beta' instead of 'estimate and 'SE.estimate'). 'true' refers to the known parameter value used to generate the data.

The computed statistics are:

Statistic	Name	Value
RB	Relative bias	(estimate - true) / true
RSE	Relative SE	SE.estimate / estimate
ERR	Absolute deviation	abs(estimate - true)
COV	Coverage	(estimate > lcl) & (estimate < ucl)

'RB', 'COV' and 'ERR' relate an estimate to the known (true) value of the parameter in `object$scenarios`. They are computed only when a model has been fitted without method = 'none'.

'COV' remains binary (0/1) in the output from `select.stats`; the result of interest is the mean of this statistic across replicates (see `summary.secrdesign`). Similarly, 'ERR' is used with field 'rms' in `summary.secrdesign` to compute the root-mean-squared-error RMSE.

`find.param` and `find.stats` may be used to 'peek' at objects of class 'estimatetables' and 'select-

edstatistics' respectively to recall the available parameter estimates or 'statistics'.

Value

For `select.stats`, an object with class c('selectedstatistics','secrdesign', 'list') suitable for numerical summarization with `summary.selectedstatistics`. The value of 'parameter' is stored as an attribute.

For `find.param`, a character vector of the names of parameters with estimates in `object`.

See Also

`run.scenarios`, `validate`

Examples

```
## using nrepl = 2 just for checking
scen1 <- make.scenarios(D = c(5,10), sigma = 25, g0 = 0.2)
traps1 <- make.grid()
tmp1 <- run.scenarios(nrepl = 2, trapset = traps1, scenarios = scen1,
    fit = TRUE, extractfn = trim)
tmp2 <- predict(tmp1)
tmp3 <- select.stats(tmp2, 'D', c('estimate','RB','RSE','COV'))
summary(tmp3)
```

---

summary.secrdesign *Generic Methods for secrdesign Objects*

Description

Methods to summarize simulated datasets.

Usage

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

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

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

## S3 method for class 'selectedstatistics'
summary(object, dec = 5, fields = c('n', 'mean',
'se'), alpha = 0.05, type = c('list','dataframe','array'), ...)

## S3 method for class 'selectedstatistics'
plot(x, scenarios, statistic, type =
c('hist', 'CI'), refline, xlab = NULL, ...)

header(object)
```

Arguments

object	object of class simulations from run.scenarios
dec	number of decimal places in output
fields	character vector; names of required summary statistics (see Details)
alpha	alpha level for confidence intervals and quantiles
type	character code for type of output (see Details)
...	other arguments – not currently used by summary but passed to hist by the plot method
x	object of class 'selectedstatistics' from run.scenarios
scenarios	integer indices of scenarios to plot (all plotted if not specified)
statistic	integer or character indices if the statistics in x for which histograms are requested
refline	logical; if TRUE a reference line is plotted at the true value of a parameter
xlab	character; optional label for x-axis

Details

If object inherits from 'selectedstatistics' then the numeric

If object inherits from 'selectedstatistics' then the numeric results from replicate simulations are summarized using the chosen 'fields' (by default, the number of non-missing values, mean and standard error), along with header information describing the simulations. Otherwise the header alone is returned.

fields is a vector of any selection from c('n', 'mean', 'sd', 'se', 'min', 'max', 'lcl', 'ucl', 'median', 'q', 'rms'), or the character value 'all'.

Field 'q' provides 1000 alpha/2 and 1000[1 - alpha/2] quantiles qxxx and qyyy.

'lcl' and 'ucl' refer to the upper and lower limits of a 100(1 - alpha)% confidence interval for the statistic, across replicates.

'rms' gives the root-mean-square of the statistic - most useful for the statistic 'ERR' (see select.stats) when it represents the overall accuracy or RMSE.

The plot method plots either (i) histograms of the selected statistics (type = 'hist') or (ii) the estimate and confidence interval for each replicate (type = 'CI'). The default for type = 'hist' is to plot the first statistic - this is usually 'n' (number of detected animals) when fit = FALSE, and 'estimate' (parameter estimate) when fit = TRUE. If length(statistic) > 1 then more than one plot will be produced, so a multi-column or multi-row layout should be prepared with par arguments 'mfcol' or 'mfrow'.

For type = 'CI' the statistics must include 'estimate', 'lcl' and 'ucl' (or 'beta', 'lcl' and 'ucl' if outputtype = 'coef').

Value

List with components 'header'

call	original function call
starttime	from object
proctime	from object
constants	small dataframe with values of non-varying inputs

varying	small dataframe with values of varying inputs
fit.args	small dataframe with values arguments for secr.fit, if specified

and 'OUTPUT', a list with one component for each field. Each component may be a list or an array.

See Also

run.scenarios, make.array, select.stats validate

Examples

```
## collect raw counts
scen1 <- make.scenarios(D = c(5,10), sigma = 25, g0 = 0.2)
traps1 <- make.grid()
tmp1 <- run.scenarios(nrepl = 50, trapset = traps1, scenarios = scen1,
    fit = FALSE)

opar <- par(mfrow=c(2,3))
plot(tmp1, statistic = 1:3)
par(opar)

summary(tmp1)

summary(tmp1, field=c('q025', 'median', 'q975'))
```

---

validate *Reject Implausible Statistics*

Description

Simulation output may contain rogue values due to idiosyncracies of model fitting. For example, nonidentifiability due to inadequate data can result in spurious extreme 'estimates' of the sampling variance. Undue influence of rogue replicates can be reduced by using the median as a summary field rather than the mean. This function is another way to deal with the problem, by setting to NA selected statistics from replicates for which some 'test' statistic is out-of-range.

Usage

```
validate(x, test, validrange = c(0, Inf), targets = test)
```

Arguments

x	object that inherits from 'selectedstatistics'
test	character; name of statistic to check
validrange	numeric vector comprising the minimum and maximum permitted values of 'test', or a matrix (see details)
targets	character vector with names of one or more statistics to set to missing (NA) when test is out-of-range

Details

Values of 'test' and 'targets' should be columns in each component 'replicate x statistic' matrix (i.e., scenario) of `x$output`. You can check for these with `find.stats`.

If `validrange` is a matrix its first and second columns are interpreted as scenario-specific bounds (minima and maxima), and the number of rows must match the number of scenarios.

If all non-missing values of 'test' are in the valid range, the effect is to force the target statistics to NA wherever 'test' is NA.

The default is to change only the test field itself. If the value of 'test' does not appear in 'targets' then the test field is unchanged.

If `targets = "all"` then all columns are set to NA when the test fails.

Value

An object of class c('selectedstatistics', secrdesign', 'list') with the same structure and header information as the input, but possibly with some values in the 'output' component converted to NA.

See Also

`select.stats`, `find.stats`

Examples

```
## Not run:

## generate some data
scen1 <- make.scenarios(D = c(5,10), sigma = 25, g0 = 0.2)
traps1 <- make.grid()
tmp1 <- run.scenarios(nrepl = 5, trapset = traps1, scenarios = scen1,
    fit = TRUE, extractfn = trim)
tmp2 <- predict(tmp1)
tmp3 <- select.stats(tmp2, 'D', c('estimate','RB','RSE','COV'))

## just for demonstration --
## apply scenario-specific +/- 20% bounds for estimated density
## set RB, RSE and COV to NA when estimate is outside this range
permitted <- outer(tmp3$scenarios$D, c(0.8,1.2))
permitted   ## a 2 x 2 matrix
tmp4 <- validate(tmp3, 'estimate', permitted, c('RB', 'RSE','COV'))

## what have we done?!
tmp4$output
summary(tmp4)


## End(Not run)
```

Index