# Package: OUwie (via r-universe)

August 27, 2024

**Version** 2.13

**Date** 2024-08-21

**Title** Analysis of Evolutionary Rates in an OU Framework

**Author** Jeremy M. Beaulieu <jmbeauli@uark.edu>, Brian O'Meara <bomeara@utk.edu>

**Maintainer** Jeremy Beaulieu <jmbeauli@uark.edu>

**Depends** R (>= 3.2.0), ape, corpcor, nloptr, geiger, RColorBrewer

**Suggests** testthat, knitr, rmarkdown

**Imports** igraph, numDeriv, phytools, paleotree, phangorn, stats, lhs, interp, grDevices, parallel, phylolm, GenSA, MASS, corHMM, data.table, expm, ggplot2, reshape2

**Description** Estimates rates for continuous character evolution under Brownian motion and a new set of Ornstein-Uhlenbeck based Hansen models that allow both the strength of the pull and stochastic motion to vary across selective regimes. Beaulieu et al (2012).

**URL** https://github.com/thej022214/OUwie

**License** GPL (>= 2)

**VignetteBuilder** knitr

**Repository** https://phylotastic.r-universe.dev

**RemoteUrl** https://github.com/thej022214/OUwie

**RemoteRef** HEAD

**RemoteSha** 892d71dacf2f45abbab4e2e9264f6902cee4b582

## Contents

---

`check.identify` *A test of regime identifiability*

---

## Description

Ho and Ane test for determining whether all regimes form connected subtrees, making both the ancestral state and the regime optima unidentifiable.

## Usage

```
check.identify(phy, data, simmap.tree=FALSE, quiet=FALSE)
```

## Arguments

| | |
|---|---|
| `phy` | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes. |
| `data` | a data.frame containing species information (see Details). |
| `simmap.tree` | a logical indicating whether the input tree is in SIMMAP format. The default is `FALSE`. |
| `quiet` | a logical indicating whether messages should be written to the screen. The default is `FALSE`. |

## Value

This returns a vector with two elements, with the first being an indicator of identifiability (0=unidentifiable, 1=identifiable, and if `get.penalty=TRUE`, the second is the penalty used for the modified BIC.

## Author(s)

Jeremy M. Beaulieu and Brian C. O'Meara

## References

Ho, L.S.T., and C. Ane. 2014. Intrinsic inference difficulties for trait evolution with Ornstein-Uhlenbeck models. Methods in Ecology and Evolution, 5: 1133-1146.

---

`dent_propose` *Propose new values This proposes new values using a normal distribution centered on the original parameter values, with desired standard deviation. If any proposed values are outside the bounds, it will propose again.*

---

## Description

Propose new values This proposes new values using a normal distribution centered on the original parameter values, with desired standard deviation. If any proposed values are outside the bounds, it will propose again.

## Usage

```
dent_propose(old_params, lower_bound = -Inf, upper_bound = Inf, sd = 1)
```

## Arguments

| | |
|---|---|
| `old_params` | The original parameter values |
| `lower_bound` | Minimum parameter values to try. One for all or a vector of the length of par. |
| `upper_bound` | Maximum parameter values to try. One for all or a vector of the length of par. |
| `sd` | Standard deviation to use for the proposals. One for all or a vector of the length of par. |

## Value

A vector of the new parameter values

dent_walk *Sample points from along a ridge*

## Description

This "dents" the likelihood surface by reflecting points better than a threshold back across the threshold (think of taking a hollow plastic model of a mountain and punching the top so it's a volcano). It then uses essentially a Metropolis-Hastings walk to wander around the new rim. It adjusts the proposal width so that it samples points around the desired likelihood. This is better than using the curvature at the maximum likelihood estimate since it can actually sample points in case the assumptions of the curvature method do not hold. It is better than varying one parameter at a time while holding others constant because that could miss ridges: if I am fitting 5=x+y, and get a point estimate of (3,2), the reality is that there are an infinite range of values of x and y that will sum to 5, but if I hold x constant it looks like y is estimated very precisely. Of course, one could just fully embrace the Metropolis-Hastings lifestyle and use a full Bayesian approach.

While running, it will display the current range of likelihoods in the desired range (by default, the best negative log likelihood + 2 negative log likelihood units) and the parameter values falling in that range. If things are working well, the range of values will stabilize during a search.

## Usage

```
dent_walk(
  par,
  fn,
  best_neglnL,
  delta = 2,
  nsteps = 1000,
  print_freq = 50,
  lower_bound = 0,
  upper_bound = Inf,
  adjust_width_interval = 100,
  badval = 1e+09,
  sd_vector = NULL,
  debug = FALSE,
  restart_after = 50,
  ...
)
```

## Arguments

| | |
|---|---|
| par | Starting parameter vector, generally at the optimum. If named, the vector names are used to label output parameters. |
| fn | The likelihood function, assumed to return negative log likelihoods |
| best_neglnL | The negative log likelihood at the optimum; other values will be greater than this. |
| delta | How far from the optimal negative log likelihood to focus samples |

| | |
|---|---|
| nsteps | How many steps to take in the analysis |
| print_freq | Output progress every print_freq steps. |
| lower_bound | Minimum parameter values to try. One for all or a vector of the length of par. |
| upper_bound | Maximum parameter values to try. One for all or a vector of the length of par. |
| adjust_width_interval | When to try automatically adjusting proposal widths |
| badval | Bad negative log likelihood to return if a non-finite likelihood is returned |
| sd_vector | Vector of the standard deviations to use for proposals. Generated automatically if NULL |
| debug | If TRUE, prints out much more information during a run |
| restart_after | Sometimes the search can get stuck outside the good region but still accept moves. After this many steps without being inside the good region, restart from one of the past good points |
| ... | Other arguments to fn. |

## Details

The algorithm tunes: if it is moving too far away from the desired likelihoods, it will decrease the proposal width; if it staying in areas better than the desired likelihood, it will increase the proposal width. It will also expand the proposal width for parameters where the extreme values still appear good enough to try to find out the full range for these values.

In general, the idea of this is not to give you a pleasingly narrow range of possible values – it is to try to find the actual uncertainty, including finding any ridges that would not be seen in univariate space.

## Value

A dentist object containing results, the data.frame of negative log likelihoods and the parameters associated with them; acceptances, the vector of whether a proposed move was accepted each step; best_neglnL, the best value passed into the analysis; delta, the desired offset; all_ranges, a summary of the results.

---

# Example *An example dataset*

---

## Description

An example dataset containing a 64-tip birth-death tree with internal node labels denoting two selective regimes, and a trait file in the proper format: 1) Genus_species, 2) current selective regime, 3) continuous trait data.

## Format

a tree of class "phylo" and a data frame with 3 columns and 64 rows

`fix.kappa` *Adjust tree for matrix condition*

## Description

Iteratively deletes taxa with shortest tip length to try to get a variance covariance matrix with good matrix condition.

## Usage

```
fix.kappa(phy, data, threshold = log(40))
```

## Arguments

| | |
|---|---|
| `phy` | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes. |
| `data` | a data.frame containing species information (see Details). |
| `threshold` | log(condition), as measured by kappa(), which is too large |

## Details

Internally, OUwie uses an algorithm that can perform poorly when the variance covariance matrix is poorly conditioned (which can happen if two columns are very similar, as when the divergence depth of two species is very recent). This does not mean there is anything wrong with the biology, just that the numerical algorithms perform poorly in that case. If it's a model that can be fit in phylolm or geiger, those packages use a differnt algorithm that is more robust to this. What this function does is take your original tree and data and deletes taxa with the shortest branches, in order, to try to get a starting tree with generally good condition. Deleting data is always a sad thing, but this can result in a more accurate estimate of the likelihood and parameter values.

## Value

This returns a list with two elements:

| | |
|---|---|
| `$phy` | the phylogeny with taxa deleted. |
| `$data` | the data with taxa deleted. |

## Author(s)

Brian C. O'Meara

`getModelAvgParams` *Model average the parameter estimates over severl hOUwie fits.*

## Description

This function takes as input a list of hOUwie fits and will automatically model average the parameter estimates. It returns model averaged parameter values.

## Usage

```
getModelAvgParams(model.list,
                  BM_alpha_treatment = "zero",
                  type               = "BIC",
                  force              = TRUE)
```

## Arguments

| | |
|---|---|
| `model.list` | A list of model results from several fits of the `hOUwie` model. All elements of the list must be of class "houwie". |
| `BM_alpha_treatment` | A boolean indicating whether alpha from BM models should be treated as zero or those models should be removed when averaging alpha values. Currently zero is the only allowed value. |
| `type` | One of AIC, BIC, or AICc for use during evaluation of relative model fit. AICc is the best option for datasets with a few number of species. |
| `force` | A boolean indicating whether to force potentially failed model fits to be included in the model averaging. |

## Details

The AIC weight of each model is first evaluated and used as the weighting for each models' contribution to the overall parameter value (Burnham and Anderson 2002). Tip values are estimated based on the weighted average of the joint probability of the fitted stochastic maps.

## Value

Returns a named list with the following elements:

`weighted_tip_values`
: Model averaged parameter estimate for each tip.

## Author(s)

James D. Boyko

## References

Beaulieu J.M., Jhwueng D.-C., Boettiger C., O'Meara B.C. 2012. Modeling Stabilizing Selection: Expanding the Ornstein–Uhlenbeck Model of Adaptive Evolution. Evolution. 66:2369–2383.

Burnham K.P., Anderson D.R. 2002. Model selection and multimodel inference: a practical information-theoretic approach. New York: Springer.

Butler M.A., King A.A. 2004. Phylogenetic Comparative Analysis: A Modeling Approach for Adaptive Evolution. The American Naturalist. 164:683–695.

Hansen T.F. 1997. Stabilizing Selection and the Comparative Analysis of Adaptation. Evolution. 51:1341–1351.

## Examples

```
## Not run:
# fit several possible models (we're using fixed parameters here)
p <- c(0.01664314, 0.39631218, 0.18684476, 2.25121568, 0.82495093) # MLE
Model_X <- hOUwie(tree, trait, rate.cat = 1, discrete_model = "ER",
                  continuous_model = "OUM", nSim = 25, p = p)
p <- c(0.01664314, 0.39631218, 0.18684476, 0.25, 2.25121568, 0.82495093) # not MLE
Model_Y <- hOUwie(tree, trait, rate.cat = 1, discrete_model = "ER",
                  continuous_model = "OUMV", nSim = 25, p = p)
p <- c(0.01664314, 0.39631218, 0.09631218, 0.18684476, 2.25121568, 0.82495093) # not MLE
Model_Z <- hOUwie(tree, trait, rate.cat = 1, discrete_model = "ER",
                  continuous_model = "OUMA", nSim = 25, p = p)

# put the model results into a list
model_list <- list(Model_X = Model_X, Model_Y = Model_Y, Model_Z = Model_Z)

# model average the parameters
getModelAvgParams(model_list)

## End(Not run)
```

---

`getModelTable` *Generate a table from a set of hOUwie models describing their relative fit to data.*

---

## Description

This function takes as input a list of hOUwie models and outputs their relative fit to a dataset.

## Usage

```
getModelTable(model.list,
              type = "BIC")
```

## Arguments

| | |
|---|---|
| model.list | A list of model results from several fits of the hOUwie model. All elements of the list must be of class "houwie". |
| type | One of AIC, BIC, or AICc for use during evaluation of relative model fit. AICc or BIC is the best option for datasets with a few number of species. |

## Details

Models are named based either on the names of the list or the order in which they are provided. If names are given, the rownames will reflect the input. If no names are given, model are assigned names M1 to Mn for the 1 to n models.

## Value

| | |
|---|---|
| model_table | A data.frame containing the number of parameters (np), total joint log likelihood (lnLik), marginal discrete log likelihood (DiscLik), marginal continuous log likelihood (ContLik), Akaike Information Criterion (AIC), difference in AIC (dAIC), and AIC weight (AICwt). |

## Author(s)

James D. Boyko

## References

Akaike H. 1998. Information Theory and an Extension of the Maximum Likelihood Principle. :15.

Burnham K.P., Anderson D.R. 2002. Model selection and multimodel inference: a practical information-theoretic approach. New York: Springer.

## Examples

```
## Not run:
# fit several possible models (we're using fixed parameters here)
p <- c(0.01664314, 0.39631218, 0.18684476, 2.25121568, 0.82495093) # MLE
Model_X <- hOUwie(tree, trait, rate.cat = 1, discrete_model = "ER",
                  continuous_model = "OUM", nSim = 25, p = p)
p <- c(0.01664314, 0.39631218, 0.18684476, 0.25, 2.25121568, 0.82495093) # not MLE
Model_Y <- hOUwie(tree, trait, rate.cat = 1, discrete_model = "ER",
                  continuous_model = "OUMV", nSim = 25, p = p)
p <- c(0.01664314, 0.39631218, 0.09631218, 0.18684476, 2.25121568, 0.82495093) # not MLE
Model_Z <- hOUwie(tree, trait, rate.cat = 1, discrete_model = "ER",
                  continuous_model = "OUMA", nSim = 25, p = p)

# put the model results into a list
model_list <- list(Model_X = Model_X, Model_Y = Model_Y, Model_Z = Model_Z)

# get a model table describing the relative fits to the data
getModelTable(model_list)

## End(Not run)
```

`getOUParamStructure` *Generate a continuous model parameter structure*

## Description

Given a specific model type (see Details) and number of observed states create a matrix which describes the continuous model structure for use in `hOUwie`. This function can be used to generate a set of character-dependent and character-independent models for multi-model inferences.

## Usage

```
getOUParamStructure(model,
                    nObsState,
                    rate.cat   = 1,
                    null.model = FALSE)
```

## Arguments

| | |
|---|---|
| `model` | One of "BM1", "BMV", "OU1", "OUA", "OUV", "OUM", "OUVA", "OUMV", "OUMA", "OUMVA". |
| `nObsState` | An integer specifying the total number of observed states. |
| `rate.cat` | An integer specifying the number of rate categories/ hidden states. If `null.model = TRUE`, then this should be at least 2 to allow for character-independent rate heterogeneity. |
| `null.model` | A boolean indicating whether the model is character-independent with rate heterogeneity (`TRUE`) or character-dependent (`FALSE`). |

## Details

The basic models structures supported and their primary reference (so far as I know) can be found below. Additionally, many intermediate models can be formed by users who modify the default rate matricies when there are multiple rate categories or discrete states. E.g., it may be that states 1 and 3 share an optimum that is unique from state 2. This model is not automatically created by `getOUParamStructure`, but can be easily created by the user.

BM1: Brownian motion model with a single evolutionary rate (Felsenstein 1985).

BMV: Brownian motion model with multiple, regime dependent, evolutionary rates (O'Meara et al. 2006).

OU1: Ornstein-Uhlenbeck model with a single optima, alpha, and sigma.sq (Hansen 1997).

OUA: Ornstein-Uhlenbeck model with a single optima, variable alpha, and single sigma.sq (Boyko et al. 2022).

OUV: Ornstein-Uhlenbeck model with a single optima, single alpha, and variable sigma.sq (Boyko et al. 2022).

OUM: Ornstein-Uhlenbeck model with variable optima, single alpha, and single sigma.sq (Butler and King 2004).

OUVA: Ornstein-Uhlenbeck model with single optima, variable alpha, and variable sigma.sq (Boyko et al. 2022).

OUMV: Ornstein-Uhlenbeck model with variable optima, single alpha, and variable sigma.sq (Beaulieu et al. 2012).

OUMA: Ornstein-Uhlenbeck model with variable optima, variable alpha, and single sigma.sq (Beaulieu et al. 2012).

OUMVA: Ornstein-Uhlenbeck model with variable optima, variable alpha, and variable sigma.sq (Beaulieu et al. 2012).

## Value

Returns a matrix which describes the continuous model structure for use in `hOUwie`.

## Author(s)

James D. Boyko

## References

Beaulieu J.M., Jhwueng D.-C., Boettiger C., O'Meara B.C. 2012. Modeling Stabilizing Selection: Expanding the Ornstein–Uhlenbeck Model of Adaptive Evolution. Evolution. 66:2369–2383.

Boyko J.D., O'Meara B.C., Beaulieu J.M. 2022. In prep.

Butler M.A., King A.A. 2004. Phylogenetic Comparative Analysis: A Modeling Approach for Adaptive Evolution. The American Naturalist. 164:683–695.

Felsenstein J. 1985. Phylogenies and the Comparative Method. Am. Nat. 125:1–15.

Hansen T.F. 1997. Stabilizing Selection and the Comparative Analysis of Adaptation. Evolution. 51:1341–1351.

O'Meara B.C., Ane C., Sanderson M.J., Wainwright P.C. 2006. Testing for Different Rates of Continuous Trait Evolution Using Likelihood. Evolution. 60:922–933.

## Examples

```
## Not run:
# for the following example we will imagine we have a single discrete character with 3 states.
# these matrices could be assigned to a variable and input into the continuous_model arguement
# in hOUwie for later model comparison.

# there are several model structures but they can be broken into:
# Character dependent (CD) which
getOUParamStructure("OUMA", 3, 1, FALSE)
# Character independent (CID) which can have rate heterogeneity
getOUParamStructure("OUMA", 3, 2, TRUE)
# Hybrid model (HYB) which has both character dependent and character independent rate heterogeneity
getOUParamStructure("OUMA", 3, 2, FALSE)

# from the different uses of the function above, notice how the parameters are associated
```

```
# with particular regimes to create the different classes of model. CD has a separate
# parameter for each state, whereas CID always has the different observed states associated
# with the same parameter value. Obviously, different mixes of these are allowed and the
# hybrid model is the most general form which can then be further constrained.

## End(Not run)
```

---

hOUwie *Fit a joint model of discrete and continuous characters via maximum-likelihood.*

---

## Description

The hOUwie model jointly estimates the likelihood of a discrete and continuous character by combining the probability of the continuous character given a particular regime and the probability of that discrete regime painting, integrated over many regime paintings. Specifically, we combine hidden Markov models of discrete character evolution (Beaulieu et al. 2013; Boyko and Beaulieu 2021) with generalized Ornstein-Uhlenbeck models (Hansen 1997; Butler and King 2004; Hansen et al. 2008; Beaulieu et al. 2012; Ho and Ane 2014a).

This function takes as main input a phylogenetic tree and a data.frame containing species, character values, and potentially measurement error. The rate category is a way to specify the number of hidden states to be incldued, with one meaning no hidden states present. The structure of the model and association between discrete and continuous characters is specified via discrete_model and continuous model (see Details).

## Usage

```
hOUwie(phy,
       data,
       rate.cat,
       discrete_model,
       continuous_model,
       null.model          = FALSE,
       nSim                = 100,
       root.p              = "yang",
       dual                = FALSE,
       collapse            = TRUE,
       root.station        = FALSE,
       get.root.theta      = FALSE,
       tip.fog             = "none",
       lb_discrete_model   = NULL,
       ub_discrete_model   = NULL,
       lb_continuous_model = NULL,
       ub_continuous_model = NULL,
       recon               = FALSE,
       nodes               = "internal",
```

```
p                   = NULL,
ip                  = NULL,
optimizer           = "nlopt_ln",
opts                = NULL,
quiet               = FALSE,
sample_tips         = FALSE,
sample_nodes        = FALSE,
adaptive_sampling   = FALSE,
diagn_msg           = FALSE,
n_starts            = 1,
ncores              = 1)
```

## Arguments

| | |
|---|---|
| phy | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes. |
| data | A data frane containing species information. The first column is always species names matching the tip labels of phy. The 2nd to ith columns are independent discrete characters. The ith+1 column is the continuous character value. Finally, there can be an ith+2 column for measurement error (optional). |
| rate.cat | Specifies the number of rate categories (see Details). |
| discrete_model | Either a user-supplied index of parameters to be optimized or one of "ARD", "SYM", or "ER". ARD: all rates differ. SYM: rates between any two states do not differ. ER: all rates are equal. |
| continuous_model | Either a user-supplied index matrix specifying the continuous model parameters to be estimated or one of "BM1", "BMV", "OU1", "OUA", "OUV", "OUM", "OUVA", "OUMV", "OUMA", "OUMVA" (See also getOUParamStructure). |
| null.model | A boolean indicating whether the model being run is a character-independent model with rate heterogeneity. Rate.cat must be greater than 1. |
| nSim | The number of stochastic maps evaluated per iteration of the ML search. |
| root.p | a vector used to fix the probabilities at the root, but "yang" can also be supplied to use the method of Yang (2006) (see Details). |
| dual | A boolean indicating whether or not to include dual transitions. If true, then transitions two or more states may change at any instant of time. For example, X0Y0 can change directly to X1Y1 without first going through X1Y0 or X0Y1. |
| collapse | A boolean indicating whether to collapse multiple character combinations into only the observed states. For example, if true a two character dataset contained (0,0), (1,0), and (1,1), this would be collapsed into 1,2,3. However, if set to false it would 1,2,4. In combination with a custom rate matrix this allows for the estimation of transitions between unobserved character combinations. Transitions to and from unobserved state combinations may not be estimatable: See Boyko and Beaulieu (2022). |
| root.station | A boolean indicating whether the starting state, $\theta_0$, should be estimated (see Details of OUwie function). |

`get.root.theta` A boolean indicating whether the starting state, $\theta_0$, should be estimated (see Details of OUwie function).

`tip.fog` designates whether a fourth column in the data matrix contains within species variation for each species value ("known"). The tip fog is assumed to be the standard error of the species mean. The default is "none", but can be estimated by setting the option to "estimate".

`lb_discrete_model`
A single value for the lower bound of discrete character evolution during the ML search. The default is `1/(Tmax*10000)`.

`ub_discrete_model`
A single value for the upper bound of discrete character evolution during the ML search. The default is `1/(Tmax*0.0001)`.

`lb_continuous_model`
Three values specifying the lower bound of continuous character evolution during the ML search. Should be provided in the order of lower.bound for alpha, sigma squared, theta. The default is `c(1e-10, 1e-10, smallest_continuous_value/10)`.

`ub_continuous_model`
Three values specifying the upper bound of continuous character evolution during the ML search. Should be provided in the order of upper.bound for alpha, sigma squared, theta. The default is `c(log(2)/(0.01 * Tmax), log(2)/(0.01 * Tmax), largest_continuous_value*10)`.

`recon` A boolean indicating whether marginal ancestral state reconstruction should take place following the ML search.

`nodes` If `recon=TRUE`, a character or vector indicating which nodes should have their discrete ancestral state characters estimated. If a vector, nodes specified will be reconstructed with 1:Ntip representing tip reconstructions and Ntip+1 representing the root. A user can also indicate "all", "internal", or "external" to reconstruct all nodes, ancestral nodes, or extant nodes respectively. Reconstructing tip states ("external") is only useful if hidden states are included in the model.

`p` A numeric vector of fixed parameters to be optimized. Parameters are organized such following `c(p_trans, p_alpha, p_sigma.squared, p_theta)`

`ip` A numeric vector indicating initial parameters to start the ML search. Parameters are organized such following `c(p_trans, p_alpha, p_sigma.squared, p_theta)`

`optimizer` One of "nlopt_ln", "nlopt_gn", or "sann" for a local search of parameter space, global search of paramater space, or simmulated annealing. The default is "nlopt_ln".

`opts` A list of options to be passed to nloptr.

`quiet` A boolean indicating whether or not to print messages.

`sample_tips` A boolean indicating whether or not to sample tip probabilities when generating underlying regimes (see Deatils, but recommended to be set to FALSE).

`sample_nodes` A boolean indicating whether or not to sample node probabilities when generating underlying regimes. Recommended that this is set to TRUE when evaluating character independent models, but can be set to FALSE for character dependent models.

`adaptive_sampling`
: A boolean indicating whether each iteration will continue sampling stochastic maps until the likelihood of a set of parameters does not improve. Recommended that this is set to TRUE when evaluating character independent models, but can be set to FALSE for character dependent models.

`diagn_msg`
: A boolean indiciating whether a message of loglikelihoods and parameters should be printed at each iteration. Can be useful for diagnosing problems.

`n_starts`
: An integer, specifying the number of trials for fitting the model, using alternative (randomized) starting parameters at each trial. A larger n_starts reduces the risk of landing on a local non-global optimum of the likelihood function, and thus increases the chances of finding the MLE.

`ncores`
: An integer, specifying the number of cores for running multiple fitting n_starts in parallel. Should generally not exceed the number of CPU cores on a machine, but must be a least 1.

## Details

This model is composed of two processes: one that describes the evolution of a discrete character and the other describes the evolution of a continuous character. To model the evolution of a single continuous character we use an Ornstein-Uhlenbeck (OU) model (Hansen 1997; Butler and King 2004; Hansen et al. 2008; Beaulieu et al. 2012; Ho and Ane 2014a). This model combines the stochastic evolution of a trait through time with a deterministic component which models the tendency for a trait to evolve towards an optimum. In this model, the value of a trait, `X_t`, is pulled towards an optimum, `theta`, at a rate scaled by the parameter `alpha`. The optimum, `theta`, is a piecewise constant on intervals and takes values in a finite set `theta`. This can represent the set of "selective regimes", "regimes", or Simpson's "adaptive zones" (Cressler et al. 2015), though it is consistent with a variety of true underlying microevolutionary models (Hansen 2014). Additionally, random deviations are introduced by Gaussian white noise dB(t), which is distributed as a normal random variable with mean zero and variance equal to `sigma^2`. Thus, `sigma^2` is a constant describing the rate of stochastic evolution around the optimum. We use the set of extensions introduced by Beaulieu et al. (2012). This allows for multiple primary optima `theta` in which both the pull strength (`alpha`) and the rate of stochastic evolution (`sigma^2`) can vary across the phylogeny.

This model allows for multiple rate categories (specified by the rate.cat argument). The default is one rate class in which only observed discrete states evolve. However, there are two main reasons one may be interested in increasing the number of rate categories. First, rate heterogeneity throughout the tree of life is more of a rule than a possibility. Not all things will evolve in the same way at the same time. For example, woody and herbaceous plants may evolve one way on the mainland and a completely different way on island. The challenge then is to allow for heterogeneity in the evolutionary process when we do not know what the 'mainland' or 'island' variables are. This is what hidden Markov models allow us to do (Boyko and Beaulieu 2021). The second reason to include hidden rate categories is to reduce the error rates of finding a false correlation. This has been discussed elsewhere in depth (see Maddison and FitzJohn 2015, Uyeda et al. 2018, Boyko and Beaulieu 2022). The problem lies in if we compare models with rate heterogeneity to models without it. Most character dependent models (correlation models) allow for different ways for the characters to evolve. In fact, their reliable inference depends on being able to define the differences between, for example, body size evolution on islands and mainlands. However, most character-independent models have no way to allow for variable ways for body size to evolve. So, whether or

not the evolution of body size is connected to island systems, we may be biased towards selecting correlation models simply because they allow body size to have different rates of evolution.

It is not uncommon to map a discrete character on a phylogeny using a model of evolution for that discrete character and then run various continuous trait models using that mapping (or a stochastic set of these mappings), and OUwie has ways to use such mappings. However, that mapping is completely uninformed by the continuous trait – it could be that a nearly identical mapping fits the continuous trait far better. A different approach is to do the regime detection using the continuous trait only, as in SURFACE or OUwie.dredge. But if there is a reasonable discrete character (perhaps with hidden states), it can make the most sense to get the likelihood integrating across all different discrete mappings, summing the likelihood for the discrete and continuous traits while doing so. Because looking at all possible reconstructions is infeasible, and looking at them from a distribution from stochastic mapping biases it towards discrete-only mappings, we approximate this by summing over a large number of potential mappings centered on those that seem a good fit to the data.

One way that hOUwie differs from other implementations of automatic regime detection is that we explicitly calculate the the probability of discrete characters (D) and stochastic mapping (z). The difficulties of calculating the pathway probabilities explicitly are outlined in Boyko et al. (2022), but ultimately led us to use an approximation. This approximation relies on a finite number of degree-2 internodes and uses the standard Chapman-Kolmgorov equation to calculate the probabilities of beginning in a particular state i and ending in state j (Pagel 1994) and is identical to a joint probability of a set of state reconstructions (Yang 2006).

`discrete_model` defines the model between discrete states. This option controls the number of independent rates and the correspondence between independent and dependent rates. If a character, then it must be one of "ER", "SYM", "ARD". If an integer matrix, then it defines a custom parametric structure for the transition rates, by mapping entries of the transition matrix to a set of independent transition-rate parameters (numbered 1,2, and so on). All cells with the number 3 represent transitions that have the same estimated rate. Additionally, ambiguities (polymorphic taxa or taxa missing data) are assigned likelihoods following Felsenstein (2004, p. 255). Taxa with missing data are coded "?" with all states observed at a tip. Polymorphic taxa are coded with states separated by an "&". For example, if a trait has four states and taxon A is observed to be in state 1 and 3, the character would be coded as "1&3". Missing data are treated as ambiguous for all states, thus all states for taxa missing data are assigned a likelihood of 1.0 (some software may incorrectly assign a likelihood of 1.0/number of possible states; note this if comparing likelihoods). For example, for a four-state character (i.e. DNA), a taxon missing data will have likelihoods of all four states equal to 1.0 [e.g. L(A)=1.0, L(C)=1.0, L(G)=1.0, L(T)=1.0].

`continuous_model` option controls the number of independent continuous model parameters and the correspondence between continuous model parameters and underlying regime. If a character, then it must be one of "BM1", "BMV", "OU1", "OUM", "OUA", "OUV", "OUMV", "OUMA", "OUVA", or "OUMVA". These correspond to allowing theta ("OUM"), sigma_square ("BMV", "OUV"), or alpha ("OUA") to vary. If an integer matrix, then it defines a custom parametric structure for the transition rates, by mapping entries of the continuous model parameters to the states of the underlying regimes. Note that when some taxa have phenotypic values below 0, we add 50 to all continuous trait values to assist the optimization, but the estimated parameter values will be scaled back to the original values.

For each set of parameters evaluated during the maximum likelihood search, a set of stochastic mappings are generated to evaluate the discrete and continuous likelihoods. To do this efficiently, we first approximate of the conditional state probabilities at nodes. This is what happens if `sample_nodes=TRUE`. The conditional state probability, unlike the more common marginal re-

construction or joint state reconstruction, calculates the probability that a node has a particular state value conditioned only on the observations of its descendants. For a particular focal node, we calculate the probability of the observing all pairwise descendant values given the OU model parameters, integrated over all possible rootward node states, and observed tipward discrete states. This process is repeated for each iteration of parameters if `adaptive_sample=TRUE`, in which case it is repeated until the joint likelihood for a given set of mappings does not improve. We recommend that both of these arugements are set to TRUE for character independent models because it is often very difficult to generate mappings based soley on a discrete process that are good when the discrete character is predicted to be unlinked to the continuous character (i.e., when you're fitting a character independent model). In the case of a character dependent model, the continuous and discrete character are expected to influence one another and actually basing maps on the discrete character only (setting sample_nodes and adaptive_sampling to FALSE) can work fairly well.

IMPORTANT NOTE: The algorithm used to calculate the likelihood described in Beaulieu et al. (2012) involves matrix inversion – a computationally costly procedure. Therefore, we implement a linear-time computation of the likelihood of Gaussian trait models following (Ho and Ane 2014). For this we rely on changes in the most recent version of the phylolm package which is only available on github (not CRAN). Therefore, in order for hOUwie to function properly make sure that the most recent version of phylolm is installed which can be done by running the following code: `devtools::install_github("lamho86/phylolm")`. For more information on phylolm installation see: https://github.com/lamho86/phylolm.

## Value

A named list with the following elements:

| | |
|---|---|
| `loglik` | The maximum log-likelihood. |
| `DiscLik` | The marginal log-likelihood of the discrete state probability for the `nSim` maps. |
| `ContLik` | The marginal log-likelihood of the continuous value probability for the `nSim` maps. |
| `AIC` | Akaike information criterion. |
| `AICc` | Akaike information criterion corrected for sample-size. |
| `BIC` | Bayesian information criterion. |
| `param.count` | Number of parameters in the model. |
| `solution.disc` | The maximum likelihood estiamte of the transition rate matrix. |
| `solution.cont` | The maximum likelihood estimate of parameters which describe continuous evolution. Each column corresponds to a discrete state and each row is a different continuous model parameter. Estimated continuous parameters include alpha, sigma.squared, and theta. |
| `recon` | A data.frame of marginal probabilities of ancestral states given the best fitting model. If ancestral state reconstruction was not specified then `NULL`. |
| `index.disc` | An index matrix specifying which discrete model parameters are estimated. |
| `index.cont` | An index matrix specifying which continuous model parameters are estimated. |
| `phy` | User provided phylogenetic tree. |
| `legend` | A named vector with elements corresponding to input states and names corresponding to a numeric version of those discrete states. Useful for interpreting hOUwie solutions. |

`expected_vals` Expected values given OU parameters averaged over the joint likelihood of the `nSim` regime mappings.

`data` User provided data.frame.

`hOUwie.dat` Internally used data.frame (provided in case a user is concerned about character matching).

`rate.cat` The number of rate categories.

`discrete_model` The user provided index matrix or character string describing the discrete model.

`continuous_model` The user provided index matrix or character string describing the continuous model.

`root.p` The root prior used in model estimation.

`time_slice` Sets how often internodes are sampled. At the moment it is automatically set to be the `max(branching.times(phy))+1`.

`root.station` A boolean indicating whether the starting state, $\theta_0$, was estimated (recommended FALSE).

`get.root.theta` A boolean indicating whether the root.theta was included in the model.

`tip.fog` Either includes tip fog for each species value ("known"). The default is "none".

`sample_tips` A boolean indicating whether tips probabilities were sampled each iteration (recommended FALSE).

`sample_nodes` A boolean indicating whether node probabilities were sampled each iteration (recommended TRUE).

`adaptive_sampling` A boolean indicating whether each iteration continued sampling stochastic maps until the likelihood of a set of parameters did not improve (recommended TRUE).

`lb_discrete_model` A vector of lower bounds on discrete character evolution.

`ub_discrete_model` A vector of upper bounds on discrete character evolution.

`lb_continuous_model` A vector of lower bounds of alpha, sigma square, and theta.

`ub_continuous_model` A vector of lower bounds of alpha, sigma square, and theta.

`p` Vector of maximum likeihood parameters if not given.

`ip` Vector of starting parameters

`nSim` Number of stochastic maps for each iteration.

`opts` List of options input for nloptr optimization.

`quiet` A boolean indicating whether quiet was TRUE or FALSE.

`all_disc_liks` A vector of discrete probabilities in the order of the `nSim` simmaps.

`all_cont_liks` A vector of continuous probabilities in the order of the `nSim` simmaps.

`simmaps` `nSim` stochastic maps from the best fitting hOUwie parameters (to be removed).

`run_time` Time to complete the model fit.

## Author(s)

James D. Boyko

## References

Beaulieu J.M., Jhwueng D.-C., Boettiger C., O'Meara B.C. 2012. Modeling Stabilizing Selection: Expanding the Ornstein–Uhlenbeck Model of Adaptive Evolution. Evolution. 66:2369–2383.

Beaulieu J.M., O'Meara B.C., Donoghue M.J. 2013. Identifying Hidden Rate Changes in the Evolution of a Binary Morphological Character: The Evolution of Plant Habit in Campanulid Angiosperms. Syst Biol. 62:725–737.

Boyko J.D., Beaulieu J.M. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods Ecol Evol. 12:468–478.

Butler M.A., King A.A. 2004. Phylogenetic Comparative Analysis: A Modeling Approach for Adaptive Evolution. The American Naturalist. 164:683–695.

Cressler C.E., Butler M.A., King A.A. 2015. Detecting Adaptive Evolution in Phylogenetic Comparative Analysis Using the Ornstein–Uhlenbeck Model. Systematic Biology. 64:953–968.

Felsenstein J. 2004. Inferring phylogenies. Sunderland MA: Sinauer Associates.

Hansen T.F. 1997. Stabilizing Selection and the Comparative Analysis of Adaptation. Evolution. 51:1341–1351.

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

Hansen T.F. 2014. Use and Misuse of Comparative Methods in the Study of Adaptation. In: Garamszegi L.Z., editor. Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology. Berlin, Heidelberg: Springer Berlin Heidelberg. p. 351–379.

Ho L. si, Ane C. 2014. A Linear-Time Algorithm for Gaussian and Non-Gaussian Trait Evolution Models. Syst Biol. 63:397–408.

---

`hOUwie.fixed` *Fit a joint model of discrete and continuous characters via maximum-likelihood with fixed regimes.*

---

## Description

The hOUwie model jointly estimates the likelihood of a discrete and continuous character by combining the probability of the continuous character given a particular regime and the probability of that discrete regime painting, integrated over many regime paintings. In the standard `hOUwie` function, regimes are generated for each iteration of parameters and then evaluated. However, `hOUwie.fixed` allows users to provide a map or set of maps which will then be optimized based on their joint probability. Most other inputs and outputs remain the same.

This function takes as main input a list of stochastic maps and a data.frame containing species, character values, and potentially measurement error. The rate category is a way to specify the number of hidden states to be incldued, with one meaning no hidden states present. The structure of the model and association between discrete and continuous characters is specified via discrete_model and continuous model (see Details).

## Usage

```
hOUwie.fixed(simmaps,
             data,
             rate.cat,
             discrete_model,
             continuous_model,
             null.model          = FALSE,
             root.p              = "yang",
             dual                = FALSE,
             collapse            = TRUE,
             root.station        = FALSE,
             get.root.theta      = FALSE,
             tip.fog             = "none",
             lb_discrete_model   = NULL,
             ub_discrete_model   = NULL,
             lb_continuous_model = NULL,
             ub_continuous_model = NULL,
             recon               = FALSE,
             nodes               = "internal",
             p                   = NULL,
             ip                  = NULL,
             optimizer           = "nlopt_ln",
             opts                = NULL,
             quiet               = FALSE,
             sample_tips         = FALSE,
             sample_nodes        = TRUE,
             adaptive_sampling   = FALSE,
             diagn_msg           = FALSE,
             make_numeric        = TRUE,
             n_starts            = 1,
             ncores              = 1)
```

## Arguments

| | |
|---|---|
| simmaps | A list of stochastic maps to be evaluated. |
| data | A data frane containing species information. The first column is always species names matching the tip labels of phy. The 2nd to ith columns are independent discrete characters. The ith+1 column is the continuous character value. Finally, there can be an ith+2 column for measurement error (optional). |
| rate.cat | Specifies the number of rate categories (see Details). |
| discrete_model | Either a user-supplied index of parameters to be optimized or one of "ARD", "SYM", or "ER". ARD: all rates differ. SYM: rates between any two states do not differ. ER: all rates are equal. |
| continuous_model | Either a user-supplied index matrix specifying the continuous model parameters to be estimated or one of "BM1", "BMV", "OU1", "OUA", "OUV", "OUM", "OUVA", "OUMV", "OUMA", "OUMVA" (See also getOUParamStructure). |

`null.model` A boolean indicating whether the model being run is a character-independent model with rate heterogeneity. Rate.cat must be greater than 1.

`root.p` a vector used to fix the probabilities at the root, but "yang" can also be supplied to use the method of Yang (2006) (see Details).

`dual` A boolean indicating whether or not to include dual transitions. If true, then transitions two or more states may change at any instant of time. For example, X0Y0 can change directly to X1Y1 without first going through X1Y0 or X0Y1.

`collapse` A boolean indicating whether to collapse multiple character combinations into only the observed states. For example, if true a two character dataset contained (0,0), (1,0), and (1,1), this would be collapsed into 1,2,3. However, if set to false it would 1,2,4. In combination with a custom rate matrix this allows for the estimation of transitions between unobserved character combinations. Transitions to and from unobserved state combinations may not be estimatable: See Boyko and Beaulieu (2022).

`root.station` A boolean indicating whether the starting state, $\theta_0$, should be estimated (see Details of OUwie function).

`get.root.theta` A boolean indicating whether the starting state, $\theta_0$, should be estimated (see Details of OUwie function).

`tip.fog` designates whether a fourth column in the data matrix contains within species variation for each species value ("known"). The tip fog is assumed to be the standard error of the species mean. The default is "none".

`lb_discrete_model`
A single value for the lower bound of discrete character evolution during the ML search. The default is `1/(Tmax*10000)`.

`ub_discrete_model`
A single value for the upper bound of discrete character evolution during the ML search. The default is `1/(Tmax*0.0001)`.

`lb_continuous_model`
Three values specifying the lower bound of continuous character evolution during the ML search. Should be provided in the order of lower.bound for alpha, sigma squared, theta. The default is `c(1e-10, 1e-10, smallest_continuous_value/10)`.

`ub_continuous_model`
Three values specifying the upper bound of continuous character evolution during the ML search. Should be provided in the order of upper.bound for alpha, sigma squared, theta. The default is `c(log(2)/(0.01 * Tmax), log(2)/(0.01 * Tmax), largest_continuous_value*10)`.

`recon` A boolean indicating whether marginal ancestral state reconstruction should take place following the ML search.

`nodes` If `recon=TRUE`, a character or vector indicating which nodes should have their discrete ancestral state characters estimated. If a vector, nodes specified will be reconstructed with 1:Ntip representing tip reconstructions and Ntip+1 representing the root. A user can also indicate "all", "internal", or "external" to reconstruct all nodes, ancestral nodes, or extant nodes respectively. Reconstructing tip states ("external") is only useful if hidden states are included in the model.

`p` A numeric vector of fixed parameters to be optimized. Parameters are organized such following `c(p_trans, p_alpha, p_sigma.squared, p_theta)`

| | |
|---|---|
| ip | A numeric vector indicating initial parameters to start the ML search. Parameters are organized such following c(p_trans, p_alpha, p_sigma.squared, p_theta). |
| optimizer | One of "nlopt_ln", "nlopt_gn", or "sann" for a local search of parameter space, global search of paramater space, or simmulated annealing. The default is "nlopt_ln". |
| opts | A list of options to be passed to nloptr. |
| quiet | A boolean indicating whether or not to print messages. |
| sample_tips | A boolean indicating whether or not to sample tip probabilities when generating underlying regimes (see Deatils, but recommended to be set to FALSE). |
| sample_nodes | A boolean indicating whether or not to sample node probabilities when generating underlying regimes (see Deatils, but recommended to be set to TRUE). |
| adaptive_sampling | A boolean indicating whether each iteration will continue sampling stochastic maps until the likelihood of a set of parameters does not improve (recommended TRUE). |
| diagn_msg | A boolean indiciating whether a message of loglikelihoods and parameters should be printed at each iteration. Can be useful for diagnosing problems. |
| make_numeric | A boolean indicating whether the given simmap needs to be converted to numeric map edges matching observed character data. Setting this to FALSE can be useful if you are interested in looking at hidden states since you will can have more mapped states than observed states. |
| n_starts | An integer, specifying the number of trials for fitting the model, using alternative (randomized) starting parameters at each trial. A larger n_starts reduces the risk of landing on a local non-global optimum of the likelihood function, and thus increases the chances of finding the MLE. |
| ncores | An integer, specifying the number of cores for running multiple fitting n_starts in parallel. Should generally not exceed the number of CPU cores on a machine, but must be a least 1. |

## Details

The difference between hOUwie.fixed and hOUwie is that regimes are provided apriori to hOUwie.fixed and then evaluated. One thing to note is that the discrete probabilities will be treated on a node-by-node basis. That is to say, the exact path probability of the stochastic map is not calculated. Rather, the discrete probability uses the same estimation as hOUwie where we simply evaluate the joint probability of that particular mapping. However, the continuous probability does reflect the exact mapping. This may lead to differences in how hOUwie and hOUwie.fixed preform in empirical situations.

See hOUwie for general details about the hOUwie model.

## Value

A named list with the following elements:

| | |
|---|---|
| loglik | The maximum log-likelihood. |

| | |
|---|---|
| DiscLik | The marginal log-likelihood of the discrete state probability for the nSim maps. |
| ContLik | The marginal log-likelihood of the continuous value probability for the nSim maps. |
| AIC | Akaike information criterion. |
| AICc | Akaike information criterion corrected for sample-size. |
| BIC | Bayesian information criterion. |
| param.count | Number of parameters in the model. |
| solution.disc | The maximum likelihood estiamte of the transition rate matrix. |
| solution.cont | The maximum likelihood estimate of parameters which describe continuous evolution. Each column corresponds to a discrete state and each row is a different continuous model parameter. Estimated continuous parameters include alpha, sigma.squared, and theta. |
| recon | A data.frame of marginal probabilities of ancestral states given the best fitting model. If ancestral state reconstruction was not specified then NULL. |
| index.disc | An index matrix specifying which discrete model parameters are estimated. |
| index.cont | An index matrix specifying which continuous model parameters are estimated. |
| phy | User provided phylogenetic tree. |
| legend | A named vector with elements corresponding to input states and names corresponding to a numeric version of those discrete states. Useful for interpreting hOUwie solutions. |
| expected_vals | Expected values given OU parameters averaged over the joint likelihood of the nSim regime mappings. |
| data | User provided data.frame. |
| hOUwie.dat | Internally used data.frame (provided in case a user is concerned about character matching). |
| rate.cat | The number of rate categories. |
| discrete_model | The user provided index matrix or character string describing the discrete model. |
| continuous_model | The user provided index matrix or character string describing the continuous model. |
| root.p | The root prior used in model estimation. |
| time_slice | Sets how often internodes are sampled. At the moment it is automatically set to be the max(branching.times(phy))+1. |
| root.station | A boolean indicating whether the starting state, $\theta_0$, was estimated (recommended FALSE). |
| get.root.theta | A boolean indicating whether the root.theta was included in the model. |
| tip.fog | Either includes measurement error for each species value ("known"). The default is "none". |
| sample_tips | A boolean indicating whether tips probabilities were sampled each iteration (recommended FALSE). |

`sample_nodes` A boolean indicating whether node probabilities were sampled each iteration (recommended TRUE).

`adaptive_sampling` A boolean indicating whether each iteration continued sampling stochastic maps until the likelihood of a set of parameters did not improve (recommended TRUE).

`lb_discrete_model` A vector of lower bounds on discrete character evolution.

`ub_discrete_model` A vector of upper bounds on discrete character evolution.

`lb_continuous_model` A vector of lower bounds of alpha, sigma square, and theta.

`ub_continuous_model` A vector of lower bounds of alpha, sigma square, and theta.

`p` Vector of maximum likeihood parameters if not given.

`ip` Vector of starting parameters

`nSim` Number of stochastic maps for each iteration.

`opts` List of options input for nloptr optimization.

`quiet` A boolean indicating whether quiet was TRUE or FALSE.

`all_disc_liks` A vector of discrete probabilities in the order of the `nSim` simmaps.

`all_cont_liks` A vector of continuous probabilities in the order of the `nSim` simmaps.

`simmaps` `nSim` stochastic maps from the best fitting hOUwie parameters (to be removed).

`run_time` Time to complete the model fit.

## Author(s)

James D. Boyko

## References

Beaulieu J.M., Jhwueng D.-C., Boettiger C., O'Meara B.C. 2012. Modeling Stabilizing Selection: Expanding the Ornstein–Uhlenbeck Model of Adaptive Evolution. Evolution. 66:2369–2383.

Beaulieu J.M., O'Meara B.C., Donoghue M.J. 2013. Identifying Hidden Rate Changes in the Evolution of a Binary Morphological Character: The Evolution of Plant Habit in Campanulid Angiosperms. Syst Biol. 62:725–737.

Boyko J.D., Beaulieu J.M. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods Ecol Evol. 12:468–478.

Butler M.A., King A.A. 2004. Phylogenetic Comparative Analysis: A Modeling Approach for Adaptive Evolution. The American Naturalist. 164:683–695.

Hansen T.F. 1997. Stabilizing Selection and the Comparative Analysis of Adaptation. Evolution. 51:1341–1351.

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

Ho L. si, Ane C. 2014. A Linear-Time Algorithm for Gaussian and Non-Gaussian Trait Evolution Models. Syst Biol. 63:397–408.

`hOUwie.recon` *Reconstruct the marginal probability of discrete node states under the hOUwie model.*

## Description

Uses the output object from `hOUwie` to conduct an ancestral state reconstruction of the discrete regime states.

## Usage

```
hOUwie.recon(houwie_obj,
             nodes      = "all")
```

## Arguments

| | |
|---|---|
| `houwie_obj` | A list of class "houwie". |
| `nodes` | One of "internal", "external", or "all" to represent reconstructing internal nodes, external (tips) nodes, or all nodes. "External" is only useful if the model has hidden states or there is uncertainty in the tip states. |

## Details

Reconstructs the marginal probability of a particular state at a given node. To do this, all possible states at the given node are fixed and the joint likelihoods are evaluated for each possibility. Those joint probabilities are then normalized by the total likelihood so that the probability sums to one.

## Value

| | |
|---|---|
| `recon_matrix` | a data.frame containing the marginal probabilities of each state (given by the column). |

## Author(s)

James D. Boyko

## References

Yang Z. 2006. Computational molecular evolution. Oxford University Press Oxford.

`hOUwie.sim` *Simulate a discrete and continuous character following a Markov and Ornstein-Uhlenbeck model.*

## Description

Simulates a discrete and continuous character following the hOUwie model. The function first evolves a discrete character based on the Q matrix provided. Next, it will evolve a continuous character following the given OU parameters and simulated discrete character. Like the hOUwie model, transitions between discrete states are assumed to take place half-way between nodes.

## Usage

```
hOUwie.sim(phy,
           Q,
           root.freqs,
           alpha,
           sigma.sq,
           theta0,
           theta)
```

## Arguments

| | |
|---|---|
| `phy` | A phylogenetic tree, in `ape` "phylo" format. |
| `Q` | A transition rate matrix with dimensions `nStates` by `nStates` describing rates of change between discrete states. |
| `root.freqs` | A vector `nStates` long with probabilities of the root being in a particular state. For example, for a binary discrete character a root prior of `c(1, 0)` would fix the root state in state 1. |
| `alpha` | A vector `nStates` long which gives alpha parameter of the OU model. For a BM model set this to be near 0. |
| `sigma.sq` | A vector `nStates` long which gives the evolutionary rate parameter of the OU model. |
| `theta0` | A numeric value giving the starting value of the continuous character at the root. |
| `theta` | A vector `nStates` long which gives the phenotypic optima for each regime state. |

## Details

The simulation protocol follows the hOUwie model where stochastic maps being produced are based on a node-by-node simulation. I.e., we first simulate the node states given the parameters and then branches are paintined based on transitions occuring half-way between the nodes.

## Value

| | |
|---|---|
| `data` | a dataframe of sp (species), reg (discrete character states), and x (continuous character values). |
| `simmap` | the history of the discrete character presented as a stochastic map. |

### Author(s)

James D. Boyko

### Examples

```
data(tworegime)
# simulate an OUM model
Q <- matrix(c(-1,1,1,-1), 2, 2)
root.freqs <- c(1, 0)
alpha <- c(2, 2)
sigma.sq <- c(1,1)
theta0 <- 5
theta <- c(5, 10)

simulated_data <- hOUwie.sim(tree, Q, root.freqs, alpha, sigma.sq, theta0, theta)
```

---

## hOUwie.thorough *Rerun a set of hOUwie models with the best mappings of the set.*

---

### Description

This function takes as input a list of hOUwie fits and will automatically model rerun the model set using the best mappings. Given the complexity of a hOUwie model, it is often beneficial to run this to ensure the MLE has been found.

### Usage

```
hOUwie.thorough(model.list, ncores = 1)
```

### Arguments

| | |
|---|---|
| `model.list` | A list of model results from several fits of the `hOUwie` model. All elements of the list must be of class "houwie". |
| `ncores` | An integer, specifying the number of cores for running multiple fitting n_starts in parallel. Should generally not exceed the number of CPU cores on a machine, but must be a least 1. |

### Details

This function will find the nSim best unique mappings from the input model set and rerun all models based on those mappings alone. Unlike the usual hOUwie, the mappings themselves are not infered and take as given. This makes this function most similar to hOUwie.fixed.

### Value

Returns a new model list with maps from the input list.

## Author(s)

James D. Boyko

---

`hOUwie.walk` *Sample points from along a ridge for a hOUwie model*

---

## Description

This "dents" the likelihood surface by reflecting points better than a threshold back across the threshold (think of taking a hollow plastic model of a mountain and punching the top so it's a volcano). It then uses essentially a Metropolis-Hastings walk to wander around the new rim. It adjusts the proposal width so that it samples points around the desired likelihood. This is better than using the curvature at the maximum likelihood estimate since it can actually sample points in case the assumptions of the curvature method do not hold. It is better than varying one parameter at a time while holding others constant because that could miss ridges: if I am fitting 5=x+y, and get a point estimate of (3,2), the reality is that there are an infinite range of values of x and y that will sum to 5, but if I hold x constant it looks like y is estimated very precisely. Of course, one could just fully embrace the Metropolis-Hastings lifestyle and use a full Bayesian approach.

While running, it will display the current range of likelihoods in the desired range (by default, the best negative log likelihood + 2 negative log likelihood units) and the parameter values falling in that range. If things are working well, the range of values will stabilize during a search.

## Usage

```
hOUwie.walk(houwie_obj,
            delta                = 2,
            nsteps               = 1000,
            print_freq           = 50,
            lower_bound          = 0,
            upper_bound          = Inf,
            adjust_width_interval = 100,
            badval               = 1e9,
            sd_vector            = NULL,
            debug                = FALSE,
            restart_after        = 50)
```

## Arguments

| | |
|---|---|
| `houwie_obj` | A list of class "houwie". |
| `delta` | How far from the optimal negative log likelihood to focus samples. |
| `nsteps` | How many steps to take in the analysis. |
| `print_freq` | Output progress every print_freq steps. |
| `lower_bound` | Minimum parameter values to try. One for all or a vector of the length of par. |
| `upper_bound` | Maximum parameter values to try. One for all or a vector of the length of par. |

adjust_width_interval
: When to try automatically adjusting proposal widths.

badval
: Bad negative log likelihood to return if a non-finite likelihood is returned.

sd_vector
: Vector of the standard deviations to use for proposals. Generated automatically if NULL.

debug
: If TRUE, prints out much more information during a run.

restart_after
: Sometimes the search can get stuck outside the good region but still accept moves. After this many steps without being inside the good region, restart from one of the past good points.

## Details

The algorithm tunes: if it is moving too far away from the desired likelihoods, it will decrease the proposal width; if it staying in areas better than the desired likelihood, it will increase the proposal width. It will also expand the proposal width for parameters where the extreme values still appear good enough to try to find out the full range for these values.

In general, the idea of this is not to give you a pleasingly narrow range of possible values – it is to try to find the actual uncertainty, including finding any ridges that would not be seen in univariate space.

## Value

A dentist object containing results, the data.frame of negative log likelihoods and the parameters associated with them; acceptances, the vector of whether a proposed move was accepted each step; best_neglnL, the best value passed into the analysis; delta, the desired offset; all_ranges, a summary of the results.

## Examples

```
# ## Finally, it is important to get estimates of your parameter uncertainty.
# ## For that we have imported the functions of Brian O'Meara's dentist package
# data(tworegime)
#
#
# ##Third step is to fit our model (here we are using fixed params from a previous ML search):
# p <- c(0.01664314, 0.39631218, 0.18684476, 2.25121568, 0.82495093) # MLE
# pp_cd <- hOUwie(tree, trait, rate.cat = 1, discrete_model = "ER",
#                 continuous_model = "OUM", nSim = 25, p = p)
#
# ## We can compare this to the result from OUwie on a fixed map to see a difference
# dent_res <- hOUwie.walk(pp_cd, nsteps = 25, debug =TRUE)
# plot(dent_res)
```

OUwie *Generalized Hansen models*

## Description

Fits generalized Ornstein-Uhlenbeck-based Hansen models of continuous characters evolving under discrete selective regimes.

## Usage

```
OUwie(phy, data, model=c("BM1","BMS","OU1","OUM","OUMV","OUMA","OUMVA",
"TrendyM","TrendyMS"), simmap.tree=FALSE, root.age=NULL,scaleHeight=FALSE,
root.station=FALSE, get.root.theta=FALSE, shift.point=0.5, clade=NULL, tip.fog="none",
starting.vals=NULL, check.identify=TRUE, algorithm=c("invert", "three.point"),
diagn=FALSE, quiet=FALSE, warn=TRUE, lb = NULL, ub = NULL, opts = list(algorithm =
"NLOPT_LN_SBPLX", maxeval = "1000", ftol_rel = .Machine$double.eps^0.5))
```

## Arguments

| | |
|---|---|
| phy | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes. |
| data | a data.frame containing species information (see Details). |
| model | models to fit to comparative data (see Details). |
| simmap.tree | a logical indicating whether the input tree is in SIMMAP format. The default is FALSE. |
| root.age | indicates the age of the tree. This is to be used in cases where the "tips" are not contemporary, such as in cases for fossil trees. Default is NULL meaning latest tip is modern day. |
| scaleHeight | a logical indicating whether the total tree height should be scaled to 1 (see Details). The default is FALSE. |
| root.station | a logical indicating whether to assume a random starting point (TRUE) or a fixed starting point (FALSE) (see Details). |
| get.root.theta | a logical indicating whether the starting state, $\theta_0$, should be estimated (see Details). |
| shift.point | the point along a branch where a regime change is assumed to have occurred (if SIMMAP=FALSE. The default is set to 0.5, or halfway along a branch. |
| clade | a list containing a pair of taxa whose MRCA is the clade of interest (see Details). |
| tip.fog | designates whether a fourth column in the data matrix contains within species variation for each species value ("known"). The tip fog is assumed to be the standard error of the species mean. The default is "none", but can be estimated by setting the option to "estimate". |
| starting.vals | a vector of initial values for the optimization search. For OU models, two must be supplied, with the first being the initial alpha value and the second being the initial sigma squared. For BM models, just a single value is needed. |

| | |
|---|---|
| check.identify | a logical indicating whether to check that the user-supplied regime paintings will produce identifiable theta estimates. The default is TRUE. |
| algorithm | designates whether the standard matrix inversion ('invert') or the faster 'three-point' algorithm of Ho and Ane (2013) should be used. |
| diagn | a logical indicating whether the full diagnostic analysis should be carried out. The default is FALSE. |
| quiet | a logical indicating whether progress should be written to the screen. The default is FALSE. |
| warn | a logical indicating whether a warning should be printed if the number of parameters exceeds ntips/10. The default is TRUE. |
| lb | if algorithm == "invert" a single value indicating the lower bound for the parameter values. if algorithm == "three.point", a vector of length 3 with position 1 as alpha lower bound, position 2 as sigma.sq's lower bound, position 3 as theta's lower bound. when set to NULL it will be a default value of 1e-9. Note that even if the model you're using doesn't include alpha (e.g. BM1), it must be included in this vector, but it will not be used to set any bounds. |
| ub | if algorithm == "invert" a single value indicating the upper bound for the parameter values. if algorithm == "three.point", a vector of length 3 with position 1 as alpha upper bound, position 2 as sigma.sq's upper bound, position 3 as theta's upper bound. when set to NULL it will be a default value of 100. Note that even if the model you're using doesn't include alpha (e.g. BM1), it must be included in this vector, but it will not be used to set any bounds. |
| opts | a list of options to pass to nloptr for the optimization: useful to adjust for faster, coarser searches |

## Details

This function fits various likelihood models for continuous characters evolving under discrete selective regimes. The function returns parameter estimates and their approximate standard errors. The R package nloptr provides a common interface to NLopt, an open-source library for nonlinear optimization. The likelihood function is maximized using the bounded subplex optimization routine (NLOPT_LN_SBPLX). As input all OUwie requires is a tree and a trait data.frame. The tree must be of class "phylo" and must contain the ancestral selective regimes as internal node labels. Internal node labels can be applied manually or from some sort of ancestral state reconstruction procedure (BayesTraits, ape, diversitree, SIMMAP, etc.), which would then be brought into OUwie. This is essentially what is required by ouch and Brownie (though Brownie provides built-in ancestral state reconstruction capabilities). The trait data.frame must have column entries in the following order: [,1] species names, [,2] current selective regime, and [,3] the continuous trait of interest. Alternatively, if the user wants to incorporate tip.fog (tip.fog="known"), then a fourth column, [,4] must be included that provides the standard error estimates for each species mean. However, a global tip.fog for all taxa can be estimated from the data (tip.fog="estimate"). Also, a user can specify a particular clade as being in a different selective regime, by specifying a pair of species whose mrca is the root of the clade of interest [e.g., clade=c("taxaA","taxaB")]. OUwie will automatically assign internal node labels and update the data matrix according to this clade designation.

Possible models are as follows: single-rate Brownian motion (model=BM1), Brownian motion with different rate parameters for each state on a tree (model=BMS), Ornstein-Uhlenbeck model with a

single optimum for all species (model=OU1), Ornstein-Uhlenbeck model with different state means and a single $\alpha$ and $\sigma^2$ acting all selective regimes (model=OUM), and new Ornstein-Uhlenbeck models that assume different state means as well as either multiple $\sigma^2$ (model=OUMV), multiple $\alpha$ (model=OUMA), or multiple $\alpha$ and $\sigma^2$ per selective regime (model=OUMVA).

By default, we drop the root optima and absorb the weight into whatever regime the root is in. In previous version we used to incorrectly refer to this as "stationarity". True stationarity assumes that the starting state comes from a distribution, and the covariance requires an additional variance term to account for the fact that, up until T=0, the lineage is assumed to have been evolving in the ancestral regime. We have added this in for the OU1 and OUM models only (root.station=TRUE).

Note, too, that when specifying the BMS model also be mindful of the root.station flag. When root.station=FALSE, the non-censored model of O'Meara et al. 2006 is invoked (i.e., a single regime at the root is estimated), and when root.station==TRUE the group mean model of Thomas et al. 2006 (i.e., the number of means equals the number of regimes). The latter case appears to be a strange special case of OU, in that it behaves similarly to the OUMV model, but without selection. I would say that this is more consistent with the censored test of O'Meara et al. (2006), as opposed to having any real connection to OU. In any case, more work is clearly needed to understand the behavior of the group means model, and therefore, I recommend setting root.station=FALSE in the BMS case.

The Hessian matrix is used as a means to estimate the approximate standard errors of the model parameters and to assess whether they are the maximum likelihood estimates. The variance-covariance matrix of the estimated values of $\alpha$ and $\sigma^2$ are computed as the inverse of the Hessian matrix and the standard errors are the square roots of the diagonals of this matrix. The Hessian is a matrix of second-order derivatives and is approximated in the R package numDeriv. So, if changes in the value of a parameter results in sharp changes in the slope around the maximum of the log-likelihood function, the second-order derivative will be large, the standard error will be small, and the parameter estimate is considered stable. On the other hand, if the second-order derivative is nearly zero, then the change in the slope around the maximum is also nearly zero, indicating that the parameter value can be moved in any direction without greatly affecting the log-likelihood. In such situations, the standard error of the parameter will be large.

For models that allow $\alpha$ and $\sigma^2$ to vary (i.e., OUMV, OUMA, and OUMVA), the complexity of the model can often times be greater than the information that is contained within the data. As a result one or many parameters are poorly estimated, which can cause the function to return a log-likelihood that is suboptimal. This has great potential for poor model choice and incorrect biological interpretations. An eigendecomposition of the Hessian can provide an indication of whether the search returned the maximum likelihood estimates. If all the eigenvalues of the Hessian are positive, then the Hessian is positive definite, and all parameter estimates are considered reliable. However, if there are both positive and negative eigenvalues, then the objective function is at a saddlepoint and one or several parameters cannot be estimated adequately. One solution is to just fit a simpler model. Another is to actually identify the offending parameters. This can be done through the examination of the eigenvectors. The row order corresponds to the entries in index.matrix, the columns correspond to the order of values in eigval, and the larger the value of the row entry the greater the association between the corresponding parameter and the eigenvalue. Thus, the largest values in the columns associated with negative eigenvalues are the parameters that are causing the objective function to be at a saddlepoint.

## Value

OUwie returns an object of class OUwie. This is a list with elements:

| | |
|---|---|
| `$loglik` | the maximum log-likelihood. |
| `$AIC` | Akaike information criterion. |
| `$AICc` | Akaike information criterion corrected for sample-size. |
| `$BIC` | Bayesian information criterion. |
| `$mBIC` | modified Bayesian information criterion of Ho and Ane (2014). |
| `$model` | The model being fit. |
| `$param.count` | The number of parameters counted in the model. |
| `$solution` | a matrix containing the maximum likelihood estimates of $\alpha$ and $\sigma^2$. |
| `$theta` | a matrix containing the maximum likelihood estimates of $\theta$ and its standard error. |
| `$solution.se` | a matrix containing the approximate standard errors of $\alpha$ and $\sigma^2$. The standard error is calculated as the diagonal of the inverse of the Hessian matrix. |
| `$tot.state` | A vector of names for the different regimes |
| `$index.mat` | The indices of the parameters being estimated are returned. The numbers correspond to the row in the `eigvect` and can useful for identifying the parameters that are causing the objective function to be at a saddlepoint (see Details). |
| `$simmap.tree` | A logical indicating whether the input phylogeny is a SIMMAP formatted tree. |
| `$root.age` | The user-supplied age at the root of the tree. |
| `$split.point` | The user-supplied point at which regime changes are assumed to have occurred. |
| `$opts` | Internal settings of the likelihood search. |
| `$data` | User-supplied dataset. |
| `$phy` | User-supplied tree. |
| `$root.station` | A logical indicating whether the starting state, $\theta_0$, was estimated. |
| `$starting.vals` | A vector of user-supplied initial search parameters. |
| `$lb` | The lower bound set. |
| `$ub` | The upper bound set. |
| `$iterations` | Number of iterations of the likelihood search that were executed. |
| `$get.root.theta` | Indicates whether the root.theta was included in the model. |
| `$regime.weights` | A table containing parameter estimates and the weights for time spent in each regime for each tip. |
| `$eigval` | The eigenvalues from the decomposition of the Hessian of the likelihood function. If any `eigval<0` then one or more parameters were not optimized during the likelihood search (see Details). |
| `$eigvect` | The eigenvectors from the decomposition of the Hessian of the likelihood function is returned (see Details). |
| `$new.start` | The vector of values to use if you want to restart the run from this point (starting.vals for a new run). |
| `$algorithm` | The algorithm used to estimate parameters |

## Author(s)

Jeremy M. Beaulieu and Brian C. O'Meara

## References

Beaulieu J.M., Jhwueng D.C., Boettiger C., and O'Meara B.C. 2012. Modeling stabilizing selection: Expanding the Ornstein-Uhlenbeck model of adaptive evolution. Evolution 66:2369-2383.

O'Meara B.C., Ane C., Sanderson P.C., Wainwright P.C. 2006. Testing for different rates of continuous trait evolution using likelihood. Evolution 60:922-933.

Butler M.A., King A.A. 2004. Phylogenetic comparative analysis: A modeling approach for adaptive evolution. American Naturalist 164:683-695.

Ho, L.S.T., and C. Ane. 2014. Intrinsic inference difficulties for trait evolution with Ornstein-Uhlenbeck models. Methods in Ecology and Evolution, 5: 1133-1146.

Thomas G.H., Freckleton R.P., and Szekely T. 2006. Comparative analysis of the influence of developmental mode on phenotypic diversification rates in shorebirds. Proceedings of the Royal Society, B. 273:1619-1624.

## Examples

```
data(tworegime)

#Plot the tree and the internal nodes to highlight the selective regimes:
select.reg<-character(length(tree$node.label))
select.reg[tree$node.label == 1] <- "black"
select.reg[tree$node.label == 2] <- "red"
plot(tree)
nodelabels(pch=21, bg=select.reg)



## Not run:
#To see the first 5 lines of the data matrix to see what how to
#structure the data:
trait[1:5,]

#Now fit an OU model that allows different sigma^2:
OUwie(tree,trait,model=c("OUMV"))

#Fit an OU model based on a clade of interest:
OUwie(tree,trait,model=c("OUMV"), clade=c("t50", "t64"), algorithm="three.point")

#For large trees, it may be useful to have ways to restart the search (due to
#finite time per run on a computing cluster, for example). You can do this
#by changing settings of OUwie runs. For example:

run1 <- OUwie(tree,trait,model=c("OUMV"), root.station=FALSE, algorithm="invert",
opts = list("algorithm"="NLOPT_LN_SBPLX", "maxeval"="500", "ftol_abs"=0.001))

save(run1, file="run1.rda")
```

```
#Then, later or in a different session:

load("run1.rda")

run2 <- OUwie(tree, trait, model=c("OUMV"), algorithm="three.point",
opts = list("algorithm"="NLOPT_LN_SBPLX", "maxeval"="500", "ftol_abs"=0.001),
starting.vals=run1$new.start)

#run2 will start off where run1 stopped.

## End(Not run)
```

---

OUwie.anc *Estimate ancestral states given a fitted OUwie model*

---

## Description

Fits ancestral states (a joint estimate) given a fitted OUwie model. Currently, only works for trees with regimes painted on as node labels (rather than a simmap tree). The intended use case is just to visualize what the model is saying about evolution to help intuition (is the model something you can believe in?) rather than a firm estimate you should use to say, Yes, 56.4 MY, the ancestral body size was 17.34 mm. It could likely be anywhere from 1 to 100 mm with about equal chance. Please read details before using this function.

## Usage

```
OUwie.anc(fitted.OUwie.object, opts = list("algorithm"="NLOPT_LN_BOBYQA",
"maxeval"="1000", "ftol_abs"=0.001), knowledge=FALSE, multiple_starts=1)
```

## Arguments

| | |
|---|---|
| `fitted.OUwie.object` | an object returned from the OUwie function |
| `opts` | a list of options to pass to nloptr for the ancestral state optimization |
| `knowledge` | a logical indicating whether or not you have read the documentation The default is FALSE. |
| `multiple_starts` | an integer indicating how many times to start the optimization. The default is 1. |

## Details

You probably DON'T want to use this function for anything more serious than poking around to make sure the data and model look right (oh, golly: my tips are all between 5 and 15 mm in size, and the ancestral states are 674 mm.). The request to implement ancestral state estimation resulted in many important comments from experts in the public R-SIG-PHYLO discussion forum. Here is a sampling; to see them all, go to R-SIG-PHYLO.

"So in short, yes, you can do it, with any number of methods. But why? If you can answer your biological question with methods that do not involve estimation of a parameter that is inherently

fraught with error, it might be better to go another way. Bottom line - use caution and be thoughtful!" – Marguerite Butler

"I would add an extra caveat to Marguerite's excellent post: Most researchers work with extant taxa only, ignoring extinction. This causes a massive ascertainment bias, and the character states of the extinct taxa can often be very different to the ancestral state reconstructions, particularly if the evolutionary model is wrong. Eg. there has been an evolutionary trend for example. Ancestral state reconstructions based only on extant taxa should be treated as hypotheses to be tested with fossil data. I wouldn't rely on them for much more." – Simone Blomberg

"While I am at it, let me echo Simone and Marguerite's warnings. The predicted ancestral states will reflect the process you assumed to predict them. Hence, if you use them to make inferences about evolution, you will recover your own assumptions. I.e. if you predict from a model with no trend, you will find no trend, etc. Many comparative studies are flawed for this reason." – Thomas Hansen

"Let me add more warnings to Marguerite and Thomas's excellent responses. People may be tempted to infer ancestral states and then treat those inferences as data (and also to infer ancestral environments and then treat those inferences as data). In fact, I wonder whether that is not the main use people make of these inferences. But not only are those inferences very noisy, they are correlated with each other. So if you infer the ancestral state for the clade (Old World Monkeys, Apes) and also the ancestral state for the clade (New World Monkeys, (Old World Monkeys, Apes)) the two will typically not only be error-prone, but will also typically be subject to strongly correlated errors. Using them as data for further inferences is very dubious. It is better to figure out what your hypothesis is and then test it on the data from the tips of the tree, without the intermediate step of taking ancestral state inferences as observations. The popular science press in particular demands a fly-on-the-wall account of what happened in evolution, and giving them the ancestral state inferences as if they were known precisely is a mistake." – Joe Felsenstein

"The minor twist I would throw in is that it's difficult to make universal generalizations about the quality of ancestral state estimation. If one is interested in the ancestral state value at node N, it might be reasonably estimated if it is nested high up within the phylogeny, if the rates of change aren't high, etc. And (local) trends etc might well be reliably inferred. We are pretty confident that the common ancestor of humans and chimps was larger than many deeper primate ancestors, for instance. If N is the root of your available phylogeny, however, you have to be much more cautious." – Nick Matzke

"I'll also add that I think there's a great deal to be skeptical of ancestral trait reconstruction even when large amounts of fossil data is available. You can try the exercise yourself: simulate pure BM on a non-ultrametric tree with lots of 'extinct' tips, and you'll still find pretty large confidence intervals on the estimates of the trait values. What does it mean to do ancestral trait reconstruction, if our calculations of uncertainty are that broad?" – Dave Bapst

These are some of the people who best know the power and limitations of the OU model in phylogenetics. Heed them!

To ensure that you've read this before use, please pass knowledge=TRUE as an argument to the function.

## Value

`OUwie.anc` returns an object of class `OUwie.anc`. This is an OUwie object but with terminal species added at each node representing the ancestral states at each node (which are also included in the

data object). There is also a NodeRecon element in the list that has the optimal ancestral states. There is not currently an estimate of uncertainty, but it is substantial.

## Author(s)

Brian C. O'Meara

---

OUwie.boot *Parametric bootstrap function*

---

## Description

A function that performs a parametric bootstrap for a set of user-specified model parameters

## Usage

```
OUwie.boot(phy, data, model=c("BM1","BMS","OU1","OUM","OUMV","OUMA","OUMVA"),
 nboot=100, alpha, sigma.sq, theta, theta0, simmap.tree=FALSE, root.age=NULL,
 scaleHeight=FALSE, root.station=FALSE, get.root.theta=FALSE, shift.point=0.5,
 clade=NULL, tip.fog="none", algorithm=c("invert", "three.point"),
 diagn=FALSE, quiet=TRUE, warn=FALSE)
```

## Arguments

| | |
|---|---|
| phy | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes. |
| data | a data matrix containing species information. |
| model | models to fit to comparative data. |
| nboot | The number of bootstrap replicates. |
| alpha | a numeric vector giving the values of $\alpha$ for each selective regime. |
| sigma.sq | a numeric vector giving the values of $\sigma^2$ for each selective regime. |
| theta | a numeric vector giving the values of $\theta$ for each selective regime. |
| theta0 | a numeric indicating the starting state, $\theta_0$ |
| simmap.tree | a logical indicating whether the input tree is in SIMMAP format. The default is FALSE. |
| root.age | indicates the age of the tree. This is to be used in cases where the "tips" are not contemporary, such as in cases for fossil trees. Default is NULL meaning latest tip is modern day. |
| get.root.theta | a logical indicating whether the starting state, $\theta_0$, should be estimated (see Details). |
| scaleHeight | a logical indicating whether the total tree height should be scaled to 1 (see Details). The default is FALSE. |
| root.station | a logical indicating whether the starting state, $\theta_0$, should be estimated (see Details). |

| | |
|---|---|
| shift.point | the point along a branch where a regime change is assumed to have occurred (if SIMMAP=FALSE. The default is set to 0.5, or halfway along a branch. |
| clade | a list containing a pair of taxa whose MRCA is the clade of interest. |
| tip.fog | designates whether a fourth column in the data matrix contains tip fog for each species value ("known"). The tip fog is assumed to be the standard error of the species mean. The default is "none". |
| algorithm | designates whether the standard matrix inversion ('invert') or the faster 'three-point' algorithm of Ho and Ane (2013) should be used. |
| diagn | a logical indicating whether the full diagnostic analysis should be carried out. The default is FALSE. |
| quiet | a logical indicating whether progress should be written to the screen. The default is TRUE. |
| warn | a logical indicating whether a warning should be printed if the number of parameters exceeds ntips/10. The default is FALSE. |

## Details

A simple function for conducting a parametric bootstrap on parameters estimated in OUwie. As before, the input is a tree and a data file. The tree must be of class "phylo" and if simmap=FALSE must contain the ancestral selective regimes as internal node labels. The data file is a dataframe that must have column entries in the following order: [,1] species names and [,2] their current selective regime. The user specifies the simulated parameter values (i.e. $\alpha$, $\sigma^2$, $\theta_0$, $\theta$), which is assumed to be the maximum likelihood estimates obtained from an OUwie run.

Note that if root.station is TRUE (the default), $\theta_0$ was dropped from the model. In this case, then, $\theta_0$ should be set to the value of the selective regime mapped at the root (i.e., state 1 in the "tworegime" example dataset).

## Value

OUwie.boot returns an object of class OUwie.boot. This is a matrix of column length equal to the number of parameters, and row length of the number of bootstrap replicates specified.

## Author(s)

Jeremy M. Beaulieu

## References

Beaulieu J.M., Jhwueng D.C., Boettiger C., and O'Meara B.C. 2012. Modeling stabilizing selection: Expanding the Ornstein-Uhlenbeck model of adaptive evolution. Evolution 66:2369-2383.

O'Meara B.C., Ane C., Sanderson P.C., Wainwright P.C. 2006. Testing for different rates of continuous trait evolution using likelihood. Evolution 60:922-933.

Butler M.A., King A.A. 2004. Phylogenetic comparative analysis: A modeling approach for adaptive evolution. American Naturalist 164:683-695.

## Examples

```
data(tworegime)

##First step is estimate parameters under a particular model:
pp <- OUwie(tree,trait,model=c("OUMV"),root.station=FALSE, algorithm="three.point")

##Second step is to run bootstrap replicates:
boot.reps <- OUwie.boot(tree,trait,model="OUMV", nboot=10, alpha=pp$solution[1,],
sigma.sq=pp$solution[2,],theta=pp$theta[,1], theta0=pp$theta[1,1],
algorithm="three.point")

##Finally summarize to obtain the desired confidence -- here is the 95% CI:
apply(boot.reps, 2, quantile, probs=c(0.025,0.975))
```

---

`OUwie.contour` *Generates data for contour plot of likelihood surface*

---

## Description

Generates the likelihood surface for pairs of free parameters for generalized Ornstein-Uhlenbeck-based Hansen models of continuous characters evolving under discrete selective regimes.

## Usage

```
OUwie.contour(OUwie.obj, focal.params=c("alpha_1", "sigma.sq_1"),
focal.params.lower=c(0,0), focal.params.upper=c(5,5), nreps=1000, n.cores=NULL)
```

## Arguments

| | |
|---|---|
| `OUwie.obj` | an object of class "OUwie" that contains the focal parameters for conducting the likelihood surface search. |
| `focal.params` | a vector specifying the parameters that you would like the likelihood surface. The format is parameter_regime – that is, for theta in regime 1, the input would be "theta_1". |
| `focal.params.lower` | a vector specifying the lower bounds for the parameters. The values need to be in the order as the focal parameters. |
| `focal.params.upper` | a vector specifying the upper bounds for the parameters. The values need to be in the order as the focal parameters. |
| `nreps` | the number points to use to estimate the likelihood surface (see Details). |
| `n.cores` | specifies the number of independent processors to conduct the analysis. The default is `NULL`. |

## Details

This function samples a set of points to estimate the likelihood surface for any pair of parameters, letting the other parameters find their own optima. This process can be very slow, as it involves optimization nrep times (though with two fewer parameters than with the chosen model, as the focal parameter values are fixed). It uses a latin hypercube design to sample points across the user-defined range of the focal parameters.

The pair of parameters to examine is passed by focal.param. The parameters need to be one of three: theta, alpha, sigma.sq. For example, to do a plot of sigma.sq from the first regime and alpha from the second regime, one would pass `focal.param = c( "sigma.sq_1", "alpha_2")`. As another example, if the regimes are characters like, flower color, the focal parameter would be `focal.param = c( "sigma.sq_Red", "sigma.sq_Blue")`.

This returns a data.frame with the last two columns being the values of the points examined and the first column the loglik of those points. The first row contains the MLE. The data.frame can be incorporated into a plotting function to obtain a contour plot (see plot.OUwie.contour).

## Value

`surface.data` the parameter values and loglik

`focal.params` the vector specifying the parameter pair for which likelihood surface is evaluated

`focal.params.lower` the vector specifying the lower bounds for the parameter pair.

`focal.params.upper` the vector specifying the upper bounds for the parameter pair.

## Author(s)

Jeremy M. Beaulieu

## References

Beaulieu J.M., Jhwueng D.C., Boettiger C., and O'Meara B.C. 2012. Modeling stabilizing selection: Expanding the Ornstein-Uhlenbeck model of adaptive evolution. Evolution 66:2369-2383.

---

`OUwie.dredge` *Generalized Detection of shifts in OU process*

---

## Description

Allows the hypothesis free detection of shifts in the OU process. The number and location of shifts is estimated using a user-specified information criterion.

## Usage

```
OUwie.dredge(phy, data, criterion=c("AIC", "AICc", "BIC", "mBIC"), shift.max=3,
sigma.sq.max.k=3, alpha.max.k=3, root.age=NULL, scaleHeight=FALSE, root.station=FALSE,
shift.point=0.5, tip.fog="none", algorithm=c("invert", "three.point"), lb=NULL, ub=NULL,
opts = list("algorithm"="NLOPT_LN_SBPLX", "maxeval"="1000",
"ftol_rel"=.Machine$double.eps^0.5), verbose=FALSE)
```

## Arguments

| | |
|---|---|
| phy | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes |
| data | a dataframe containing two columns, taxon names in the first column, and species trait information in the second column. |
| criterion | information criterion to use for shift detection. |
| shift.max | maximum allowed number of shifts. |
| sigma.sq.max.k | maximum allowed number of sigma.sq parameters. |
| alpha.max.k | maximum allowed number of alpha parameters. |
| root.age | indicates the age of the tree. This is to be used in cases where the "tips" are not contemporary, such as in cases for fossil trees. Default is NULL meaning latest tip is modern day. |
| scaleHeight | a logical indicating whether the total tree height should be scaled to 1. The default is FALSE. |
| root.station | a logical indicating whether the starting state, $\theta_0$, should be estimated. |
| shift.point | the point along a branch where a regime change is assumed to have occurred (if SIMMAP=FALSE. The default is set to 0.5, or halfway along a branch. |
| tip.fog | designates whether a fourth column in the data matrix contains tip fog for each species value ("known"). The tip fog is assumed to be the standard error of the species mean. The default is "none". |
| algorithm | designates whether the standard matrix inversion ('invert') or the faster 'three-point' algorithm of Ho and Ane (2013) should be used. |
| lb | if algorithm == "invert" a single value indicating the lower bound for the parameter values. if algorithm == "three.point", a vector of length 3 with position 1 as alpha lower bound, position 2 as sigma.sq's lower bound, position 3 as theta's lower bound. when set to NULL it will be a default value of 1e-9. Note that even if the model you're using doesn't include alpha (e.g. BM1), it must be included in this vector, but it will not be used to set any bounds. |
| ub | if algorithm == "invert" a single value indicating the upper bound for the parameter values. if algorithm == "three.point", a vector of length 3 with position 1 as alpha upper bound, position 2 as sigma.sq's upper bound, position 3 as theta's upper bound. when set to NULL it will be a default value of 100. Note that even if the model you're using doesn't include alpha (e.g. BM1), it must be included in this vector, but it will not be used to set any bounds. |
| opts | a list of options to pass to nloptr for the optimization: useful to adjust for faster, coarser searches |
| verbose | a logical indicating whether to print incremental steps |

## Details

This is an expanded version of the shift point model of Ho and Ane (2014). This is currently being tested, but as of now we strongly recommend using the mBIC criterion when searching for shifts.

## Value

`OUwie.dredge` returns an object of class `OUwie.dredge`. This is a list with elements:

| | |
|---|---|
| `$loglik` | the maximum log-likelihood. |
| `$criterion` | the information criterion to use for shift detection. |
| `$criterion.score` | the information criterion score used for shift detection. |
| `$shift.model` | The shift model estimated from the data. |
| `$solution` | a matrix containing the maximum likelihood estimates of $\alpha$ and $\sigma^2$. |
| `$tip.fog.est` | indicates value of the measurement error if it was estimated from the data. |
| `$theta` | a matrix containing the maximum likelihood estimates of $\theta$. |
| `$tot.states` | A vector of names for the different regimes |
| `$index.mat` | The indices of the parameters being estimated are returned. The numbers correspond to the row in the `eigvect` and can useful for identifying the parameters that are causing the objective function to be at a saddlepoint (see Details) |
| `$simmap.tree` | A logical indicating whether the input phylogeny is a SIMMAP formatted tree. |
| `$root.age` | The user-supplied age at the root of the tree. |
| `$scaleHeight` | Indicates whether the tree was constrained to a total height of 1. |
| `$shift.point` | The user-specified portion of the branch where a regime shift occurs. |
| `$opts` | Settings used for optimization routine. |
| `$data` | The shift model dataset, which includes regime painting for each tip. |
| `$phy` | The shift model painted phylogeny. |
| `$root.station` | A logical indicating whether the starting state, $\theta_0$, was estimated |
| `$starting.vals` | the starting values used for the parameter search. |
| `$regime.weights` | A table containing parameter estimates and the weights for time spent in each regime for each tip. |

## Author(s)

Jeremy M. Beaulieu

## References

Ho, L.S.T., and C. Ane. 2014. Intrinsic inference difficulties for trait evolution with Ornstein-Uhlenbeck models. Methods in Ecology and Evolution, 5: 1133-1146.

OUwie.fixed *Generalized Hansen model likelihood calculator*

## Description

Allows the user to calculate the likelihood given a specified set of parameter values

## Usage

```
OUwie.fixed(phy, data, model=c("BM1","BMS","OU1","OUM","OUMV","OUMA","OUMVA"),
simmap.tree=FALSE, root.age=NULL, scaleHeight=FALSE, root.station=FALSE,
get.root.theta=FALSE, shift.point=0.5, alpha=NULL, sigma.sq=NULL, theta=NULL,
clade=NULL, tip.fog="none", check.identify=TRUE, algorithm=c("invert", "three.point"),
tip.paths=NULL, quiet=FALSE)
```

## Arguments

| | |
|---|---|
| phy | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes |
| data | a dataframe containing species information (see Details) |
| model | models to fit to comparative data (see Details). |
| simmap.tree | a logical indicating whether the input tree is in SIMMAP format. The default is FALSE. |
| root.age | indicates the age of the tree. This is to be used in cases where the "tips" are not contemporary, such as in cases for fossil trees. Default is NULL meaning latest tip is modern day. |
| scaleHeight | a logical indicating whether the total tree height should be scaled to 1 (see Details). The default is FALSE. |
| root.station | a logical indicating whether the starting state, $\theta_0$, should be estimated. |
| get.root.theta | a logical indicating whether the starting state, $\theta_0$, should be estimated (see Details). |
| shift.point | the point along a branch where a regime change is assumed to have occurred (if SIMMAP=FALSE. The default is set to 0.5, or halfway along a branch. |
| alpha | a numeric vector giving the values of $\alpha$ for each selective regime |
| sigma.sq | a numeric vector giving the values of $\sigma^2$ for each selective regime |
| theta | a numeric vector giving the values of $\theta$ for each selective regime |
| clade | a list containing a pair of taxa whose MRCA is the clade of interest. |
| tip.fog | designates whether a fourth column in the data matrix contains tip fog for each species value ("known"). The tip fog is assumed to be the standard error of the species mean. If a single numeric value is supplied it will be applied to all species uniformly. The default is "none", meaning no tip fog is used. |
| check.identify | a logical indicating whether to check that the user-supplied regime paintings will produce identifiable theta estimates. The default is TRUE. |

| | |
|---|---|
| algorithm | designates whether the standard matrix inversion ('invert') or the faster 'three-point' algorithm of Ho and Ane (2013) should be used. |
| tip.paths | an optional list that can be provided by the user where each element from 1:nTips is the path from tip to root by labeled node number. The default is NULL and this option is used for internal speedups. |
| quiet | a logical indicating whether or not to print progress to the screen. The default is "FALSE". |

## Details

The input is a tree and a data file. The tree must be of class "phylo" and must contain the ancestral selective regimes as internal node labels. The data file is a data.frame that must have column entries in the following order: [,1] species names and [,2] their current selective regime. The user specifies the parameter values (i.e. $\alpha$, $\sigma^2$, and $\theta$).

## Value

OUwie.fixed returns an object of class OUwie.fixed. This is a list with elements:

| | |
|---|---|
| $loglik | the maximum log-likelihood. |
| $AIC | Akaike information criterion. |
| $AICc | Akaike information criterion corrected for sample-size. |
| $BIC | Schwartz information criterion. |
| $model | The model being fit |
| $param.count | The number of parameters counted in the model. |
| $solution | a matrix containing the maximum likelihood estimates of $\alpha$ and $\sigma^2$. |
| $theta | a matrix containing the maximum likelihood estimates of $\theta$. |
| $tot.state | A vector of names for the different regimes |
| $index.mat | The indices of the parameters being estimated are returned. The numbers correspond to the row in the eigvect and can useful for identifying the parameters that are causing the objective function to be at a saddlepoint (see Details) |
| $simmap.tree | A logical indicating whether the input phylogeny is a SIMMAP formatted tree. |
| $root.age | The user-supplied age at the root of the tree. |
| $shift.point | The user-specified portion of the branch where a regime shift occurs. |
| $data | User-supplied dataset |
| $phy | User-supplied tree |
| $root.station | A logical indicating whether the starting state, $\theta_0$, was estimated |
| $scaleHeight | Indicates whether the tree was constrained to a total height of 1. |
| $get.root.theta | Indicates whether the root.theta was included in the model. |
| $regime.weights | A table containing parameter estimates and the weights for time spent in each regime for each tip. |
| $algorithm | The algorithm used to estimate parameters. |

## Author(s)

Jeremy M. Beaulieu

## Examples

```
data(tworegime)

#Calculate the likelihood based on known values of
#alpha, sigma^2, and theta:
alpha=c(0.5632459,0.1726052)
sigma.sq=c(0.1064417,0.3461386)
theta=c(1.678196,0.4185894)

OUwie.fixed(tree,trait,model=c("OUMVA"), simmap.tree=FALSE, scaleHeight=FALSE,
clade=NULL, alpha=alpha,sigma.sq=sigma.sq,theta=theta, algorithm="three.point")
```

---

# OUwie.format *Format data and tree for OUwie*

---

## Description

Simplifies conversion of a tree and, optionally, data, for OUwie.

## Usage

```
OUwie.format(phy, tip.regimes=NULL, tip.data=NULL, tip.fog=NULL,
tip.fog.percentage=NULL, verbose=TRUE)
```

## Arguments

| | |
|---|---|
| phy | a phylogenetic tree, in ape "phylo" format and optionally with internal nodes labeled denoting the ancestral selective regimes |
| tip.regimes | a named vector or one-column data.frame with rownames with the regimes for each tip |
| tip.data | a named vector or one-column data.frame with rownames with the trait values for each tip |
| tip.fog | a named vector or one-column data.frame with rownames with the tip fog for each tip |
| tip.fog.percentage | a value for the percentage of the trait value to use as the tip fog |
| verbose | a logical indicating whether or not to print messages |

## Details

OUwie expects a tree with regimes mapped on nodes or a simmap tree. It also wants a data.frame with a column for taxon names, one for regime, one (JUST one) for trait value, and optionally one for tip fog. This function gets all of those formatted.

If you pass in no data, just the tree, the function will return a tree with regime 1 appearing everywhere, as well as a trait data.frame with regime 1 and trait value NA everywhere. It also includes zero tip fog for all taxa. This is very unrealistic, of course, and creates a bias in results (any variance in tips must be caused by the variance in the evolutionary process if tip fog is assumed to be zero).

For 'tip.regimes', 'tip.data', and 'tip.fog' the user can pass in a vector with the values and the names corresponding to the names of the tips on the tree or a single column data.frame where the rownames match the taxon names on the tree. The names of the input data and the names on the tree do not need to be in the same order, but they do need to match: "Homo sapiens" and "Homo_sapiens" do not match, for example.

Another way of handling tip dfog is assuming it's a fixed percentage of the tip value: "My uncertainty in squid arm lengths is 67 percent". To do this, pass in the percentage: 'tip.fog.percentage=67'. Note that it is per_cent: that is, for 67 percent put in 67, not 0.67. It will only work if there is already a measurement for the trait.

## Value

A list with the tree ('tree') and the data ('data')

## Author(s)

Brian C. O'Meara

---

`OUwie.sim` *Generalized Hansen model simulator*

---

## Description

Simulates the Ornstein-Uhlenbeck process of continuous characters evolving under discrete selective regimes.

## Usage

```
OUwie.sim(phy=NULL, data=NULL, simmap.tree=FALSE, root.age=NULL, scaleHeight=FALSE,
alpha=NULL, sigma.sq=NULL, theta0=NULL, theta=NULL, tip.fog="none", shift.point=0.5,
fitted.object=NULL, get.all=FALSE)
```

## Arguments

| | |
|---|---|
| `phy` | a phylogenetic tree, in ape "phylo" format and with internal nodes labeled denoting the ancestral selective regimes |
| `data` | a dataframe containing species information (see Details). Not necessary to include if `simmap=TRUE`. |

| | |
|---|---|
| simmap.tree | a logical indicating whether the input tree is in SIMMAP format. The default is FALSE. |
| root.age | indicates the age of the tree. This is to be used in cases where the "tips" are not contemporary, such as in cases for fossil trees. Default is NULL meaning latest tip is modern day. |
| scaleHeight | a logical indicating whether the total tree height should be scaled to 1 (see Details). The default is FALSE. |
| alpha | a numeric vector giving the values of $\alpha$ for each selective regime (see Details) |
| sigma.sq | a numeric vector giving the values of $\sigma^2$ for each selective regime (see Details) |
| theta0 | a numeric indicating the starting state, $\theta_0$ |
| theta | a numeric vector giving the values of $\theta$ for each selective regime (see Details) |
| tip.fog | designates whether a third column in the data matrix contains tip fog for each species value ("known"). The tip fog is assumed to be the standard error of the species mean. The default is "none". |
| shift.point | the point along a branch where a regime change is assumed to have occurred (if SIMMAP=FALSE. The default is set to 0.5, or halfway along a branch. |
| fitted.object | a model fit from OUwie to use for simulation. |
| get.all | a logical indicating whether or not the entire simulation history is to be returned. The default is FALSE meaning that only the tips are returned. |

## Details

The input is a tree and a data file OR a fitted OUwie object. The tree must be of class "phylo" and if simmap=FALSE must contain the ancestral selective regimes as internal node labels. The data file is a dataframe that must have column entries in the following order: [,1] species names and [,2] their current selective regime. If tip.fog="known" then a third column can be added which contains the tip fog for each species. Note that if simmap=TRUE no data file is needed. The user specifies the simulated parameter values (i.e. $\alpha$, $\sigma^2$, $\theta_0$, $\theta$). Assuming two selective regimes, possible models can be specified as follows (Note that this assumes a stationary distribution at the root):

a. Single rate Brownian motion (BM1): alpha=c(1e-10,1e-10); sigma.sq=c(0.45,0.45); theta0=1.0; theta=c(0,0).

b. Brownian motion with different rate parameters for each state on a tree (BMS): alpha=c(1e-10,1e-10) sigma.sq=c(0.45,0.90); theta0=1.0; theta=c(0,0).

c. Ornstein Uhlenbeck with a single optimum for all species (OU1): alpha=c(0.1,0.1); sigma.sq=c(0.9,0.9); theta0=1; theta=c(1.0,1.0).

d. Ornstein Uhlenbeck model that assumes different state means and a single $\alpha$ and $\sigma^2$ (OUM): alpha=c(1.0,1.0); sigma.sq=c(0.45,0.45); theta0=1.0; theta=c(1.0,2.0).

e. Ornstein Uhlenbeck model that assumes different state means and multiple $\sigma^2$ (OUMV): alpha=c(1.0,1.0); sigma.sq=c(0.45,0.90); theta0=1.0; theta=c(1.0,2.0).

f. Ornstein Uhlenbeck model that assumes different state means and multiple $\alpha$ (OUMA): alpha=c(1.0,0.5); sigma.sq=c(0.45,0.45); theta0=1.0; theta=c(1.0,2.0).

g. Ornstein Uhlenbeck model that assumes different state means and multiple $\sigma^2$ and $\alpha$ (OUMVA): alpha=c(1.0,0.5); sigma.sq=c(0.45,0.9); theta0=1.0; theta=c(1.0,2.0).

With a fitted OUwie model, it just uses the parameters from that, ignoring any alpha, theta, etc. set in the function.

## Value

A dataframe containing, as column entries, [,1] species names, [,2] current regime, [,3] simulated continuous trait, x.

## Author(s)

Jeremy M. Beaulieu and Brian C. O'Meara

## Examples

```
data(sim.ex)

#Simulate an Ornstein-Uhlenbeck model with different state means
#and a separate alpha and sigma^2 per selective regime
alpha=c(1.0,0.5)
sigma.sq=c(0.45,0.9)
theta0=1.0
theta=c(1.0,2.0)

sim.data<-OUwie.sim(tree,trait,simmap.tree=FALSE,scaleHeight=FALSE,
alpha=alpha,sigma.sq=sigma.sq,theta0=theta0,theta=theta)

#Now fit a model to this and simulate from the fitted results
result <- OUwie(tree, sim.data, model="OUMVA", simmap.tree=FALSE,scaleHeight=FALSE)
sim.data.2 <- OUwie.sim(fitted.object=result)
```

---

# plot.dentist *Plot the dented samples This will show the univariate plots of the parameter values versus the likelihood as well as bivariate plots of pairs of parameters to look for ridges.*

---

## Description

Plot the dented samples This will show the univariate plots of the parameter values versus the likelihood as well as bivariate plots of pairs of parameters to look for ridges.

## Usage

```
## S3 method for class 'dentist'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object of class dentist |
| ... | Other arguments to pass to plot |

`plot.OUwie.contour` *Contour plot*

## Description

A plotting function for visualizing likelihood surface for a pair of parameters using OUwie.contour data.

## Usage

```
## S3 method for class 'OUwie.contour'
plot(x, mle.point=NULL, levels=c(0:20*0.1), xlab=NULL,
ylab=NULL, xlim=NULL, ylim=NULL, col=grey.colors(21, start=0, end=1), ...)
```

## Arguments

| | |
|---|---|
| `x` | a `OUwie.contour` object. |
| `mle.point` | specifies the color for the maximum likelihood set of parameters. By default the point is not plotted (i.e., set to `NULL`. |
| `levels` | the levels at which to draw contour lines, measured as lnL units away from the best values. |
| `xlab` | allows users to specify the x-axis label. |
| `ylab` | allows users to specify the y-axis label. |
| `xlim` | allows users to specify the lower and upper limits of the x-axis. |
| `ylim` | allows users to specify the lower and upper limits of the y-axis. |
| `col` | indicates the color gradient. |
| `...` | additional parameters to control the plot. |

## Author(s)

Jeremy M. Beaulieu

## References

Beaulieu J.M., Jhwueng D.C., Boettiger C., and O'Meara B.C. 2012. Modeling stabilizing selection: Expanding the Ornstein-Uhlenbeck model of adaptive evolution. Evolution 66:2369-2383.

## print.dentist *Print dentist print summary of output from dent_walk*

### Description

Print dentist print summary of output from dent_walk

### Usage

```
## S3 method for class 'dentist'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class dentist |
| ... | Other arguments (not used) |

## summary.dentist *Summarize dentist Display summary of output from dent_walk*

### Description

Summarize dentist Display summary of output from dent_walk

### Usage

```
## S3 method for class 'dentist'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | An object of class dentist |
| ... | Other arguments (not used) |

# Index