Package 'evoTS'

January 17, 2023

Title Analyses of Evolutionary Time-Series

Version 1.0.0

Description Facilitates univariate and multivariate analysis of evolutionary sequences of phenotypic change. The package extends the modeling framework available in the 'paleoTS' package. Please see <https://klvoje.github.io/evoTS/index.html> for information about the package and the implemented models.

License GPL (>= 2)

Encoding UTF-8

Imports paleoTS (>= 0.4-4), mvtnorm, plotly, pracma, MASS, stats, utils

Maintainer Kjetil Lysne Voje <k.l.voje@nhm.uio.no>

LazyData true

RoxygenNote 7.2.1

Suggests rmarkdown, knitr

Depends R (>= 3.5.0)

URL https://klvoje.github.io/evoTS/index.html

BugReports https://github.com/klvoje/evoTS/issues

NeedsCompilation no

Author Kjetil Lysne Voje [aut, cre]

Repository CRAN

Date/Publication 2023-01-17 17:40:05 UTC

R topics documented:

`as.evoTS.multi.BW.acceldecel.fit` *Class for fit to evolutionary sequence (time-series) models*

Description

A function that combines useful information summarizing model fit.

Usage

```
as.evoTS.multi.BW.acceldecel.fit(
  converge,
  modelName,
  logL,
  ancestral.values,
  SE.anc,
  r,
  SE.r,
  R,
  SE.R,
  method,
  K,
  n,
  iter
)
```

Arguments

`converge`	info on model convergence
`modelName`	description of the model.
`logL`	log-likelihood of model
`ancestral.values`	maximum-likelihood estimates of the ancestral trait values
`SE.anc`	standard errors of the estimated ancestral states
`r`	maximum-likelihood estimates of the r parameter
`SE.r`	standard error of the r parameter
`R`	maximum-likelihood estimates of the parameters in the R matrix
`SE.R`	standard errors of the parameters in the R matrix
`method`	the parameterization used: Joint
`K`	number of parameters in the model
`n`	sample size
`iter`	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.

Details

This function is used by the model-fitting routines for the Unbiased Random Walk models with an accelerated and decelerated rate of evoluton to create standardized output

Note

This function is not likely to be called directly by the user.

Author(s)

Kjetil Lysne Voje

`as.evoTS.multi.BW.fit` *Class for fit to evolutionary sequence (time-series) models*

Description

A function that combines useful information summarizing model fit.

Usage

```
as.evoTS.multi.BW.fit(
  converge,
  modelName,
  logL,
  ancestral.values,
  SE.anc,
  R,
  SE.R,
  method,
  K,
  n,
  iter
)
```

Arguments

`converge`	info on model convergence
`modelName`	description of the model
`logL`	log-likelihood of model
`ancestral.values`	maximum-likelihood estimates of the ancestral trait values
`SE.anc`	standard errors of the estimated ancestral states
`R`	maximum-likelihood estimates of the parameters in the R matrix
`SE.R`	standard errors of the parameters in the R matrix

method	the parameterization used: Joint
K	number of parameters in the model
n	sample size
iter	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.

Details

This function is used by the model-fitting routines for the multivariate Unbiased Random Walk models to create standardized output

Note

This function is not likely to be called directly by the user.

Author(s)

Kjetil Lysne Voje

---

as.evoTS.multi.OU.fit *Class for fit to evolutionary sequence (time-series) models*

Description

A function that combines useful information summarizing model fit.

Usage

```
as.evoTS.multi.OU.fit(
  converge,
  logL,
  ancestral.values,
  SE.anc,
  optima,
  SE.optima,
  A,
  SE.A,
  half.life,
  R,
  SE.R,
  method,
  K,
  n,
  iter
)
```

Arguments

converge	info on model convergence
logL	log-likelihood of model
ancestral.values	maximum-likelihood estimates of the ancestral trait values
SE.anc	standard errors of the estimated ancestral states
optima	maximum-likelihood estimates of the optima
SE.optima	standard errors of the estimated optimal trait values
A	maximum-likelihood estimates of the parameters in the A matrix
SE.A	standard errors of the estimated A matrix
half.life	the calculated half-life of the evolutionary process
R	maximum-likelihood estimates of the parameters in the R matrix
SE.R	standard errors of the parameters in the R matrix
method	the parameterization used: Joint
K	number of parameters in the model
n	sample size
iter	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.

Details

This function is used by the model-fitting routines for the multivariate Ornstein-Uhlenbeck models to create standardized output

Note

This function is not likely to be called directly by the user.

Author(s)

Kjetil Lysne Voje

---

as.evoTSfit.OUBM *Class for fit to evolutionary sequence (time-series) models*

Description

A function that combines useful information summarizing model fit.

Usage

```
as.evoTSfit.OUBM(logL, parameters, modelName, method, K, n, iter, se)
```

Arguments

logL	log-likelihood of model
parameters	maximum-likelihood estimates of the parameters
modelName	description of the model
method	the parameterization used: Joint
K	number of parameters in the model
n	sample size
iter	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
se	standard errors of parameter estimates

Details

This function is used by the model-fitting routines for the univariate Ornstein-Uhlenbeck model where the optimum evolves as an Unbiased Random Walk to create standardized output

Note

This function is not likely to be called directly by the user.

Author(s)

Kjetil Lysne Voje

---

diameter_S.yellowstonensis

Evolutionary sequence (time-series) of phenotypic change in diameter in the lineage Stephanodiscus yellowstonensis

---

Description

Phenotypic data (diameter) from a centric diatom lineage Stephanodiscus yellowstonensis. The time series spans about 14 000 years. The data set contains data on valve diameter (measured in micrometres). The data consists of an object of class paleoTS (diameter_S.yellowstonensis). Objects of class paleoTS can be analyzed in evoTS. The object (trait data set) contains a vector of sample means (mm), sample variances (vv), sample sizes (nn) and sample ages (tt). The oldest sample is listed first. The data spans an interval of 13728 years.

Usage

```
data(diameter_S.yellowstonensis)
```

Format

An object of class "paleoTS".

References

Theriot et al. 2006. Late Quaternary rapid morphological evolution of an endemic diatom in Yellowstone Lake, Wyoming. Paleobiology 32:38-54

Examples

```
ln.diameter<-paleoTS::ln.paleoTS(diameter_S.yellowstonensis)
ln.diameter$tt<-ln.diameter$tt/(max(ln.diameter$tt))
opt.joint.decel(ln.diameter)
```

`fit.all.univariate` *Fit all univariate models to an evolutionary sequence (time-series).*

Description

Wrapper function to find maximum likelihood solutions for all univariate models (excluding models with mode shifts) to an evolutionary sequence (time-series).

Usage

```
fit.all.univariate(y, pool = TRUE)
```

Arguments

`y`	an univariate paleoTS object.
`pool`	indicating whether to pool variances across samples

Value

The function returns a list of all investigated models and their highest log-likelihood (and their corresponding AICc and AICc weight).

Author(s)

Kjetil Lysne Voje

References

Hunt, G. 2006. Fitting and comparing models of phyletic evolution: random walks and beyond. *Paleobiology* 32:578–601

Hunt, G., Bell, M. A. & Travis, M. P. Evolution towards a new adaptive optimum: Phenotypic evolution in a fossil stickleback lineage. *Evolution* 62:700–710 (2008)

Examples

```
## ##Generate a paleoTS object.
x <- paleoTS::sim.GRW(30)

## Fit univariate models to the data.
fit.all.univariate(x, pool = TRUE)
```

`fit.mode.shift` *Fit two models to two separate segments to an evolutionary sequence (time-series).*

Description

Wrapper function to find maximum likelihood solutions to two models to an evolutionary sequence.

Usage

```
fit.mode.shift(
  y,
  model1 = c("Stasis", "URW", "GRW", "OU"),
  model2 = c("Stasis", "URW", "GRW", "OU"),
  fit.all = FALSE,
  minb = 7,
  shift.point = NULL,
  pool = TRUE,
  silent = FALSE,
  hess = FALSE
)
```

Arguments

`y`	an univariate evoTS object.
`model1`	the model fitted to the first segment. Options are Stasis, URW, GRW, OU.
`model2`	the model fitted to the second segment. Options are Stasis, URW, GRW, OU.
`fit.all`	logical indicating whether to fit all pairwise combinations of the four models to the evolutionary sequence (time-series).
`minb`	the minimum number of samples within a segment to consider
`shift.point`	The sample that split the time-series into two segments. The samples are passed to the argument as a vector. Default is NULL, which means all possible shift points will be assessed constrained by how minb is defined.
`pool`	logical indicating whether to pool variances across samples
`silent`	if TRUE, less information is printed to the screen as the model is fit
`hess`	logical, indicating whether to calculate standard errors from the Hessian matrix. #'

Value

the function returns a list of all investigated models and their highest log-likelihood (and their corresponding AICc and AICc weight).

`logL`	the log-likelihood of the optimal solution
`AICc`	AIC with a correction for small sample sizes
`parameters`	parameter estimates
`modelName`	abbreviated model name
`method`	Joint consideration of all samples
`K`	number of parameters in the model
`n`	the number of observations/samples
`all.logl`	log-likelihoods for all tested partitions of the series into segments. Will return a single value if shift points have been given
`GG`	matrix of indices of initial samples of each tested segment configuration; each column of GG corresponds to the elements of all.logl

In addition, if fit.all=TRUE the function also returns a list of all investigated models and their highest log-likelihood (and their corresponding AICc and AICc weight).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Hunt, G. 2006. Fitting and comparing models of phyletic evolution: random walks and beyond. *Paleobiology* 32:578–601

Hunt, G., Bell, M. A. & Travis, M. P. Evolution towards a new adaptive optimum: Phenotypic evolution in a fossil stickleback lineage. *Evolution* 62:700–710 (2008)

Examples

```
##Generate a paleoTS object.
x <- paleoTS::sim.GRW(30)

## Fit a mode-shift model without defining a shift point (the example may take > 5 seconds to run)
fit.mode.shift(x, model1="URW", model2="Stasis")
```

`fit.multivariate.OU` *Fit predefined multivariate Ornstein-Uhlenbeck models to multivariate evolutionary sequence (time-series) data.*

Description

Function to find maximum likelihood solutions to a large suite of predefined multivariate Ornstein-Uhlenbeck model fitted to multivariate evolutionary sequence (time-series) data.

Usage

```
fit.multivariate.OU(
  yy,
  A.matrix = "diag",
  R.matrix = "symmetric",
  method = "Nelder-Mead",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL,
  user.init.diag.A = NULL,
  user.init.diag.R = NULL,
  user.init.off.diag.A = NULL,
  user.init.off.diag.R = NULL,
  user.init.theta = NULL,
  user.init.anc = NULL
)
```

Arguments

`yy`	a multivariate evoTS object.
`A.matrix`	the pull matrix. The options are "diag", "upper.tri", "lower.tri", and "full". Default is "diag". See details (or vignette) for more info on what the different options mean.
`R.matrix`	the drift matrix. The options are "diag" and "symmetric".
`method`	optimization method, passed to function optim. Default is "Nelder-Mead".
`hess`	logical, indicating whether to calculate standard errors from the Hessian matrix.
`pool`	indicating whether to pool variances across samples
`trace`	logical, indicating whether information on the progress of the optimization is printed.
`iterations`	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
`iter.sd`	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

`user.init.diag.A`
: starting values for the optimization routine of the diagonal elements of the A matrix. Default is NULL.

`user.init.diag.R`
: starting values for the optimization routine of the diagonal elements of the R matrix. Default is NULL.

`user.init.off.diag.A`
: starting values for the optimization routine of the off-diagonal elements of the A matrix. Default is NULL.

`user.init.off.diag.R`
: starting values for the optimization routine of the off-diagonal elements of the R matrix. Default is NULL.

`user.init.theta`
: starting values for the optimization routine of the optima. Default is NULL.

`user.init.anc`
: starting values for the optimization routine of the ancestral values. Default is NULL.

Details

A detailed explanation of the predefined models that can be fitted using the function is given in the online vignette (https://klvoje.github.io/evoTS/index.html), but a short summary is provided here. Note that this function provides the user with fixed options for how to parameterize the A and R matrices. For full flexibility, the user is allowed to customize the parameterization of the A and R matrix in the 'fit.multivariate.OU.user.defined' function. The type of trait dynamics is defined based on how the pull matrix (A) and drift matrix (R) are defined. The function allows testing four broad categories of models: 1 Independent evolution (A.matrix ="diag", R.matrix = "diag"); 2 Independent adaptation (A.matrix ="diag", R.matrix = "symmetric"); 3 Non-independent adaptation (A.matrix = "upper.tri"/"lower.tri"/full", R.matrix = "diagonal"); 4 Non-independent evolution (A.matrix = "upper.tri"/"lower.tri"/"full", R.matrix = "symmetric"). Setting the A.matrix to "diagonal" means the traits do not affect each others optimum (A matrix). A "diagonal" R matrix means the stochastic changes in the traits are assumed to be uncorrelated. A "symmetric" R matrix means the stochastic changes in the traits are assumed to be correlated, i.e. that they are non-independent. A "full" parameterization of A estimates the effect of each trait on the optima on the other traits. The "upper.tri" option parameterize the model in such a way that the first layer (first trait in the data set) adapts non-independently because its optimum is affected by all other traits included in the data set, while the bottom layer (the last trait in the data set) adapts independently (as an Ornstein Uhlenbeck process). Layers in between the upper- and lower layer (not the first or last trait in the data set (if there are more than two traits in the data set)) evolve non-independently as their optimum is affected by all layers/traits below themselves. The option "lower.tri" defines the causality the opposite way compared to "upper.tri". It is also possible to implement a model where the bottom layer (last trait in the data set) evolve as an Unbiased random walk (akin to a Brownian motion) which affects the optima for all other traits in the data set (i.e. all layers except the bottom layer). This model can be fitted by defining A.matrix ="OUBM", which will override how the R matrix is defined.

The function searches - using an optimization routine - for the maximum-likelihood solution for the chosen multivariate Ornstein-Uhlenbeck model. The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. Note that the the default method (Nelder-Mead) seems to work for most evolutionary sequences. The method L-BFGS-B allows box-constraints on some parameters (e.g. non-negative variance

parameters) and is faster than Nelder-Mead, but is less stable than the default method (Nelder-Mead).

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Ornstein-Uhlenbeck model (from the paleoTS package) fitted to each time-series separately.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

There is no guarantee that the likelihood can be computed with the initial parameters provided by the function. The starting values for fitting the multivariate OU model are based on maximum likelihood parameter estimates for the univariate OU model fitted to each trait separately, which seems to provide sensible (and working) initial parameter estimates for almost all tested data sets. However, the provided initial parameters may fail depending on the nature of the data. If an error message is returned saying "function cannot be evaluated at initial parameters", the user can try to start the optimization procedure from other initial parameter values using "user.init.diag.A", "user.init.diag.R", "user.init.off.diag.A", "user.init.off.diag.R", "user.init.theta", and "user.init.anc." It is usually the initial guess of the off-diagonal elements of the A and R matrices that prevents the optimization routine to work. It is therefore recommended to only try to change these initial values before experimenting with different starting values for the diagonal of the A and R matrices.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, optima, A, and R). The half-life is also provided, which is the The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Reitan, T., Schweder, T. & Henderiks, J. Phenotypic evolution studied by layered stochastic differential equations. *Ann Appl Statistics* 6, 1531–1551 (2012).

Bartoszek, K., Pienaar, J., Mostad, P., Andersson, S. & Hansen, T. F. A phylogenetic comparative method for studying multivariate adaptation. *J Theor Biol* 314, 204–215 (2012).

Clavel, J., Escarguel, G. & Merceron, G. mvmorph: an r package for fitting multivariate evolutionary models to morphometric data. *Methods Ecol Evol 6*, 1311–1319 (2015).

Examples

```
## Generate a evoTS object by simulating a multivariate dataset
x <- sim.multi.OU(15)


##Fit a multivariate Ornstein-Uhlenbeck model to the data. This example will run for a long time.
fit.multivariate.OU(x, A.matrix="diag", R.matrix="symmetric")
```

---

fit.multivariate.OU.user.defined

Fit user-defined multivariate Ornstein-Uhlenbeck models to multivariate evolutionary sequence (time-series) data.

Description

Function to find maximum likelihood solutions to a multivariate Ornstein-Uhlenbeck model fitted using user-defined A and R matrices.

Usage

```
fit.multivariate.OU.user.defined(
  yy,
  A.user = NULL,
  R.user = NULL,
  method = "Nelder-Mead",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL,
  user.init.diag.A = NULL,
  user.init.upper.diag.A = NULL,
  user.init.lower.diag.A = NULL,
  user.init.diag.R = NULL,
  user.init.off.diag.R = NULL,
  user.init.theta = NULL,
  user.init.anc = NULL
)
```

Arguments

yy	a multivariate evoTS object.
A.user	the pull matrix. A user-defined A matrix.
R.user	the drift matrix. A user-defined R matrix.

`method` optimization method, passed to function optim. Default is "Nelder-Mead".

`hess` logical, indicating whether to calculate standard errors from the Hessian matrix.

`pool` indicating whether to pool variances across samples

`trace` logical, indicating whether information on the progress of the optimization is printed.

`iterations` the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.

`iter.sd` defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

`user.init.diag.A`
starting values for the optimization routine of the diagonal elements of the A matrix. Default is NULL.

`user.init.upper.diag.A`
starting values for the optimization routine of the upper diagonal elements of the A matrix. Default is NULL.

`user.init.lower.diag.A`
starting values for the optimization routine of the lower diagonal elements of the A matrix. Default is NULL.

`user.init.diag.R`
starting values for the optimization routine of the diagonal elements of the R matrix. Default is NULL.

`user.init.off.diag.R`
starting values for the optimization routine of the off-diagonal elements of the R matrix. Default is NULL.

`user.init.theta`
starting values for the optimization routine of the optima. Default is NULL.

`user.init.anc` starting values for the optimization routine of the ancestral values. Default is NULL.

Details

This function provides users the flexibility to define their own A and R matrices. The possibility to define any A matrices enable detailed investigation of specific evolutionary hypotheses. The parameters to be estimated in the matrices are indicated by the value 1. All other entries in the matrix must be 0.

The function searches - using an optimization routine - for the maximum-likelihood solution for the chosen multivariate Ornstein-Uhlenbeck model. The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. Note that the the default method (Nelder-Mead) seems to work for most evolutionary sequences. The method L-BFGS-B allows box-constraints on some parameters (e.g. non-negative variance parameters) and is faster than Nelder-Mead, but is less stable than the default method (Nelder-Mead).

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Ornstein-Uhlenbeck model (from the paleoTS package) fitted to each time-series separately.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more

than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

There is no guarantee that the likelihood can be computed with the initial parameters provided by the function. The starting values for fitting the multivariate OU model are based on maximum likelihood parameter estimates for the univariate OU model fitted to each trait separately, which seems to provide sensible (and working) initial parameter estimates for almost all tested data sets. However, the provided initial parameters may fail depending on the nature of the data. If an error message is returned saying "function cannot be evaluated at initial parameters", the user can try to start the optimization procedure from other initial parameter values using "user.init.diag.A", "user.init.upper.diag.A", "user.init.lower.diag.A", "user.init.diag.R", "user.init.off.diag.R", "user.init.theta", and "user.init.anc." It is usually the initial guess of the off-diagonal elements of the A and R matrices that prevents the optimization routine to work. It is therefore recommended to only try to change these initial values before experimenting with different starting values for the diagonal of the A and R matrices.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, optima, A, and R). The half-life is also provided, which is the The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Reitan, T., Schweder, T. & Henderiks, J. Phenotypic evolution studied by layered stochastic differential equations. *Ann Appl Statistics* 6, 1531–1551 (2012).

Bartoszek, K., Pienaar, J., Mostad, P., Andersson, S. & Hansen, T. F. A phylogenetic comparative method for studying multivariate adaptation. *J Theor Biol* 314, 204–215 (2012).

Clavel, J., Escarguel, G. & Merceron, G. mvmorph: an r package for fitting multivariate evolutionary models to morphometric data. *Methods Ecol Evol 6*, 1311–1319 (2015).

Examples

```
## Generate a evoTS object by simulating a multivariate dataset
x <- sim.multi.OU(15)
```

```
## Define an A matrix that is lower diagonal.
A <- matrix(c(1,0,1,1), nrow=2, byrow=TRUE)

## Define a diagonal R matrix.
R <- matrix(c(1,0,0,1), nrow=2, byrow=TRUE)


## Fit the multivariate Ornstein-Uhlenbeck model to the data. This example will run for a long time.
fit.multivariate.OU.user.defined(x, A.user=A, R.user=R, trace=TRUE)
```

`fit.multivariate.URW` *Fit multivariate Unbiased Random Walk models to multivariate evolutionary sequence (time-series) data.*

Description

Function to find maximum likelihood solutions to a multivariate Unbiased Random Walk model.

Usage

```
fit.multivariate.URW(
  yy,
  R = "symmetric",
  r = "fixed",
  method = "L-BFGS-B",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

yy	a multivariate evoTS object.
R	the drift matrix. The options are "diagonal" and "symmetric"
r	parameter describing the exponential increase/decrease in rate across time. The options are "fixed", "accel" and "decel".
method	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.
pool	indicating whether to pool variances across samples
trace	logical, indicating whether information on the progress of the optimization is printed.

`iterations`	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
`iter.sd`	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function allows the users to test six variants of multivariate Unbiased Random Walk models. There are two options for the structure of the R matrix. A "diagonal" R matrix means the stochastic changes in the traits are assumed to be uncorrelated. A "symmetric" R matrix means the stochastic changes in the traits are assumed to be correlated, i.e. that they are non-independent.

There are three options for the 'r' parameter. The "fixed" option means there is no change in the rate of change across time (r = 0). Setting r to "fixed" therefore fits a regular multivariate Unbiased Random Walk. The "decel" and "accel" options make the rate of change (the R matrix) decay (r < 0) and increase (r > 0) exponentially through time, respectively.

The function searches - using an optimization routine - for the maximum-likelihood solution for the chosen multivariate Unbiased Random Walk model. The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the auto-correlation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Revell, L. J. & Harmon, L. Testing quantitative genetic hypotheses about the evolutionary rate matrix for continuous characters. *Evolutionary Ecology Research* 10, 311–331 (2008).

Voje, K. L. Testing eco-evolutionary predictions using fossil data: Phyletic evolution following ecological opportunity. *Evolution* 74, 188–200 (2020).

Examples

```
## Generate an evoTS object by simulating a multivariate dataset
x <- sim.multi.URW(30)

## Fit a multivariate Unbiased Random Walk model to the data, allowing for correlated changes.
fit.multivariate.URW(x, R = "symmetric", r = "fixed")
```

---

fit.multivariate.URW.shift

Fit separate multivariate Unbiased Random Walk models to two different segments of a multivariate evolutionary sequence (time-series).

Description

Function to find maximum likelihood solutions for multivariate Unbiased Random Walk models fitted to two different segments of a multivariate evolutionary sequence (time-series).

Usage

```
fit.multivariate.URW.shift(
  yy,
  minb = 10,
  hess = FALSE,
  pool = TRUE,
  shift.point = NULL,
  method = "L-BFGS-B",
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

yy	a multivariate evoTS object.
minb	the minimum number of samples within a segment to consider.
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.
pool	indicating whether to pool variances across samples

shift.point	the sample in the time series that represents the first sample in the second segment.
method	optimization method, passed to function optim. Default is "L-BFGS-B".
trace	logical, indicating whether information on the progress of the optimization is printed.
iterations	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
iter.sd	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model ti two non-overlapping segments in the time series.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Revell, L. J. & Harmon, L. Testing quantitative genetic hypotheses about the evolutionary rate matrix for continuous characters. *Evolutionary Ecology Research* 10, 311–331 (2008).

Examples

```
## Generate an evoTS object by simulating a multivariate dataset
x <- sim.multi.URW(60)

## Fit two multivariate Unbiased Random Walk models to separate parts of the time-series.
fit.multivariate.URW.shift(x, shift.point = 31)
```

`logL.joint.accel.decel.single.R`
Log-likelihoods for evolutionary models

Description

Returns log-likelihood for a multivariate Unbiased Random Walk model with accelerating or decelerating rates of evolution through time.

Usage

```
logL.joint.accel.decel.single.R(init.par, y, m, n, anc.values, yy)
```

Arguments

`init.par`	initial (starting) parameters values
`y`	vector containing all trait values from all traits
`m`	number of traits
`n`	number of populations
`anc.values`	initial values for the ancestral trait values
`yy`	a multivariate evoTS object

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

`logL.joint.accel.decel.single.R.zero.corr` *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for a multivariate Unbiased Random Walk model with uncorrelated changes and with accelerating or decelerating rates of evolution through time.

Usage

```
logL.joint.accel.decel.single.R.zero.corr(init.par, y, m, n, anc.values, yy)
```

Arguments

`init.par`	initial (starting) parameters values
`y`	vector containing all trait values from all traits
`m`	number of traits
`n`	number of populations
`anc.values`	initial values for the ancestral trait values
`yy`	a multivariate evoTS object

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

logL.joint.accel_decel *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for an Unbiased Random Walk with an accelerating or decelerating rate of change through time.

Usage

```
logL.joint.accel_decel(p, y)
```

Arguments

p	parameters of the model to be optimized
y	a paleoTS object

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

logL.joint.multi.OUOU *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for a multivariate Ornstein-Uhlenbeck model.

Usage

```
logL.joint.multi.OUOU(init.par, yy, A.matrix, R.matrix)
```

Arguments

init.par	initial (starting) parameters values
yy	a multivariate evoTS object
A.matrix	the pull matrix.
R.matrix	the drift matrix..

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

---

logL.joint.multi.OUOU.user

Log-likelihoods for evolutionary models

Description

Returns log-likelihood for a multivariate Ornstein-Uhlenbeck model with used defined A and R matrices..

Usage

```
logL.joint.multi.OUOU.user(
  init.par,
  yy,
  A.user,
  R.user,
  locations.A,
  location.diag.A,
  location.upper.tri.A,
  location.lower.tri.A,
  locations.R,
  location.diag.R,
  location.upper.tri.R
)
```

Arguments

`init.par`	initial (starting) parameters values
`yy`	a multivariate evoTS object
`A.user`	the pull matrix.
`R.user`	the drift matrix.
`locations.A`	location (row and column) of parameters (elements) in the A matrix that is estimated
`location.diag.A`	location (row and column) of parameters (elements) in the diagonal of the A matrix that is estimated
`location.upper.tri.A`	location (row and column) of parameters (elements) in the upper triangle of the A matrix that is estimated
`location.lower.tri.A`	location (row and column) of parameters (elements) in the lower triangle of the A matrix that is estimated
`locations.R`	location (row and column) of parameters (elements) in the R matrix that is estimated
`location.diag.R`	location (row and column) of parameters (elements) in the diagonal of the R matrix that is estimated
`location.upper.tri.R`	location (row and column) of parameters (elements) in the upper triangle of the R matrix that is estimated

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

`logL.joint.multi.R` *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for a multivariate Unbiased Random Walk model fitted to separate segments of a multivariate time series.

Usage

```
logL.joint.multi.R(init.par, C, y, m, n, anc.values, yy)
```

Arguments

`init.par`	initial (starting) parameters values
`C`	distance matrix
`y`	vector containing all trait values from all traits
`m`	number of traits
`n`	number of populations
`anc.values`	initial values for the ancestral trait values
`yy`	a multivariate evoTS object

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

`logL.joint.OU.BM` *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for an Ornstein-Uhlenbeck model where the optimum evolves as a Unbiased Random Walk. The movement of the optimum is not parameterized based on separate data.

Usage

```
logL.joint.OU.BM(p, y, opt.anc)
```

Arguments

`p`	parameters of the model to be optimized
`y`	a paleoTS object
`opt.anc`	logical, indicating if the ancestral trait value is at the optimum (TRUE) or displaced from the optimum (FALSE)

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

References

Hansen, T. F., Pienaar, J. & Orzack, S. H. A Comparative Method for Studying Adaptation to a Randomly Evolving Environment. *Evolution* 62, 1965–1977 (2008).

`logL.joint.single.R` *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for a multivariate Unbiased Random Walk model.

Usage

```
logL.joint.single.R(init.par, C, y, m, n, anc.values, yy)
```

Arguments

`init.par`	initial (starting) parameters values
`C`	distance matrix
`y`	vector containing all trait values from all traits
`m`	number of traits
`n`	number of populations
`anc.values`	initial values for the ancestral trait values
`yy`	a multivariate evoTS object

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

`logL.joint.single.R.zero.corr` *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for a multivariate Unbiased Random Walk model with uncorrelated changes.

Usage

```
logL.joint.single.R.zero.corr(init.par, C, y, m, n, anc.values, yy)
```

Arguments

`init.par`	initial (starting) parameters values
`C`	distance matrix
`y`	vector containing all trait values from all traits
`m`	number of traits
`n`	number of populations
`anc.values`	initial values for the ancestral trait values
`yy`	a multivariate evoTS object

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

logL.joint.Stasis.OU *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for a model with stasis in the first segment and an Ornstein-Uhlenbeck process in the second segment. .

Usage

```
logL.joint.Stasis.OU(p, y, gg)
```

Arguments

p	parameters of the model to be optimized
y	a paleoTS object
gg	numeric vector indicating membership of each sample in a segment

Details

In general, users will not be access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

logL.joint.URW.URW *Log-likelihoods for evolutionary models*

Description

Returns log-likelihood for a model with an Unbiased Random Walk in the first segment and an Unbiased Random Walk in the second segment.

Usage

```
logL.joint.URW.URW(p, y, gg)
```

Arguments

p	parameters of the model to be optimized
y	a paleoTS object
gg	numeric vector indicating membership of each sample in a segment

Details

In general, users will not access these functions directly, but instead use the optimization functions, which use these functions to find the best-supported parameter values.

Value

The log-likelihood of the parameter estimates, given the data.

Author(s)

Kjetil Lysne Voje

---

loglik.surface.accel *Calculate the log-likelihood surface for a part of parameter space*

Description

Function to calculate the log-likelihood surface for a part of parameter space for an Unbiased Random Walk with an accelerated rate of evolution.

Usage

```
loglik.surface.accel(y, vstep.vec, r.vec, pool = TRUE)
```

Arguments

y	an univariate paleoTS object.
vstep.vec	vector containing the parameter values of the variance parameter to be evaluated
r.vec	vector containing the parameter values of the r parameter to be evaluated
pool	indicating whether to pool variances across samples

Value

the function returns the range of parameter values that are within two log-likelihood units from the best (maximum) parameter estimate and a log-likelihood surface.

Note

How fine-scaled the estimated log-likelihood surface is depends on the step size between the values in the input-vectors. The step-size therefore determines how accurate the representation of the support surface is, including the returned upper and lower estimates printed in the console. The range of the input vectors needs to be increased if the confidence interval includes the boundary of the input vector. Note also that it might be wise to include the maximum likelihood estimates as part of the input vectors. The computed support surface is conditional on the best estimates of the other model parameters that are not part of the support surface (e.g. the estimated ancestral trait value).

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS objects
x <- sim.accel.decel(50)

## Fit the model to the data.
x1<-opt.joint.accel(x)


## Create log-likelihood surface (the example may take > 5 seconds to run)
loglik.surface.accel(x, vstep.vec = seq(0,4,0.005), r.vec = seq(0.15,0.25,0.005))
```

---

loglik.surface.decel *Calculate the log-likelihood surface for a part of parameter space*

Description

Function to calculate the log-likelihood surface for a part of parameter space for an Unbiased Random Walk with an decelerated rate of evolution.

Usage

```
loglik.surface.decel(y, vstep.vec, r.vec, pool = TRUE)
```

Arguments

y	an univariate paleoTS object.
vstep.vec	vector containing the parameter values of the variance parameter to be evaluated
r.vec	vector containing the parameter values of the r parameter to be evaluated
pool	indicating whether to pool variances across samples

Value

the function returns the range of parameter values that are within two log-likelihood units from the best (maximum) parameter estimate and a log-likelihood surface.

Note

How fine-scaled the estimated log-likelihood surface is depends on the step size between the values in the input-vectors. The step-size therefore determines how accurate the representation of the support surface is, including the returned upper and lower estimates printed in the console. The range of the input vectors needs to be increased if the confidence interval includes the boundary of the input vector. Note also that it might be wise to include the maximum likelihood estimates as part of the input vectors. The computed support surface is conditional on the best estimates of the other model parameters that are not part of the support surface (e.g. the estimated ancestral trait value).

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS objects
x <- sim.accel.decel(30, r=-0.5)

## Fit the model to the data.
x1<-opt.joint.decel(x)


## Create log-likelihood surface (the example may take > 5 seconds to run)
loglik.surface.decel(x, vstep.vec = seq(0, 5, 0.1), r.vec = seq(-1, 0, 0.01))
```

---

loglik.surface.GRW *Calculate the log-likelihood surface for a part of parameter space*

Description

Function to calculate the log-likelihood surface for a part of parameter space for a General Random Walk.

Usage

```
loglik.surface.GRW(y, mstep.vec, vstep.vec, pool = TRUE)
```

Arguments

y	an univariate paleoTS object.
mstep.vec	vector containing the parameter values of the mstep parameter to be evaluated
vstep.vec	vector containing the parameter values of the variance parameter to be evaluated
pool	indicating whether to pool variances across samples

Value

the function returns the range of parameter values that are within two log-likelihood units from the best (maximum) parameter estimate and a log-likelihood surface.

Note

How fine-scaled the estimated log-likelihood surface is depends on the step size between the values in the input-vectors. The step-size therefore determines how accurate the representation of the support surface is, including the returned upper and lower estimates printed in the console. The range of the input vectors needs to be increased if the confidence interval includes the boundary of the input vector. Note also that it might be wise to include the maximum likelihood estimates as part of the input vectors. The computed support surface is conditional on the best estimates of the other model parameters that are not part of the support surface (e.g. the estimated ancestral trait value).

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS objects
x <- paleoTS::sim.GRW(30)

## Fit the the model to the data.
x1<-paleoTS::opt.joint.GRW(x)


## Create log-likelihood surface (the example may take > 5 seconds to run)
loglik.surface.GRW(x, mstep.vec= seq(0,0.3,0.01), vstep.vec = seq(0,0.3,0.01))
```

---

loglik.surface.OU *Calculate the log-likelihood surface for a part of parameter space*

Description

Function to calculate the log-likelihood surface for a part of parameter space for a Ornstein-Uhlenbeck model.

Usage

```
loglik.surface.OU(
  y,
  stat.var.vec,
  h.vec,
  anc = NULL,
  theta = NULL,
  pool = TRUE
)
```

Arguments

y	an univariate paleoTS object.
stat.var.vec	vector containing the parameter values of the stationary variance to be evaluated
h.vec	vector containing the parameter values of the half life to be evaluated
anc	the ancestral state
theta	the optimum
pool	indicating whether to pool variances across samples

Value

the function returns the range of parameter values that are within two log-likelihood units from the best (maximum) parameter estimate and a log-likelihood surface.

Note

How fine-scaled the estimated log-likelihood surface is depends on the step size between the values in the input-vectors. The step-size therefore determines how accurate the representation of the support surface is, including the returned upper and lower estimates printed in the console. The range of the input vectors needs to be increased if the confidence interval includes the boundary of the input vector. Note also that it might be wise to include the maximum likelihood estimates as part of the input vectors. The computed support surface is conditional on the best estimates of the other model parameters that are not part of the support surface (e.g. the estimated ancestral trait value).

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS objects
x <- paleoTS::sim.OU(40)

## Fit the model to the data.
x1<-paleoTS::opt.joint.OU(x)
```

```
##calculate half-life from model output
log(2)/x1$parameters[4]

##calculate stationary variance from model output
x1$parameters[2]/(2*x1$parameters[4])


## Create log-likelihood surface (the example may take > 5 seconds to run)
loglik.surface.OU(x, stat.var.vec=seq(0.001,0.5,0.01), h.vec=seq(0.01,10, 0.1))
```

`loglik.surface.OUBM` *Calculate the log-likelihood surface for a part of parameter space*

Description

Function to calculate the log-likelihood surface for a part of parameter space for a Ornstein-Uhlenbeck model where the optimum changes according to an Unbiased Random Walk.

Usage

```
loglik.surface.OUBM(
  y,
  stat.var.vec,
  h.vec,
  anc = NULL,
  theta.0 = NULL,
  vo = NULL,
  opt.anc = TRUE,
  pool = TRUE
)
```

Arguments

`y`	an univariate paleoTS object.
`stat.var.vec`	vector containing the parameter values of the stationary variance to be evaluated
`h.vec`	vector containing the parameter values of the half life to be evaluated
`anc`	the ancestral state
`theta.0`	the optimum
`vo`	the variance (vstep) parameter for the optimum
`opt.anc`	logical, indicating whether the the ancestral trait state is at the optimum.
`pool`	indicating whether to pool variances across samples

Value

the function returns the range of parameter values that are within two log-likelihood units from the best (maximum) parameter estimate and a log-likelihood surface.

Note

How fine-scaled the estimated log-likelihood surface is depends on the step size between the values in the input-vectors. The step-size therefore determines how accurate the representation of the support surface is, including the returned upper and lower estimates printed in the console. The range of the input vectors needs to be increased if the confidence interval includes the boundary of the input vector. Note also that it might be wise to include the maximum likelihood estimates as part of the input vectors. The computed support surface is conditional on the best estimates of the other model parameters that are not part of the support surface (e.g. the estimated ancestral trait value).

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS objects
x <- sim.OUBM(40)

## Fit the model.
x1<-opt.joint.OUBM(x)

##calculate half-life from model output
log(2)/x1$parameters[3]

##calculate stationary variance from model output
x1$parameters[2]/(2*x1$parameters[3])


## Create log-likelihood surface (the example may take > 5 seconds to run)
loglik.surface.OUBM(x, stat.var.vec=seq(0,4,0.01), h.vec=seq(0.0,5, 0.1))
```

---

loglik.surface.stasis *Calculate the log-likelihood surface for a part of parameter space*

Description

Function to calculate the log-likelihood surface for a part of parameter space for the Stasis model.

Usage

```
loglik.surface.stasis(y, theta.vec, omega.vec, pool = TRUE)
```

Arguments

y	an univariate paleoTS object.
theta.vec	vector containing the parameter values of the theta parameter to be evaluated
omega.vec	vector containing the parameter values of the omega parameter to be evaluated
pool	indicating whether to pool variances across samples

Value

the function returns the range of parameter values that are within two log-likelihood units from the best (maximum) parameter estimate and a log-likelihood surface.

Note

How fine-scaled the estimated log-likelihood surface is depends on the step size between the values in the input-vectors. The step-size therefore determines how accurate the representation of the support surface is, including the returned upper and lower estimates printed in the console. The range of the input vectors needs to be increased if the confidence interval includes the boundary of the input vector. Note also that it might be wise to include the maximum likelihood estimates as part of the input vectors. The computed support surface is conditional on the best estimates of the other model parameters that are not part of the support surface (e.g. the estimated ancestral trait value).

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS objects
x <- paleoTS::sim.Stasis(30)

## Fit the the model to the data.
x1<-paleoTS::opt.joint.Stasis(x)


## Create log-likelihood surface (the example may take > 5 seconds to run)
loglik.surface.stasis(x, theta.vec= seq(-0.15,0.1,0.001), omega.vec = seq(0,0.1,0.001))
```

---

loglik.surface.URW *Calculate the log-likelihood surface for a part of parameter space*

Description

Function to calculate the log-likelihood surface for a part of parameter space for a Unbiased Random Walk.

Usage

```
loglik.surface.URW(y, vstep.vec, pool = TRUE)
```

Arguments

y	an univariate paleoTS object.
vstep.vec	vector containing the parameter values of the variance parameter to be evaluated
pool	indicating whether to pool variances across samples

Value

the function returns the range of parameter values that are within two log-likelihood units from the best (maximum) parameter estimate and a log-likelihood surface.

Note

How fine-scaled the estimated log-likelihood surface is depends on the step size between the values in the input-vectors. The step-size therefore determines how accurate the representation of the support surface is, including the returned upper and lower estimates printed in the console. The range of the input vectors needs to be increased if the confidence interval includes the boundary of the input vector. Note also that it might be wise to include the maximum likelihood estimates as part of the input vectors. The computed support surface is conditional on the best estimates of the other model parameters that are not part of the support surface (e.g. the estimated ancestral trait value).

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS objects
x <- paleoTS::sim.GRW(30)

## Fit a the model to the data by defining shift points.
x1<-paleoTS::opt.joint.URW(x)

## Create log-likelihood surface (the example may take > 5 seconds to run)
loglik.surface.URW(x, vstep.vec = seq(0,0.5,0.001))
```

`make.multivar.evoTS` *Makes a multivariate data set of*

Description

Function to make a multivariate data set consisting of two or more evolutionary sequences (time-series).

Usage

```
make.multivar.evoTS(
  evoTS.1 = NULL,
  evoTS.2 = NULL,
  evoTS.3 = NULL,
  evoTS.4 = NULL,
  evoTS.5 = NULL,
  evoTS.6 = NULL,
  evoTS.7 = NULL,
  evoTS.8 = NULL,
  evoTS.9 = NULL,
  evoTS.10 = NULL
)
```

Arguments

`evoTS.1`	an univariate evolutionary sequences (time-series) on the format used in paleoTS
`evoTS.2`	an univariate evolutionary sequences (time-series) on the format used in paleoTS
`evoTS.3`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)
`evoTS.4`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)
`evoTS.5`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)
`evoTS.6`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)
`evoTS.7`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)
`evoTS.8`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)
`evoTS.9`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)
`evoTS.10`	an univariate evolutionary sequences (time-series) on the format used in paleoTS (optional)

Details

See the function as.paleoTS for details. See also read.paleoTS, which is often a more convenient way for getting the relevant data from text files.

Value

a multivariate evoTS object that can be analysed with functions fitting multivariate models (e.g. fit.multivariate.OU, fit.multivariate.URW)

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate two evolutionary sequences (time-series)
x1 <- paleoTS::sim.GRW(60)
x2 <- paleoTS::sim.GRW(60)


## Make a multivariate data set
x1_x2<-make.multivar.evoTS(x1, x2)
```

`opt.accel.single.R` *Fit multivariate Unbiased Random Walk with increasing (exponential accelerating) rate of change through time.*

Description

Function to find maximum likelihood solution to a multivariate Unbiased Random Walk model with increasing (exponential accelerating) rate of change through time.

Usage

```
opt.accel.single.R(
  yy,
  method = "L-BFGS-B",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

yy	a multivariate evoTS object.
method	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.
pool	indicating whether to pool variances across samples
trace	logical, indicating whether information on the progress of the optimization is printed.
iterations	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
iter.sd	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model with increasing (exponential accelerating) rate of change through time.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately. The starting value for r = 1.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R, r). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the auto-correlation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate an evoTS object by simulating a multivariate dataset
indata <- sim.multi.URW(60)

## Fit a multivariate Unbiased Random Walk with an increasing rate of change through time.
opt.accel.single.R(indata)
```

`opt.accel.single.R.zero.corr` *Fit multivariate Unbiased Random Walk model with uncorrelated trait changes and with increasing (exponential accelerating) rate of change through time.*

Description

Function to find maximum likelihood solution to a multivariate Unbiased Random Walk model with uncorrelated trait changes and with increasing (exponential accelerating) rate of change through time.

Usage

```
opt.accel.single.R.zero.corr(
  yy,
  method = "L-BFGS-B",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

`yy`	a multivariate evoTS object.
`method`	optimization method, passed to function optim. Default is "L-BFGS-B".
`hess`	logical, indicating whether to calculate standard errors from the Hessian matrix.
`pool`	indicating whether to pool variances across samples
`trace`	logical, indicating whether information on the progress of the optimization is printed.
`iterations`	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
`iter.sd`	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model with uncorrelated trait changes and with increasing (exponential accelerating) rate of change through time.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately. The starting value for r = 1.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R, r). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the auto-correlation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate an evoTS object by simulating a multivariate dataset
x <- sim.multi.URW(30)

## Fit a multivariate Unbiased Random Walk model with an increasing rate of change through time.
opt.accel.single.R.zero.corr(x)
```

`opt.decel.single.R` *Fit multivariate Unbiased Random Walk with decreasing (exponential decaying) rate of change through time.*

Description

Function to find maximum likelihood solution to a multivariate Unbiased Random Walk model with decreasing (exponential decaying) rate of change through time.

Usage

```
opt.decel.single.R(
  yy,
  method = "L-BFGS-B",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

`yy`	a multivariate evoTS object.
`method`	optimization method, passed to function optim. Default is "L-BFGS-B".
`hess`	logical, indicating whether to calculate standard errors from the Hessian matrix.
`pool`	indicating whether to pool variances across samples
`trace`	logical, indicating whether information on the progress of the optimization is printed.
`iterations`	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
`iter.sd`	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model with decreasing (exponential decaying) rate of change through time.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately. The starting value for r = -1.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R, r). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Voje, K. L. 2020. Testing eco-evolutionary predictions using fossil data: Phyletic evolution following ecological opportunity.*Evolution* 74:188–200.

Examples

```
## Generate an evoTS object by simulating a multivariate dataset
indata <- sim.multi.URW(30)

## Fit a multivariate Unbiased Random Walk model with a decreasing rate of change through time.
opt.accel.single.R(indata)
```

`opt.decel.single.R.zero.corr`

Fit multivariate Unbiased Random Walk model with uncorrelated trait changes and with decreasing (exponential decaying) rate of change through time.

Description

Function to find maximum likelihood solution to a multivariate Unbiased Random Walk model with uncorrelated trait changes and with decreasing (exponential decaying) rate of change through time.

Usage

```
opt.decel.single.R.zero.corr(
  yy,
  method = "L-BFGS-B",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

yy	a multivariate evoTS object.
method	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.
pool	indicating whether to pool variances across samples
trace	logical, indicating whether information on the progress of the optimization is printed.
iterations	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
iter.sd	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model with uncorrelated trait changes and with increasing (exponential accelerating) rate of change through time.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately. The starting value for r = -1.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R, r). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Voje, K. L. 2020. Testing eco-evolutionary predictions using fossil data: Phyletic evolution following ecological opportunity.*Evolution* 74:188–200.

Examples

```
## Generate an evoTS object by simulating a multivariate dataset.
indata <- sim.multi.URW(30)

## Fit a multivariate Unbiased Random Walk model with a decreasing rate of change through time.
opt.decel.single.R.zero.corr(indata)
```

---

opt.joint.accel — *Fit an Unbiased Random Walk with an accelerating rate of change through time.*

Description

Function to find maximum likelihood solutions to a Unbiased Random Walk with an accelerating or decelerating rate of change through time.

Usage

```
opt.joint.accel(y, pool = TRUE, meth = "L-BFGS-B", hess = FALSE)
```

Arguments

y	an univariate evoTS object.
pool	logical indicating whether to pool variances across samples
meth	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.

Value

logL	the log-likelihood of the optimal solution
AICc	AIC with a correction for small sample sizes
parameters	parameter estimates
modelName	abbreviated model name
method	Joint consideration of all samples
K	number of parameters in the model
n	the number of observations/samples

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate a paleoTS object by simulating a univariate evolutionary sequence
y <- paleoTS::sim.GRW(30)

## Fit the model
opt.joint.accel(y)
```

---

opt.joint.decel *Fit an Unbiased Random Walk with an decelerating rate of change through time.*

Description

Function to find maximum likelihood solutions to a Unbiased Random Walk with an decelerating or decelerating rate of change through time.

Usage

```
opt.joint.decel(y, pool = TRUE, meth = "L-BFGS-B", hess = FALSE)
```

Arguments

y	an univariate evoTS object.
pool	logical indicating whether to pool variances across samples
meth	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.

Value

logL	the log-likelihood of the optimal solution
AICc	AIC with a correction for small sample sizes
parameters	parameter estimates
modelName	abbreviated model name
method	Joint consideration of all samples
K	number of parameters in the model
n	the number of observations/samples

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Voje, K. L. 2020. Testing eco-evolutionary predictions using fossil data: Phyletic evolution following ecological opportunity.*Evolution* 74:188–200.

Examples

```
## Generate a paleoTS object by simulating a univariate evolutionary sequence
x <- paleoTS::sim.GRW(30)

## Fit the model
opt.joint.decel(x)
```

`opt.joint.OUBM` *Fit an Ornstein-Uhlenbeck model with an optimum that evolves according to a Unbiased Random Walk.*

Description

Function to find maximum likelihood solutions to an Ornstein-Uhlenbeck model with an optimum that evolves according to a Unbiased Random Walk.

Usage

```
opt.joint.OUBM(
  y,
  pool = TRUE,
  meth = "L-BFGS-B",
  hess = FALSE,
  iterations = NULL,
  iter.sd = NULL,
  opt.anc = TRUE
)
```

Arguments

`y`	an univariate paleoTS object.
`pool`	logical indicating whether to pool variances across samples
`meth`	optimization method, passed to function optim. Default is "L-BFGS-B".
`hess`	logical, indicating whether to calculate standard errors from the Hessian matrix.
`iterations`	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
`iter.sd`	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.
`opt.anc`	logical, indicating whether the the ancestral trait state is at the optimum.

Value

`logL`	the log-likelihood of the optimal solution
`AICc`	AIC with a correction for small sample sizes
`parameters`	parameter estimates
`modelName`	abbreviated model name
`method`	Joint consideration of all samples
`K`	number of parameters in the model
`n`	the number of observations/samples

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Hansen, T. F., Pienaar, J. & Orzack, S. H. 2008. A Comparative Method for Studying Adaptation to a Randomly Evolving Environment. *Evolution* 62:1965–1977.

Examples

```
## Generate a paleoTS object by simulating a univariate evolutionary sequence
x <- paleoTS::sim.GRW(60)

## Fit the model
opt.joint.OUBM(x)
```

`opt.joint.URW.Stasis` *Optimization and log-likelihoods for pairs of models.*

Description

A collections of functions that serves the function fit.mode.shift. See fit.mode.shift for info.

Usage

```
opt.joint.URW.Stasis(
  y,
  gg,
  cl = list(fnscale = -1),
  pool = TRUE,
  meth = "L-BFGS-B",
  hess = FALSE
)
```

Arguments

y	a paleoTS object.
gg	numeric vector indicating membership of each sample in segments
cl	control list to be passed to optim
pool	logical indicating whether to pool variances across samples
meth	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.

Details

In general, users will not be access these functions directly, but instead use the wrapper function, which use these functions to find the best-supported parameter values.

Note

This function is not likely to be called directly by the user.

Author(s)

Kjetil Lysne Voje

---

opt.multi.R *Fit separate multivariate Unbiased Random Walk models to two different segments of a multivariate evolutionary sequence (time-series).*

Description

This function is used internally by evoTS and will generally not be used directly by users.

Usage

```
opt.multi.R(
  yy,
  gg,
  method,
  hess = FALSE,
  pool = TRUE,
  trace,
  iterations,
  iter.sd
)
```

Arguments

yy	a multivariate evoTS object.
gg	numeric vector indicating membership of each sample in the two segments.
method	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.
pool	indicating whether to pool variances across samples
trace	logical, indicating whether information on the progress of the optimization is printed.
iterations	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
iter.sd	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

This function is used internally by evoTS and will generally not be used directly by users.

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the auto-correlation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Revell, L. J. & Harmon, L. Testing quantitative genetic hypotheses about the evolutionary rate matrix for continuous characters. *Evolutionary Ecology Research* 10, 311–331 (2008).

`opt.single.R` *Fit multivariate Unbiased Random Walk model.*

Description

Function to find maximum likelihood solution to a multivariate Unbiased Random Walk model.

Usage

```
opt.single.R(
  yy,
  method = "L-BFGS-B",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

`yy`	a multivariate evoTS object.
`method`	optimization method, passed to function optim. Default is "L-BFGS-B".
`hess`	logical, indicating whether to calculate standard errors from the Hessian matrix.
`pool`	indicating whether to pool variances across samples
`trace`	logical, indicating whether information on the progress of the optimization is printed.
`iterations`	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
`iter.sd`	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Revell, L. J. & Harmon, L. Testing quantitative genetic hypotheses about the evolutionary rate matrix for continuous characters. *Evolutionary Ecology Research* 10, 311–331 (2008).

Examples

```
## Generate an evoTS objects by simulating a multivariate dataset
x <- sim.multi.URW(30)

## Fit a multivariate Unbiased Random Walk model.
opt.single.R(x)
```

`opt.single.R.zero.corr` *Fit multivariate Unbiased Random Walk model with uncorrelated trait changes.*

Description

Function to find maximum likelihood solution to a multivariate Unbiased Random Walk model with uncorrelated trait changes.

Usage

```
opt.single.R.zero.corr(
  yy,
  method = "L-BFGS-B",
  hess = FALSE,
  pool = TRUE,
  trace = FALSE,
  iterations = NULL,
  iter.sd = NULL
)
```

Arguments

yy	a multivariate evoTS object.
method	optimization method, passed to function optim. Default is "L-BFGS-B".
hess	logical, indicating whether to calculate standard errors from the Hessian matrix.
pool	indicating whether to pool variances across samples
trace	logical, indicating whether information on the progress of the optimization is printed.
iterations	the number of times the optimization method is run from different starting points. Default is NULL, meaning the optimization is run once.
iter.sd	defines the standard deviation of the Gaussian distribution from which starting values for the optimization routine is run. Default is 1.

Details

The function searches - using an optimization routine - for the maximum-likelihood solution for a multivariate Unbiased Random Walk model with uncorrelated trait changes.

The argument 'method' is passed to the 'optim' function and is included for the convenience of users to better control the optimization routine. The the default method (L-BFGS-B) seems to work for most evolutionary sequences.

Initial estimates to start the optimization come from maximum-likelihood estimates of the univariate Unbiased Random Walk model (from the paleoTS package) fitted to each time-series separately.

It is good practice to repeat any numerical optimization procedure from different starting points. This is especially important for complex models as the log-likelihood surface might contain more than one peak. The number of iterations is controlled by the argument 'iterations'. The function will report the model parameters from the iteration with the highest log-likelihood.

Value

First part of the output reports the log-likelihood of the model and its AICc score. The second part of the output is the maximum log-likelihood model parameters (ancestral.values, R). The last part of the output gives information about the number of parameters in the model (K), number of samples in the data (n) and number of times the optimization routine was run (iter).

Note

The models have been implemented to be compatible with the joint parameterization routine in the package paleoTS. The optimization is therefore fit using the actual sample values, with the autocorrelation among samples accounted for in the log-likelihood function. The joint distribution of sample means is multivariate normal, with means and variance-covariances determined by evolutionary parameters and sampling errors.

Author(s)

Kjetil Lysne Voje

References

Revell, L. J. & Harmon, L. Testing quantitative genetic hypotheses about the evolutionary rate matrix for continuous characters. *Evolutionary Ecology Research* 10, 311–331 (2008).

Examples

```
## Generate an evoTS object by simulating a multivariate dataset
x <- sim.multi.URW(30)

## Fit a multivariate Unbiased Random Walk model with uncorrelated trait changes.
opt.single.R.zero.corr(x)
```

---

plotevoTS *Plot a paleoTS object*

Description

Plot a paleoTS object (slightly modified version of the same function in paleoTS)

Usage

```
plotevoTS(
  x,
  nse = 1,
  pool = FALSE,
  add = FALSE,
  modelFit = NULL,
  pch = 19,
  lwd = 1.5,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

x	a paleoTS object
nse	the number of standard errors represented by the error bars on the plot; defaults to 1
pool	logical indicating if variances should be pooled across samples for the purposes of displaying error bars; defaults to FALSE
add	logical, if TRUE, adds to existing plot
modelFit	optional model fit from fitting functions
pch	plotting symbol, defaults to 19
lwd	line width, defaults to 1.5
ylim	optional, y-limits of the plot
xlab	a title for the x axis
ylab	a title for the y axis
...	other arguments passed to plotting functions

Value

The results are plotted.

Author(s)

Kjetil Lysne Voje

---

plotevoTS.multivariate

Plots multivariate evolutionary sequence (time-series) data set

---

Description

Function to plot multivariate evolutionary sequence (time-series), showing trait means over time.

Usage

```
plotevoTS.multivariate(
  yy,
  nse = 1,
  col = NULL,
  lty = NULL,
  lwd = NULL,
  pch = NULL,
  x.label = NULL,
  y.label = NULL,
  y_min = NULL,
  y_max = NULL,
```

```
  cex.axis = NULL,
  cex.lab = NULL,
  cex.main = NULL,
  axes = NULL
)
```

Arguments

yy	a multivariate evoTS object
nse	the number of standard errors represented by the error bars on the plot; default is 1
col	vector indicating colors
lty	line type
lwd	line width
pch	plotting symbols
x.label	label on x axis
y.label	label on y axis
y_min	minimum value of y axis
y_max	maximum value of y axis
cex.axis	Specify the size of the tick label numbers/text
cex.lab	specify the size of the axis label
cex.main	specify the size of the title text
axes	logical, whether to plot axes or not

Value

The results are plotted.

Author(s)

Kjetil Lysne Voje

Examples

```
## Generate two evolutionary sequences (time-series)
x1 <- paleoTS::sim.Stasis(60, vp=1)
x2 <- paleoTS::sim.Stasis(60, vp=1)

## Make a multivariate data set
x1_x2<-make.multivar.evoTS(x1, x2)

## Plot the data
plotevoTS.multivariate(x1_x2, y_min=-1, y_max=1)
```

ribs_S.yellowstonensis

Evolutionary sequence (time-series) of phenotypic change in the number of ribs in the lineage Stephanodiscus yellowstonensis

Description

Phenotypic data (ribs) from a centric diatom lineage Stephanodiscus yellowstonensis. The time series spans about 14 000 years. The data set contains data on the number of costae (ribs) per valve. The data consists of an objects of class paleoTS (ribs_S.yellowstonensis). Objects of class paleoTS can be analyzed in evoTS. The object (trait data set) contains a vector of sample means (mm), sample variances (vv), sample sizes (nn) and sample ages (tt). The oldest sample is listed first. The data spans an interval of 13728 years.

Usage

```
data(ribs_S.yellowstonensis)
```

Format

An object of class "paleoTS".

References

Theriot et al. 2006. Late Quaternary rapid morphological evolution of an endemic diatom in Yellowstone Lake, Wyoming. Paleobiology 32:38-54

Examples

```
ln.ribs<-paleoTS::ln.paleoTS(ribs_S.yellowstonensis)
ln.ribs$tt<-ln.ribs$tt/(max(ln.ribs$tt))
opt.joint.decel(ln.ribs)
```

sim.accel.decel

Simulate an Unbiased Random Walk with an accelerating or decelerating rate of change through time.

Description

Function to simulate an evolutionary sequence data set according to an Unbiased Random Walk with an accelerating or decelerating rate of change through time.

Usage

```
sim.accel.decel(
  ns = 20,
  vs = 0.5,
  r = 0.2,
  vp = 0.2,
  nn = rep(20, ns),
  tt = 0:(ns - 1)
)
```

Arguments

ns	number of samples in time-series
vs	step variance of the trait
r	the parameter controlling the exponential decay (if negative) or increase (if positive) of the rate (vs) through time.
vp	phenotypic variance of each sample
nn	vector of the number of individuals in each sample (identical sample sizes for all time-series is assumed)
tt	vector of sample times (ages

Value

An evolutionary sequence (time-series) data set (a paleoTS object)

Author(s)

Kjetil Lysne Voje

Examples

```
##Simulate an unbiased random walk where the rate decelerates through time.
x<-sim.accel.decel(40, r=-0.5)

## Plot the data
plotevoTS(x)
```

---

sim.multi.OU — *Simulate multivariate Ornstein-Uhlenbeck evolutionary sequence data sets*

Description

Function to simulate a multivariate Ornstein-Uhlenbeck evolutionary sequence data set.

Usage

```
sim.multi.OU(
  ns = 30,
  anc = c(0, 0),
  optima = c(3, 2),
  A = matrix(c(7, 0, 0, 6), nrow = 2, byrow = TRUE),
  R = matrix(c(0.05, 0, 0, 0.05), nrow = 2, byrow = TRUE),
  vp = 0.1,
  nn = rep(30, ns),
  tt = 0:(ns - 1)
)
```

Arguments

ns	number of samples in time-series
anc	the ancestral trait values
optima	the optimal trait values
A	the pull matrix.
R	the drift matrix
vp	within-population trait variance
nn	vector of the number of individuals in each sample (identical sample sizes for all time-series is assumed)
tt	vector of sample ages, increases from oldest to youngest

Value

A multivariate evolutionary sequence (time-series) data set.

Note

The Ornstein Uhlenbeck model is reduced to an Unbiased Random Walk when the alpha parameter is zero. It is therefore possible to let a trait evolve as an Unbiased Random Walk by setting the diagonal element for that trait to a value close to zero (e.g. 1e-07). Elements in the diagonal of A cannot be exactly zero as this will result in a singular variance-covariance matrix.

Author(s)

Kjetil Lysne Voje

Examples

```
##Define the A and R matrices

A_matrix<-matrix(c(4,-2,0,3), nrow=2, byrow = TRUE)
R_matrix<-matrix(c(4,0.2,0.2,4), nrow=2, byrow = TRUE)

## Generate an evoTS object by simulating a multivariate dataset
```

```
data_set<-sim.multi.OU(40, optima = c(1.5,2),A=A_matrix , R = R_matrix)

## plot the data
plotevoTS.multivariate(data_set)
```

`sim.multi.URW` *Simulate multivariate evolutionary sequence data that evolve according to an Unbiased Random Walk*

Description

Function to simulate multivariate evolutionary sequence data that evolve according to an Unbiased Random Walk

Usage

```
sim.multi.URW(
  ns = 30,
  anc = c(0, 0),
  R = matrix(c(0.5, 0, 0, 0.5), nrow = 2, byrow = TRUE),
  vp = 0.1,
  nn = rep(30, ns),
  tt = 0:(ns - 1)
)
```

Arguments

`ns`	number of samples in time-series
`anc`	the ancestral trait values
`R`	the drift matrix
`vp`	within-population trait variance
`nn`	vector of the number of individuals in each sample (identical sample sizes for all time-series is assumed)
`tt`	vector of sample ages, increases from oldest to youngest

Value

A multivariate evolutionary sequence (time-series) data set.

Author(s)

Kjetil Lysne Voje

Examples

```
## Create a multivariate dataset
data_set<-sim.multi.URW(40, R = matrix(c(0.2,0.1,0.1,0.3), nrow=2, byrow = TRUE))

## plot the data
plotevoTS.multivariate(data_set)
```

`sim.OUBM` *Simulate an Ornstein-Uhlenbeck process with optimum changing according to an unbiased random walk*

Description

Function to simulate an Ornstein-Uhlenbeck evolutionary sequence data set with an optimum moving according to an unbiased random walk.

Usage

```
sim.OUBM(
  ns = 20,
  anc = 0,
  theta.0 = 1,
  alpha = 0.3,
  vstep.trait = 0.1,
  vstep.opt = 0.1,
  vp = 1,
  nn = rep(20, ns),
  tt = 0:(ns - 1)
)
```

Arguments

`ns`	number of samples in time-series
`anc`	the ancestral trait values
`theta.0`	the ancestral value for the optimum
`alpha`	strength of attraction to the optimum
`vstep.trait`	step variance of the trait
`vstep.opt`	step variance of the optimum
`vp`	phenotypic variance of each sample
`nn`	vector of the number of individuals in each sample (identical sample sizes for all time-series is assumed)
`tt`	vector of sample times (ages

Value

An evolutionary sequence (time-series) data set (a paleoTS object)

Author(s)

Kjetil Lysne Voje

Examples

```
##Create data
x<-sim.OUBM(50, theta.0 = 5, alpha = 0.6, vstep.opt = 0.5)

## plot the data
plot(x)
```

Index