Package 'paleoTS'

March 12, 2019

Title Analyze Paleontological Time-Series

Version 0.5.2

Description Facilitates analysis of paleontological sequences of trait values. Functions are provided to fit, using maximum likelihood, simple evolutionary models (including unbiased random walks, directional evolution,stasis, Ornstein-Uhlenbeck, covariate-tracking) and complex models (punctuation, mode shifts).

Depends R (>= 3.4.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports mnormt, foreach, iterators, parallel, doParallel

RoxygenNote 6.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Gene Hunt [aut, cre]

Maintainer Gene Hunt <hunte@si.edu>

Repository CRAN

Date/Publication 2019-03-12 20:10:03 UTC

R topics documented:

`as.paleoTS` *Make a Paleontological Time-series object*

Description

Combines information into an object of class `paleoTS`

Usage

```
as.paleoTS(mm, vv, nn, tt, MM = NULL, genpars = NULL, label = NULL,
  start.age = NULL, oldest = c("first", "last"), reset.time = TRUE)
```

Arguments

mm	vector of sample means
vv	vector of sample variances
nn	vector of sample sizes
tt	vector of sample ages
MM	vector of true means (simulated data)
genpars	generating parameters (simulated data)
label	optional, label for time-series
start.age	optional, age of oldest sample
oldest	value indicating if the oldest sample is first or last in the sequence
reset.time	logical; if TRUE, then change time scale to start at t=0 and adjust start.age accordingly

Details

This function combines data into a paleoTS object. For empirical data it may be more convenient to use read.paleoTS.

If sample ages decrease through the sequence, as if given in millions of years ago, tt will automatically be converted to time elapsed from the beginning of the sequence as long as reset.time = TRUE.

Value

a paleoTS object

Note

All model-fitting functions estimate the contribution of sampling noise to the observed differences between samples. They do this assuming that the trait is represented by sample means, which have sampling variances equal to variance divided by sample size, vv/nn. If one is interested in analyzing statistics other than the sample mean (medians, quantiles, or other statistics), use the the following procedure: set the statistic in question as the mm values, replace vv with a vector of the squared standard errors for each estimate (generated by other means, for example bootstrapping), and set all values of nn to one.

See Also

read.paleoTS

Examples

```
x <- as.paleoTS(mm = rnorm(20), vv = rep(1, 20), nn = rep(25, 20), tt=1:20)
plot(x) # easier to use sim.Stasis()
```

as.paleoTSfit *Create a* paleoTSfit *object*

Description

Create a paleoTSfit object

Usage

```
as.paleoTSfit(logL, parameters, modelName, method, K, n, se)
```

Arguments

logL	model log-likelihood
parameters	model parameter estimates
modelName	model name
method	parameterization, either "AD" or "Joint"
K	number of model parameters
n	sample size
se	standard errors of parameter estimates

Value

a paleoTSfit object

Note

All model-fitting functions use this function to package the resulting data-model fits. Users will not need to call this function.

bootSimpleComplex *Bootstrap test to see if a complex model is significantly better than a simple model.*

Description

Bootstrap test to see if a complex model is significantly better than a simple model.

Usage

```
bootSimpleComplex(y, simpleFit, complexFit, nboot = 99, minb = 7,
  ret.full.distribution = FALSE, parallel = FALSE, ...)
```

Arguments

y	a paleoTS object
simpleFit	a paleoTSfit object, representing the model fit of a simple model
complexFit	a paleoTSfit object, representing the model fit of a complex model
nboot	number of replications for parametric bootstrapping
minb	minimum number of populations within each segment
ret.full.distribution	logical, indicating if the null distribution for the likelihood ratio from the parametric bootstrap should be returned
parallel	logical, if TRUE, the bootstrapping is done using parallel computing
...	further arguments, passed to optimization functions

Details

Simulations suggest that AICc can be overly liberal with complex models with mode shifts or punctuations (Hunt et al., 2015). This function implements an alternative of parametric boostrapping to compare the fit of a simple model with a complex model. It proceeds in five steps:

1. Compute the observed gain in support from the simple to complex model as the likelihood ratio, $LR_obs = -2(logL_simple - logL_complex)$
2. Simulate trait evolution under the specified simple model nboot times
3. Fit to each simulated sequence the specified simple and complex models
4. Measure the gain in support from simple to complex as the bootstrap likelihood ratio for each simulated sequence
5. Compute the P-value as the percentile of the bootstrap distribution corresponding to the observed LR.

Argument simpleFit should be a paleoTS object returned by the function fitSimple or similar functions (e.g., opt.joint.GRW,opt.GRW, etc.). Argument complexFit must be a paleoTS object returned by fitGpunc or fitModeShift.

Calculations can be speeded up by setting parallel = TRUE, which uses functions from the doParallel package to run the bootstrap replicates in parallel, using one fewer than the number of detected cores.

Value

A list of the observed likelihood ratio statistic, LRobs, the P-value of the test, and the number of bootstrap replicates. If ret.full.distribution = TRUE, the null distribution of likelihood ratios generated by parametric bootstrapping is also returned.

References

Hunt, G., M. J. Hopkins and S. Lidgard. 2015. Simple versus complex models of trait evolution and stasis as a response to environmental change. PNAS 112(16): 4885-4890.

See Also

sim.Stasis.RW, fitModeShift

Examples

```
## Not run:
x <- sim.Stasis.RW(ns = c(15, 15), omega = 0.5, ms = 1, order = "Stasis-RW")
ws <- fitSimple(x)
wc <- fitModeShift(x, order = "Stasis-RW", rw.model = "GRW")
bootSimpleComplex(x, ws, wc, nboot = 50, minb = 7)  # nboot too low for real analysis!

## End(Not run)
```

---

cantius_L *Time-series of the length of lower first molar for the Cantius lineage*

Description

Time-series of the length of lower first molar for the Cantius lineage

Usage

```
cantius_L
```

Format

a paleoTS object with the data

Source

Clyde, W. C. and P. D. Gingerich (1994). Rates of evolution in the dentition of early Eocene Cantius: comparison of size and shape. *Paleobiology* 20(4): 506-522.

---

compareModels *Compare model fits for a paleontological time-series*

Description

Takes output from model-fitting functions and compiles model-fit information (log-likelihood, AICc, etc.) into a convenient table

Usage

```
compareModels(..., silent = FALSE, sort = FALSE)
```

Arguments

...	any number of model fit (as.paleoTSfit) objects
silent	if TRUE, suppresses printing
sort	if TRUE, the table sorts models from best to worst

Value

if silent = FALSE, the table is printed and nothing is returned. If silent = TRUE, printing is suppressed and a list of two objects is returned: the table of model fits, modelFits, and a list of parameter estimates, pl.

Examples

```
x <- sim.GRW(ns = 40, ms = 0.5, vs = 0.1)
m1 <- fitSimple(x, model = "GRW")  # the true model
m2 <- fitSimple(x, model = "URW")
plot(x, modelFit = m1)
compareModels(m1, m2)
```

---

dorsal.spines *Time-series of dorsal spine data from a fossil stickleback lineage*

Description

Time-series of dorsal spine data from a fossil stickleback lineage

Usage

```
dorsal.spines
```

Format

a paleoTS object of the mean number of dorsal spines (log-transformed)

Source

Bell, M.A., M.P. Travis and D.M. Blouw 2006. Inferring natural selection in a fossil threespine stickleback. *Paleobiology* **32**:562-577.

Hunt, G., M. A. Bell and M. P. Travis (2008). Evolution toward a new adaptive optimum: phenotypic evolution in a fossil stickleback lineage. *Evolution* 62(3): 700-710.

ESD *Compute Expected Squared Divergence (ESD) for Evolutionary Models*

Description

Computes for a specified model and duration of time the expected squared divergence (ESD), which is a useful measure of the magnitude or rate of change across different models.

Usage

```
ESD(y, dt, model = c("GRW", "URW", "Stasis", "allThree"),
  method = c("Joint", "AD"), pool = TRUE, ...)
```

Arguments

y	a paleoTS object
dt	the time interval to evaluate ESD
model	the model of evolution to assume; see Details
method	Joint or AD parameterization
pool	logical, if TRUE, variances are averaged (pooled) across samples
...	other arguments to the model-fitting functions

Details

Hunt (2012) argued that rate metrics make sense only in the context of specific models of evolution. It is thus difficult to meaningfully compare rates across sequences generated by different evolutionary processes. ESD values can be used for a specified model and duration as a comparable measure of the amount of evolutionary change that is expected. Acceptable values for the model argument can be "GRW" for the general random walk (directional change), "URW" for the unbiased random walk, and "Stasis." In addition, one can also specify "allThree", in which case all these models will be fit and the resulting ESD will be the weighted average of them, using model support (Akaike weights) for the weighting (see Hunt [2012], p. 370)

Value

the ESD value

References

Hunt, G. 2012. Measuring rates of phenotypic evolution and the inseparability of tempo and mode. *Paleobiology* 38:351–373.

Examples

```
x<- sim.GRW(ns=20)
esd.urw<- ESD(x, dt=10, model="URW")
esd.all<- ESD(x, dt=10, model="allThree")
```

fit.sgs *Fit a model of trait evolution with a protracted punctuation.*

Description

This function fits a model of punctuated change that is is protracted enough that it is captured by multiple transitional populations. Trait evolution starts in stasis, shifts to a general random walk, and then shifts back into stasis.

Usage

```
fit.sgs(y, minb = 7, oshare = TRUE, pool = TRUE, silent = FALSE,
  hess = FALSE, meth = "L-BFGS-B", model = "GRW")
```

Arguments

y	a paleoTS object
minb	minimum number of populations within each segment
oshare	logical, if TRUE, variance assumed to be shared (equal) across segments
pool	if TRUE, sample variances are substituted with their pooled estimate
silent	logical, if TRUE, progress updates are suppressed
hess	if TRUE, standard errors computed from the Hessian matrix are returned
meth	optimization method, passes to optim
model	type of random walk: "URW", unbiased random walk, or "GRW", a general (directional) random walk

Value

a paleoTSfit object

See Also

fitGpunc

Examples

```
x <- sim.sgs(ns = c(15, 15, 15))  # default values OK
w <- fit.sgs(x, minb = 10)  # increase minb so example takes less time; not recommended!
plot(x)
abline(v = c(16, 31), lwd = 3)  # actual shifts
abline(v = c(w$parameters[6:7]), lwd = 2, lty = 3, col = "red")  # inferred shifts
```

`fit3models` *Fit a set of standard evolutionary models*

Description

Fit a set of standard evolutionary models

Usage

```
fit3models(y, silent = FALSE, method = c("Joint", "AD"), ...)

fit4models(y, silent = FALSE, method = c("Joint", "AD"), ...)
```

Arguments

`y`	a `paleoTS` object
`silent`	if TRUE, results are returned as a list and not printed
`method`	"Joint" or "AD", see `fitSimple`
`...`	other arguments passed to model fitting functions

Details

Function `fit3models` fits the general (biased) random walk (GRW), unbiased random walk (URW), and Stasis models. In addition to these three, `fit4models` also fits the model of Strict Stasis.

Value

if silent = FALSE, a table of model fit statistics, also printed to the screen. if silent = TRUE, a list of the model fit statistics and model parameter values.

Functions

- `fit4models`: add model of "Strict Stasis" to the three models

See Also

`fitSimple`

Examples

```
x <- sim.GRW(ns = 50, ms = 0.2)
fit4models(x)
```

`fit9models` *Fit large set of models to a time-series*

Description

This function fits nine models to a time-series following Hunt et al. (2015). These include the simple models fit by `fit4models` along with mode shift and punctuation models.

Usage

```
fit9models(y, silent = FALSE, method = c("Joint", "AD"), ...)
```

Arguments

`y`	a `paleoTS` object
`silent`	logical, if TRUE, progress updates are suppressed
`method`	parameterization to use: `Joint` or `AD`; see Details
`...`	other arguments, passed to optimization functions

Value

if silent = FALSE, a table of model fit statistics, also printed to the screen. if silent = TRUE, a list of the model fit statistics and model parameter values.

References

Hunt, G., M. J. Hopkins and S. Lidgard. 2015. Simple versus complex models of trait evolution and stasis as a response to environmental change. PNAS 112(16): 4885-4890.

Examples

```
## Not run:
x <- sim.Stasis.RW(ns = c(15, 15), omega = 0.5, ms = 1, order = "Stasis-RW")
plot(x)
fit9models(x)

## End(Not run)
```

`fitGpunc` *Fit trait evolution model with punctuations estimated from the data*

Description

Fit trait evolution model with punctuations estimated from the data

Usage

```
fitGpunc(y, ng = 2, minb = 7, pool = TRUE, oshare = TRUE,
  method = c("Joint", "AD"), silent = FALSE, hess = FALSE,
  parallel = FALSE, ...)
```

Arguments

`y`	a `paleoTS` object
`ng`	number of groups (segments) in the sequence
`minb`	minimum number of populations within each segment
`pool`	if TRUE, sample variances are substituted with their pooled estimate
`oshare`	logical, if TRUE, variance assumed to be shared (equal) across segments
`method`	parameterization to use: `Joint` or `AD`; see Details
`silent`	logical, if TRUE, progress updates are suppressed
`hess`	if TRUE, standard errors computed from the Hessian matrix are returned
`parallel`	logical, if TRUE, the analysis is done in parallel
`...`	other arguments, passed to optimization functions

Details

This function tests all possible shift points for punctuations, subject to the constraint that the number of populations in each segment is always >= `minb`. The shiftpoint yielding the highest log-likelihood is returned as the solution, along with the log-likelihoods (`all.logl`) of all tested shift points (`GG`).

The function uses `opt.punc` (if `method = "AD"`) or `opt.joint.punc` (if `method = "Joint"`) to do the fitting.

Value

a `paleoTSfit` object with the results of the model-fitting.

Note

Calculations can be speeded up by setting `parallel = TRUE`, which uses functions from the `doParallel` package to run the bootstrap replicates in parallel, using one fewer than the number of detected cores.

See Also

fit9models, sim.punc

Examples

```
x <- sim.punc(ns = c(15, 15), theta = c(0,3), omega = c(0.1, 0.1))
w.punc <- fitGpunc(x, oshare = TRUE)
plot(x, modelFit = w.punc)
```

`fitModeShift` *Fit model in which the mode of trait evolution shifts once*

Description

Trait evolution is modeled as a shift from a random walk (general or unbiased) to stasis, or vice versa.

Usage

```
fitModeShift(y, minb = 7, pool = TRUE, order = c("Stasis-RW",
  "RW-Stasis"), rw.model = c("URW", "GRW"), method = c("Joint", "AD"),
  silent = FALSE, hess = FALSE, ...)
```

Arguments

y	paleoTS object
minb	minimum number of populations within each segment
pool	if TRUE, sample variances are substituted with their pooled estimate
order	whether stasis or random walk come first, one of Stasis-RW or RW-Stasis
rw.model	whether the random walk segment is an unbiased random walk, URW or a general random walk, GRW
method	parameterization to use: Joint or AD
silent	logical, if TRUE, progress updates are suppressed
hess	if TRUE, standard errors computed from the Hessian matrix are returned
...	other arguments, passed to optimization functions

Value

a paleoTSfit object

See Also

sim.Stasis.RW

Examples

```
x <- sim.Stasis.RW(ns = c(15, 15), omega = 0.5, ms = 1, order = "Stasis-RW")
plot(x)
w <- fitModeShift(x, order = "Stasis-RW", rw.model = "GRW")
abline(v = x$tt[15], lwd = 3)  # actual shift point
abline(v = x$tt[w$par["shift1"]], lty = 3, lwd = 2, col = "red") # inferred shift
```

---

fitMult *Fit the same simple model across multiple time-series*

Description

Fit the same simple model across multiple time-series

Usage

```
fitMult(yl, model = c("GRW", "URW", "Stasis", "covTrack"),
  method = c("Joint", "AD"), pool = TRUE, zl = NULL, hess = FALSE)
```

Arguments

yl	a list of paleoTS objects
model	the model to fit; see Details
method	parameterization to use: Joint or AD
pool	if TRUE, sample variances are substituted with their pooled estimate
zl	for the covTrack model only, a list of covariate vectors, one each paleoTS object in yl
hess	if TRUE, standard errors computed from the Hessian matrix are returned

Details

This function fits a model with shared parameters across multiple trait time-series. The most likely application would be to model a common evolutionary dynamic across different sequences, perhaps representing time-series of the same trait and lineage from different localities or time intervals.

Four simple models are currently implemented:

- **GRW**: parameters mstep and vstep of the general random walk are shared across sequences.
- **URW**: parameter vstep of the unbiased random walk is shared across sequences.
- **Stasis**: parameter omega of stasis is shared across sequences.
- **covTrack**: parameters b0, b1, and evar of the covariate-tracking model are shared across sequences.

Under the joint parameterization, method = "Joint", an additional parameter, anc is fit, representing the ancestral (starting) trait value. This parameter is estimated separately in each sequence so it is not assumed that they all start at the same trait value.

Value

a paleoTSfit object with the results of the model-fitting

Note

The models are described in the help for fitSimple and the functions linked from there.

See Also

fitSimple

Examples

```
x1 <- sim.GRW(ms = 1, vs = 0.2)
x2 <- sim.GRW(ms = 1, vs = 0.2)
fitMult(list(x1, x2), model = "GRW")
```

---

fitSimple *Fit simple models of trait evolution*

Description

Fit simple models of trait evolution

Usage

```
fitSimple(y, model = c("GRW", "URW", "Stasis", "StrictStasis", "OU",
  "covTrack"), method = c("Joint", "AD"), pool = TRUE, z = NULL,
  hess = FALSE)
```

Arguments

y	a paleoTS object
model	the model to be fit, one of "GRW", "URW", "Stasis", "OU","covTrack"
method	parameterization to use: Joint or AD; see Details
pool	if TRUE, sample variances are substituted with their pooled estimate
z	a vector of a covariate, used only for the "covTrack" model
hess	if TRUE, standard errors computed from the Hessian matrix are returned

Details

This is a convenience function that calls the specific individual functions for each model and parameterization, such as `opt.GRW` and `opt.joint.GRW`. The models that this function can fit are:

- **GRW**: General Random Walk. Under this model, evolutionary changes, or "steps" are drawn from a distribution with a mean of `mstep` and variance of `vstep`. `mstep` determines directionality and `vstep` determines volatility (Hunt, 2006).
- **URW**: Unbiased Random Walk. Same as GRW with `mstep` = 0, and thus evolution is non-directional. For a URW, `vstep` is the rate parameter.
- **Stasis**: This parameterization follows Sheets & Mitchell (2001), with a constant mean `theta` and variance `omega` (equivalent to white noise).
- **Strict Stasis**: Same as Stasis with `omega` = 0, indicating no real evolutionary differences; all observed variation is sampling error (Hunt et al. 2015).
- **OU**: Ornstein-Uhlenbeck model (Hunt et al. 2008). This model is that of a population ascending a nearby peak in the adaptive landscape. The optimal trait value is `theta`, `alpha` indicates the strength of attraction to that peak (= strength of stabilizing selection around `theta`), `vstep` measures the random walk component (from genetic drift) and `anc` is the trait value at the start of the sequence.
- **covTrack**: Covariate-tracking (Hunt et al. 2010). The trait tracks a covariate with slope `b1`, consistent with an adaptive response. `evar` is the residual variance, and, under `method = "Joint"`, `b0` is the intercept of the relationship between trait and covariate. model.

Value

a `paleoTSfit` object with the model fitting results

Note

For the covariate-tracking model, z should be a vector of length *n* when `method = "Joint"` and *n* - 1 when `method = "AD"`, where *n* is the number of samples in `y`.

Method = `"Joint"` is a full likelihood approach, considering each time-series as a joint sample from a multivariate normal distribution. Method = `"AD"` is a REML approach that uses the differences between successive samples. They perform similarly, but the Joint approach does better under some circumstances (Hunt, 2008).

References

Hunt, G. 2006. Fitting and comparing models of phyletic evolution: random walks and beyond. *Paleobiology* 32(4): 578-601.
Hunt, G. 2008. Evolutionary patterns within fossil lineages: model-based assessment of modes, rates, punctuations and process. p. 117-131 *In* From Evolution to Geobiology: Research Questions Driving Paleontology at the Start of a New Century. Bambach, R. and P. Kelley (Eds).
Hunt, G., M. A. Bell and M. P. Travis. 2008. Evolution toward a new adaptive optimum: phenotypic evolution in a fossil stickleback lineage. *Evolution* 62(3): 700-710.
Sheets, H. D., and C. Mitchell. 2010. Why the null matters: statistical tests, random walks and evolution. *Genetica* 112– 113:105–125.

See Also

opt.GRW, opt.joint.GRW, opt.joint.OU, opt.covTrack

Examples

```
y <- sim.Stasis(ns = 20, omega = 2)
w1 <- fitSimple(y, model = "GRW")
w2 <- fitSimple(y, model = "URW")
w3 <- fitSimple(y, model = "Stasis")
compareModels(w1, w2, w3)
```

IC *Compute Information Criteria*

Description

Compute Information Criteria

Usage

```
IC(logL, K, n = NULL, method = c("AICc", "AIC", "BIC"))
```

Arguments

logL	log-likelihood
K	number of parameters
n	sample size
method	either "AIC", "AICc", or "BIC"

Value

the value of the specified information criterion

Note

This function is used internally by the model-fitting functions. It will not generally be called directly by the user.

`ln.paleoTS` *Approximate log-transformation of time-series data*

Description

Approximate log-transformation of time-series data

Usage

```
ln.paleoTS(y)
```

Arguments

`y`	a `paleoTS` object

Details

For a random variable x, its approximate mean on a natural log scale is the log of its untransformed mean. The approximate variance on a log scale is equal to the squared coefficient of variation.

Value

the converted `paleoTS` object

Note

This transformation only makes sense for variables with dimension and a true zero point, such as lengths and areas.

References

Hunt, G. 2006. Fitting and comparing models of phyletic evolution: random walks and beyond. *Paleobiology* 32:578-601.
Lewontin, R. 1966. On the measurement of relative variability. *Systematic Zoology* 15:141-142.

Examples

```
x <- sim.Stasis(ns = 10, theta = 20, omega = 1)
plot(x)
xl <- ln.paleoTS(x)
plot(xl)
```

LRI *Log-rate, Log-interval (LRI) method of Gingerich*

Description

Gingerich (1993) introduced a method that plots on log-log scale, the rate and interval for each pair of samples in an evolutionary sequence. On this plot, the slope is interpreted as an indicator of evolutionary mode (-1 for stasis, 0.5 for random walk, 0 for directional), and the intercept is interpreted as a measure of the rate of evolution over one generation.

Usage

```
LRI(y, gen.per.t = 1e+06, draw = TRUE)
```

Arguments

y	a paleoTS object
gen.per.t	the number of generations per unit time
draw	logical, if TRUE, a plot is produced

Details

Following Gingerich (1993), a robust line is fit through the points by minimizing the sum of absolute deviations. If generations are one year long and time is measured in Myr, gen.per.t= 1e6.

Value

A named vector with three elements: Intercept, slope, and GenerationalRate

Note

This method was important in early attempts to disentangle evolutionary tempo and mode. I view likelihood-based methods as more informative, and in particular the estimation of 'Generational Rates' using LRI is compromised by sampling error; see Hunt (2012) and the example below.

References

Gingerich, P.D. 1993. Quantification and comparison of evolutionary rates. *American Journal of Science* 293-A:453–478.

Hunt, G. 2012. Measuring rates of phenotypic evolution and the inseparability of tempo and mode. *Paleobiology* 38:351–373.

See Also

lynchD

Examples

```
set.seed(1)
xFast <- sim.GRW(ns = 20, ms = 0.5, vs = 0.2)  # fast evolution
xSlow <- sim.Stasis(ns = 20, omega = 0)        # strict stasis (zero rates)
lri.Fast <- LRI(xFast, draw = FALSE)
lri.Slow <- LRI(xSlow, draw = FALSE)
print(lri.Fast[3], 4)
print(lri.Slow[3], 4)  # LRI thinks strict stasis rates are faster!
```

---

lynchD *Compute Lynch's Delta rate metric*

Description

This function computes D, the rate metric proposed by Lynch (1990). This metric derives from the random walk model, with `D = Vstep/(2Vp)`, where `Vstep` is the step variance of the unbiased random walk, and `Vp` is the within sample variance, pooled among samples. Under mutation - drift equilibrium, `D` is expected to range approximately between 5e-5 and 5e-3.

Usage

```
lynchD(y, gen.per.t = 1e+06, pool = TRUE, method = c("Joint", "AD"),
  ...)
```

Arguments

`y`	a `paleoTS` object
`gen.per.t`	the number of generations per unit time
`pool`	logical, whether variances should be pooled over samples
`method`	parameterization to use: based on ancestor-descendant (AD) differences, or Joint consideration of all samples
`...`	further arguments, passed to `opt.URW` or `opt.joint.URW`

Value

`D`	value of rate metric
`pooled.var`	value of pooled within-sample variance
`gen.per.t`	number of generations per unit time
`vstep`	computed `Vstep`, at the original time scale of `y`
`drift.range`	expected minimum and maximum values of `D` consistent with neutral evolution
`result`	conclusion reached about the plausibility of neutral evolution

References

Lynch (1990). The rate of morphological evolution in mammals from the standpoint of the neutral expectation. *The American Naturalist* 136:727-741. Hunt, G. 2012. Fitting and comparing models of phyletic evolution: random walks and beyond. *Paleobiology* 38:351-373.

Examples

```
y <- sim.GRW(ns = 20, ms = 0, vs = 1e-4, tt=seq(0, 1e6, length.out=20))  # per-year simulation
lynchD(y, gen.per.t = 1)
```

`mle.GRW` *Analytical ML estimator for random walk and stasis models*

Description

Analytical ML estimator for random walk and stasis models

Usage

```
mle.GRW(y)

mle.URW(y)

mle.Stasis(y)
```

Arguments

y	a paleoTS object

Value

a vector of `mstep` and `vstep` for `mle.GRW`, `vstep` for `mle.URW`, and `theta` and `omega` for `mle.Stasis`

Functions

- `mle.URW`: ML parameter estimates for URW model
- `mle.Stasis`: ML parameter estimates for Stasis model

Note

These analytical solutions assume even spacing of samples and equal sampling variance in each, which will usually be violated in real data. They are used here mostly to generate initial parameter estimates for numerical optimization; they not likely to be called directly by the user.

See Also

`fitSimple`

opt.covTrack *Fit a model in which a trait tracks a covariate*

Description

Fit a model in which a trait tracks a covariate

Usage

```
opt.covTrack(y, z, pool = TRUE, cl = list(fnscale = -1),
  meth = "L-BFGS-B", hess = FALSE)

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

Arguments

y	a paloeTS object
z	a vector of covariate values
pool	if TRUE, sample variances are substituted with their pooled estimate
cl	optional control list, passed to optim()
meth	optimization algorithm, passed to optim()
hess	if TRUE, return standard errors of parameter estimates from the hessian matrix

Details

In this model, changes in a trait are linearly related to changes in a covariate with a slope of b and residual variance evar: dx = b * dz + eps, where eps ~ N(0, evar). This model was described, and applied to an example in which body size changes tracked changes in temperature, by Hunt et al. (2010).

For the AD version (opt.covTrack), a trait sequence of length ns, the covariate, z, can be of length ns - 1, interpreted as the vector of *changes*, dx. If z is of length ns, differences are taken and these are used as the dx's, with a warning issued.

The Joint version (opt.joint.covTrack), z should be of length ns and there is an additional parameter that is the intercept of the linear relationship between trait and covariate. See warning below about using the Joint version.

Value

a paleoTSfit object with the results of the model fitting

Functions

- opt.joint.covTrack: fits the covTrack model using the joint parameterization

Warning

The Joint parameterization of this model can be fooled by temporal autocorrelation and, especially, trends in the trait and the covariate. The latter is tested for, but the AD parameterization is generally safer for this model.

References

Hunt, G, S. Wicaksono, J. E. Brown, and K. G. Macleod. 2010. Climate-driven body size trends in the ostracod fauna of the deep Indian Ocean. *Palaeontology* 53(6): 1255-1268.

See Also

fitSimple

Examples

```
set.seed(13)
z <- c(1, 2, 2, 4, 0, 8, 2, 3, 1, 9, 4, 3)
x <- sim.covTrack(ns = 12, z = z, b = 0.5, evar = 0.2)
w.urw <- opt.URW(x)
w.cov <- opt.covTrack(x, z = z)
compareModels(w.urw, w.cov)
```

`opt.GRW` *Fit evolutionary model using "AD" parameterization*

Description

Fit evolutionary model using "AD" parameterization

Usage

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

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

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

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

Arguments

y	a paleoTS object
pool	if TRUE, sample variances are substituted with their pooled estimate
cl	optional control list, passed to optim()
meth	optimization algorithm, passed to optim()
hess	if TRUE, return standard errors of parameter estimates from the hessian matrix

Details

These functions use differences between consecutive populations in the time series in order to remove temporal autocorrelation. This is referred to as the "Ancestor-Descendant" or "AD" parameterization by Hunt [2008], and it is a REML approach (like phylogenetic independent contrasts). A full ML approach, called "Joint" was found to have somewhat better performance (Hunt, 2008) and generally should be used instead.

Value

a paleoTSfit object with the model fitting results

Functions

- opt.URW: fit the URW model by the AD parameterization
- opt.Stasis: fit the Stasis model by the AD parameterization
- opt.StrictStasis: fit the Strict Stasis model by the AD parameterization

Note

It is easier to use the convenience function fitSimple.

References

Hunt, G. 2006. Fitting and comparing models of phyletic evolution: random walks and beyond. *Paleobiology* 32(4): 578-601.

See Also

fitSimple, opt.joint.GRW

Examples

```
x <- sim.GRW(ns = 20, ms = 1)  # strong trend
plot(x)
w.grw <- opt.GRW(x)
w.urw <- opt.URW(x)
compareModels(w.grw, w.urw)
```

`opt.GRW.shift` *Fit random walk model with shift(s) in generating parameters*

Description

Fit random walk model with shift(s) in generating parameters

Usage

```
opt.GRW.shift(y, ng = 2, minb = 7, model = 1, pool = TRUE,
  silent = FALSE)
```

Arguments

`y`	a `paloeTS` object
`ng`	number of segments in the sequence
`minb`	minimum number of populations in each segment
`model`	numeric, specifies exact evolutionary model; see Details
`pool`	if TRUE, sample variances are substituted with their pooled estimate
`silent`	logical, if TRUE, progress updates are suppressed

Details

Fits a model in which a sequence is divided into two or more segments and trait evolution proceeds as a general random walk, with each segment (potentially) getting its own generating parameters (`mstep`, `vstep`).

This function tests for shifts after each population, subject to the constraint that the number of populations in each segment is always >= `minb`. The shiftpoint yielding the highest log-likelihood is returned as the solution, along with the log-likelihoods (`all.logl` of all tested shift points (`GG`).

Different variants of the model can be specified by the `model` argument:

- `model = 1:` `mstep` is separate across segments; `vstep` is shared
- `model = 2:` `mstep` is shared across segments; `vstep` is separate
- `model = 3:` `mstep` is set to zero (unbiased random walk); `vstep` is separate across segments
- `model = 4:` `mstep` and `vstep` are both separate across segments

Value

a `paleoTSfit` object

See Also

sim.GRW.shift

Examples

```
x <- sim.GRW.shift(ns = c(15,15), ms = c(0, 1), vs = c(0.1,0.1))
w.sep <- opt.GRW.shift(x, ng = 2, model = 4)
w.sameVs <- opt.GRW.shift(x, ng = 2, model = 1)
compareModels(w.sep, w.sameVs)
plot(x)
abline(v = x$tt[16], lwd = 3)  # actual shift point
abline(v = x$tt[w.sameVs$par["shift1"]], lty = 3, col = "red", lwd = 2) # inferred shift point
```

---

opt.joint.GRW *Fit evolutionary models using the "Joint" parameterization*

Description

Fit evolutionary models using the "Joint" parameterization

Usage

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

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

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

opt.joint.StrictStasis(y, pool = TRUE, cl = list(fnscale = -1),
  hess = FALSE)
```

Arguments

y	a paleoTS object
pool	if TRUE, sample variances are substituted with their pooled estimate
cl	optional control list, passed to optim()
meth	optimization algorithm, passed to optim()
hess	if TRUE, return standard errors of parameter estimates from the hessian matrix

Details

These functions use the joint distribution of population means to fit models using a full maximum-likelihood approach. This approach was found to have somewhat better performance than the "AD" approach, especially for noisy trends (Hunt, 2008).

Value

a paleoTSfit object with the model fitting results

Functions

- opt.joint.URW: fit the URW model by the Joint parameterization
- opt.joint.Stasis: fit the Stasis model by the Joint parameterization
- opt.joint.StrictStasis: fit the Strict Stasis model by the Joint parameterization

Note

It is easier to use the convenience function fitSimple.

References

#' Hunt, G., M. J. Hopkins and S. Lidgard. 2015. Simple versus complex models of trait evolution and stasis as a response to environmental change. PNAS 112(16): 4885-4890.

See Also

fitSimple, opt.GRW

Examples

```
x <- sim.GRW(ns = 20, ms = 1)  # strong trend
plot(x)
w.grw <- opt.joint.GRW(x)
w.urw <- opt.joint.URW(x)
compareModels(w.grw, w.urw)
```

---

opt.joint.OU *Fit Ornstein-Uhlenbeck model using the "Joint" parameterization*

Description

Fit Ornstein-Uhlenbeck model using the "Joint" parameterization

Usage

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

Arguments

y	a paleoTS object
pool	if TRUE, sample variances are substituted with their pooled estimate
cl	optional control list, passed to optim()
meth	optimization algorithm, passed to optim()
hess	if TRUE, return standard errors of parameter estimates from the hessian matrix

Details

This function fits an Ornstein-Uhlenbeck (OU) model to time-series data. The OU model has four generating parameters: an ancestral trait value (anc), an optimum value (theta), the strength of attraction to the optimum (alpha), and a parameter that reflects the tendency of traits to diffuse (vstep). In a microevolutionary context, these parameters can be related to natural selection and genetic drift; see Hunt et al. (2008).

Value

a paleoTSfit object with the model fitting results

Note

It is easier to use the convenience function fitSimple. Note also that preliminary work found that the "AD" parameterization did not perform as well for the OU model and thus it is not implemented here.

References

Hunt, G., M. A. Bell and M. P. Travis. 2008. Evolution toward a new adaptive optimum: phenotypic evolution in a fossil stickleback lineage. *Evolution* 62(3): 700-710.

See Also

fitSimple, opt.joint.GRW

Examples

```
x <- sim.OU(vs = 0.5)  # most defaults OK
w <- opt.joint.OU(x)
plot(x, modelFit = w)
```

opt.punc *Fit a model of trait evolution with specified punctuation(s)*

Description

Fit a model of trait evolution with specified punctuation(s)

Usage

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

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

Arguments

y	a paleoTS object
gg	vector of indices indicating different segments
pool	if TRUE, sample variances are substituted with their pooled estimate
cl	optional control list, passed to optim()
meth	optimization algorithm, passed to optim()
hess	if TRUE, return standard errors of parameter estimates from the
oshare	logical, if TRUE, variance assumed to be shared (equal) across segments

Details

The sequence is divided into segments, which are separated by punctuations. Means for each segment are given by the vector theta with variances given by the vector omega (or a single value if oshare = TRUE). This function calls optim to numerically fit this model to a time-series, y.

Value

a paleoTSfit object with the results of the model fitting

Functions

- opt.joint.punc: fits the punctuation model using the joint parameterization

Note

These functions would be used in the uncommon situation in which there is a prior hypothesis as to where the punctuation(s) take place. Normally users will instead use the function fitGpunc, which uses these functions to fit a range of possible timings for the punctuations.

See Also

fitGpunc

Examples

```
x <- sim.punc(ns = c(15, 15), theta = c(0,3), omega = c(0.1, 0.1))
w.sta  <- fitSimple(x, model = "Stasis", method = "Joint")
w.punc <- opt.joint.punc(x, gg = rep(1:2, each = 15), oshare = TRUE)
compareModels(w.sta, w.punc)
```

---

plot.paleoTS *Plot a paleoTS object*

Description

Plot a paleoTS object

Usage

```
## S3 method for class 'paleoTS'
plot(x, nse = 1, pool = FALSE, add = FALSE,
  modelFit = NULL, pch = 21, lwd = 1.5, ylim = 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
...	other arguments passed to plotting functions

Value

none.

Examples

```
x <- sim.GRW(ns = 30)
w <- fitSimple(x, model = "GRW", method = "Joint")
plot(x, modelFit = w)
```

pool.var *Compute a pooled variance*

Description

Computes a pooled variance from samples in a paleontological time-series

Usage

```
pool.var(y, nn = NULL, minN = NULL, ret.paleoTS = FALSE)
```

Arguments

y	either a paleoTS object, or a vector of sample variances
nn	a vector of sample sizes
minN	sample size below which variances are replaced with pooled variances. See Details.
ret.paleoTS	if TRUE, a paleoTS object is returned. If FALSE, the value of the pooled variance is returned.

Details

A pooled variance of a set of populations is the weighted average of the individual variances of the populations, with the weight for each population equal to its sample size minus one.

For many kinds of traits, variation levels tend to be similar among closely related populations. When this is true and sample sizes are low, much of the observed differences in variance among samples will be due to the high noise of estimated the variances. Replacing the observed variances of all populations (or only those with `nn < minN`) with the estimated pooled variance can reduce this noise.

Value

if `ret.paleoTS = TRUE` a paleoTS object with all (or some) variances replaced with the pooled variance; otherwise the pooled variance

Examples

```
data(cantius_L)
cant_all <- pool.var(cantius_L, ret.paleoTS = TRUE)  # replace all variances with pooled variance
cant_n5  <- pool.var(cantius_L, minN = 5, ret.paleoTS = TRUE)  # replace only pops with n < 5
```

read.paleoTS *Read a text-file with data from a paleontological time-series*

Description

Read a text-file with data from a paleontological time-series

Usage

```
read.paleoTS(file = NULL, oldest = "first", reset.time = TRUE, ...)
```

Arguments

file	file name; if not supplied, an interactive window prompts the user to navigate to the text file
oldest	"first" if samples are in order from oldest to youngest, "last" if the opposite
reset.time	logical; see as.paleoTS
...	other arguments, passed to read.table

Details

This function reads a text file with a specified format and converts it into a paleoTS object. It will often be the easiest way for users to import their own data. The text file should have four columns without headers, in this order: sample size, sample means, sample variances, sample ages.

Value

a paleoTS object

See Also

as.paleoTS

sim.covTrack *Simulate trait evolution that tracks a covariate*

Description

Simulate trait evolution that tracks a covariate

Usage

```
sim.covTrack(ns = 20, b = 1, evar = 0.1, z, nn = rep(20, times =
  ns), tt = 0:(ns - 1), vp = 1)
```

Arguments

ns	number of populations in a sequence
b	slope of the relationship between the change in the covariate and the change in the trait
evar	residual variance of the same relationship
z	vector of covariate that the trait tracks
nn	vector of sample sizes for populations
tt	vector of times (ages) for populations
vp	phenotypic trait variance within each population

Details

In this model, changes in a trait are linearly related to changes in a covariate with a slope of b and residual variance evar: `dx = b * dz + eps`, where `eps ~ N(0, evar)`. This model was described, and applied to an example in which body size changes tracked changes in temperature, by Hunt et al. (2010).

Value

a paleoTS object

Note

For a trait sequence of length ns, the covariate, z, can be of length ns - 1,in which case it is interpreted as the vector of *changes*, dz. If z is of length ns, differences are taken and these are used as the dz's.

References

Hunt, G, S. Wicaksono, J. E. Brown, and K. G. Macleod. 2010. Climate-driven body size trends in the ostracod fauna of the deep Indian Ocean. *Palaeontology* 53(6): 1255-1268.

Examples

```
set.seed(13)
z <- c(1, 2, 2, 4, 0, 8, 2, 3, 1, 9, 4, 3)
x <- sim.covTrack(ns = 12, z = z, b = 0.5, evar = 0.2)
plot(x, ylim = c(-1, 10))
lines(x$tt, z, col = "blue")
```

`sim.GRW` *Simulate random walk or directional time-series for trait evolution*

Description

Simulate random walk or directional time-series for trait evolution

Usage

```
sim.GRW(ns = 20, ms = 0, vs = 0.1, vp = 1, nn = rep(20, ns),
  tt = 0:(ns - 1))
```

Arguments

`ns`	number of populations in the sequence
`ms`	mean of evolutionary "steps"
`vs`	variance of evolutionary "steps"
`vp`	phenotypic variance within populations
`nn`	vector of population sample sizes
`tt`	vector of population times (ages)

Details

The general random walk model considers time in discrete steps. At each time step, an evolutionary change is drawn at random from a distribution of possible evolutionary "steps." It turns out that the long-term dynamics of an evolving lineage depend only on the mean and variance of this step distribution. The former, `mstep`, determined the directionality in a sequence and the latter, `vstep`, determines its volatility.

Value

a `paleoTS` object

Note

This function simulates an unbiased random walk if `ms` is equal to zero and a general (or biased) random walk otherwise.

See Also

`sim.Stasis`, `sim.OU`, `as.paleoTS`

Examples

```
x.grw <- sim.GRW(ms = 0.5)
x.urw <- sim.GRW(ms = 0)
plot(x.grw, ylim = range(c(x.grw$mm, x.urw$mm)))
plot(x.urw, add = TRUE, col = "blue")
legend(x = "topleft", c("GRW", "URW"), col = c("black", "blue"), lty = 1)
```

`sim.GRW.shift` *Simulate (general) random walk with shift(s) in generating parameters*

Description

Simulate (general) random walk with shift(s) in generating parameters

Usage

```
sim.GRW.shift(ns = c(10, 10), ms = c(0, 1), vs = c(0.5, 0.5),
  nn = rep(30, sum(ns)), tt = 0:(sum(ns) - 1), vp = 1)
```

Arguments

`ns`	vector of the number of samples in each segment
`ms`	vector of mean step parameter in each segment
`vs`	vector of step variance parameter in each segment
`nn`	vector of sample sizes, one for each population
`tt`	vector of samples times (ages)
`vp`	phenotypic variance in each sample

Details

Simulates under a model in which a sequence is divided into two or more segments. Trait evolution proceeds as a general random walk, with each segment getting its own generating parameters (`mstep`, `vstep`).

Value

a `paleoTS` object with the simulated time-series

See Also

`sim.GRW`, `sim.sgs`, `opt.GRW.shift`

Examples

```
x <- sim.GRW.shift(ns = c(10,10,10), ms = c(0, 1, 0), vs = c(0.1,0.1,0.1))
plot(x)
abline(v = c(9.5, 19.5), lty = 3, lwd = 2, col = "blue")  # shows where dynamics shift
text (c(5, 15, 25), c(2,2,2), paste("segement", 1:3, sep =" "), col = "blue")
```

sim.OU *Simulate an Ornstein-Uhlenbeck time-series*

Description

Simulate an Ornstein-Uhlenbeck time-series

Usage

```
sim.OU(ns = 20, anc = 0, theta = 10, alpha = 0.3, vstep = 0.1,
  vp = 1, nn = rep(20, ns), tt = 0:(ns - 1))
```

Arguments

ns	number of populations in the sequence
anc	ancestral phenotype
theta	OU optimum (long-term mean)
alpha	strength of attraction to the optimum
vstep	step variance
vp	phenotypic variance of each sample
nn	vector of sample sizes
tt	vector of sample times (ages)

Details

This function simulates an Ornstein-Uhlenbeck (OU) process. In microevolutionary terms, this models a population ascending a nearby peak in the adaptive landscape. The optimal trait value is theta, alpha indicates the strength of attraction to that peak (= strength of stabilizing selection around theta), vstep measures the random walk component (from genetic drift) and anc is the trait value at the start of the sequence.

Value

a paleoTS object

References

Hunt, G., M. A. Bell and M. P. Travis. 2008. Evolution toward a new adaptive optimum: phenotypic evolution in a fossil stickleback lineage. *Evolution* 62(3): 700-710.

See Also

opt.joint.OU

Examples

```
x1 <- sim.OU(alpha = 0.8)  # strong alpha
x2 <- sim.OU(alpha = 0.1)  # wearker alpha
plot(x1)
plot(x2, add = TRUE, col = "blue")
```

`sim.punc` *Simulate a punctuated time-series*

Description

Simulates punctuated trait evolution with punctuations that are rapid relative to the spacing of samples. In practice, the time-series is divided into two or more segments, each of which has its own mean and variance.

Usage

```
sim.punc(ns = c(10, 10), theta = c(0, 1), omega = rep(0,
  length(theta)), nn = rep(30, sum(ns)), tt = 0:(sum(ns) - 1),
  vp = 1)
```

Arguments

`ns`	vector of the number of samples in each segment
`theta`	vector of means, one for each segment
`omega`	vector of variances, one for each segment.
`nn`	vector of sample sizes, one for each population
`tt`	vector of times (ages), one for each population
`vp`	phenotypic variance within each population

Details

Segments are separated by punctuations. Population means in the ith segment are drawn randomly from a normal distribution with a mean equal to ith element of `theta` and variance equal to the ith element of `omega`. The magnitudes of punctuations are determined by the differences in adjacent `theta` values.

Value

a `paleoTS` object with the simulated time-series.

See Also

`fitGpunc`

Examples

```
x <- sim.punc(ns = c(15, 15), theta = c(0,3), omega = c(0.1, 0.1))
plot(x)
```

`sim.sgs` *Simulate protracted punctuation*

Description

This function simulates a punctuated change that is is protracted enough that it is captured by multiple transitional populations. Trait evolution starts in stasis, shifts to a general random walk, and then shifts back into stasis.

Usage

```
sim.sgs(ns = c(20, 20, 20), theta = 0, omega = 1, ms = 1,
  vs = 0.1, nn = rep(30, sum(ns)), tt = 0:(sum(ns) - 1), vp = 1)
```

Arguments

`ns`	vector with the number of samples in each segment
`theta`	trait mean for initial stasis segment
`omega`	trait variance for stasis segments
`ms`	step mean during random walk segment
`vs`	step variance during random walk segment
`nn`	vector of sample sizes for each population
`tt`	vector of times (ages) for each population
`vp`	phenotypic trait variance for each population

Details

Trait evolution proceeds in three segments: Stasis, General random walk, stasis (sgs). The initial stasis segment has a mean of `theta` and variance `omega` before shifting in the second segment to a general random walk with parameters `ms` and `vs`. Finally, the third segment is a return to stasis, centered around the trait value of the last population of the random walk.

Value

a `paleoTS` object

Examples

```
x <- sim.sgs()  # default values OK
plot(x)
```

`sim.Stasis` *Simulate Stasis time-series for trait evolution*

Description

Simulate Stasis time-series for trait evolution

Usage

```
sim.Stasis(ns = 20, theta = 0, omega = 0, vp = 1, nn = rep(20,
  ns), tt = 0:(ns - 1))
```

Arguments

`ns`	number of populations in the sequence
`theta`	mean of populations
`omega`	variance among populations
`vp`	phenotypic variance within populations
`nn`	vector of population sample sizes
`tt`	vector of population times (ages)

Value

a `paleoTS` object

See Also

`sim.GRW`, `sim.OU`, `as.paleoTS`

Examples

```
x <- sim.Stasis(omega = 0.5, vp = 0.1)
w.sta <- fitSimple(x, model = "Stasis")
w.ss  <- fitSimple(x, model = "StrictStasis")
compareModels(w.sta, w.ss)
```

`sim.Stasis.RW` *Simulate trait evolution with a mode shift*

Description

Trait evolution is modeled as a shift from a random walk (general or unbiased) to stasis, or vice versa.

Usage

```
sim.Stasis.RW(ns = c(20, 20), order = c("Stasis-RW", "RW-Stasis"),
  anc = 0, omega = 1, ms = 0, vs = 1, vp = 1, nn = 30,
  tt = NULL)
```

Arguments

`ns`	vector of the number of samples in each segment
`order`	whether stasis or random walk come first, one of `"Stasis-RW"` or `"RW-Stasis"`
`anc`	starting trait value
`omega`	variance of stasis segment
`ms`	step mean during random walk segment
`vs`	step variance during random walk segment
`vp`	phenotypic trait variance for each population
`nn`	vector of sample sizes for each population
`tt`	vector of times (ages) for each population

Details

The `anc` argument is the starting trait value, and if the first segment is stasis, this is also the value of the stasis mean. When the first segment is a random walk, the stasis mean in the second segment is equal to the true trait mean at the end of the initial random walk.

Value

a `paleoTSfit` object

See Also

`fitModeShift`

Examples

```
x1 <- sim.Stasis.RW(omega = 0.1, ms = 5, order = "Stasis-RW")
x2 <- sim.Stasis.RW(omega = 0.1, ms = 5, order = "RW-Stasis")
plot(x1)
plot(x2, add = TRUE, col = "blue")
abline(v = 19, lty=3)
```

`std.paleoTS` *Convert time-series to standard deviation units*

Description

Convert time-series to standard deviation units

Usage

```
std.paleoTS(y, center = c("mean", "start"))
```

Arguments

`y`	a `paleoTS` object
`center`	optional translation of time-series according to "mean" or "start"; see Details

Details

The standardization expresses each sample mean as the deviation from the overall mean, divided by the pooled within-sample variance. Sample variances are also divided by the pooled sample variance.

Essentially, this converts paleontological time-series data into standard deviation units, similar to the computation of evolutionary rates in haldanes. This operation *does not* change the relative fit of models, but it does facilitate the comparison of parameter estimates across time-series of traits measured in different units.

If argument `center` = "start" the time-series is translated such that the trait mean of the first sample is zero.

Value

the converted `paleoTS` object

Examples

```
x <- sim.Stasis(ns = 8, theta = 1, omega = 4, vp = 2)
xs <- std.paleoTS(x, center = "start")
plot(x, ylim = range(c(x$mm, xs$mm)))
plot(xs, col = "red", add = TRUE)
legend(x = "topright", c("unstandardized", "standardized"), lty=1, col=c("black", "red"), cex=0.7)
```

sub.paleoTS *Subsample a paleontological time-series*

Description

Subsampling is done according to the supplied logical vector or, if none is supplied, as a proportion of samples, randomly chosen.

Usage

```
sub.paleoTS(y, ok = NULL, k = 0.1, reset.time = TRUE)
```

Arguments

y	a paleoTS object
ok	a logical vector, TRUE for populations to retain
k	proportion of samples to retain, with the samples chosen randomly
reset.time	if TRUE, resets the time so that the first population time is zero

Value

the sub-sampled paleoTS object

Examples

```
x <- sim.GRW(ns=20)
plot(x)
xs1 <- sub.paleoTS(x, k = 0.5)
plot(xs1, add = TRUE, col="green")
keep <- rep(c(TRUE, FALSE), 10)
xs2 <- sub.paleoTS(x, ok = keep)
plot(xs2, add = TRUE, col = "red")
```

test.var.het *Test for heterogeneity of variances among samples in a time-series*

Description

Test for heterogeneity of variances among samples in a time-series

Usage

```
test.var.het(y, method = "Bartlett")
```

Arguments

y	a paleoTS object
method	test to use; currently only "Bartlett" is implemented

Value

a list with the test statistic, degrees of freedom, and p-value

Note

Most often, this function will be used to assess if it is reasonable to pool variances across samples using pool.var. A significant result means that the null hypothesis of equal variances across samples is rejected. Even in this case, however, it may still be preferable to pool variances, at least for some populations, if sample sizes are quite low.

References

Sokal, R. R and F. J. Rohlf. 1995. *Biometry* 3rd Ed.

See Also

pool.var

Examples

```
data(cantius_L)
test.var.het(cantius_L)  # significant, but still may want to pool variances
```

Index