Package: diverge (via r-universe)

September 17, 2024

Version 2.0.6

Date 2022-12-14

Title Evolutionary Trait Divergence Between Sister Species and Other Paired Lineages

Maintainer Sean A. S. Anderson <sean.as.anderson@gmail.com>

Author Sean A. S. Anderson and Jason T. Weir

Description Compares the fit of alternative models of continuous trait differentiation between sister species and other paired lineages. Differences in trait means between two lineages arise as they diverge from a common ancestor, and alternative processes of evolutionary divergence are expected to leave unique signatures in the distribution of trait differentiation in datasets comprised of many lineage pairs. Models include approximations of divergent selection, drift, and stabilizing selection. A variety of model extensions facilitate the testing of process-to-pattern hypotheses. Users supply trait data and divergence times for each lineage pair. The fit of alternative models is compared in a likelihood framework.

Imports parallel, truncnorm

Depends R (>= 3.4)

License GPL (>= 2)

NeedsCompilation no

LazyData true

Repository CRAN

Date/Publication 2022-12-15 08:40:06 UTC

Contents

`diverge-package` *Evolutionary Trait Divergence Between Sister Species and Other Paired Lineages*

Description

This package can be used to study the dynamics of continuous trait divergence in sister species and other paired lineages. The primary aim is to characterize the evolutionary processes responsible for continuous trait differentiation between lineages following their departure from a common ancestor. Alternative mechanisms of trait divergence such as divergent selection, drift, and stabilizing selection are expected to leave unique signatures in the distribution of trait differentiation in datasets comprised of many species pairs. The package uses maximum likelihood to estimate parameters and compare the fits of alternative process-based models to such distributions. Key functions for most users will be 'model_select' and 'find_mle'. See 'details' below for a description of the modelling framework, data requirements, and other considerations.

Details

Package:	diverge
Type:	Package
Version:	2.0.6
Date:	2022-12-14
License:	GPL (>= 2)

BASIC MODELS:
Alternative mechanisms of trait divergence are approximated here by stochastic process models. We use Brownian Motion (BM) models to approximate divergence under random drift or fluctuating selection; Ornstein-Uhlenbeck (OU) models in which two diverging lineages share an optima to approximate divergence under parallel and/or shared stabilizing selection; and OU models in which two diverging lineages are pulled toward alternative optima to approximate divergent adaptation (this last suite of models we refer to as "DA" models). In the DA models, the difference between

optima of paired lineages (psi) measures the extent of divergent selection. A more detailed explanation of the model variants used in this package is provided in the model_select man page.

MIXTURE MODELS:
Of course, different pairs in any given dataset may be diverging under different processes. We thus include mixture models (NEW TO VERSION 2.0) in which pairs in a dataset can diverge under one of two processes, and the proportion of pairs diverging under a given processes is estimated as a model parameter. The two basic mixture models are DA-OU and OU-BM, and the proportion parameter returned is that for the more complex model (i.e. Proportion DA in a DA-OU model and Proportion OU in an OU-BM model). A DA-BM model is also possible and is encoded in these functions, but including this model in comparative tests leads to higher error rates in model selection, and the model behaviour is often biologically unrealistic, so we recommend against its usage (tests to be released in supp.mat of Anderson et al. 2021 in review).

MODEL EXTENSIONS:
(1) Effect of continuous variables on model parameters.
Rates of evolution, patterns of divergent selection, and proportions of pairs diverging under a given process (in mixture models) may vary across continuous biological or ecological gradients such as latitude, elevation, and body mass. We follow the approach used in the EvoRAG package (Weir 2014), whereby model parameters are represented as linear functions of the continuous gradient. Model fitting requires, as input, the gradient value for each pair.
(2) Effect of categorical variables on model parameters.
Just as patterns of divergence can vary across continuous ecological gradients, they can also vary across categorical variables such as relative geographic range (i.e. allopatric versus sympatric), and different pollinator types. We model this variation by allowing parameters to vary among the user-defined categories. Tests of categorical models require as input a vector containing the category of each pair (coded as 0, 1, 2, for a maximum of 3 categories).
(3) Discrete temporal change in extent of divergent selection.
In models of divergent selection (DA models), lineages in a pair are pulled toward alternative adaptive optima following their initial divergence from an ancestor. Often one or both of these lineages will experience a secondary shift in their adaptive environment which changes the position of their optima. This manifests in the DA models as a discrete shift in psi, referred to as an "epoch shift". WAIT-TIME models (DA_wt, DA_wt_linear) estimate the timing of this shift as a wait time shared by all pairs in the dataset. BREAKPOINT models (DA_bp, DA_bp_ilnear) require users to provide a set of breakpoint values, which are the times after divergence at which a shift in psi is hypothesized to occur. These will be unique to pairs of different ages. Pairs in which no shift is expected to have occurred are assigned breakpoint value of zero. See the model_select description of DA_bp for more details.

DATA REQUIREMENTS:
diverge is designed to work with datasets of lineage-pair contrasts in a continuous trait. At minimum, datasets must contain two components for each lineage pair: (1) a measure of the difference between the two lineages in a trait, and (2) an estimate of time since the two lineages initially diverged. Extended versions of models require additional info on the gradient value or categorical value for each pair. Power and model performance increase with the number of lineage pairs in a dataset. Several functions in this package use the same arguments for required data. These include:

div - a vector of trait differences for each pair in the dataset. Required for all evolutionary models. These are calculated for each pair as abs(trait_val_lineage_2 - trait_val_lineage1). Raw values (i.e. not absolute values) can also be used but this must be noted by the user in the absolute argument of

most functions.

ages - a vector containing the age (i.e. estimated time since divergence) for each pair in the dataset.

me1 - measurement error (standard error of mean) in species 1s of each pair (variance/n-measurements)

me1 - measurement error (standard error of mean) in species 2s of each pair (variance/n-measurements)

GRAD - a vector containing the gradient position of each pair. This is the value of a continuous variable such as latitude or body size across which parameters are hypothesized to vary. Required for all models with the 'linear' suffix.

cats - a vector containing the category code (0, 1, or 2) for each pair in the dataset. Required when using the DA-cat model only.

breakpoint - a vector of breakpoint times for each pair in the dataset. These are the times after divergence at which a shift occurs in the psi parameter of a DA model. Required for DA_bp and DA_bp_linear. See model_select for details on how to calculate.

IMPORTANT NOTE:
div, ages, ms1, ms2, GRAD, cats, and breakpoint vectors must be aligned such that div[i], age[i], grad[i], cat[i], and breakpoint[i] represent values for the same i'th pair.

PARALLELIZATION:
Many functions in 'diverge' can be run in parallel across multiple cores through the 'parallel' logical argument. This functionality uses forking and is therefore unavailable on windows machines. All examples shown in function manuals are run in serial. We encourage users to insert parallel=TRUE in these commands where resources allow.

CUSTOMIZATION:
The code for several key functions has been lightly commented to aid users wishing to hack functions for custom use.

Author(s)

Sean A. S. Anderson and Jason T. Weir

Maintainer: Sean A.S. Anderson <sean.as.anderson@gmail.com>

`bootstrap_ci` *Estimate confidence intervals using bootstrap*

Description

Calculates 95 percent confidence intervals of parameter estimates through bootstrap resampling

Usage

```
bootstrap_ci(div, ages, me1=NULL, me2=NULL, GRAD = NULL, cats=NULL, breakpoint = NULL,
  domain = NULL, model, N, parallel = FALSE, cores = NULL, starting = NULL)
```

Arguments

div	Vector of trait divergences for a set of lineage pairs. Calculated for each pair as abs(trait_val_lineage_2 - trait_val_lineage1). Raw values (i.e. not absolute values) can also be used but must be noted by the user in the argument 'absolute'.
ages	Vector containing the age (i.e. estimated time since divergence) for each pair in the dataset. IMPORTANT: div, ages, GRAD, and breakpoint vectors must be aligned such that div[i] age[i] grad[i] and breakpoint[i] represent values for the same lineage pair.
me1	Vector containing the measurement error (standard error of mean) for species 1s of each pair (for one species = variance/No.measurements)
me2	Vector containing the measurement error (standard error of mean) for species 2s of each pair (for one species = variance/No.measurements)
GRAD	Vector containing the gradient position of each pair. This is the value of a continuous variable such as latitude or body size across which parameters are hypothesized to vary. Required for models with 'linear' suffix.
cats	Vector containing the category code (0, 1, or 2) for each pair (see package help page for details).
breakpoint	Vector of breakpoint times for each pair in the dataset. These are the times AFTER divergence at which a shift occurs in the psi parameter of DA model. Required for DA_bp and DA_bp_linear. See find_mle for details on how to calculate.
domain	Vector of length 2 defining the low and high ends of the gradient domain. Essentially identical to the 'xlim' argument in plotting functions. Required for models with 'linear' suffix.
model	Character string defining one of ten models of trait divergence (options: "BM_null", "BM_linear", "OU_null", "OU_linear", "DA_null", "DA_linear", "DA_wt", "DA_bp", "DA_wt_linear", "DA_bp_linear"). See find_mle for model descriptions.
N	Number of bootstrap replicates to create (use integers only).
parallel	Logical indicating whether likelihood searches should be conducted in parallel across multiple cores. Not available on windows machines. Defaults to FALSE.
cores	If parallel=TRUE, the number of cores on which to run the function. Defaults to all virtual cores.
starting	Optional matrix of customized parameter values from which to launch likelihood searches. Must match the structure required for the chosen model. See the model descriptions in find_mle for details. If starting=NULL, default starting values are used.

Details

WARNING: estimating parameters for a large number of bootstrap datasets is computationally intensive. Run times can be exceedingly high for the breakpoint models – in particular DA_bp and especially DA_bp_linear, DA_wt, DA_wt_linear. When running breakpoint models, we strongly recommend using the parallel option where resources allow, ideally on a multi-core server.

Value

A named list of length 4 containing [[1]] the model chosen, [[2]] all parameter estimates, [[3]], the number of replicates, and [[4]] summary statistics for parameter estimation.

Author(s)

Sean A.S. Anderson and Jason T. Weir

References

Efron, B. and Tibshirani, R. (1986). The Boostrap Method for standard errors, confidence intervals, and other measures of statistical accuracy. Statistical Science, Vol 1., No. 1, pp 1-35.

Examples

```
## Estimate confidence intervals for sigma^2 under a single-process BM model
## 10 bootstrap replicates

# simulate dataset
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
sig2 = 0.2
sis_div = simulate_div(model="BM_null", ages=ages, pars=sig2)

# Run bootstrap_ci
N=100
res = bootstrap_ci(div=sis_div, ages=ages, model=("BM_null"), N=N)
res


## Estimate confidence intervals for all parameters under a DA_linear model
## 10 bootstrap replicates.

# simulate dataset under DA_linear
# pairs are evenly distributed across a 0-60 degree latitudinal gradient
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
grad_cats = rep(c(0, 15, 30, 45, 60), 30)
grad=c(rep(grad_cats[1], 30), rep(grad_cats[2],30), rep(grad_cats[3],30),
  rep(grad_cats[4],30), rep(grad_cats[5],30))
alpha = 0.8
sig2 = 0.2
psi_sl = -0.01
psi_int = 2
sis_div = simulate_div(model="DA_linear", ages=ages, pars=c(alpha, sig2, psi_sl, psi_int),
  GRAD=grad)

# Run bootstrap_ci
N = 10
res = bootstrap_ci(div=sis_div, ages=ages, GRAD=grad, domain=c(0,60),
model=("DA_linear"), N=N)
res

## Estimate confidence intervals for psi1 and psi2 under a 2-category DA_cat
```

```
## model given 10 bootstrap replicates.
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
grad_cats = rep(c(0, 15, 30, 45, 60), 30)
cats = c(rep(0, 75), rep(1, 75))
alpha = 0.8
sig2 = 0.2
psi1 = 0.5
psi2 = 1
N = 2
sis_div = simulate_div(model="DA_cat", ages=ages, pars=c(alpha, sig2, psi1, psi2), cats=cats)
res = bootstrap_ci(div=sis_div, ages=ages, cats=cats, model=("DA_cat"), N=N, parallel=FALSE)
res
```

---

diverge-data *Example Tree*

Description

A phylogenetic tree of birds of the world.

Usage

```
data(all_birds)
```

Details

all_birds is the maximum clade credibility tree of 9982 avian species (and 3301 sister pairs) from Pulido-Santacruz and Weir (2016). Branch lengths are time-calibrated in millions of years.

Value

A phylo object.

Author(s)

Sean A. S. Anderson

References

Pulido-Santacruz, P. and J. T. Weir. 2016. Extinctioin as a driver of avian latitudinal diversity gradients. Evolution 70: 860-872.

Examples

```
str(all_birds)
```

`expected_val` *Expected value through time*

Description

Calculate and plot expected value of trait divergence through time

Usage

```
expected_val(model, sig2, alpha = NULL, psi = NULL, time_span = c(0, 10),
  quantile = FALSE, plot = FALSE, labels = TRUE, bm_col = "darkgoldenrod1",
  ou_col = "firebrick2", da_col = "navy", exval_lwd = 5, ylim = NULL, ...)
```

Arguments

`model`	Name of the model whose expected values are to be calculated. Must be one of "BM_null", "OU_null", "DA_null", "DA_OU", "DA_BM", or "OU_BM".
`sig2`	Value of the sigma-squared parameter. Required for all models.
`alpha`	Value of the alpha parameter. Required for all OU and DA models.
`psi`	Value of psi parameter. Required for DA models. NOTE: if using a breakpoint or wait time model, this is the psi for the first evolutionary epoch
`time_span`	The time after initial divergence at which to calculate the expected value. If time_span is a vector lf length 2, the expectation is calculated continuously between the two times. If time_span is any other length, the expected value is calculated only at the given times. Defaults to c(0,10)
`quantile`	Logical indicating whether to additionally calculate (and optionally plot) the expected quantiles (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95) of trait divergence. Defaults to FALSE.
`plot`	Logical indicating whether to plot the expected values through time. If quantile=TRUE, then quantile curves are also plotted. Defaults to FALSE.
`labels`	Logical indicating whether x and y axes are to be labelled.
`bm_col`	Color of the line (and shaded if quantile=TRUE) area showing expected value through time under a BM model.
`ou_col`	Color of the line (and shaded if quantile=TRUE) area showing expected value through time under an OU model.
`da_col`	Color of the line (and shaded if quantile=TRUE) area showing expected value through time under an OU model.
`exval_lwd`	Width in which the line showing the expected value through time should be plotted. Defaults to 5.
`ylim`	Custom y-axis for graph in the form 'ylim = c(miny, maxy)'
`...`	Additional arguments to be passed to the 'plot' function for custom visualization.

Details

Calculates and plots the expected value of continuous trait divergence under different models of trait evolution. Time begins when lineages initially depart form a common ancestor. If users provide upper and lower bounds for time_span, then that span is divided into 10,001 time steps at which expectations are calculated to approximate a continuous tracking of expectation through time. Expectations for the 2.5th-97.5th quantile can also be estimated and 95 percent confidence intervals plotted to visualize how the distribution of trait divergence changes.

IMPORTANT: Not all models can be selected for this function. The expected value of divergence under one of the "linear" models, for instance, would be different for each position on a continuous gradient and thus can't be calculated or plotted in this way. Acceptable models are ("BM_null", "OU_null", "DA_null", "DA_OU", "DA_BM", "OU_BM").

Value

Returns either (1) a matrix of 2 columns in which col1 = time and col2 = the expected trait divergence at that time, or (2) a matrix of 12 columns that additionally contains the expectation for the 10th-95th quantile of trait divergence at each time (if quantiles=TRUE is indicated in the function call). The number of rows of the output matrix equals the length of time_span EXCEPT if length(time_span) == 2, in which case the output matrix has 10,001 rows.

Author(s)

Sean A.S. Anderson and Jason T. Weir

Examples

```
# Ex. 1. Calculate the expected value of divergence after 5my under DA_null
sig2 = 0.2
alpha = 0.8
psi = 0.9
exval = expected_val(model="DA_null", sig2=sig2, alpha=alpha, psi=psi, time_span=5)

# Ex. 2. calculate and plot expected trait divergence and expected quantiles through time over 8my
# under a DA_null model
sig2 = 0.2
alpha = 0.8
psi = 0.3

exval = expected_val(model="DA_null", sig2=sig2, alpha=alpha, psi=psi, time_span=c(0,8),
  quantile=TRUE, plot=TRUE)


# Ex. 3. same as above but with customized graphical parameters
sig2 = 0.2
alpha = 0.8
psi = 0.3

exval = expected_val("DA_null", sig2=sig2, alpha=alpha, psi=psi, time_span=c(0,8), quantile=TRUE,
 plot=TRUE, da_col="green", exval_lwd = 4, ylim = c(0, 3), axes=FALSE, labels=FALSE)
box()
```

```
axis(1, labels=NA)
axis(1, lwd = 0, line = -0.6)
axis(2, labels = NA)
axis(2, lwd = 0, line = -0.6)
title(line = 1.9, xlab = "Custom X Axis Title")
title(line = 1.9, ylab = "Custom Y AXis Title")


# Ex. 4. calculate and plot expected trait divergence and expected quantiles through time over 8my
# under a DA-OU mixture model
sig2 = 0.05
alpha = 0.8
psi = 1

exval = expected_val(model="DA_OU", sig2=sig2, alpha=alpha, psi=psi, time_span=c(0,8),
  quantile=TRUE, plot=TRUE)
# add a legend
legend("topleft", legend=c("ExVal DA", "ExVal OU"), lwd=2, col=c("navy", "firebrick2"),
    bty="n", cex=0.8)
```

`extract_sisters` *Extract sister pairs*

Description

Find the names and divergence times for all sister pairs in phylogeny

Usage

```
extract_sisters(tree, sis_age=FALSE, mol_clock=NULL, crown_age=NULL)
```

Arguments

`tree`	A phylo object. Branch lengths must represent time or genetic distances with known molecular clock.
`sis_age`	Logical indicating whether or not sister pair ages are to be estimated.
`mol_clock`	The estimated constant rate of sequence evolution per million years.
`crown_age`	The estimated crown age in millions of years.

Details

extract_sisters takes a phylo object and identifies all sister pairs by name. If sis_age=TRUE, then extract_sisters also estimates the age of each pair (in Ma) based on branch lengths and either 1) a given (constant) molecular clock, or 2) an estimate for the crown age. If no molecular clock rate or crown age is supplied by the user, the branch lengths are assumed to represent time in Ma. CAUTION: age estimates will be returned for any tree that contains branch lengths, so it's up to the user to ensure that the tree is ultrametric.

Value

Returns a data frame in which each row corresponds to a sister pair and contains the taxon names for the two lineages, and if sis_age=TRUE, then the estimated divergence time (i.e. age) for that pair.

Author(s)

Sean A.S. Anderson

Examples

```
# Note: this function automatically generates a warning to remind users that it is
# their responsibility to ensure the tree is ultrametric. This warning can be suppressed
# with suppressWarnings

# extract avian sister pairs and calculate ages based on branch lengths in Ma
bird_sis = extract_sisters(tree=all_birds, sis_age=TRUE)


# extract avian sister pairs and calculate ages based on molecular clock of 1.1
# note: the actual branch lengths are in Ma, this is just a toy example
bird_sis = extract_sisters(tree=all_birds, sis_age=TRUE, mol_clock = 1.1)

# extract avian sister pairs and calculate ages based on a crown age of 30 Ma
# note: the actual branch lengths are in Ma, this is just a toy example
bird_sis = extract_sisters(tree=all_birds, sis_age=TRUE, crown_age=30)
```

---

find_mle *Find Maximum Likelihood Estimates*

Description

Find maximum likelihood parameter estimates for an evolutionary model given data

Usage

```
find_mle(model, p_starting = NULL, div, ages, me1=NULL, me2=NULL, GRAD = NULL, cats,
  breakpoint = NULL, domain = NULL, absolute = TRUE, parallel = FALSE, cores = NULL)
```

Arguments

model	Character string defining one of 11 models of trait divergence (options: "BM_null", "BM_linear", "OU_null", "OU_linear", "DA_null", "DA_linear", "DA_cat", "DA_wt", "DA_bp", "DA_wt_linear", "DA_bp_linear"). See Details for info on each model.
p_starting	Optional matrix of customized parameter values from which to launch likelihood searches. Must match the structure required for the chosen model. See model descriptions in the model_select man page for details on providing a custom starting matrix.

div	Vector of trait divergences for a set of lineage pairs. Calculated for each pair as abs(trait_val_lineage_2 - trait_val_lineage1). Raw values (i.e. not absolute values) can also be used but must be noted by the user in the argument 'absolute'.
ages	Vector containing the age (i.e. estimated time since divergence) for each pair in the dataset. IMPORTANT: div, ages, GRAD, and breakpoint vectors must be aligned such that div[i] age[i] grad[i] and breakpoint[i] represent values for the same lineage pair.
me1	Vector containing the measurement error (standard error of mean) for species 1s of each pair (for one species = variance/No.measurements)
me2	Vector containing the measurement error (standard error of mean) for species 2s of each pair (for one species = variance/No.measurements)
GRAD	Vector containing the gradient position of each pair. This is the value of a continuous variable such as latitude or body size across which parameters are hypothesized to vary. Required for models with 'linear' suffix.
cats	Vector containing the category code (0, 1, or 2) for each pair
breakpoint	Vector of breakpoint times for each pair in the dataset. These are the times AFTER divergence at which a shift occurs in the psi parameter of DA model. Required for DA_bp and DA_bp_linear. See find_mle for details on how to calculate.
domain	Vector of length 2 defining the low and high ends of the gradient domain. Essentially identical to the 'xlim' argument in plotting functions. Required for models with 'linear' suffix.
absolute	Logical indicating whether 'div' represents absolute value of trait divergence or the raw divergence values.
parallel	Logical indicating whether likelihood searches should be conducted in parallel across multiple cores. Not available on windows machines. Defaults to FALSE.
cores	If parallel=TRUE, the number of cores on which to run the function. Defaults to all virtual cores.

Details

This function can be used to find the maximum likelihood parameter estimates for one of ten evolutionary models given a lineage-pair dataset.'find_mle' uses 'nlminb' to optimize the likelihood function of the selected model. Since optimization algorithms can often get stuck on local optima, find_mle works by feeding the optimizer a large set of starting parameter values and ranking the likelihoods of estimates that are returned by each. Many sets of starting parameters should converge on the MLE, and find_mle returns the output of nlminb given one of these optimal starting sets.

RUN TIME CONSIDERATIONS
The set of starting parameter values grows rapidly with parameter number, so calculating MLEs in the most complex models (esp DA_wt, DA_wt_linear) can take a few minutes. We therefore recommend using the parallel=TRUE option when using any model more complex than DA_null. Users can define their own starting parameter sets but should keep in mind this tradeoff between run time and the breadth of parameter space through which to search.

Value

A list containing maximum likelihood parameter estimates, the maximumum likelihood, and likelihood search diagnostics. This is the output of nlminb as returned from an optimal starting parameter set.

Author(s)

Sean A.S. Anderson and Jason T. Weir

Examples

```
## Find the maximum likelihood and parameters for a DA_linear model
# assume an elevational gradient from 0-1000m

# simulate a dataset
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
grad_cats = rep(c(0, 250, 500, 750, 1000), 30)
grad=c(rep(grad_cats[1], 30), rep(grad_cats[2],30), rep(grad_cats[3],30),
  rep(grad_cats[4],30), rep(grad_cats[5],30))
alpha = 0.8
sig2 = 0.2
psi_sl = -0.01
psi_int = 2
sis_div = simulate_div(model="DA_linear", ages=ages,
  pars=c(alpha, sig2, psi_sl, psi_int), GRAD=grad)

# find max. likelihood and estimate parameters
res = find_mle(model="DA_linear", div=sis_div, ages=ages, GRAD=grad, domain=c(0,1000))
```

---

likelihood_func *Functions for negative log likelihood*

Description

Calculate negative log likelihood for one of fourteen trait divergence models given a dataset and user-defined parameters.

Details

This function will typically will not be called directly by the user. Typing 'likelihood_func' may, however, be of interest to those wishing see/verify how we coded up the likelihood functions for each evolutionary model. The model returns the negative log-likelihood of a chosen model based on a dataset and a set of parameters. Likelihood_func is called by 'find_mle', which uses the non-linear search algorithm 'nlminb' to minimize its output.

`model_error_rate` *Calculate error rate in model selection*

Description

Estimate error rates in model selection using replicate datasets simulated under a known model

Usage

```
model_error_rate(datasets, ages, sim_model, alternatives, type, me1=NULL, me2=NULL,
  GRAD = NULL, cats=NULL, breakpoint = NULL, domain = NULL, threshold = 0,
  parallel = FALSE, cores = NULL)
```

Arguments

`datasets`	List containing replicate datasets of trait divergence simulated under a single model
`ages`	Vector of lineage-pair ages for which trait divergence was simulated to create the 'datasets'
`sim_model`	Character vector naming the model under which trait divergence was simulated (options: "BM_null", "BM_linear", "OU_null", "OU_linear", "DA_null", "DA_linear", "DA_wt", "DA_bp", "DA_wt_linear", "DA_bp_linear"). See find_mle for info on each model.
`alternatives`	Character vector naming alternative models to be included in the model comparison
`type`	Numeric "1" or "2". Indicates the whether test is for type 1 error rate (i.e. false rejection of true null) or type 2 error rate (i.e. failure to reject incorrect null).
`me1`	Vector containing the measurement error (standard error of mean) for species 1s of each pair (for one species = variance/No.measurements)
`me2`	Vector containing the measurement error (standard error of mean) for species 2s of each pair (for one species = variance/No.measurements)
`GRAD`	Vector of gradient values. Required when any model with linear gradients is included in the tests (see find_mle). Must match length of 'ages' vector.
`cats`	Vector containing the category code (0, 1, or 2) for each pair (see package help page for details).
`breakpoint`	Vector of breakpoints. Required when DA_bp or DA_bp_linear is included in the tests. Must match length of 'ages' vector.
`domain`	Vector of length 2 defining the low and high ends of the gradient domain. Essentially identical to the 'xlim' argument in plotting functions. Required when any models with 'linear' suffix is included in the tests.
`threshold`	Numeric value for the delta_AICc threshold corresponding to an error in model selection (see details). Defaults to zero.
`parallel`	Logical indicating whether likelihood searches should be conducted in parallel across multiple cores. Not available on windows machines. Defaults to FALSE.

cores — If parallel=TRUE, the number of cores on which to run the function. Defaults to all virtual cores.

Details

'model_error_rate' calculates the proportion of replicate runs of model_select that return an incorrect model based on a user-defined AICc threshold. A commonly-used threshold is 2, and the default threshold is zero. Since users must know which of the 11 evolutionary models produced the replicate sets of trait divergence, the function is typically used in conjunction with simulate_div.

Users must designate whether type 1 or type 2 error rate is to be calculated via the 'type' argument. If type = 1, sim_model must be the simplest model against which more general alternatives are compared. If type = 2, sim_model must be a more complex model against which a simpler null alternative is compared. Type 1 error rate is calculated as the proportion of model_select runs for which delta_AICc of the sim model exceeds the threshold. Type 2 error rate is calculated as the proportion of model_select runs for which the difference between AICc of the sim_model and that of the best null model is greater than the threshold*(-1).

TIME CONSIDERATIONS
Warning: this function runs model selection tests on replicate datasets. Model selection can be time intensive in its own right, especially when more complex models are involved. Running model selection trials across a large number of replicate datasets can therefore be extremely time intensive. For example, calculating error rate based on 1000 datasets when the models include DA_wt, DA_bp, DA_wt_linear, and DA_bp_linear will often require parallel runs on a multi-core server over several hours

Value

The error rate in decimal form.

Author(s)

Sean A.S. Anderson

Examples

```
## Calculate type I error rate for BM_null versus BM_linear and OU_linear
# simulate 10 replicate sets under BM_null
sig2 = 0.2
N = 10
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
dsets = simulate_div(model="BM_null", ages=ages, pars=sig2, N=N)

# generate a continuous gradient (in this case from 0-60 degrees latitude)
grad_cats = rep(c(0, 15, 30, 45, 60), 30)
grad=c(rep(grad_cats[1], 30), rep(grad_cats[2],30), rep(grad_cats[3],30),
  rep(grad_cats[4],30), rep(grad_cats[5],30))

# calculate type I error rate given the dataset and gradient
err_rateT1 = model_error_rate(datasets=dsets, ages=ages, sim_model="BM_null",
  alternatives=c("BM_linear","OU_linear"), type=1, GRAD=grad,
  domain=c(0,60), threshold=2)
```

```
err_rateT1

## Calculate type II error rate of DA_linear v DA_null
# simulate 10 replicate sets under DA_linear
alpha = 0.8
sig2 = 0.2
psi_sl = -0.01
psi_int = 2
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
grad_cats = rep(c(0, 15, 30, 45, 60), 30)
grad=c(rep(grad_cats[1], 30), rep(grad_cats[2],30), rep(grad_cats[3],30),
  rep(grad_cats[4],30), rep(grad_cats[5],30))
dsets = simulate_div(model="DA_linear", ages=ages, pars=c(alpha, sig2, psi_sl, psi_int),
  N=10, GRAD=grad)

# calculate type II error rate
# NOTE: this can take ~1-2 mins
err_rateT2 = model_error_rate(datasets=dsets, ages=ages, sim_model="DA_linear",
  alternatives="DA_null", type=2, GRAD=grad, domain=c(0,60), threshold=2)
err_rateT2
```

`model_generator` *Generate parameter~gradient linear models*

Description

Generate a set of parameter~gradient linear models over the domain of a continuous gradient

Usage

```
model_generator(domain, par_low, par_high)
```

Arguments

`domain`	Vector of length 2 defining the low and high ends of the gradient domain. Essentially identical to the 'xlim' argument in plotting functions. E.g. domain = c(0,60) may be suitable when investigating a latitudinal gradient.
`par_low`	Parameter value or vector of parameter values at lowest end of domain
`par_high`	Parameter value or vector of parameter values at hightest end of domain. NOTE: linear models returned are restricted such that the parameter is never more extreme than the values given in par_high and par_low across the domain.

Details

An internal utility function called by param_grid that may nonetheless have value to some users. This function generates a set of linear models for a parameter value over a continuous gradient. The slopes and intercepts generated are ultimately used as starting parameter values in the likelihood search function 'find_mle'.

model_generator works by taking a user-defined parameter value or a set of values at opposite ends of a user-defined domain and calculating the slope of each straight line between them. The slopes thus generated create lines such that parameter values across the domain are never more extreme than the most extreme values supplied by the user. This helps avoid issues like nonsensical negative parameters. The number of models returned = length(par_low)*length(par_high).

Value

Returns a matrix of slope and intercept values where each row is the slope and intercept of one linear model.

Author(s)

Sean A.S. Anderson

Examples

```
# Generate and plot 100 linear models for a parameter across a latitudinal gradient.
# Param values must be between 1 and 10 at the equator and between 0 and 9 at 60 degrees.

mods = model_generator(domain=c(0,60), par_low=seq(1, 10), par_high=seq(0,9))
curve(expr=(y=mods[1,1]*x + mods[1,2]), ylim=c(0,10), xlim=c(0,60), ylab="Parmam Value",
  xlab="Abs. Latitude")
for(i in 2:nrow(mods)) curve(expr=(y=mods[i,1]*x + mods[i,2]), add=TRUE)
```

---

model_select *Select the best fit model of trait divergence*

Description

Compare likelihood and AICc support among several models of sister pair trait divergence given a sister pair dataset.

Usage

```
model_select(div, ages, me1 = NULL, me2 = NULL, GRAD =NULL, cats=NULL, breakpoint=NULL,
   domain=NULL, models, starting = NULL, absolute=TRUE, parallel=FALSE, cores=NULL)
```

Arguments

div	Vector of trait divergences for a set of lineage pairs. Calculated for each pair as abs(trait_val_lineage_2 - trait_val_lineage1). Raw values (i.e. not absolute values) can also be used but must be noted by the user in the argument 'absolute'.
ages	Vector containing the age (i.e. estimated time since divergence) for each pair in the dataset. IMPORTANT: div, ages, GRAD, and breakpoint vectors must be aligned such that div[i] age[i] GRAD[i] and breakpoint[i] represent values for the same lineage pair.

me1	Vector containing the measurement error (standard error of mean) for species 1s of each pair (for one species = variance/No.measurements)
me2	Vector containing the measurement error (standard error of mean) for species 2s of each pair (for one species = variance/No.measurements)
GRAD	Vector containing the gradient position of each pair. This is the value of a continuous variable such as latitude or body size across which parameters are hypothesized to vary. Required for models with 'linear' suffix.
cats	Vector containing the category code (0, 1, or 2) for each pair (see package help page for details).
breakpoint	Vector of breakpoint times for each pair in the dataset. These are the times AFTER divergence at which a shift occurs in the psi parameter of DA model. Required for DA_bp and DA_bp_linear. See find_mle for details on how to calculate.
domain	Vector of length 2 defining the low and high ends of the gradient domain. Essentially identical to the 'xlim' argument in plotting functions. Required for models with 'linear' suffix.
models	Character vector naming models for which to compare likelihood estimates and AICc values (options: "BM_null", "BM_linear", "BM_cat", "OU_null", "OU_linear", "OU_linear_sig", "OU_cat", "DA_null", "DA_linear", "DA_cat", "DA_wt", "DA_bp", "DA_wt_linear", "DA_bp_linear", "DA_OU", "DA_OU_linear", "DA_OU_cat", "DA_BM", "DA_BM_linear", "DA_BM_cat", "OU_BM", "OU_BM_linear", "OU_BM_cat"). See find_mle for model descriptions.
starting	Optional argument for providing customized set of parameters from which to launch likelihood searches. If only one model is specified in "models", then 'starting' should be a matrix in which each row is a complete set of starting parameters for that given model (see parameter requirements for each model below). If more than one model is specific in "models", then "starting" must be a list in which each element is a matrix of customized parameter values. The list must be ordered such that the nth element contains the starting parameter values for the nth model named in the "models" argument. The nth elements in the list must match the structure required for starting parameters for the nth model in "models". If starting=NULL, default starting values are used.
absolute	Logical indicating whether 'div' represents absolute value of trait divergence or the raw divergence values.
parallel	Logical indicating whether likelihood searches should be conducted in parallel across multiple cores. Not available on windows machines. Defaults to FALSE.
cores	If parallel=TRUE, the number of cores on which to run the function. Defaults to all virtual cores.

Details

OVERVIEW
model_select estimates parameters and compares the fit of up to 20 evolutionary models to lineage-pair datasets based on likelihood, AIC, and AICc. It is the key function in this package for most users. We strongly recommend that users define the models they want to fit to the data (i.e. only

those models that are relevant to distinguishing among hypotheses of interest) rather than simply fitting all models (the default option).

In most cases, if you find that a model fails to fit (e.g. NaN is being returned for likelihood for one or more models), then the default starting parameters are not well-suited to your dataset and you will need to make a list of custom starting parameters (see 'starting' argument above).

Note on troubleshooting: if models fail to converge or return nonsensical results, it is usually the case that the default starting parameter values are no good (i.e. the true parameter values are well outside the default values). Start your troubleshooting by creating custom starting parameter matrices for the different models (see 'starting' argument above).

RUN TIME CONSIDERATIONS
model_select calls the underlying function find_mle, which launches likelihood searches from a large set of starting parameter values to avoid getting stuck on local optima (see find_mle for more details). This set of starting parameter values grows rapidly with parameter number, so model comparisons that include the most complex evolutionary models (esp DA_wt, DA_wt_linear) can take several minutes. We recommend using the parallel=TRUE option where resources allow when including any model more complex than DA_null. Users can define their own starting parameter sets but should keep in mind this tradeoff between run time and the breadth of parameter space through which to search.

MODELS OF EVOLUTIONARY TRAIT DIVERGENCE
Users can select up to twenty evolutionary models that differ in how a continuous trait in two lineages "i" and "j" evolves after their initial departure from a common ancestor at T=0. All models require div and age vectors, but some models require additional data. The models are:

(1) BM_null Description: The trait in both lineages evolves under Brownian Motion (BM) processes. Parameter: sig2 (i.e. sigma^2, the dispersion parameter of a BM process; this parameter the same in both lineages). Optional custom starting matrix: A user-defined starting matrix must be a 1 column matrix containing sig2 starting values.

(2) BM_linear Description: Same as BM_null but sig2 varies as a linear function of a continuous variable such as latitude, elevation, or body size. Additional Data Required: values of the continuous variable for each pair must be provided in the 'GRAD' vector. Parameters: sig2_slope (slope of sig2~gradient relationship), sig2_int (intercept of sig2~gradient relationship). Optional custom starting matrix: A user-defined starting matrix must be 2 column matrix with col1 = sig2_slope and col2 = sig2_int.

(3) BM_cat Description: similar to BM_null but sig2 varies among discrete categories to which different pairs belong, such as allopatric versus sympatric ranges, or different pollinator type. Additional Data Required: The numeric category code of each pair must be provided in the 'cats' vector. A value must be provided for every pair (i.e. the cats vector must be as long as the data vectors). For example, in a 2-category dataset (e.g. a dataset containing allopatric and sympatric pairs), all pairs of one category can be assigned a 'cats' value of 0, and all pairs of the other category are assigned a 'cats' value of 1. If a third category existed, those pairs would be assigned a 'cats' value of 2, and so on with additional categories. Parameters = sig2_1, sig2_2, ... sig2_n, where n is the number of categories. Optional custom starting matrix: A user-defined p_starting matrix must be an n column matrix where col1 = sig2_1, col2 = sig2_2, ..., coln= sig2_n.

(4) OU_null Description: The trait in both lineages evolves under Ornstein-Uhlenbeck (OU) processes. Parameters: alpha (the OU constraint parameter), sig2 - both parameters are shared by the two lineages Optional custom starting matrix: A user-defined p_starting matrix must be 2 column matrix in the order col1 = alpha and col2 = sig2.

(5) OU_linear Description: Same as OU_null but alpha varies as a linear function of a continuous variable such as latitude, elevation, or body size. Additional Data Required: Values of the continuous variable for each pair must be provided in the 'GRAD' vector. Parameters: alpha_int (intercept of alpha~gradient relationship), alpha_slope (slope of alpha~gradient relationship), sig2. Optional custom starting matrix: A user-defined p_starting matrix must be 3 column matrix where col1 = alpha_int, col2 = alpha_slope, and col3 = sig2.

(6) OU_linear_sig Description: Same as OU_linear but sig2, not alpha, varies as a linear function of a continuous variable such as latitude, elevation, or body size. Additional Data Required: Values of the continuous variable for each pair must be provided in the 'GRAD' vector. Parameters: alpha_int (intercept of alpha~gradient relationship), alpha_slope (slope of alpha~gradient relationship), sig2. Optional custom starting matrix: A user-defined p_starting matrix must be 3 column matrix where col1 = alpha, col2 = sig2_slope, and col3 = sig2_int.

(7) OU_cat Description: similar to BM_cat but alpha (not sig2) varies among discrete categories. Additional Data Required: The numeric category code of each pair must be provided in the 'cats' vector. A value must be provided for every pair (i.e. the cats vector must be as long as the data vectors). For example, in a 2-category dataset (e.g. a dataset containing allopatric and sympatric pairs), all pairs of one category can be assigned a 'cats' value of 0, and all pairs of the other category are assigned a 'cats' value of 1. If a third category existed, those pairs would be assigned a 'cats' value of 2, and so on with additional categories. Parameters = sig2, alpha_1, alpha_2, ... alpha_n, where n is the number of categories. Optional custom starting matrix: A user-defined p_starting matrix must be an n+1 column matrix where col1 = sig2, col2 = alpha_1, col3 = alpha_2, ..., col(n+1)= alpha_n.

(8) DA_null Description: Traits evolve under independent OU processes in lineages i and j after they depart from an ancestor, and these processes differ only in the value of their optima (see Anderson and Weir, 2020, 'Inferring speciation drivers from functional trait divergence'. Am Nat, 196, 429-442.). Parameters: alpha, sig2, and psi (the distance between optima of independent OU processes). Optional custom starting matrix: A user-defined starting matrix must be 3 column matrix where col1 = alpha, col2 = sig2, and col3 = psi.

(9) DA_linear Description: Same as DA_null but psi varies as a linear function of a continuous variable such as latitude, elevation, or body size. Additional Data Required: Values of the continuous variable for each pair must be provided in the 'GRAD' vector. Parameters: alpha, sig2, psi_slope (slope of psi~gradient relationship), and psi_int (intercept of psi~gradient relationship) Optional custom starting matrix: A user-defined starting matrix must be 4 column matrix where col1 = alpha, col2 = sig2, col3 = psi_sl, and col4=psi_int

(10) DA_cat Description: A DA model in which psi varies among discrete categories. Additional Data Required: The numeric category code of each pair must be provided in the 'cats' vector. A value must be provided for every pair (i.e. the cats vector must be as long as the data vectors). For example, in a 2-category dataset (e.g. a dataset containing allopatric and sympatric pairs), all pairs of one category can be assigned a 'cats' value of 0, and all pairs of the other category are assigned a 'cats' value of 1. If a third category existed, those pairs would be assigned a 'cats' value of 2, and so on with additional categories. Parameters = alpha, sig2, psi1 (psi for variable category 1), psi2 (psi for variable category 2), ..., psi_n (psi for variable category n). Optional custom starting matrix: A user-defined p_starting matrix must be a n+2 column matrix where col1 = alpha, col2 = sig2, col3 = psi1, col4 = psi2, ..., col(n+2) = psi_n.

(11) DA_OU Description: mixture model in which some proportion of pairs are diverging under a DA process while the rest diverge under an OU process. Parameters: alpha, sig2, psi, and 'prop' (the proportion of pairs diverging under a DA process, with 1-prop being the proportion of pairs

diverging under the OU process) Optional custom starting matrix: A user-defined starting matrix must be a 4 column matrix where col1=alpha, col2=sig2, col3=psi, and col4=prop

(12) DA_OU_linear Description: mixture model in which the proportion of pairs diverging under a DA process (i.e. 'prop' parameter) varies as a linear function of a continuous variable such as latitude, elevation, or body size. Additional Data Required: Values of the continuous variable for each pair must be provided in the 'GRAD' vector. Parameters: alpha, sig2, psi, prop_slope, and prop_int Optional custom starting matrix: A user-defined starting matrix must be a 5 column matrix where col1=alpha, col2=sig2, col3=psi, col4=prop_slope, and col5=prop_int

(13) DA_OU_cat Description: similar to other categorical models but the parameter that varies among categories is 'prop', the proportion of pairs diverging under a DA process (where 1-prop is the proportion diverging under the OU process) Additional Data Required: Values of the category variable for each pair must be provided in the 'cat' vector. For example, in a 2-category dataset (e.g. a dataset containing allopatric and sympatric pairs), all pairs of one category can be assigned a 'cats' value of 0, and all pairs of the other category are assigned a 'cats' value of 1. If a third category existed, those pairs would be assigned a 'cats' value of 2, and so on with additional categories. Parameters = alpha, sig2, psi, prop_1, prop_2, ..., prop_n (for variable category n). Optional custom starting matrix: A user-defined p_starting matrix must be a n+3 column matrix where col1 = alpha, col2 = sig2, col3 = psi, col4 = prop_1, col5=prop_2, ..., col(n+3) = prop_n.

(14) OU_BM Description: mixture model in which some proportion of pairs ('prop' parameter) are diverging under an OU process while the rest diverge under a BM process. Parameters: alpha, sig2, and prop (the proportion of pairs diverging under an OU process, with 1-prop being the proportion of pairs diverging under the BM process) Optional custom starting matrix: A user-defined starting matrix must be a 3 column matrix where col1=alpha, col2=sig2, and col3=prop

(15) OU_BM_linear Description: mixture model in which the proportion of pairs diverging under an OU process (i.e. 'prop' parameter) varies as a linear function of a continuous variable such as latitude, elevation, or body size. Additional Data Required: Values of the continuous variable for each pair must be provided in the 'GRAD' vector. Parameters: alpha, sig2, prop_slope, and prop_int Optional custom starting matrix: A user-defined starting matrix must be a 4 column matrix where col1=alpha, col2=sig2, col3=prop_slope, and col4=prop_int

(16) OU_BM_cat Description: similar to other categorical models but the parameter that varies among categories is 'prop', the proportion of pairs diverging under an OU process (where 1-prop is the proportion diverging under the OU process) Additional Data Required: Values of the category variable for each pair must be provided in the 'cat' vector. For example, in a 2-category dataset (e.g. a dataset containing allopatric and sympatric pairs), all pairs of one category can be assigned a 'cats' value of 0, and all pairs of the other category are assigned a 'cats' value of 1. If a third category existed, those pairs would be assigned a 'cats' value of 2, and so on with additional categories. Parameters = alpha, sig2, prop_1, prop_2, ..., prop_n (for variable category n). Optional custom starting matrix: A user-defined p_starting matrix must be a n+2 column matrix where col1 = alpha, col2 = sig2, col3 = prop_1, col4=prop_2, ..., col(n+2) = prop_n.

(17) DA_wt Description: Like DA_null but with a discrete shift in psi that occurs after a wait time shared by all pairs in the dataset. DA_wt estimates this wait time based on divergence levels of pairs of different ages, so datasets should contain many pairs from a broad range of ages. Parameters: alpha, sig2, psi1 (distance between optima prior to the wait time), psi2 (distance between optima after the wait time), and wt (the wait time to a shift in psi; the time after initial divergence from a common ancestor at which a shift occurs). Optional custom starting matrix: A user-defined p_starting matrix must be 5 column matrix where col1 = alpha, col2 = sig2, col3 = psi1, col4=psi2, and col5=wt.

(18) DA_bp Description: Like DA_null but with a discrete shift in psi that occurs after some "breakpoint" time (bp) known to the user. Differs from DA_wt in two ways: (1) the timing of the discrete shift isn't isn't estimated as a parameter; it is instead provided by the user in the breakpoint vector that contains a bp value for each pair, and (2) the timing of the discrete shift is not shared by all pairs. Additional Data Required: Values of bp for each pair must be provided in the breakpoint vector. Parameters: alpha, sig2, psi1, psi2 Optional custom starting matrix: A user-defined p_starting matrix must be 4 column matrix where col1 = alpha, col2 = sig2, col3 = psi1, and col4=psi2 Additional Notes: This model is useful for testing hypotheses related to dated biogeographic events. For example, consider a set of lineage pairs with diverge times between 3 and 7my. If a river formed 1mya and isolated some of these already-diverging pairs on either side, we might hypothesize that this caused a shift in psi for those pairs, and a breakpoint model can test this. Bp is then calculated as bp = AgeOfPair - 1my for each pair divided by the river. For pairs that aren't divided by the river, we set bp = 0. IMPORTANT: datasets must contain pairs that HAVE and HAVE NOT experienced the hypothesized shift in psi. In the river example, the dataset must contain pairs whose lineages were NOT separated by the river. BP IS SET TO 0 FOR ALL SUCH PAIRS. Parameter estimation and model_selection is most accurate when ~60-75 percent of the dataset is comprised of bp=0 (aka "single-epoch") pairs.

(19) DA_wt_linear Description: Like DA_wt except psi varies with a continuous gradient both before and after the wait time. Additional Data Required: Values of the continuous variable for each pair must be provided in the 'GRAD' vector. Parameters: alpha, sig2, psi1_sl (slope of psi1~gradient relationship), psi1_int (intercept of psi1~gradient relationship), psi2_sl (slope of psi2~gradient relationship), psi2_int (intercept of psi2~gradient relationship), wt Optional custom starting matrix: A user-defined p_starting matrix must be 7 column matrix where col1 = alpha, col2 = sig2, col3 = psi1_slope, col4=psi1_int, col5=psi2_slope, col6=psi1_int, and col7=wt

(20) DA_bp_linear Description: Like DA_bp except psi varies with a continuous gradient both before and after the wait time. Additional Data Required: Values of the continuous variable for each pair must be provided in the 'GRAD' vector AND values of bp for each pair must be provided in the breakpoint vector. Parameters: alpha, sig2, psi1_slope (distance between optima prior to bp), psi1_int, psi2_slope (slope of distance beteen optima after bp), psi2_int (intercept of distance between optima after bp) Optional custom starting matrix: A user-defined p_starting matrix must be 6 column matrix where col1 = alpha, col2 = sig2, col3 = psi1_slope, col4=psi1_int, col5=psi2_slope, and col6=psi2_int

In general, BM-models may be interpreted as representing random drift or fluctuating selection in trait evolution, OU-models may be interpreted as representing parallel adaptation or shared stabilizing selection (these are mathematically identical in this context), and DA-models may be interpreted as representing divergent adaptation (i.e. evolution in which lineages are generally pulled by selection toward alternative adaptive optima following initial divergence from a common ancestor)

Value

Returns a summary matrix of likelihood results, AIC calculations, and maximum likelihood parameter estimates for each chosen model.

Author(s)

Sean A.S. Anderson and Jason T. Weir

Examples

```
# to fit model, need a dataset
# for illustrative purposes, can simulate dataset under DA_linear
# assume a 0-1000m elevational gradient.
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
GRAD_cats = rep(c(0, 250, 500, 750, 1000), 30)
GRAD=c(rep(GRAD_cats[1], 30), rep(GRAD_cats[2],30), rep(GRAD_cats[3],30),
  rep(GRAD_cats[4],30), rep(GRAD_cats[5],30))
alpha = 0.8
sig2 = 0.2
psi_sl = -0.01
psi_int = 2
sis_div = simulate_div(model="DA_linear", ages=ages, GRAD=GRAD,
  pars=c(alpha, sig2, psi_sl, psi_int))


# run model comparison with DA_null and DA_linear

res <- model_select(div=sis_div, ages=ages, GRAD=GRAD, domain=c(0, 1000),
  models=c("DA_null", "DA_linear"))
```

---

param_grid *Generate parameter grid*

Description

Generate a grid of starting parameter values for each model of trait divergence

Usage

```
param_grid(model, domain = NULL, ncats=NULL)
```

Arguments

model	Character string defining one of ten models of trait divergence (options: "BM_null", "BM_linear", "OU_null", "OU_linear", "DA_null", "DA_linear", "DA_wt", "DA_bp", "DA_wt_linear", "DA_bp_linear"). See find_mle for model descriptions.
domain	Vector of length 2 defining the low and high ends of the gradient domain. Essentially identical to the 'xlim' argument in plotting functions. Required for models with 'linear' suffix.
ncats	A number (either 2 or 3) indicating the number of categories in a DA_cat model.

Details

Primarily a utility function but might be useful in some other cases. Non-linear optimizers can often get stuck on local optima when finding the maximum likelihood parameter set, especially when calculating likelihoods with complex models. find_mle solves this problem by feeding the optimizer a grid of parameter values from which to launch its algorithm. While users can determine their own starting parameters, default parameter grids in model_select and find_mle are calculated with this function. To see/measure/assess the default starting parameter grid for a function of interest, users can use this function directly.

Value

Returns a matrix of starting parameter values. Each olumn contains different values for one parameter and each row is a unique parameter combination in the correct order for likelihood estimation.

Author(s)

Sean A.S. Anderson and Jason T. Weir

Examples

```
# Call the default parameter grid for the "DA_linear" model
# asssume we are testing for a latitudinal gradient over 0-60 degrees.
par_grd = param_grid(model="DA_linear", domain=c(0,60))
dim(par_grd)
head(par_grd)

# Call the default parameter grid for a 3-category "DA_cat" model.
par_grd = param_grid(model="DA_cat", ncat=3)
dim(par_grd)
head(par_grd)
```

---

random_walks *Simulate and plot trait evolution through time*

Description

Simulate and plot continuous trait values as they evolve through time in two diverging lineages under one of four general models.

Usage

```
random_walks(model, TIME, nsim, sig2, alpha = NULL, psi = NULL, psi2 = NULL,
 wt = NULL, theta = 0, centre = 0, steps = 100, plot = TRUE, col = c("black", "red"),
  labels = TRUE, ylim = NULL, ... )
```

Arguments

model	Character vector naming the model under which traits are evolving (one of "BM_null", "OU_null", "DA_null", "DA_wt")
TIME	Numeric value in units of millions of years over which to plot trait evolution
nsim	Number of replicate walks to simulate. If N=40, then 20 replicates of lineages i and 20 replicates of lineage j will be simulated.
/	
sig2	Numeric value of sigma squared parameter. Required for all models.
alpha	Numeric value of alpha parameter for OU and DA models
psi	Numeric value of psi parameter for DA models. If using a breakpoint or wait time model, this is the psi for the first evolutionary epoch
psi2	Numeric value of psi for the second evolutionary epoch if using DA_wt
wt	Numeric value for wait time to a shift in the adaptive regime (aka epoch shift)
theta	Optimum trait value to which both lineages are drawn in a single-process OU model. Defaults to zero.
centre	For DA models, the centre value between the optima of two lineages. Defaults to zero.
steps	Numeric indicating the number of steps per unit TIME over which to calculate simulated trait values - greater numbers of steps create more continuous-looking trait curves, but also larger image files
plot	Logical indicating whether to plot replicate walks.
col	Character vector containing the names of the two colours in which to plot replicate random walks for the two lineages.
labels	Logical indicating whether x and y axes are to be labelled
ylim	Vector of length 2 indicating the max and min y values for the plot. Same argument as in plotting functions. Defaults so that trait curves fill the plot area
...	Additional parameters for the 'plot' function for custom visualization

Details

This function is primarily a visualization aid that will help users intuit the processes under which lineages in a pair evolve after departing from an ancestor. In BM and OU models, lineages in a pair undergo identical evolutionary processes after separating and trait differentiation is generated by random deviations in evolutionary trajectory (i.e. differences are solely generated by the stochastic component of stochastic process models). In the DA models, lineages are pulled deterministically toward alternative adaptive optima, though stochastic noise is still present. The x-label in the default graph is the number of time steps. NOTE: To plot a single random process rather than two diverging lineages (possible for BM and OU models only), simply give the same colour for the lines of both lineages (e.g. col=c("black","black"))

Value

The object returned depends on the model selected. For BM_null and OU_null, random_walks returns an N*M matrix of simulated trait values, where N is the number of replicate walks and M is the number of time steps. For DA_null and DA_bp, random_walks returns a list of length two, where the first element is a K*M matrix (where K=N/2) of simulated trait values for lineage "i" and the second element is a K*M matrix of simulated trait values for lineage "j".The number of rows of the output matrix is equal to the number of time steps.

Author(s)

Sean A.S. Anderson

Examples

```
# plot replicate runs of two lineages evolving under Brownian motion
rw = random_walks(model="BM_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, plot=TRUE)

# plot replicate runs of two lineages evolving under an OU process
rw = random_walks(model="OU_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, alpha=0.8,
plot=TRUE)

# same as above but with custom y axis
rw = random_walks(model="OU_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, alpha=0.8,
plot=TRUE, ylim=c(-2,2))

# plot replicate runs of two lineages evolving under an divergent selection
rw = random_walks(model="DA_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, alpha=0.8,
psi=1, plot=TRUE)

# plot replicate runs of two lineages evolving under an divergent selection
# with a breakpoint in the strength of divergent selection at 2.5my
rw = random_walks(model="DA_wt", TIME=5, steps=200, nsim=40, sig2 = 0.2, alpha=0.8,
psi=0, psi2=1.5, wt=2.5, plot=TRUE)

# play with colours
rw = random_walks(model="DA_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, alpha=0.8,
psi=1.5, plot=TRUE, col=c("purple","orange"), ylim=c(-2,2))

# use custom axes and show time in units of millions of years
rw = random_walks(model="DA_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, alpha=0.8,
psi=1.5, plot=TRUE, col=c("purple","orange"), labels = FALSE, ylim=c(-2,2), axes=FALSE)
box()
axis(1, labels=NA)
axis(1, lwd = 0, line = -0.6, at = seq(0, 1000, 200), labels = seq(0, 10, 2))
axis(2, labels = NA)
axis(2, lwd = 0, line = -0.6)
title(line = 1.9, xlab = "Time (Ma)")
title(line = 1.9, ylab = "Trait Value (mm)")

# plot a single BM process
rw = random_walks(model="BM_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, plot=TRUE,
```

```
col=c("black","black"))

# plot a single OU process
rw = random_walks(model="OU_null", TIME=5, steps=200, nsim=40, sig2 = 0.2, alpha=0.9,
plot=TRUE, ylim=c(-2,2), col=c("black","black"))
```

`re_estimator` *Estimate model parameters*

Description

Re-estimate known model parameters from replicate datasets simulated under a given model. Facilitates power analyses and the measure of accuracy and precision in parameter estimation.

Usage

```
re_estimator(model, div, ages, me1=NULL, me2=NULL, GRAD = NULL, cats=NULL,
 breakpoint = NULL, domain = NULL, p_starting = NULL, parallel = FALSE, cores = NULL,
  absolute = TRUE)
```

Arguments

`model`	Character string defining one of ten models of trait divergence (options: "BM_null", "BM_linear", "OU_null", "OU_linear", "DA_null", "DA_linear", "DA_wt", "DA_bp", "DA_wt_linear", "DA_bp_linear"). See find_mle for model descriptions.
`div`	Vector of trait divergences for a set of lineage pairs. Calculated for each pair as abs(trait_val_lineage_2 - trait_val_lineage1). Raw values (i.e. not absolute values) can also be used but must be noted by the user in the argument 'absolute'.
`ages`	Vector containing the age (i.e. estimated time since divergence) for each pair in the dataset. IMPORTANT: div, ages, GRAD, and breakpoint vectors must be aligned such that div[i] age[i] grad[i] and breakpoint[i] represent values for the same lineage pair.
`me1`	Vector containing the measurement error (standard error of mean) for species 1s of each pair (for one species = variance/No.measurements)
`me2`	Vector containing the measurement error (standard error of mean) for species 2s of each pair (for one species = variance/No.measurements)
`GRAD`	Vector containing the gradient position of each pair. This is the value of a continuous variable such as latitude or body size across which parameters are hypothesized to vary. Required for models with 'linear' suffix.
`cats`	Vector containing the category code (0, 1, or 2) for each pair (see model_select help page for details).
`breakpoint`	Vector of breakpoint times for each pair in the dataset. These are the times AFTER divergence at which a shift occurs in the psi parameter of DA model. Required for DA_bp and DA_bp_linear. See find_mle for details on how to calculate.

domain	Vector of length 2 defining the low and high ends of the gradient domain. Essentially identical to the 'xlim' argument in plotting functions. Required for models with 'linear' suffix.
p_starting	Optional matrix of customized parameter values from which to launch likelihood searches. Must match the structure required for the chosen model. See the model descriptions in find_mle for details. If p_starting=NULL, default starting values are used.
parallel	Logical indicating whether likelihood searches should be conducted in parallel across multiple cores. Not available on windows machines. Defaults to FALSE.
cores	If parallel=TRUE, the number of cores on which to run the function. Defaults to all virtual cores.
absolute	Logical indicating whether 'div' represents absolute value of trait divergence or the raw divergence values.

Details

re_estimator is a wrapper to find_mle that returns just the parameter values. The argument 'div' can be a vector of trait divergences or a list containing several vectors of trait divergence. When 'div' is a list, the function can be used to generate a distribution of parameter values. These can in turn be used to evaluate precision and accuracy of likelihood searches when replicate datasets are simulated under a known model.

TIME CONSIDERATIONS Warning: re-estimating parametres from a large number of replicate datasets can be extremely time intensive when DA_wt, DA_bp, DA_wt_linear, or DA_bp_linear are the model in question. Re-estimating parameters to assess model performance using a reasonable number of replicates (~1000), from one of these four complex models, can take hours on a multi-core server.

Value

Returns a matrix of parameter estimates. Each column contains replicate estimates of one parameter. The matrix has one row if div is a vector and N rows if div is a list of length N.

Author(s)

Sean A.S. Anderson

Examples

```
# simulate a dataset of trait divergence with 150 lineage pairs with a range of ages.
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)
alpha = 0.8
sig2 = 0.2
psi=0.8
N=10
sis_div = simulate_div(model="DA_null", ages=ages, pars=c(alpha, sig2, psi), N=N)

# estimate pars from one trait divergence dataset
par_es = re_estimator(model = "DA_null", div=sis_div[[1]], ages=ages)
par_es
```

```
# estimate pars from a list of trait divergence datasets and find the median
# of each estimate

par_es = re_estimator(model = "DA_null", div=sis_div, ages=ages)
head(par_es)
meds = apply(par_es, 2, median)
meds
```

---

simulate_div *Simulate trait divergence*

Description

Simulate replicate sets of lineage pair trait divergence under one of ten evolutionary models.

Usage

```
simulate_div(model, pars, ages, me_prop=NULL, GRAD = NULL, cats=NULL,
  breakpoint = NULL, Nsets = 1)
```

Arguments

model	Character string defining one of 11 models of trait divergence (options: "BM_null", "BM_linear", "OU_null", "OU_linear", "DA_null", "DA_linear", "DA_cat", "DA_wt", "DA_bp", "DA_wt_linear", "DA_bp_linear", "DA_OU", "DA_OU_linear", "DA_OU_cat", "DA_BM", "DA_BM_linear", "DA_BM_cat", "OU_BM", "OU_BM_linear", "OU_BM_cat"). See model_select for model descriptions.
pars	Vector of model parameters to use in sumulation. Parameters must be in the correct order in the vector for each model. The parameter orders are BM_null: sig2, BM_linear: sig2_slope, sig2_int OU_null: alpha, sig2 OU_linear: alpha_int, alpha_slope, sig2 DA_null: alpha, sig2, psi DA_linear: alpha, sig2, psi_slope, psi_int DA_cat: alpha, sig2, psi1, psi2, psi3(optional) DA_wt: alpha, sig2, psi1, psi2, wt DA_bp: alpha, sig2, psi1, psi2 DA_wt_linear: alpha, sig2, psi1_slope, psi1_int, psi2_slope, psi2_int, wt DA_bp_linear: alpha, sig2, psi1_slope, psi1_int, psi2_slope, psi2_int
ages	Vector of 'ages' (i.e. times since divergence) of lineage pairs for which to simulate trait divergence.
me_prop	Induced measurement error as a proportion of the expected variance. Can be a single value applied to all simulated species or a vector of values of length equal to length(ages).

GRAD	Vector containing the gradient position of each pair to be simulated. This is the value of a continuous variable such as latitude or body size across which parameters will vary. Required for all models with 'linear' suffix.
cats	Vector containing the category code (0, 1, or 2) for each pair (see model_select help page for details).
breakpoint	Vector of breakpoint times (times after 0 at which a discrete shift occurs in psi) for each lineage pair to be simulated. Required for all "bp" models. Must align with "ages" vector such that ages[i] and breakpoint[i] correspond to the same lineage pair. When using bp models, datasets can and should be simulated with a mix of pairs that have and have not experienced a shift in psi (see model_select for details on bp models). Set bp=0 for pairs that experience no shift.
Nsets	Number of datasets to simulate.

Details

Simulates replicate sets of trait divegence under one of ten evolutionary models given a user-defined set of lineage pair ages. Since all evolutionary models are stochastic process models, replicate datasets simulated under the same model parameters are not identical. Users can simulate the addition of measurement error using the ms_prop argument, which denotes measurement error as a proportion of the expected variance (e.g. ms_prop = 0.1 corresponds to a measurement error in each species equal to 10 percent of the expected variance in the trait in that species, given the parameters).

Value

Returns a vector (if Nsets=1) representing divergences for one dataset or a list of vectors (if Nsets>1) for replicate datasets of lineage-pair trait divergences.

Author(s)

Sean A. S. Anderson

Examples

```
## Simulate datasets of trait divergence for 150 lineage pairs
# under different evolutionary models

# Define vector of 150 lineage pair ages
ages = rep(c(0.5, 1, 1.5, 2, 3, 8), 25)

## simulate a dataset of trait divergence under a single-process Brownian Motion model
sig2 = 0.2
sis_div_BM = simulate_div(model="BM_null", ages=ages, pars=c(sig2))
summary(sis_div_BM)
sis_div_BM

## simulate 1000 divergence datasets under the null model of divergent adaptation
sig2 = 0.2
alpha = 0.8
psi = 0.8
```

```
sis_div_DA = simulate_div(model="DA_null", ages=ages, pars=c(alpha, sig2, psi), N=1000)
length(sis_div_DA)
sis_div_DA[[1]]

## Simulate 1000 divergence datasets under the breakpoint model
# note: a breakpoint vector must be created
# pairs that experience no epoch shift (no shift in the psi parameter) are assigned bp=0
# for all two-epoch pairs, the breakpoint time must be lower than the age of the pair
# here we make half the dataset into two-epoch pairs, half into one-epoch pairs
# we set the breakpoint time equal to half the age of each two-epoch pair
bp = c(ages[1:75]/2, rep(0, 75))

# run the simulation
sig2 = 0.2
alpha = 0.8
psi1 = 0.3
psi2 = 0.9
sis_div_bp = simulate_div(model="DA_bp", ages=ages, breakpoint=bp,
  pars=c(alpha, sig2, psi1, psi2), N=1000)
length(sis_div_bp)
sis_div_bp[[1]]
```

Index