Package 'sra'

July 23, 2025

Type Package

Title Selection Response Analysis

Version 0.1.4.1

Date 2024-02-14

Author Arnaud Le Rouzic

Maintainer Arnaud Le Rouzic <arnaud.le-rouzic@universite-paris-saclay.fr>

Imports stats, stats4, graphics

URL https://github.com/lerouzic/sra

Description

Artificial selection through selective breeding is an efficient way to induce changes in traits of interest in experimental populations. This package (sra) provides a set of tools to analyse artificial-selection response datasets. The data typically feature for several generations the average value of a trait in a population, the variance of the trait, the population size and the average value of the parents that were chosen to breed. Sra implements two families of models aiming at describing the dynamics of the genetic architecture of the trait during the selection response. The first family relies on purely descriptive (phenomenological) models, based on an autoregressive framework. The second family provides different mechanistic models, accounting e.g. for inbreeding, mutations, genetic and environmental canalization, or epistasis. The parameters underlying the dynamics of the time series are estimated by maximum likelihood. The sra package thus provides (i) a wrapper for the R functions mle() and optim() aiming at fitting in a convenient way a predetermined set of models, and (ii) some functions to plot and analyze the output of the models.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2024-02-14 09:40:13 UTC

Contents

---

1. sra package *Selection Response Analysis: a R package*

Description

This package (sra) provides a set of tools to analyse artificial-selection response datasets. The data typically feature for several generations the average value of a trait in a population, the variance of the trait, the population size and the average value of the parents that were chosen to breed. sra implements two families of models aiming at describing the dynamics of the genetic architecture of the trait during the selection response. The first family relies on purely descriptive (phenomenological) models, based on an autoregressive framework. The second family provides different mechanistic models, accounting e.g.\ for inbreeding, mutations, genetic and environmental canalization, or epistasis. The parameters underlying the dynamics of the time series are estimated by maximum likelihood. The sra package thus provides (i) a wrapper for the R functions mle and optim aiming at fitting in a convenient way a predetermined set of models, and (ii) some functions to plot and analyze the output of the models.

Details

Package:	sra
Type:	Package
Version:	0.1
License:	GPL-2

Data set The user must provide (i) a vector containing the mean phenotype for all generations, (ii) a vector containing the phenoypic variance, (iii) a vector for the population sizes, (iv) a vector for the mean phenotype of the breeders, (v) (if available) a vector of the phenotypic variances among breeders. Several time series (e.g.\ several lines submitted to similar or different selection pressures) can be analyzed.

Phenomenological models The function sraAutoreg fits an autoregressive model to the variance trends, and provides a description of the dynamics of the genetic architecture that is not based on *a priori* biological mechanisms. The complexity of the model can be adjusted by adding extra autoregressive parameters.

Scaling The relevant scale for genetic architecture models is not necessarily the original measurement scale. Autoregressive models can be run from data e.g.\ on a log scale, but the package also provides two additional 'scales' that are specific to genetic architecture properties. sraAutoregHerit fits the same models as sraAutoreg, but considering that the dynamics of environmental variance

is constrained by the quantity known as "heritability" (ratio between additive and phenotypic variances). sraAutoregEvolv proposes that both genetic and environmental variances are constrained by the mean of the population ("mean-scaled evolvability").

Mechanistic models Classical quantitative-genetics models are also provided. sraCstvar implements a "constant-variance" model, sraDrift considers the effects of inbreeding on the additive genetic variance, sraMutation introduces some mutational variance, sraCanalization illustrates the impact of a simple model of genetic and environmental canalization, sraDirepistasis considers directional epistasis, and sraSelection models the effect of unavoidable natural (stabilizing) selection competing with directional artificial selection.

Plotting and analysis All the models provide an objet of class srafit that can be plotted directly (see e.g. plot.srafit).

Author(s)

Arnaud Le Rouzic

Maintainer: Arnaud Le Rouzic <a.p.s.lerouzic@bio.uio.no>

References

Le Rouzic, A., Houle, D., and Hansen, T.F. (2011) A modelling framework for the analysis of artificial selection-response time series. Genetics Research.

Examples

```
############ Generating a dummy dataset ################

m <- c(12,11,12,14,18,17,19,22,20,19)
v <- c(53,47,97,155,150,102,65,144,179,126)
s <- c(15,14,14,17,21,20,22,25,24,NA)
n <- c(100,80,120,60,100,90,110,80,60,100)

########## Making a sra data set ######################
data <- sraData(phen.mean=m, phen.var=v, phen.sel=s, N=n)

#################### Data Analysis ####################

# Autoregressive models
autor <- sraAutoreg(data)

# AIC of the model:
AIC(autor)

# Maximum-likelihood estimates
coef(autor)

autor.herit <- sraAutoregHerit(data)
autor.evolv <- sraAutoregEvolv(data)

# Mechanistic models
```

```
# Constant variance
cstvar <- sraCstvar(data)
# Inbreeding
drift <- sraDrift(data)

# Plotting
plot(drift)
plot(drift, var=TRUE)
```

---

2. Phenomenological models

Descriptive models of artificial-selection responses: auto-regressive models

---

Description

The sraAutoreg functions are wrappers for the maximum-likelihood optimization function mle. They propose descriptive models for the dynamics of genetic architectures of different complexities based on an auto-regressive framework, additional parameters corresponding to different generation lags. The model can also be fit considering logatithmic, "heritability" and "evolvability" scales.

Usage

```
sraAutoreg(sradata, active = c(FALSE, TRUE, FALSE, FALSE),
start = NULL, fixed = NULL, negative.k = FALSE,
rand = 0, rep = 1, ...)
sraAutoregLog(sradata, active = c(FALSE, TRUE, FALSE, FALSE),
start = NULL, fixed = NULL, negative.k = FALSE,
rand = 0, rep = 1, ...)
sraAutoregHerit(sradata, active = c(FALSE, TRUE, FALSE, FALSE),
start = NULL, fixed = NULL, negative.k = FALSE,
rand = 0, rep = 1, ...)
sraAutoregEvolv(sradata, active = c(FALSE, TRUE, FALSE, FALSE),
start = NULL, fixed = NULL, negative.k = FALSE,
rand = 0, rep = 1, ...)
```

Arguments

sradata	A data object generated by sraData.
active	A vector of four booleans, corresponding to the active lags for both genetic and environmental variances (see Details). By default, only lag 1 is active, corresponding to an exponential change of the variances.
start	A named list of starting values for the convergence algorithm. NULL is allowed and lets the function sraStartingvalues calculating a (hopefully) educated guess on the starting values. See Details.
fixed	A named list of the parameters that have to be kept constant. NULL is allowed.

`negative.k` Whether or not the k parameters can take negative values. Negtive values for k lead to more complex likelihood functions, and the resulting dynamics may display cyclic patterns.

`rand` Amount of randomness for the starting values. Useful in case of convergence issues. Although this variable can take any positive value, reasonable figures should not exceed 0.2.

`rep` Number of convergence attempts. When the likelihood function is complex, which is often the case when the number of parameters exceeds 6 to 8, convergence may often fail or end up on a local maximum. When `rep` > 1, several attempts are made from different starting values (the amount of randomness for the starting values being set by the `rand` parameter), and the best fit (highest likelihood) is returned. Setting high values of `rep` may slow down significantly the convergence process. In practice, 10 to 30 repetitions with `rand` = 0.1 are generally enough to ensure convergence.

`...` Additional parameters to be passed to `mle`, and thus to `optim`.

Details

Model

The following summarizes the models developed in Le Rouzic et al. 2010.

The mean of the population μ changes according to the Lande equation (Lande and Arnold 1983):

$$\mu(t+1) = \mu(t) + VarA(t) * \beta(t),$$

where $\beta(t)$ is the selection gradient at generation t.

The genetic architecture models predict the dynamics of a parameter P as:

$$P(t+1) = k0 + k1 * P(t) + k2 * P(t-1) + k3 * P(t-2)$$

Models with time lags > 3 could be easily implemented, but convergence issues increase with the number of parameters. The first time points are calculated as if $P(t < 1) = P(1)$, e.g. $P(3) = k0 + k1 * P(2) + k2 * P(1) + k3 * P(1)$.

Each model considers the dynamics of two independent parameters, one related to the additive genetic variance ($VarA$), one related to the residual (environmental) variance $VarE$ (which actually also accounts for all non-additive genetic variance).

The default model `sraAutoreg` considers directly the dynamics of $VarA$ (parameters: `kA0`, `kA1`, `kA2`, and `kA3`) and the dynamics of $VarE$ (parameters `kE0`, `kE1`, `kE2`, and `kE3`).

The log scale turns a multiplicative trait into an additive one, and is particularly relevant for ratio-scale traits (i.e. most quantitative traits such as size, fertility, etc). The original data is transformed assuming log-normality, and the likelihood is computed based on the log-normal density function.

The "heritability" model `sraAutoregHerit` focuses on the dynamics of $h^2 = VarA/(VarA + VarE)$, described by the parameters `kA0`, `kA1`, `kA2`, and `kA3`, and considers that `kE0`, `kE1`, `kE2`, and `kE3` describe the dynamics of the phenotypic variance $VarP = VarA + VarE$. Therefore, $VarE$ is constrained both by the dynamics of $VarP$ and the independent dynamics of h^2.

The "evolvability" model considers that kA0, kA1, kA2, and kA3 describe the dynamics of $IA(t) = VarA(t)/(\mu(t)^2)$, and kE0, kE1, kE2, and kE3 the dynamics of $IE(t) = VarE(t)/(\mu(t)^2)$.

Shortcut for active and inactive parameters

The user will often have to fit models of different complexity. This can be achieve by manipulating the active vector. c(FALSE,FALSE,FALSE,FALSE) corresponds to a constant-variance model (no dynamic parameter), c(TRUE,FALSE,FALSE,FALSE) to a case in which only kA0 and kE0 are active, c(TRUE,TRUE,FALSE,FALSE) to active parameters for lags 0 and 1 only, etc. The total number of parameters in the model will be $3 + 2 * x$, where x is the number of TRUE in the vector active.

To bypass the constrains of this shortcut, it is possible to specify the active and inactive parameters through the list of starting values. A combination such as active=c(TRUE,FALSE,TRUE,FALSE), start=list(kA1=0,kE3=NA), fixed=list(kE2=1) will lead to a model with 8 active parameters (mu0, varA0, varE0, kA0, kE0, kA1 (which starting value will be 0), kA2, and kE3 (which starting value, specified as NA, will be determined via the function sraStartingvalues. All other parameters are fixed.

Parameterization

The models thus involve up to 11 parameters: three initial values ($\mu(1)$, $VarA(1)$ and $VarE(1)$), four parameters to describe the dynamics of the additive variance (or relative variable such as IA or h^2) (kA0, kA1, kA2, and kA3), and four parameters for the environmental variance (or IE, or h^2): kE0, kE1, kE2, and kE3. To make numerical convergence more efficient, the following parameterization was implemented: parameters mu0, logvarA0 and logvarE0 correspond to the estimates of the initial values of, respectively, the population mean, the logarithm of the additive variance, and the logarithm of the environmental variance. The parameters kA0 and kE0 are calculated as relative to the initial values of the dynamic variable, e.g. relativekA0 = $k0A/VarA(1)$ (so that relativekA0 has the same unit and the same order of magnitude as kA1, kA2 and kA3).

Value

The functions return objects of class srafit, a list containing information about the model, the data, and the parameter estimates. Some standard R functions can be applied to the object, including AIC (AIC.srafit), logLik (logLik.srafit), vcov (vcov.srafit), coef (coef.srafit) confint (confint.srafit), and plot (plot.srafit).

Author(s)

Arnaud Le Rouzic

References

Le Rouzic, A., Houle, D., and Hansen, T.F. (2011) A modelling framework for the analysis of artificial selection-response time series. Genetics Research.

Lande, R., and Arnold, S. (1983) The measurement of selection on correlated characters. Evolution 37:1210-1226.

See Also

sraCstvar, sraDrift and all other mechanistic models, sraAutoregTsMinuslogL and sraAutoregTimeseries for some details about the internal functions, AIC.srafit, logLik.srafit, vcov.srafit, coef.srafit, confint.srafit, plot.srafit for the analysis of the results.

Examples

```
# Making the example reproducible
########### Generating a dummy dataset ################

m <- c(12,11,12,14,18,17,19,22,20,19)
v <- c(53,47,97,155,150,102,65,144,179,126)
s <- c(15,14,14,17,21,20,22,25,24,NA)
n <- c(100,80,120,60,100,90,110,80,60,100)

########## Making a sra data set ######################
data <- sraData(phen.mean=m, phen.var=v, phen.sel=s, N=n)

################### Data Analysis ###################

# Autoregressive models
autor <- sraAutoreg(data)

# Details of the model:
AIC(autor)
coef(autor)
plot(autor)
plot(autor, var=TRUE)

# Alternative scales
autor.log <- sraAutoregLog(data)
autor.herit <- sraAutoregHerit(data)
autor.evolv <- sraAutoregEvolv(data)


# Changes in the complexity of the model:
autor0 <- sraAutoreg(data, active=c(TRUE,TRUE,FALSE,FALSE))

# In case of convergence issues
autor1 <- sraAutoreg(data, active=c(TRUE,TRUE,TRUE,TRUE), rep=2, rand=0.1)
```

---

3. Mechanistic models *Descriptive models of artificial-selection responses: quantitative genetics models*

Description

The sra functions for mechanistic model are wrappers for the maximum-likelihood optimization routine `mle`. They implement classical quantitative genetics models, fit them to artificial-selection time series, and provide estimates of e.g. the effective population size, the mutational variance, the strength of genetic / environmental canalization, the directionality and strength of epistasis, etc., given some assumptions about the properties of the genetic architecture.

Usage

```
sraCstvar(sradata, start=NULL, fixed=NULL, macroE=FALSE, Bulmer=TRUE, ...)
sraDrift(sradata, start=NULL, fixed=NULL, macroE=FALSE, Bulmer=TRUE, ...)
sraMutation(sradata, start=NULL, fixed=NULL, macroE=FALSE, Bulmer=TRUE, ...)
sraCanalization(sradata, start=NULL, fixed=NULL, macroE=FALSE, Bulmer=TRUE, ...)
sraCanalizationOpt(sradata, start=NULL, fixed=NULL, macroE=FALSE, Bulmer=TRUE, ...)
sraSelection(sradata, start=NULL, fixed=NULL, macroE=FALSE, Bulmer=TRUE, ...)
sraDirepistasis(sradata, start=NULL, fixed=NULL, macroE=FALSE, ...)
```

Arguments

sradata	A data object generated by sraData.
start	A named list of starting values for the convergence algorithm. NA is allowed and lets the function sraStartingvalues calculating a (hopefully) educated guess on the starting values. Parameters are described below.
fixed	A named list of the parameters that have to be kept constant. NA is not allowed.
macroE	Whether or not macro-environmental effects (random deviation of the phenotypic mean each generation) should be included. This might have some side effects.
Bulmer	Whether or not the impact of linkage disequilibrium (Bulmer effect) due to selection on variance should be accounted for.
...	Additional parameters to be passed to mle, and thus to optim.

Details

All functions (except sraDirepistasis) rely on the same underlying model, and thus simply provide convenient shortcuts for different sets of parameters to fit.

mu0 is the initial phenotype of the population.

logvarA0 is the logarithm of the initial additive variance in the population.

logvarE0 is the logarithm of the initial environmental variance in the population.

logNe is the logarithm of the effective population size.

logn is the logarithm of the effective number of loci.

logvarM is the logarithm of the mutational variance.

kc and kg are the strength of environmental and genetic canalization, respectively.

o corresponds to the 'optimum' phenotype. When fixed and set to NA, o is identical to mu0. For convenience, the same optimum is used for environmental canalization, genetic canalization and natural stabilizing selection.

s corresponds to the strength of natural selection.

logvarepsilon is the logarithm of the variance of the epistatic coefficient (ε). This parameter is fixed by default, since it is unlikely that realistic data sets contain enough information to estimate it properly.

The dynamic model that is fitted (except of the directional epistasis, detailed in the next paragraph) is:

$$\mu(t+1) = \mu(t) + VarA(t) * (\beta(t) + s\delta(t))$$
$$VarA(t+1) = VarM + VarA(t) * (1 - 1/(2 * N_e)) * e^{kg*(|\delta(t+1)|-|\delta(t)|)}$$
$$VarE(t+1) = VarE(t) * e^{kc*|\delta(t)|}$$

$\mu(1)$, $VarA(1)$ and $varE(1)$ are parameters of the model, $\beta(t)$ is the selection gradient, calculated for each generation from the data set, and $\delta(t) = \mu(t) - o$.

The directional epistasis model has its own setting:

$$\mu(t+1) = \mu(t) + varA(t) * \beta(t)$$
$$VarA(t+1) = VarA(t) + 2 * \beta(t) * \varepsilon * VarA(t)^2$$
$$VarE(t+1) = VarE(1) + (\varepsilon^2 + Var\varepsilon) * VarA(t)^2$$

Where `epsilon` is a key parameter, representing the directionality of epistasis (Hansen and Wagner 2001, Carter et al. 2005). The properties of the likelihood function when epsilon varies makes numerical convergence tricky, and the function `SRAdirepistasis` actually performs two model fits: one for positive epsilons (the estimated parameter being `logepsilon`) and one for negative epsilons (estimating `logminusepsilon`). The likelihood of both models is then compared, and the best model is returned, providing either `logepsilon` or `logminusepsilon`. Although part of the model, the parameter `logvarepsilon` appeared to affect environmental variance only weakly, and its estimation is problematic in most cases.

These models are extensively described in le Rouzic et al 2010.

Value

The functions return objects of class `srafit`, a list containing information about the model, the data, and the parameter estimates. Some standard R functions can be applied to the object, including `AIC` (`AIC.srafit`), `logLik` (`logLik.srafit`), `vcov` (`vcov.srafit`), `coef` (`coef.srafit`) `confint` (`confint.srafit`), and `plot` (`plot.srafit`).

Author(s)

Arnaud Le Rouzic

References

Carter, A.J.R., Hermisson, J. and Hansen, T.F. (2005) The role of epistatic genetic interactions in the response to selection and the evolution of evolvability. Theor. Pop. Biol. 68, 179-196.

Hansen, T.F. and Wagner, G.P. (2001) Modelling genetic architecture: a multilinear theory of gene interaction. Theor. Pop. biol. 59, 61-86.

Le Rouzic, A., Houle, D., and Hansen, T.F. (2010) A modelling framework for the analysis of artificial selection-response time series. in prep.

See Also

`sraAutoreg`, `sraAutoregHerit` and `sraAutoregEvolv` for phenomenological models, `sraAutoregTsMinuslogL` and `sraAutoregTimeseries` for some details about the internal functions, `AIC.srafit`, `logLik.srafit`,`vcov.srafit`, `coef.srafit`, `confint.srafit`, `plot.srafit` for the analysis of the results.

Examples

```
########### Generating a dummy dataset ################

m <- c(12,11,12,14,18,17,19,22,20,19)
v <- c(53,47,97,155,150,102,65,144,179,126)
s <- c(15,14,14,17,21,20,22,25,24,NA)
n <- c(100,80,120,60,100,90,110,80,60,100)

########## Making a sra data set ######################
data <- sraData(phen.mean=m, phen.var=v, phen.sel=s, N=n)

#################### Data Analysis ####################

cstvar <- sraCstvar(data)
drift <- sraDrift(data)
direpi <- sraDirepistasis(data)

# In case of convergence problems, better starting values can be provided:
direpi <- sraDirepistasis(data, start=list(mu0=10, logvarA0=log(20), logvarE0=NA),
fixed=list(logNe=log(50)))

plot(cstvar)

AIC(direpi)
```

---

4. Methods for srafit objects

Overloaded functions for "srafit" objects

Description

The functions return the output expected from the corresponding R functions, applied to an object of class `srafit`.

Usage

```
## S3 method for class 'srafit'
logLik(object, ...)
## S3 method for class 'srafit'
AIC(object, ...)
## S3 method for class 'srafit'
coef(object, ...)
## S3 method for class 'srafit'
confint(object, ...)
## S3 method for class 'srafit'
vcov(object, ...)
```

Arguments

`object`	An object of class `srafit`.
`...`	Any additional parameters to the corresponding functions.

Details

The confidence intervals are calculated from the estimate standard errors, and are thus different (less precise) from what would be calculated from the profile likelihood.

See Also

`sraAutoreg`, `sraCstvar` and other mechanistic models, `AIC`, `coef`, `logLik`, `confint`, `vcov`.

Examples

```
########### Generating a dummy dataset ################

m <- c(12,11,12,14,18,17,19,22,20,19)
v <- c(53,47,97,155,150,102,65,144,179,126)
s <- c(15,14,14,17,21,20,22,25,24,NA)
n <- c(100,80,120,60,100,90,110,80,60,100)

########## Making a sra data set ######################
data <- sraData(phen.mean=m, phen.var=v, phen.sel=s, N=n)

#################### Data Analysis ####################

cstvar <- sraCstvar(data)

AIC(cstvar)
logLik(cstvar)
coef(cstvar)
confint(cstvar)
vcov(cstvar)
```

---

5. plot.srafit *Plotting "srafit" objects*

Description

These functions plot in a nice way the content of objects of class `srafit`, the result of `sra` model fitting.

Usage

```
## S3 method for class 'srafit'
 plot(x, series = levels(x$data$rep), resid = FALSE,
variance = FALSE, ...)
sraPlotMean(srafit, series=levels(srafit$data$rep), legend=TRUE,
                xlim=NULL, ylim=NULL, xlab=NULL, ylab=NULL, pch=1, ...)
sraPlotMeanResid(srafit, series=levels(srafit$data$rep))
sraPlotVar (srafit, series=levels(srafit$data$rep), legend=TRUE,
             xlim=NULL, ylim=NULL, xlab=NULL, ylab=NULL, pch=1, ...)
sraPlotVarResid (srafit, series=levels(srafit$data$rep))
sraPlotlegend(labels, estimates, AIC=NULL, confint=NULL,
                         location="topleft")
sraFormatlegend(names, values, AIC=NULL, ...)
```

Arguments

x	An object of class srafit
srafit	An object of class srafit
series	The identifier (rep) of the time series that should be plotted. By default, all series are displayed.
resid	Whether or not the residuals (data - model expectation) should be displayed.
variance	If TRUE, the phenotypic variance is displayed. If FALSE (default), the phenotypic mean is displayed.
legend	If TRUE, a legend with the parameter estimates is displayed.
xlim	Same meaning as in plot. If NULL, a (hopefully) nice range is calculated.
ylim	Same meaning as in plot. If NULL, a (hopefully) nice range is calculated.
xlab	Same meaning as in plot. If NULL, a default label is provided.
ylab	Same meaning as in plot. If NULL, a default label is provided.
pch	Same meaning as in par.
labels	Formatted labels of the names of the estimates in the legend.
estimates	Values of the parameter estimates.
AIC	Value of the AIC. If NULL, the AIC is not added to the legend.
confint	Confidence intervals of the parameters. If NULL (default), they are not displayed in the legend.
location	Location of the legend, corresponding to x in legend.
names	Names of the parameters (as defined in the models).
values	Values of the parameter estimates (as defined in the models).
...	For plot.srafit, sraPlotMean, sraPlotVar: Additional parameters for the function plot. For sraFormatlegend: Additional parameters to format.

Details

The only function that should be used by the end user is plot.srafit.

Author(s)

Arnaud Le Rouzic

See Also

sraAutoreg, sraCstvar.

Examples

```
########### Generating a dummy dataset ################

m <- c(12,11,12,14,18,17,19,22,20,19)
v <- c(53,47,97,155,150,102,65,144,179,126)
s <- c(15,14,14,17,21,20,22,25,24,NA)
n <- c(100,80,120,60,100,90,110,80,60,100)

########## Making a sra data set ######################
data <- sraData(phen.mean=m, phen.var=v, phen.sel=s, N=n)

cstvar <- sraCstvar(data)


plot(cstvar)
plot(cstvar, xlim=c(3,9))
plot(cstvar, var=TRUE, ylab="This is a custom Y axis label")
plot(cstvar, resid=TRUE, legend=FALSE, main="Constant variance model fit")
```

---

6. sraData *Generates an object of class "sradata", necessary to run the models provided by the sra package.*

Description

The data necessary to analyse selection response time series are, for each generation, (i) the mean phenotype of the population, (ii) the phenotypic variance, (iii) the mean of the breeders, and (iv) the population size. These data have to be grouped into an object of class "sradata", which can be provided to the sra analysis functions such as sraAutoreg or sraCstvar.

Usage

```
sraData(phen.mean, phen.var, phen.sel, var.sel=NULL, N=NULL,
  gen=NULL, rep=NULL)
```

Arguments

phen.mean	The vector of phenotypic means.
phen.var	The vector of phenotypic variances.

phen.sel	The vector of the mean phenotype of breeders. Can be NA for the last generations.
var.sel	The vector of the phenotypic variances if breeders. If not provided (NULL), the vector is estimated assuming truncation selection and normal distribution of phenotypes.
N	The vector of population size for each generation (before selection).
gen	The generation numbers. Useful when several times series are provided.
rep	The repetition identification. Useful when several time series are provided.

Details

If not provided, the default value for N is 100. Incorrect values for N will affect the likelihood value and the maximum-likelihood estimates.

gen will be assumed to vary from 1 to the maximum number of generations by default.

If more than one time series are provided, it is dangerous not to specify gen or rep. rep can be any unique identifier. For instance, fro two times series of 3 generations each, gen can be c(1,2,3,1,2,3) and rep can be c("up","up","up","down","down","down").

Value

An object of class sradata.

See Also

sraAutoreg, sraCstvar, and other mechanistic models, sraAutoregTimeseries.

Examples

```
########### Generating a dummy dataset ################

m <- c(12,11,12,14,18,17,19,22,20,19)
v <- c(53,47,97,155,150,102,65,144,179,126)
s <- c(15,14,14,17,21,20,22,25,24,NA)
n <- c(100,80,120,60,100,90,110,80,60,100)

########## Making a sra data set ######################
data <- sraData(phen.mean=m, phen.var=v, phen.sel=s, N=n)
```

Index