Package 'PVAClone'

October 12, 2022

Type Package

Title Population Viability Analysis with Data Cloning

Version 0.1-6

Date 2016-03-11

Author Khurram Nadeem, Peter Solymos

Maintainer Peter Solymos <solymos@ualberta.ca>

Description Likelihood based population viability analysis in the presence of observation error and missing data.
The package can be used to fit, compare, predict, and forecast various growth model types using data cloning.

License GPL-2

Depends R (>= 2.15.0), dcmle (>= 0.3-0), dclone, stats4

Suggests parallel

Imports methods, coda

SystemRequirements JAGS (>= 3.0.0)

URL https://github.com/psolymos/PVAClone

BugReports https://github.com/psolymos/PVAClone/issues

LazyLoad yes

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2016-03-12 08:51:09

R topics documented:

`PVAClone-package` *Population Viability Analysis with Data Cloning*

Description

Likelihood based population viability analysis in the presence of observation error and missing data. The package can be used to fit, compare, predict, and forecast various growth model types using data cloning.

Details

The package implements data cloning based population viability analysis methodology developed by Nadeem and Lele (2012). This includes model estimation, model selection and forecasting of future population abundances for estimate the extinction risk of a population of interest.

`pva`: main function for model fitting.

`model.select`: main function for model model selection.

Growth models: `gompertz`, `ricker`, `bevertonholt`, `thetalogistic`, `thetalogistic_D`.

Author(s)

Khurram Nadeem, Peter Solymos

Maintainer: Peter Solymos <solymos@ualberta.ca>

References

Nadeem, K., Lele S. R., 2012. Likelihood based population viability analysis in the presence of observation error. Oikos 121, 1656–1664.

See Also

`pva`

Examples

```
## Not run:
## model selection for data with missing observations
data(songsparrow)
## model without observation error
m1 <- pva(songsparrow, gompertz("none"), 2, n.iter=1000)
## model with Poisson observation error
m2 <- pva(songsparrow, gompertz("poisson"), 2, n.iter=1000)
## model with Poisson observation error is strongly supported
model.select(m1, m2)

## End(Not run)
```

`fancyPVAmodel` *Print fancy model names in summaries*

Description

This function prints the fancy model names in summaries.

Usage

```
fancyPVAmodel(object, initial = "PVA object:\n", part = 1:2)
```

Arguments

`object`	A fitted 'pva' object.
`initial`	A fancy header for the fancy output.
`part`	Integer, 1 = model type is printed, 2 = data info is printed 1:2 = both are printed.

Value

Character with fancy model summary.

Author(s)

Khurram Nadeem and Peter Solymos

`generateLatent` *Generate latent variable*

Description

Generate latent variable of a hierarchical PVA model.

Usage

```
generateLatent(x, ...)
```

Arguments

`x`	A fitted PVA model object.
`...`	Arguments passed to `jags.fit`, such as `n.iter`, `n.chains`.

Details

It uses MLE from a fitted PVA model to generate values for the latent variables.

Value

A marix with `n.iter * n.chains` rows and as many columns as the length of the time series.

Author(s)

Khurram Nadeem and Peter Solymos

References

Ponciano, J. M. et al. 2009. Hierarchical models in ecology: confidence intervals, hypothesis testing, and model selection using data cloning. Ecology 90, 356–362.

See Also

`pva`

Examples

```
## Not run:
data(paurelia)
m <- pva(paurelia, gompertz("normal"), 5)
p <- generateLatent(m, n.chains=1, n.iter=1000)
summary(p)

## End(Not run)
```

growth-models *Growth models*

Description

Population growth model to be used in model fitting via pva.

Usage

```
gompertz(obs.error = "none", fixed)
ricker(obs.error = "none", fixed)
thetalogistic(obs.error = "none", fixed)
thetalogistic_D(obs.error = "none", fixed)
bevertonholt(obs.error = "none", fixed)
```

Arguments

obs.error	Character, describing the observation error. Can be "none", "poisson", or "normal".
fixed	Named numeric vector or list with fixed parameter names and values. Can be used for providing alternative prior specifications, see Details and Examples.

Details

These functions can be called in pva to fit the following growth models to a given population time series assuming both with and without observation error. When assuming the presence of observation error, either the Normal or the Poisson observation error model must be assumed within the state-space model formulation (Nadeem and Lele, 2012). The growth models are defined as follows.

Gompertz (gompertz):

$$x_t = a + x_{t-1} + bx_{t-1} + \epsilon_t$$

where x_t is log abundance at time t and $\epsilon_t \sim Normal(0, \sigma^2)$.

Ricker (ricker):

$$x_t = x_{t-1} + a + be^{x_{t-1}} + \epsilon_t$$

where x_t is log abundance at time t and $\epsilon_t \sim Normal(0, \sigma^2$.

Theta-Logistic (thetalogistic):

$$x_t = x_{t-1} + r[1 - (e^{x_{t-1}}/K)^t heta] + \epsilon_t$$

where x_t is log abundance at time t and $\epsilon_t \sim Normal(0, \sigma^2$.

Theta-Logistic with Demographic Variability (thetalogistic_D):

$$x_t = x_{t-1} + r[1 - (e^{x_{t-1}}/K)^t heta] + \epsilon_t$$

where x_t is log abundance at time t and $\epsilon_t \sim Normal(0, \sigma^2 + sigma.d^2$, where $sigma.d^2$ is the demographic variability. If $sigma.d^2$ is missing or fixed at zero, Theta-Logistic model is fitted instead.

Generilzed Beverton-Holt (`bevertonholt`):

$$x_t = x_{t-1} + r - log[1 + (e^{x_{t-1}}/K)^t heta] + \epsilon_t$$

where x_t is log abundance at time t and $\epsilon_t \sim Normal(0, \sigma^2$.

Observation error models are described in the help page of `pva`.

The argument `fixed` can be used to fit the model assuming *a priori* values of a subset of the parameters. For instance, fixing theta equal to one reduces Theta-Logistic and Gnerelized Beverton-Holt models to Logistic and Beverton-Holt models respectively. The number of parameters that should be fixed at most is $p - 1$, where p is the dimension of the full model. See examples below and in `pva` and `model.select`.

The `fixed` argument can be used to provide alternative prior specification using the BUGS language. In this case, values in `fixed` must be numeric. Use a list when real fixed values (numeric) and priors (character) are provided at the same time (see Examples). Alternative priors can be useful for testing insensitivity to priors, which is a diagnostic sign of data cloning convergence.

Value

An S4 class of 'pvamodel' (see `pvamodel-class`)

Author(s)

Khurram Nadeem and Peter Solymos

References

Nadeem, K., Lele S. R., 2012. Likelihood based population viability analysis in the presence of observation error. Oikos 121, 1656–1664.

See Also

`pvamodel-class`, `pva`

Examples

```
gompertz()
gompertz("poisson")
ricker("normal")
ricker("normal", fixed=c(a=5, sigma=0.5))
thetalogistic("none", fixed=c(theta=1))
bevertonholt("normal", fixed=c(theta=1))

## alternative priors
ricker("normal", fixed=c(a="a ~ dnorm(2, 1)"))@model
ricker("normal", fixed=list(a="a ~ dnorm(2, 1)", sigma=0.5))@model
```

Internals *Internal functions*

Description

Functions used internally.

Usage

```
ts_index(x, type=c("density", "expectation"))
```

Arguments

x	A vector of observations, possibly with missing values.
type	Character, type of index to calculate.

Details

ts_index calculates positional indices of elements of a vector that fulfill the following conditions when type = "density": (1) if there is only one observation present before the first NA, it is not selected, else, all the observations preceding to the first NA are selected; (2) if there is only one observation present after the last NA, it is not selected, else, all the observations following the last NA are selected; (3) if there is only one observation present between two consecutive NAs, it is not selected, else, all the observations falling between two consecutive NAs are selected. ts_index calculates positional indices of elements of a vector that immediately follow a missing (NA) value if type = "expectation". The reason for this is that these elements depend on missing data given a first order Markov process. As a result, these need different treatment in calculating log densities for model selection.

Value

ts_index returns an integer vector.

Author(s)

Peter Solymos and Khurram Nadeem

Examples

```
## ts_index
x <- 1:20
x[c(3,4, 6, 10, 13:15, 20)] <- NA
ts_index(x, "density")
ts_index(x, "expectation")
```

`model.select` *Model selection for 'pva' objects*

Description

Likelihood ratio calculation and model selection for (hierarchical) 'pva' objects.

`pva.llr` is the workhorse behind `model.select`. `pva.llr` can also be used for profile likelihood calculations if called iteratively (no wrapper presently).

Usage

```
pva.llr(null, alt, pred)
model.select(null, alt, B = 10^4)
## S3 method for class 'pvaModelSelect'
print(x, ...)
```

Arguments

`null`	A fitted 'pva' object representing the Null Hypothesis.
`alt`	A fitted 'pva' object representing the Alternative Hypothesis (usually broader model).
`B`	Number of replicates to be generated from the latent variables.
`pred`	A matrix of replicates from the latent variables, e.g. as returned by `generateLatent`. When there are no missing values and both the model objects for the null and alternative hypotheses are without observation error, `pred` can be missing. The log observations are used when `pred` is missing, and any user supplied values for `pred` are used if provided.
`x`	A model selection object to be printed.
`...`	Additional argument for print method.

Details

These functions implement Ponciano et. al.'s (2009) data cloning likelihood ratio algorithm (DCLR) to compute likelihood ratios for comparing hierarchical (random effect) models. In the population growth models context, these models are (1) with observation error population growth models, and/or (2) population growth models with missing observations.

The functions can also compute likelihood ratios when both of the population growth models are fixed effect models, e.g. without observation error Gompertz model Vs. without observation error Ricker model. See examples below and in `pva`.

Value

`pva.llr` returns a single numeric value, the log likelihood ratio of the two models (logLik0 - logLik1).

`model.select` returns a modified data frame with log likelihood ratio and various information criteria metrics (delta AIC, BIC, AICc).

The print method gives fancy model names and a human readable interpretation of the numbers.

Author(s)

Khurram Nadeem and Peter Solymos

References

Ponciano, J. M. et al. 2009. Hierarchical models in ecology: confidence intervals, hypothesis testing, and model selection using data cloning. Ecology 90, 356–362.

Nadeem, K., Lele S. R., 2012. Likelihood based population viability analysis in the presence of observation error. Oikos 121, 1656–1664.

See Also

`pva`

Examples

```
## Not run:
data(redstart)
m1 <- pva(redstart, gompertz("none"), 2, n.iter=1000)
m2 <- pva(redstart, gompertz("poisson"), 2, n.iter=1000)
m3 <- pva(redstart, gompertz("normal"), 2, n.iter=1000)
p <- generateLatent(m2, n.chains=1, n.iter=10000)
pva.llr(m1, m2, p)
model.select(m1, m2)
model.select(m1, m3)
model.select(m2, m3)

m1x <- pva(redstart, ricker("none"), 2, n.iter=1000)
m2x <- pva(redstart, ricker("poisson"), 2, n.iter=1000)
m3x <- pva(redstart, ricker("normal"), 2, n.iter=1000)

model.select(m1, m1x)
model.select(m2, m2x)
model.select(m3, m3x)

## missing data situation
data(paurelia)
m1z <- pva(paurelia, ricker("none"), 2, n.iter=1000)
m2z <- pva(paurelia, ricker("poisson"), 2, n.iter=1000)
m3z <- pva(paurelia, ricker("normal"), 2, n.iter=1000)

#model.select(m1z, m2z) # not yet implemented
#model.select(m1z, m3z) # not yet implemented
model.select(m2z, m3z)

## Analysis of song sparrow data in Nadeem and Lele (2012)
```

```
## use about 100 clones to get MLE's repoted in the paper.
data(songsparrow)
m1z <- pva(songsparrow,
    thetalogistic_D("normal",fixed=c(sigma2.d=0.66)),
    n.clones=5, n.adapt=3000, n.iter=1000)
m2z <- pva(songsparrow,
    thetalogistic_D("normal",fixed=c(theta=1, sigma2.d=0.66)),
    n.clones=5, n.adapt=3000,n.iter=1000)
m3z <- pva(songsparrow,
    thetalogistic_D("none",fixed=c(sigma2.d=0.66)),
    n.clones=5, n.adapt=3000,n.iter=1000)
m4z <- pva(songsparrow,
    thetalogistic_D("none",fixed=c(theta=1,sigma2.d=0.66)),
    n.clones=5, n.adapt=3000,n.iter=1000)

model.select(m2z, m1z)
model.select(m3z, m1z)
model.select(m4z, m1z)

## profile likelihood
m <- pva(redstart, gompertz("normal"), 5, n.iter=5000)
p <- generateLatent(m, n.chains=1, n.iter=10000)
m1 <- pva(redstart, gompertz("normal",
    fixed=c(sigma=0.4)), 5, n.iter=5000)
## etc for many sigma values
pva.llr(m1, m, p) # calculate log LR for each
## finally, fit smoother to points and plot

## End(Not run)
```

---

paurelia *Paramecium abundance time series*

Description

Paramecium aurelia abundance time series

Usage

```
data(paurelia)
```

Format

The format is: num [1:20] 2 NA 17 29 39 63 185 258 267 392 ...

Details

Paramecium aurelia abundance time series with a missing value.

Source

Gause (1934: Appendix I, Table 3)

References

Gause, G.F. (1934). The Struggle for Existence. Williams & Wilkins, Baltimore.

Examples

```
data(paurelia)
paurelia
plot(paurelia)
```

---

pva *Population Viability Analysis*

Description

Population Viability Analysis (PVA).

Usage

```
pva(x, model, n.clones, ...)
diagn_scale(object)
```

Arguments

`x`	Numeric, a time series. Values must be non-negative, missing values are allowed (but first and last observation must not be missing).
`model`	A 'pvamodel' object returned by a function, see Examples.
`n.clones`	Numeric, number of clones (possibly a vector).
`object`	A fitted 'pva' object returned by the `pva` function.
`...`	Arguments passed to underlying fitting functions, most notably `n.update`, `n.iter`, `n.chains`, `thin`, `cl`. See `dcmle`.

Details

The function implements the first step in PVA, i.e. to fit a given growth model to a population time series data (Nadeem and Lele, 2012). The function employs Lele et. al's (2007, 2010) data cloning (DC) algorithm for computing the maximum likelihood estimates of model parameters along with the corresponding standard errors. See Solymos (2010) for an R implementation of the DC algorithm. The growth models currently available in the package PVAClone are listed on the growthmodels page.

These models can also be fitted assuming the presence of observation error using the general state-space model formulation (Nadeem and Lele, 2012). Currently the Normal and Poisson observation error models are supported.

Normal observation error model: $y_t \sim Normal(x_t, \tau^2)$, where y_t is the estimated abundance on the log scale at time t.

Poisson observation error model: $O_t \sim Poisson(e^{x_t})$, where O_t is the estimated abundance at time t.

In addition, missing observations can be accommodated in both with or without observation error cases.

Value

An object of class 'pva', see pva-class.

Author(s)

Khurram Nadeem and Peter Solymos

References

Lele, S.R., B. Dennis and F. Lutscher, 2007. Data cloning: easy maximum likelihood estimation for complex ecological models using Bayesian Markov chain Monte Carlo methods. Ecology Letters 10, 551–563.

Lele, S. R., K. Nadeem and B. Schmuland, 2010. Estimability and likelihood inference for generalized linear mixed models using data cloning. Journal of the American Statistical Association 105, 1617–1625.

Nadeem, K., Lele S. R., 2012. Likelihood based population viability analysis in the presence of observation error. Oikos 121, 1656–1664.

Solymos, P., 2010. dclone: Data Cloning in R. The R Journal 2(2), 29–37. URL: http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Solymos.pdf

See Also

Model selection: model.select

Growth models: growthmodels

Class definitions: pva-class, pvamodel-class

Examples

```
## Not run:
data(redstart)
data(paurelia)
data(songsparrow)

## Gompertz
m1 <- pva(redstart, "gompertz", c(5,10))
m2 <- pva(redstart, gompertz("poisson"), c(5,10))
m3 <- pva(redstart, gompertz("normal"), c(5,10))
m1na <- pva(paurelia, "gompertz", c(5,10))
m2na <- pva(paurelia, gompertz("poisson"), c(5,10))
m3na <- pva(paurelia, gompertz("normal"), c(5,10))
```

```
m1x <- pva(redstart, gompertz("normal"), 5)
m2x <- pva(redstart, gompertz("normal", fixed=c(tau=0.1)), 5)

## Ricker
m1 <- pva(redstart, "ricker", c(5,10))
m2 <- pva(redstart, ricker("poisson"), c(5,10))
m3 <- pva(redstart, ricker("normal"), c(5,10))
m1na <- pva(paurelia, "ricker", c(5,10))
m2na <- pva(paurelia, ricker("poisson"), c(5,10))
m3na <- pva(paurelia, ricker("normal"), c(5,10))
m1x <- pva(redstart, ricker("normal"), 5)
m2x <- pva(redstart, ricker("normal", fixed=c(tau=0.1)), 5)

## Theta-Logistic
m1 <- pva(songsparrow, "thetalogistic", c(5,10))
m2 <- pva(songsparrow, thetalogistic("poisson"), c(2,5))
m3 <- pva(songsparrow, thetalogistic("normal"), c(2,5))
m1x <- pva(songsparrow,
    thetalogistic_D("normal", fixed=c(sigma2.d=0.66)), 5)
m2x <- pva(songsparrow,
    thetalogistic_D("none", fixed=c(theta=1, sigma2.d=0.66)), 10)

m2x
summary(m2x)
coef(m2x)
vcov(m2x)
confint(m2x)
plot(m2x)
plot(diagn_scale(m2x))

## End(Not run)
```

---

pva-class *Class* "pva"

Description

Model class for fitted PVA objects.

Objects from the Class

Objects can be created by calls of the form `new("pva", ...)`.

Slots

`observations`**:** Object of class `"numeric"`, vector of observations (must be non-negative but not necessarily integer), possibly with missing values (`NA`).

`model`**:** Object of class `"pvamodel"`, internal representation of the growth model and observation error structure.

`summary`: Object of class `"matrix"`, asymptotic Wald-type summary on the 'original' scale of the parameters (i.e. not on the scale used for model fitting and diagnostics).

`dcdata`: Object of class `"dcFit"`, internal representation of the data and JAGS model.

`call`: Object of class `"language"`, the call.

`coef`: Object of class `"numeric"`, point estimates of the model parameters.

`fullcoef`: Object of class `"numeric"`, vector possibly containing fixed parameter values.

`vcov`: Object of class `"matrix"`, variance covariance matrix of the estimates.

`details`: Object of class `"dcCodaMCMC"`, MCMC output from data cloning.

`nobs`: Object of class `"integer"`, number of observations (excluding missing values).

`method`: Object of class `"character"`, optimization method (data cloning).

Extends

Class `"dcmle"`, directly.

Methods

coef `signature(object = "pva")`

confint `signature(object = "pva")`

show `signature(object = "pva")`

vcov `signature(object = "pva")`

Author(s)

Khurram Nadeem and Peter Solymos

See Also

`pva`

Examples

```
showClass("pva")
```

`pva-methods` *Methods for 'pva' objects*

Description

`coef`, `vcov`, `confint`, and `summary` methods for 'pva' objects.

Methods

`signature(object = "pva")` Methods for S4 objects of class 'pva'.

pvamodel-class *Class* "pvamodel"

Description

S4 class for predefined PVA models.

Objects from the Class

Objects can be created by calls of the form `new("pvamodel", ...)`.

Slots

`growth.model`: Object of class `"character"`, name of growth model.

`obs.error`: Object of class `"character"`, name of observation error type (`"none"`, `"poisson"`, `"normal"`).

`model`: Object of class `"dcModel"`, BUGS model for estimation.

`genmodel`: Object of class `"dcModel"`, BUGS model for prediction.

`p`: Object of class `"integer"`, number of parameters in model (including fixed parameters!).

`support`: Object of class `"matrix"`, range of support for parameters (true parameter scale).

`params`: Object of class `"character"`, parameter names (diagnostic scale).

`varnames`: Object of class `"character"`, parameter names (true parameter scale).

`fixed`: Object of class `"nClones"`, named vector of fixed parameters.

`fancy`: Object of class `"character"`, fancy model description.

`transf`: Object of class `"function"`, function to transform from true parameter scale to diagnostic scale (takes care of fixed value which are not part of the MCMC output.

`backtransf`: Object of class `"function"`, function to transform from diagnostic scale to true parameter scale (takes care of fixed value which are not part of the MCMC output.

`logdensity`: Object of class `"function"`, function to calculate log density (used in model selection).

`neffective`: Object of class `"function"`, function to calculate effective sample size from the number of observations.

Methods

No methods defined with class "pvamodel" in the signature.

Author(s)

Khurram Nadeem and Peter Solymos

See Also

`pva`

Examples

```
showClass("pvamodel")
```

`redstart` *Abundance time series of American Redstart*

Description

Counts for American Redstart (*Setophaga ruticilla*) at a survey location in the North American Breeding Bird Survey (BBS; Robbins et al. 1986, Peterjohn 1994). BBS record number 0214332808636 observed from 1966 to 1995.

Usage

```
data(redstart)
```

Format

The format is: num [1:30] 18 10 9 14 17 14 5 10 9 5 ...

Details

redstart abundance time series

Source

Data reported in B. Dennis, J. M. Ponciano, S. R. Lele, M. L. Taper, and D. F. Staples (unpublished manuscript, see Lele 2006).

References

Lele, S.R. 2006. Sampling variability and estimates of density dependence: a composite-likelihood approach. Ecology 87, 189–202.

Peterjohn, B.G. 1994. The North American Breeding Bird Survey. Birding 26, 386–398.

Robbins, C.S., D. Bystrak, and P.H. Geissler. 1986. The breeding bird survey: its first fifteen years, 1965–1979. U.S. Fish and Wildl. Serv. Resource Publ. 157. Washington, D.C. 196 pp.

Examples

```
data(redstart)
redstart
plot(redstart)
```

songsparrow *Abundance time series of Song Sparrow*

Description

Counts for Song Sparrow (*Melospiza melodia*) on Mandarte Island, British Columbia, Canada from 1975–1998 reported in Saether et al. (2000).

Usage

```
data(songsparrow)
```

Format

The format is: num [1:24] 35 31 45 48 66 9 19 26 54 ...

Details

Song Sparrow abundance time series.

Source

Peter Arcese kindly provided the Song Sparrow population counts data.

References

Nadeem, K., Lele S. R., 2012. Likelihood based population viability analysis in the presence of observation error. Oikos 121, 1656–1664.

Saether, B. et al. 2000. Estimating the time to extinction in an island population of song sparrows. Proc. R. Soc. B 267, 621–626.

Examples

```
data(songsparrow)
songsparrow
plot(songsparrow)
```

Index