Package 'EvoRAG'

February 19, 2015

Type Package

Title Evolutionary Rates Across Gradients

Version 2.0

Date 2014-07-14

Author Jason T. Weir

Maintainer Jason T. Weir <jason.weir@utoronto.ca>

Description

Uses maximum likelihood to estimate rates of trait evolution across environmental gradients.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-08 07:38:38

R topics documented:

EvoRAG-package 2
bootstrap.test 2
EvoRAG-data 5
EvoRAG-internal 6
expectation.gradient 6
expectation.time 8
model.test.sisters 10
MScorrection 14
parameter.reestimation 16
plotGradient.ci 17
power.test 18
Profile.like.CI 20
sim.sisters 22
sisterContinuous 24
starting.values 27
TypeI.error 28

Index **30**

EvoRAG-package *Evolutionary Rates Across Gradients*

Description

This packages uses maximum likelihood to estimate rates of trait evolution under several evolutionary models (Brownian Motion, Ornstein Ulhembeck) for datasets comprising many sister pairs (sister species or other sorts of sister pair data). Models in which a single evolutionary rate is applied to a dataset (null models) can be compared to models in which rates vary as a function of another continuous variable. The provided example tests to see if rates of vocal evolution in birds vary as a function of latitude. Functions are provided for simulating data under all implemented models and confidence intervals can be generated either from variances calculated via from bootstrap analysis or via profile likelihood.

Details

Package:	EvoRAG
Type:	Package
Version:	2.0
Date:	2014-01-07
License:	GPL version 2 or greater?

This package can be used to estimate rates of trait evolution across environmental or other sorts of gradients. The key function is model.test.sisters.

Author(s)

Jason T. Weir

Maintainer: Jason T. Weir <jason.weir@utoronto.ca>

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution. 66: 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B. 278:1713-1720.

bootstrap.test *Estimate confidence intervals using bootstrap*

Description

Applies bootstrap analyses to each of the packages models as an alternative way to estimate 95

Usage

```
bootstrap.test(DIST, TIME, GRAD, model, parameters, meserr1=0, meserr2=0,
   breakpoint = "NULL", N = c(1000), starting=NULL)
```

Arguments

DIST	vector of Euclidean distances for sister pair dataset
TIME	vector of evolutionary ages (i.e. node ages) for sister pair dataset
GRAD	vector of gradient values (i.e. any continuous variable) for sister pair dataset (see Details)
model	The name of the model to bootstrap.
parameters	a vector containing the maximum likelihood estimates of model parameters. These should be in the order indicated in sisterContinuous.
meserr1	a list of measurement errors that correspond to the first of each species in a sister pair. Order of sister pairs is the same as for DIST.
meserr2	a list of measurement errors that correspond to the second of each species in a sister pair. Order of sister pairs is the same as for DIST.
breakpoint	if using the models BM_2rate or OU_2rate, set this to the maximum likelihood estimate of the breakpoint.
N	The number of bootstrap replicates to perform.
starting	List of starting values. If starting=NULL, the built-in starting parameters are used.

Details

N bootstrap samples are generated, and are used to generate two estimates of the 95

Value

A matrix is returned listing the mean, median, variance, and 95

Author(s)

Jason T. Weir

References

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

See Also

sisterContinuous

Examples

```
## Not run:
###EXAMPLE 1

  ###simulate data
  set.seed(seed = 3)
  TIME = runif(n=100, min = 0, max = 10)
  GRAD = runif(n=100, min = 0, max = 60)
  DATA1 <- sim.sisters(TIME = TIME, GRAD=GRAD, parameters = c(2, -0.03),
     model=c("BM_linear"))

  ###Find the MLE of model parameters
  RESULT <- model.test.sisters(DIST=DATA1[,3], TIME=DATA1[,2],
     GRAD=DATA1[,1], models=c("BM_linear"))
  intercept <- as.numeric(RESULT[5,1])
  slope <- as.numeric(RESULT[6,1])
  model = c("BM_linear")
  parameters=c(intercept, slope)

  ###Run the bootstrap
  RR <- bootstrap.test(DIST=DATA1[,3], TIME=DATA1[,2],
     GRAD=DATA1[,1], model = "BM_linear", parameters, meserr1=0,
 meserr2=0, N = c(100))
  summary <- RR$summary #to show only the summary.
  bootstraps <- RR$bootstraps #to obtain the bootstraps


###EXAMPLE 2
  ###simulate data
  set.seed(seed = 3)
  TIME = runif(n=100, min = 0, max = 10)
  GRAD = runif(n=100, min = 0, max = 60)
  DATA1 <- sim.sisters(TIME = TIME, GRAD=GRAD, parameters = c(2, -0.03, 1,
      0.1), model=c("OU_linear"))
  ###Find the MLE of model parameters
  RESULT <- model.test.sisters(DIST=DATA1[,3], TIME=DATA1[,2],
     GRAD=DATA1[,1], models=c("OU_linear"))
  intercept_beta <- as.numeric(RESULT[5,1])
  slope_beta <- as.numeric(RESULT[7,1])
  intercept_alpha <- as.numeric(RESULT[11,1])
  slope_alpha <- as.numeric(RESULT[12,1])
  parameters=c(intercept_beta, slope_beta, intercept_alpha, slope_alpha)

  ###Run the bootstrap
  RR <- bootstrap.test(DIST=DATA1[,3], TIME=DATA1[,2],
     GRAD=DATA1[,1], model = "OU_linear", parameters, meserr1=0, meserr2=0,
     N = c(100))
  summary <- RR$summary #to show only the summary.
  bootstraps <- RR$bootstraps #to obtain the bootstraps

## End(Not run)#end dontrun
```

EvoRAG-data *example data sets of avian sister pairs with trait data and a gradient*

Description

This dataset is from Weir et al. 2012 (Evolution).

Usage

```
data(bird.pitch)
data(bird.syllables)
```

Details

bird.pitch: Euclidean distances are estimated from 6 measures of bird song pitch (using a Euclidean distance of PC1 and PC2). The gradient is midpoint latitude of the sister pair.

bird.syllables: Euclidean distances are from PC2 in a dataset which measured number of syllable types and temporal aspects of bird song. PC2 reflected the number of syllable types in a song. The gradient is midpoint latitude of the sister pair.

Value

returns the dataset in the form of a matrix

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66, 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B. 278:1713-1720.

Examples

```
data(bird.pitch)
data(bird.syllables)
```

EvoRAG-internal *internal EvoRAG functions*

Description

EvoRAG-internal functions are not typically called by the user

Details

These are internal **EvoRAG** functions, not intended to be called directly by the user. They include the following functions: sisterContinuous_logSpace, find.mle.sister, and simulation.analysis.

expectation.gradient *calculate the expected (i.e. mean) Euclidean distances for a gradient model*

Description

For models where evolutionary rate (Beta) or constraint (alpha) vary across a gradient, calculate the expected (i.e. mean) Euclidean distances at each point across the gradient after a given amount of time.

Usage

```
expectation.gradient(gradient.span = c(0, 10), model = c("BM_null",
   "BM_linear", "BM_2rate", "BM_linear_breakpoint", "BM_quadratic",
   "OU_null", "OU_linear_beta", "OU_linear", "OU_2rate",
   "OU_linear_breakpoint"), parameters, time=c(3), values=TRUE,
   plot=TRUE, quantile=FALSE)
```

Arguments

gradient.span	The gradient range over which to calculate the expectation
model	A vector listing the model name under which to simulate (e.g. model=c("OU_linear"). Any of the 10 models described in sisterContinuous may be used.
parameters	A vector listing the model parameters under which to simulate. Model parameters must be in the same order as described in sisterContinuous.
time	The time (since species slit from a common ancestor) at which to calculate the expectation.
values	TRUE (null) returns the values in matrix form.
plot	Plot the expected (solid line) Euclidean distance and optionally quantiles for a given Beta.
quantile	Calculate (and optionally plot) the expected quantiles (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99) for Euclidean distances under a given Beta.

Details

This function calculates the expectation (i.e. mean value under a half normal distribution) for Eculidean distance across a gradient where either evolutionary rate (Beta) and/or constraint (Alpha) vary as a function of the gradient. The user must specify the time at which the expectation will be calculated. The user can also have the quantiles (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99) calculated across the gradient.

Value

Returns a matrix with 1 columns corresponding to L (the gradient value) and the expectation, and an additional 11 columns with quantiles if qualtiles=TRUE. If plot=TRUE, the expectation (solid line) and optionally the quantiles (dashed lines) are plotted.

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66, 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B 278, 1713-1720.

See Also

expectation.time, sisterContinuous

Examples

```
## Not run:
##Example 1
  ###Expectation after 3 time units under BM_linear with Beta at 0 = 7, and
  ###a slope of Beta = -0.1.
  expectation.gradient(gradient.span = c(0, 60), model = c("BM_linear"),
     values = FALSE, parameters=c(7,-0.1), time=c(3),quantile=TRUE)


##Example 2
  ###Expectation after 3 time units under OU_linear with Beta constant
  ###across the gradient and alpha declining.
  expectation.gradient(gradient.span = c(0, 60), model = c("OU_linear"),
     values = FALSE, parameters=c(0.1, 0, 7, -0.1), time=c(3),quantile=TRUE)


##Example 3
  ###Expectation after 3 time units under OU_linear with Beta declining across
  ###the gradient and alpha remaining constant.
  expectation.gradient(gradient.span = c(0, 60), model = c("OU_linear"),
     values = FALSE, parameters=c(7, -0.1, 10, 0), time=c(3),quantile=TRUE)
```

```
##Example 4
  ###Expectation after 3 time units under BM_2rate with Beta 5 times higher
  ###after a breakpoint at L = 20.
  expectation.gradient(gradient.span = c(0, 60), model = c("BM_2rate"),
     values = FALSE, parameters=c(1, 20,5), time=c(3),quantile=FALSE)

##Example 5
  ###Expectation after 3 time units under BM_linear_breakpoint with the slope
  ###of Beta increasing 5 times higher after a breakpoint at L = 20.
  expectation.gradient(gradient.span = c(0, 60), model = c("BM_linear_breakpoint"),
     values = FALSE, parameters=c(0.1, 0.001, 20,0.1), time=c(3),quantile=TRUE)


##Example 6
  ###Expectation after 3 time units under BM_quadratic in which beta increases
  ###initially across the gradient and then declines. Under the quadratic,
  ###Beta_a (the third parameter) > 0 parabola curves upward, Beta_a < 0 downward.
  expectation.gradient(gradient.span = c(0, 60), model = c("BM_quadratic"),
     values = FALSE, parameters=c(10, 15, -0.2), time=c(3),quantile=TRUE)

## End(Not run)
```

`expectation.time` *calculate the expected (i.e. mean) Euclidean distances through time given a rate of evolution, Beta.*

Description

calculate the expected (i.e. mean) Euclidean distances through time given a rate of evolution, Beta.

Usage

```
expectation.time(Beta, Alpha="NULL", time.span=c(0, 10),
   values=TRUE, plot=TRUE, quantile=FALSE)
```

Arguments

`Beta`	Evolutionary rate parameter to plot
`Alpha`	Evolutionary constrain parameter tom plot (for OU model only). Leave as "NULL" to implement the BM model
`time.span`	A vector of length 1 if the expectation is calcuated for a single time; length 2 if to be calculated over a range from 0 to an upper value chosen by the user; or length > 0, where the user supplies 3 or more times over which to calculate the Expectation.
`values`	TRUE (null) returns the values in matrix form.

plot	Plot the expected (solid line) Euclidean distance and optionally quantiles for a given Beta.
quantile	Calculate (and optionally plot) the expected quantiles (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99) for Euclidean distances under a given Beta.

Details

This function calculates the expectation (i.e. mean value under a half normal distribution) for Eculidean distance across a time range and optionally the quantiles (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99).

Value

Returns a matrix with 3 columns corresponding to L, T and simulated E, and an additional 11 columns with quantiles if qualtiles=TRUE. If plot=TRUE, the expectation (solid line) and optionally the quantiles (dashed lines) are plotted.

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66, 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B 278, 1713-1720.

See Also

expectation.gradient, sisterContinuous, bootstrap.sister

Examples

```
##Example 1
  ###Compare data simulated under BM_null to the expectation and quantiles
  TIME = c(0:100) * 0.1
  GRAD = (0:100)*0 #BM_null does not require GRAD, thus simply make a dummy set of GRAD
  DATA1 <- sim.sisters(TIME=TIME, GRAD=GRAD, parameters = c(0.1),
     model=c("BM_null"), MULT=10)
  expectation.time(Beta=0.1, Alpha="NULL", time.span=c(0, 10), values=FALSE,
     plot=TRUE, quantile=TRUE)
  points(DATA1[,3] ~ DATA1[,2], col="black", cex=0.4)

##Example 2
  ###Compare data simulated under OU_null to the expectation and quantiles
  TIME = c(0:100) * 0.1
  GRAD = (0:100)*0 #GRAD is not required by these models, so a dummy set of GRAD are provided
  DATA1 <- sim.sisters(TIME=TIME, GRAD=GRAD, parameters = c(0.1, 1),
     model=c("OU_null"), MULT=10)
  expectation.time(Beta=0.1, Alpha=1, time.span=c(0, 10), values=FALSE,
```

```
    plot=TRUE, quantile=TRUE)
points(DATA1[,3] ~ DATA1[,2], col="black", cex=0.4)
```

`model.test.sisters` *Fit evolutionary models for continuous trait data*

Description

Takes a dataset of continuous trait values for a series of sister pairs (e.g. sister species) with known ages of divergence and finds the maximum likelihood fits under a series of evolutionary models.

Usage

```
model.test.sisters(DIST, TIME, GRAD, GRAD2 = "NULL", meserr1 = 0,
     meserr2 = 0, models = c("BM_null", "BM_linear", "BM_2rate",
 "BM_linear_breakpoint", "BM_quadratic", "OU_null", "OU_linear_beta",
 "OU_linear", "OU_2rate", "OU_linear_breakpoint"), starting=NULL,
 Beta_starting = NULL, Alpha_starting = NULL)
```

Arguments

`DIST`	vector of Euclidean distances for sister pair dataset
`TIME`	vector of evolutionary ages (i.e. node ages) for sister pair dataset
`GRAD`	vector of gradient values (i.e. any continuous variable) for sister pair dataset (see Details)
`GRAD2`	this is a vector of gradient values for a second continuous variable to be used for models that test for the effect of two gradients on rates of evolution. Not currently implemented.
`meserr1`	a list of measurement errors that correspond to the first of each species in a sister pair. Order of sister pairs is the same as for DIST.
`meserr2`	a list of measurement errors that correspond to the second of each species in a sister pair. Order of sister pairs is the same as for DIST.
`models`	A vector listing which models to test. By default all models are tested. Models with more than 4 parameters should be used with caution, especially with small datasets (i.e. less than 100 sister pairs) as the data may provide insufficient power to reject simpler models in favour of complex ones.
`starting`	List of starting values for each model. If starting=NULL, the built-in starting parameters are used. The method can be sensitive to starting parameters. For each model, the null method tests a large number of starting parameters, and chooses the set of starting parameters maximized the likelihood. However, the null starting parameters may not be optimized for all datasets. The user can customize starting parameters. Each element of the list that is not "NULL" is a matrix, with the number of columns equal to the number of model parameters and each row containing a different set of starting parameters. See Example 3 below.

`Beta_starting` vector of Beta starting values to test for BM_2rate and OU_2rate models. NULL uses built-in starting parameters are used. The null values are c(0.001, 0.01, 0.1, 1, 10, 100, 1000) for BM_2rate and c(0.01, 0.1, 1, 10, 100) for OU_2rate

`Alpha_starting` vector of Alpha starting values to test for OU_2rate model. NULL uses built-in starting parameters are used. The null values are c(0.01, 0.1, 1, 10)

Details

Evolutionary models include null models whereby a single set of model parameters are fit to all sister pairs and models whereby parameters are allowed to vary as a function of another continuous variable. The second continuous variable could be elevation, latitude, body mass or any other continuous variable of interest, over which rates of trait evolution might vary. This function uses the nlm optimizer to search for the maximum likelihood estimates under 10 evolutionary models. For details of the evolutionary models implemented see sisterContinuous. Running all models on a dataset of about 100 sisters should take less than 5 minutes. Excluding those models with more than 4 parameters will speed up the search considerably. For BM_2rate and OU_2rate models, Beta and alpha values before and after the breakpoint are each set to the values in Beta_starting and Alpha_starting, and all possible pairwise combinations of these are tested together with a variety of breakpoints. Thus keep the length of these vectors of starting values under 5, or expect to wait a long time for the function to execute.

Value

Returns a table with log-likelihoods, Akaike information criterion, and parameter estimates. The final column returns the exit status of the nlm function (and results should not be trusted if this value is 3 or higher; see nlm documentation). parameters are output in the same order as specified in sisterContinuous.

- **BM_null:** Beta = b1
- **OU_null:** Beta = b1, Alpha = a1
- **BM_linear:** Beta_C = b1; Beta_slope = b1_slope
- **OU_linear:** Beta_C = b1, Beta_slope = b1_slope, Alpha_C = a1, Alpha_slope = a1_slope
- **BM_linear_beta:** Beta_C = b1, Beta_slope = b1_slope
- **OU_linear_beta:** Beta_C = b1, Beta_slope = b1_slope, Alpha = a1
- **BM_2rate parameters:** Beta1 = b1, breakpoint = breakpoint, Beta2 = b2
- **OU_2rate parameters:** Beta1 = b1, breakpoint = breakpoint, Beta2 = b2, Alpha1 = a1, Alpha2 = a2
- **BM_linear_breakpoint:** Beta_C1 = b1, Beta_slope1=b1_slope, breakpoint=breakpoint, Beta_Slope2=b2_slope
- **OU_linear_breakpoint:** Beta_C1 = b1, Beta_slope1=b1_slope, breakpoint=breakpoint, Beta_Slope2=b2_slope, Alpha_C1 = a1, Alpha_slope1=a1_slope, and Alpha_slope2 = a2_slope
- **BM_quadratic:** Beta_c = Quadratic_c, Beta_b = Quadratic_b, Beta_a = Quadratic_a

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66,2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B 278,1713-1720.

See Also

sisterContinuous

Examples

```
## Not run:
##Example 1
  ###This example uses the four models used in Weir et al. 2012 to test for
  ###a latitudinal effect on Euclidean distances for bird song pitch on 87
  ###forest sister pairs.
     data(bird.pitch)
     attach(bird.pitch)

  #STEP 1 Correct Euclidean distances for sampling and measurement bias
     DIST_cor <- MScorrection(nA=bird.pitch$number_individuals_Species1,
        nB=bird.pitch$number_individuals_Species2,
        VarA=bird.pitch$Variance_PC1and2_Species1,
        VarB=bird.pitch$Variance_PC1and2_Species2,
        DIST_actual=bird.pitch$Uncorrected_Euclidean_Distance)

  #STEP 2  Extract and test only forest species
     DIST <- subset(DIST_cor, subset = (bird.pitch$Habitat == "forest"))
     TIME <- subset(bird.pitch$TIME,subset = (bird.pitch$Habitat == "forest"))
     GRAD <- subset(bird.pitch$GRAD,
        subset = (bird.pitch$Habitat == "forest"))
     models = c("BM_null", "BM_linear", "OU_null", "OU_linear")
     #The following generally takes 1 to 2 minutes to run
     FIT1 <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models)

  ###The best fit model for forest species is the OU_linear model in which
  ###rates of evolution increase with latitude (b1_slope is positive) and
  ###evolutionary constraint declines with increasing latitude (a1_slope is
  ###negative).High latitude species are evolving faster and in a less
  ###constrained fashion.

##Example 2
  ###This example tests to see if allopatric and sympatric species pairs
  ###have significantly different rates under the BM_null model

  #STEP 1  Correct Euclidean distances for sampling and measurement bias
     DIST_cor <- MScorrection(nA=bird.pitch$number_individuals_Species1,
        nB=bird.pitch$number_individuals_Species2,
        VarA=bird.pitch$Variance_PC1and2_Species1,
        VarB=bird.pitch$Variance_PC1and2_Species2,
```

```
            DIST_actual=bird.pitch$Uncorrected_Euclidean_Distance)

  #STEP 2  First, fit BM_linear to the entire dataset
     DIST <- DIST_cor
     TIME <- bird.pitch$TIME
     GRAD <- bird.pitch$GRAD
     models = c("BM_null")
     FIT2a <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models)

  #STEP 3  Next, fit BM_linear to the allopatric subset
     DIST <- subset(DIST_cor, subset = (bird.pitch$Patry == "allopatric"))
     TIME <- subset(bird.pitch$TIME,
        subset = (bird.pitch$Patry == "allopatric"))
     GRAD <- subset(bird.pitch$GRAD,
        subset = (bird.pitch$Patry == "allopatric"))
     models = c("BM_null")
     FIT2b <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models)

  #STEP 4  Finally, fit BM_linear to the sympatric subset
     DIST <- subset(DIST_cor, subset = (bird.pitch$Patry == "sympatric"))
     TIME <- subset(bird.pitch$TIME,
        subset = (bird.pitch$Patry == "sympatric"))
     GRAD <- subset(bird.pitch$GRAD,
        subset = (bird.pitch$Patry == "sympatric"))
     models = c("BM_null")
     FIT2c <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models)

  #STEP 5  Compare the AIC of the model fit to the entire dataset to the model
  #with separate rates for allopatric and sympatric subsets.
     ###To calculate AIC for the allopatric and sympatric model
     ###the loglikelihoods for the subsets are summed
     logLikelihood <- as.numeric(FIT2b[1,]) + as.numeric(FIT2c[1,])
     ###The subsets model has 2 parameters (1 for each subset)
     ###thus AIC = 2*2 - 2*logLike
     AIC_forest_nonforest <-  2*2 - 2*logLikelihood

  ###The AIC for the entire dataset is 319.86 and for the model with separate rates
  ###for allopatric and sympatric AIC is 320.13. The best fit model is the full dataset
  ###model without separate rates for different subsets, indicating a failure to reject
  ###the null hypothesis in favour of separate rates for allopatric and sympatric
  ###species pairs.


##Example 3
  ###using the same data as Example 1, this example demonstrates user control of
  ###starting parameters
  #STEP 1  generate matrices of starting values for those models which the user
  #wishes to use customized starting values
     p_matrix <- c(0.0001, 0.001, 0.01, 0.1, 1,2,3,4,5,10,100,1000)
     BM_null_starting <- matrix(p_matrix, length(p_matrix), 1, byrow=TRUE)

     p_matrix <- c(10, -1, 10, 1, 0, -0.1, 0, 0.1)
     BM_linear_starting <- matrix(p_matrix, length(p_matrix)/2, 2, byrow=TRUE)
```

```
  #first, use only 2 models, each with customize starting parameters
     models <- c("BM_null", "BM_linear")
     FIT3a <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models,
         starting = list(BM_null_starting, BM_linear_starting) )

  #next use 4 models, but customize starting parameters for only the first two
     models <- c("BM_null", "BM_linear", "OU_null", "OU_linear")
     FIT3b <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models,
         starting = list(BM_null_starting, BM_linear_starting, "NULL", "NULL") )


##EXAMPLE 4
  ###This example uses the syllable dataset for oscine songbirds Weir & Wheatcroft 2011
  data(bird.syllables)
  attach(bird.syllables)

  #STEP 1 Correct Euclidean distances for sampling and measurement bias
     DIST_cor <- MScorrection(nA=bird.syllables$number_individuals_Species1,
        nB=bird.syllables$number_individuals_Species2,
        VarA=bird.syllables$Species1_PC2_var,
        VarB=bird.syllables$Species2_PC2_var,
        DIST_actual=abs(bird.syllables$Species1_PC2_mean -
        bird.syllables$Species2_PC2_mean))

  #STEP 2  Test all models on oscines only (in which song has a strong
  #culturally transmitted component)
     DIST <- subset(DIST_cor, subset = (bird.syllables$Suboscine == "oscine"))
     TIME <- subset(bird.syllables$TIME,subset = (bird.syllables$Suboscine == "oscine"))
     GRAD <- subset(bird.syllables$GRAD,
        subset = (bird.syllables$Suboscine == "oscine"))
     FIT5 <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models)
     #The best fit model in FIT5 is BM_linear in which tropical species have a
     #much slower rate than temperate species.

## End(Not run)#end dontrun
```

`MScorrection` *Correct for finite sample size in Euclidean distances given known variances in each sample*

Description

Correct for finite sample size in Euclidean distances given known variances in each sample

Usage

```
MScorrection(nA, nB, VarA, VarB, MSwithin = NA, DIST_actual)
```

Arguments

nA	The number of individuals sampled in species A
nB	The number of individuals sampled in species B
VarA	Sample variance for species A
VarB	Sample variance for species B
MSwithin	Alternatively, if MSwithin (e.g. the error mean squared; see Sokal & Rohlf 1995 pg 214) is available, this can be used instead of VarA and VarB
DIST_actual	The uncorrected Euclidean distance between species A and B

Details

Euclidean distances are generally biased upwards by sampling and measurement error within species. This bias is typically large when few individuals are measured and the true Euclidean distance between species is small. Here I use a correction based on the ANOVA (Weir & Wheatcroft 2011) that corrects the expected bias in Euclidean distances (for full details see Weir & Whatcroft 2011, Weir et al. 2012). Corrected Euclidean distances can be used with other functions in this package. Alternatively, measurement error can be included directly in likelihood functions in model.test.sister.

Value

returns the negative log-Likelihood

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66, 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B 278, 1713-1720.

Sokal, R. R. & Rohlf, F. J. 1995 Biometry: the principles and practice of statistics in biological research, 3rd edn. New York, NY: W. H. Freeman & Co page 214.

See Also

MScorrection_MSwithin

Examples

```
data(bird.pitch)
attach(bird.pitch)
DIST_cor <- MScorrection(nA=bird.pitch$number_individuals_Species1,
   nB=bird.pitch$number_individuals_Species2,
   VarA=bird.pitch$Variance_PC1and2_Species1,
   VarB=bird.pitch$Variance_PC1and2_Species2, MSwithin = NA,
```

```
DIST_actual=bird.pitch$Uncorrected_Euclidean_Distance)
```

`parameter.reestimation` *uses simulation to determine robustness of parameter estimates under a model*

Description

uses simulation to determine robustness of parameter estimates under a model

Usage

```
parameter.reestimation(GRAD, TIME, model, PARAMETERS, N, REP = 1)
```

Arguments

GRAD	vector of gradient values (i.e. any continuous variable) for sister pair dataset
TIME	vector of evolutionary ages (i.e. node ages) for sister pair dataset
model	any model implemented in EvoRAG
PARAMETERS	A vector listing the model parameters under which to simulate. Model parameters must be in the same order as described in sisterContinuous.
REP	How many replicated datasets of TIME and GRAD to use. Default = 1. Example: REP=3 generates a dataset with each element in TIME and GRAD repeated 3 times. This option will be used primarily for calculating statistical power as a function of increasing number of sister pairs
N	The number of simulations to perform

Details

Simulates data under a model, and re-estimates model parameters using the same model. A model performs well if the parameters it is simulated under are similar to those it re-estimates.

Value

Returns a matrix showing the mean, median, range, several percentiles and the standard error for each model parameter.

Author(s)

Jason T. Weir

Examples

```
## Not run:
  ###simulate data
  set.seed(seed = 3)
  TIME = runif(n=300, min = 0, max = 10)
  GRAD = runif(n=300, min = 0, max = 60)
  DATA1 <- sim.sisters(TIME = TIME, GRAD=GRAD, parameters = c(2, -0.03),
     model=c("BM_linear"))

  ###run parameter.reestimation
  model = c("BM_linear")
  parameter.reestimation(GRAD, TIME, model=model, PARAMETERS=c(2, -0.03),
     N=100, REP = 1)

## End(Not run)#end dontrun
```

---

plotGradient.ci *Plot evolutionary rates and their confidence regions across a gradient*

Description

Takes the output from bootstrap.test for the BM_linear model and plots how evolutionary rates and their confidence regions change across the gradient.

Usage

```
plotGradient.ci(bootstraps1, bootstraps2=c("FALSE"), Lmin, Lmax, ylim,
   MLE = FALSE, MLE1, MLE2, xlab="Gradient")
```

Arguments

bootstraps1	the raw bootstraps output by bootstrap.test
bootstraps2	same as bootstraps1, but for an optional second dataset.
Lmin	minimum gradient value for graph
Lmax	maximum gradient value for graph
ylim	maximum y-axis value
MLE	Logical. If MLE=TRUE, then the maximum likelihood values are plotted. If MLE=FALSE, then the mean bootstrap values are plotted.
MLE1	A list of the maximum likelihood parameter values for dataset 1
MLE2	A list of the maximum likelihood parameter values for dataset 2, if a second dataset provided
xlab	A title for the x-axis.

Details

Currently, only works for the BM_linear model.

Value

A plot of the bootstrap 95

Author(s)

Jason T. Weir

See Also

bootstrap.test

Examples

```
## Not run:
  ###simulate data
  set.seed(seed = 3)
  TIME = runif(n=200, min = 0, max = 10)
  GRAD = runif(n=200, min = 0, max = 60)
  DATA1 <- sim.sisters(GRAD, TIME, parameters = c(0.1, 0.065), model=c("BM_linear"))

  ###Find the MLE of model parameters
  RESULT <- model.test.sisters(DIST=DATA1[,3], TIME=DATA1[,2],
     GRAD=DATA1[,1], models=c("BM_linear"))
  intercept <- as.numeric(RESULT[5,1])
  slope <- as.numeric(RESULT[6,1])
  model = c("BM_linear")
  parameters=c(intercept, slope)

  ###Run the bootstrap
  RR <- bootstrap.test(DIST=DATA1[,3], TIME=DATA1[,2],
     GRAD=DATA1[,1], model = "BM_linear", parameters, meserr1=0,
 meserr2=0, N = c(100))
  summary <- RR$summary #to show only the summary.
  bootstraps <- RR$bootstraps #to obtain the bootstraps

  ###Plot data
  plotGradient.ci(bootstraps1=bootstraps,
     bootstraps2= c("FALSE"), Lmin=0, Lmax=60, ylim=c(0,10),
     MLE=TRUE, MLE1=c(0.1, 0.065), MLE2=c(0,0), xlab="Latitude")

## End(Not run)#end dontrun
```

---

power.test *performs a simulation based analysis of statistical power*

Description

performs a simulation based analysis of statistical power

Usage

```
power.test(TIME, GRAD, parameters, test.model, threshold_deltaAICc,
   REP=1, N, write = "FALSE", wd = "")
```

Arguments

TIME	vector of evolutionary ages (i.e. node ages) for sister pair dataset
GRAD	vector of gradient values (i.e. any continuous variable) for sister pair dataset
parameters	A vector listing the model parameters under which to simulate. Model parameters must be in the same order as described in sisterContinuous.
test.model	Any one of the following models are currently supported ("BM_linear", "OU_linear_beta", "OU_linear")
threshold_deltaAICc	A single threshold deltaAICc or a list of such values
REP	How many replicated datasets of TIME and GRAD to use. Default = 1. Example: REP=3 generates a dataset with each element in TIME and GRAD repeated 3 times. This option will be used primarily for calculating statistical power as a function of increasing number of sister pairs
N	The number of simulations to perform
write	If true, writes output to several files
wd	directory to write files to if other than the current working directory. (Windows example, "D:/SIMS/"

Details

Performs an analysis of statistical power (e.g. the probability of supporting a true alternative hypothesis) for a given dataset under a given model and set of model parameters. The threshold_deltaAICc should be set at a level that will maintain a type I error (probability of rejecting a true null model) of 0.05. Appropriate threshold_deltaAICc values can be determined using the function TypeI.error. The null hypothesis here tested is that rates of evolution do not vary as a function of gradient (e.g. "BM_null", and "OU_null"). The alternative, is rates do vary as a linear function of a gradient (e.g. "BM_linear", "OU_linear_beta", "OU_linear"). Several hundred or more replicates should be performed. Currently, only "BM_linear", "OU_linear_beta", "OU_linear" are included in the candidate set of gradient models.

Value

Returns a list with the following elements: test.model The model for which power was calculated parameters The parameters under which power was calculated N_sisters The number of sister pairs in the dataset N_sims The number of simulations performed power_test_hypothesis Statistical power calculated for the alternative hypothesis that rates of evolution vary as a linear function of a gradient. Power is returned for each threshold value in threshold_deltaAIC. Where appropriate, power to reject BM_null and OU_null is returned for three comparisons: 1) BMlinear_and_OUlinear_beta_vs_2null: power when simulating data either under BM_linear or OU_linear_beta, but when the OU_linear model is not included in the analysis; 2) BMlinear_and_OUlinear_vs_2null: power when OU_linear_beta is not included; 3) 3gradient_vs_2null: power when all three gradient models are included.

power_test_hypothesis The probability of the test model correctly rejecting each of the other null and gradient models on an individual basis.

Author(s)

Jason T. Weir

See Also

TypeI.error

Examples

```
## Not run:

 ###simulate data
  set.seed(seed = 3)
  TIME = runif(n=300, min = 0, max = 10)
  GRAD = runif(n=300, min = 0, max = 60)
  DATA1 <- sim.sisters(TIME = TIME, GRAD=GRAD, parameters = c(2, -0.03),
     model=c("BM_linear"))

  ###run power.test
  model = c("BM_linear")
  power.test(TIME=TIME, GRAD=GRAD, parameters = c(2, -0.03), test.model="BM_linear",
     threshold_deltaAICc = c((1:20)*0.5), REP=1, N=2, write = "FALSE", wd = "")

## End(Not run)#end dontrun
```

Profile.like.CI *Estimate confidence intervals using profile likelihood*

Description

profile likelihood is used to estimate 95

Usage

```
Profile.like.CI(DIST, TIME, GRAD, meserr1 = 0, meserr2 = 0, like, par,
   MODEL, test.values.par1, test.values.par2, p_starting="NULL")
```

Arguments

DIST	vector of Euclidean distances for sister pair dataset
TIME	vector of evolutionary ages (i.e. node ages) for sister pair dataset
GRAD	vector of gradient values (i.e. any continuous variable) for sister pair dataset (see Details)

meserr1	a list of measurement errors that correspond to the first of each species in a sister pair. Order of sister pairs is the same as for DIST.
meserr2	a list of measurement errors that correspond to the second of each species in a sister pair. Order of sister pairs is the same as for DIST.
like	loglike at the MLE as returned by model.test.sisters
par	a vector containing the maximum likelihood estimates of model parameters. These should be in the order indicated in sisterContinuous.
MODEL	The name of the gradient model to perform profile likelihood on. Currently implemented models are "BM_linear", "OU_linear_beta", and "OU_linear"
test.values.par1	a vector of values to calculate likelihoods for parameter 1
test.values.par2	a vector of values to calculate likelihoods for parameter 2
p_starting	List of starting values for the model. If starting=list("NULL"), the built-in starting parameters are used, and is generally recommended.

Details

This function uses profile likelihood to estimate confidence for select parameters. Likelihood surfaces often have ridges (e.g. the "OU_linear" model), and the resulting confidence intervals are not always symmetric around the MLE. Profile likelihood generates confidence intervals appropriate in such cases. However, the code is computationally demanding. Currently, profile likelihood is only implemented for the two parameters of BM_linear and for the slope parameters of OU_linear_beta (1 parameter) and OU_linear (2 parameters).

Value

Returns a list with the following elements: profile.likelihoods_par1, 2, 3 etc For each parameter, a matrix showing the range of values tested (test.value), the log likelihoods of each value in the range (logLike), the difference in likelihood from the MLE and each value (logLikeDifference). The final column (CI_range) gives a 1 if the value was less than 1.92 loglikelihood units below the MLE, and thus outside the 95 model The model used MLE_par1 The MLE for parameter 1 CI_par1 The lower and upper 95 warnings_par1 A warning is returned only if the lower or upper limit of the CI has not been reached by the range of tested values. Otherwise, returns NA

Author(s)

Jason T. Weir

See Also

bootstrap.test

Examples

```
## Not run:

  ###This example uses the syllable dataset for oscine songbirds Weir & Wheatcroft 2011
```

```
data(bird.syllables)
attach(bird.syllables)

#STEP 1 Correct Euclidean distances for sampling and measurement bias
   DIST_cor <- MScorrection(nA=bird.syllables$number_individuals_Species1,
      nB=bird.syllables$number_individuals_Species2,
      VarA=bird.syllables$Species1_PC2_var,
      VarB=bird.syllables$Species2_PC2_var,
      DIST_actual=abs(bird.syllables$Species1_PC2_mean -
      bird.syllables$Species2_PC2_mean))

#STEP 2  Test all models on oscines only (in which song has a strong
#culturally transmitted component)
   DIST <- subset(DIST_cor, subset = (bird.syllables$Suboscine == "oscine"))
   TIME <- subset(bird.syllables$TIME,subset = (bird.syllables$Suboscine == "oscine"))
   GRAD <- subset(bird.syllables$GRAD,
      subset = (bird.syllables$Suboscine == "oscine"))
   FIT5 <- model.test.sisters(DIST=DIST, TIME=TIME, GRAD=GRAD, models=models)
   #The best fit model in FIT5 is BM_linear in which tropical species have a
   #much slower rate than temperate species.

#STEP 3 run the profile likelihood
Profile.like.CI(DIST=DIST, TIME=TIME, GRAD=GRAD, meserr1 = 0, meserr2 = 0,
   like=FIT5[1,2], par=c(FIT5[5,2], FIT5[6,2]), MODEL="BM_linear", MULT=1,
test.values.par1 = c((0:100)*0.001), test.values.par2 = c((33:100)*0.0001),
p_starting="NULL")


## End(Not run)#end dontrun
```

`sim.sisters` *simulate Euclidean distances for sister pair data under 10 evolutionary models*

Description

simulate Euclidean distances for sister pair data under 10 evolutionary models

Usage

```
sim.sisters(TIME, GRAD, GRAD2 = "NULL", parameters, model, MULT=1)
```

Arguments

TIME	vector of evolutionary ages (i.e. node ages) for sister pair dataset
GRAD	vector of gradient values (i.e. any continuous variable) for sister pair dataset
GRAD2	this is a vector of gradient values for a second continuous variable to be used for models that test for the effect of two gradients on rates of evolution.

parameters	A vector listing the model parameters under which to simulate. Model parameters must be in the same order as described in sisterContinuous.
model	A vector listing the model name under which to simulate (e.g. model=c("OU_linear"). Any of the 10 models described in sisterContinuous may be used.
MULT	How many replicated simulations per set of GRAD and TIME. Default = 1

Details

This function is called by bootstrap.sister, but can also be used for customized routines to explore model power and to visualize what data is expected to look like under different evolutionary rates.

Value

Returns a matrix with 3 columns corresponding to GRAD, TIME and simulated DIST.

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66, 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B 278, 1713-1720.

See Also

sisterContinuous, bootstrap.sister

Examples

```
##Example 1
  ###This example graphically compares the distributions of simulated Euclidean
  ###distances under BM_null when Beta (evolutionary rate) is 0.1 and 0.2
  TIME = c(0:100) * 0.1
  GRAD = (0:100)*0 #BM_null does not require GRAD, thus simply make a dummy set of GRAD
  DATA1 <- sim.sisters(TIME=TIME, GRAD=GRAD, parameters = c(0.2),
     model=c("BM_null"), MULT=10)
  DATA2 <- sim.sisters(TIME=TIME, GRAD=GRAD, parameters = c(0.1),
     model=c("BM_null"), MULT=10)
  plot(DATA1[,3] ~ DATA1[,2], xlab="Genetic distance of sister pair",
     ylab = "Euclidean distance", cex=0.5)
  expectation1 <- expectation.time(Beta = 0.2, Alpha="NULL", time.span=c(0, 10),
     values="TRUE", plot=FALSE, quantile=FALSE)
  lines(expectation1[,2] ~ expectation1[,1], lwd=2)
  points(DATA2[,3] ~ DATA2[,2], col="red",  cex=0.5)
  expectation2 <- expectation.time(Beta = 0.1, Alpha="NULL", time.span=c(0, 10),
      values="TRUE", plot=FALSE, quantile=FALSE)
  lines(expectation2[,2] ~ expectation2[,1],col="red", lwd=2)
```

```
  ###Notice that doubling Beta still results in largely overlapping distributions
  ###of DIST at any given TIME, and the expectation (shown by lines) is not doubled.

##Example 2
  ###graphically compare data simulated with the same evolutionary rate (Beta)
  ###under BM_null versus OU_null to see the effect of constraint (Alpha)
  TIME = c(0:100) * 0.1
  GRAD = (0:100)*0 #GRAD is not required by these models, so a dummy set of GRAD are provided
  DATA1 <- sim.sisters(TIME=TIME, GRAD=GRAD, parameters = c(0.2),
     model=c("BM_null"), MULT=10)
  DATA2 <- sim.sisters(TIME=TIME, GRAD=GRAD, parameters = c(0.2, 1),
     model=c("OU_null"), MULT=10)
  plot(DATA1[,3] ~ DATA1[,2], xlab="Genetic distance of sister pair",
     ylab = "Euclidean distance", cex=0.5)
  expectation1 <- expectation.time(Beta = 0.2, Alpha="NULL", time.span=c(0, 10),
     values="TRUE", plot=FALSE, quantile=FALSE)
  lines(expectation1[,2] ~ expectation1[,1], lwd=2)
  points(DATA2[,3] ~ DATA2[,2], col="red", cex=0.5)
  expectation2 <- expectation.time(Beta = 0.2, Alpha=1, time.span=c(0, 10),
     values="TRUE", plot=FALSE, quantile=FALSE)
  lines(expectation2[,2] ~ expectation2[,1],col="red", lwd=2)
  ###Notice that DIST increases in a similar fashion under BM and OU until about
  ###TIME = 0.5 after which point the strong constraint in OU becomesevident.
```

`sisterContinuous` *likelihood functions for continuous trait evolutionary models*

Description

Returns the negative log-likelihood of the data under an evolutionary model. Evolutionary models include

Usage

```
sisterContinuous(parameters, meserr1 = 0, meserr2 = 0, model = c("BM_null",
     "BM_2rate", "BM_linear", "BM_linear_breakpoint", "BM_quadratic",
 "OU_null", "OU_2rate", "OU_linear", "OU_linear_beta",
 "OU_linear_breakpoint"), breakpoint = "NULL", DIST, TIME,
 GRAD, GRAD2="NULL")
```

Arguments

`parameters`	a vector of parameter values to be tested.
`meserr1`	a list of measurement errors that correspond to the first of each species in a sister pair. Order of sister pairs is the same as for DIST.
`meserr2`	a list of measurement errors that correspond to the second of each species in a sister pair. Order of sister pairs is the same as for DIST.
`model`	evolutionary model to calculate log-likelihood (see Details).

breakpoint	breakpoint (along GRAD) to use for the BL_2rate and OU_2rate models.
DIST	vector of Euclidean distances for sister pair dataset.
TIME	vector of evolutionary ages (i.e. node ages) for sister pair dataset
GRAD	vector of gradient values (i.e. any continuous variable) for sister pair dataset (see Details).
GRAD2	this is a vector of gradient values for a second continuous variable to be used for models that test for the effect of two gradients on rates of evolution. Not currently implemented

Details

This function calculates the negative log-likelihood for continuous trait data for a series of sister pairs (e.g. sister species) under a variety of evolutionary models that allow rates of evolution (Beta) or evolutionary constraint (Alpha) to either remain constant or to vary as a function of another continuous variable. The second continuous variable could be elevation, latitude, body mass or any other continuous variable of interest, over which rates of trait evolution might vary. This function can be used in combination with an optimizer such as optim or nlm to find the maximum likelihood values for model parameters. These optimizers often perform poorly on more complex models, and instead we suggest that model.test.sisters be used.

Evolutionary models implemented are as follows.

- **BM_null and OU_null** Applies a simple Brownian motion (BM; 1 parameter) and Ornstein Uhlenbeck (OU; 2 parameters) model in which model parameters do not vary as a function of GRAD. Model parameters: for BM_null a single parameter describing the evolutionary rate, parameters = c(Beta); for OU_null an additional parameter describing the evolutionary constraint, parameters = c(Beta, Alpha)
- **BM_2rate and OU_2rate** Allows model parameters for BM (3 parameters) and OU (5 parameters) to differ before and after a breakpoint along the gradient GRAD. Model parameters: for BM_2rate parameters = c(Beta1, Beta2) where Beta1 and Beta2 are the rates before and after the breakpoint and breakpoint is a third parameter set using the breakpoint argument; for OU_2rate parameters = c(Beta1, breakpoint, Beta2, Alpha1, Alpha2) where Alpha1 and Alpha2 are the constraints before and after the breakpoint.
- **BM_linear and OU_linear** Allows model parameters for BM (2 parameters) and OU (4 parameters) to vary as a linear function of GRAD. Model parameters: for BM_linear parameters = c(Beta_C, Beta_slope) which describe the intercept and slope of Beta; for OU_linear parameters = c(Beta_C, Beta_slope, Alpha_C, Alpha_slope) which describe the intercept and slope of Beta and Alpha
- **OU_linear_beta** The same as OU_linear but only Beta varies linearly with GRAD, while Alpha remains constant across GRAD. Model parameters = c(Beta_C, Beta_slope, Alpha)
- **BM_linear_breakpoint and OU_linear_breakpoint** A breakpoint model whereby model parameters before and after a breakpoint vary by different linear functions of GRAD, with both linear functions intersecting at the breakpoint. Model parameters: for BM_linear_breakpoint parameters = c(Beta_C1, Beta_slope1, breakpoint, Beta_Slope2) which describe the intercept and slope of Beta prior to the breakpoint, and the slope of Beta following the breakpoint; for OU_linear_breakpoint parameters = c(Beta_C1, Beta_slope1, breakpoint, Beta_Slope2, Alpha_C1, Alpha_slope1, and Alpha_slope2) which describe the intercept and slope of Alpha prior to the breakpoint, and the slope of Alpha following the breakpoint.

- **BM_quadratic** Model parameters for BM (3 parameters) change as a quadratic function of GRAD. Model parameters = c(Beta_c, Beta_b, Beta_a) where Beta = Beta_c + Beta_b * GRAD + Beta_a * GRAD^2.

Value

returns the negative log-Likelihood

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66, 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B 278, 1713-1720.

See Also

model.test.sisters

Examples

```
data(bird.pitch)
attach(bird.pitch)

###The following example uses optim to find the maximum likelihood estimate
###on data from Weir et al 2012.

#STEP 1: Correct Euclidean distances for sampling and measurement bias
DIST_cor <- MScorrection(nA=bird.pitch$number_individuals_Species1,
   nB=bird.pitch$number_individuals_Species2,
   VarA=bird.pitch$Variance_PC1and2_Species1,
   VarB=bird.pitch$Variance_PC1and2_Species2,
   DIST_actual=bird.pitch$Uncorrected_Euclidean_Distance)

#STEP 2: Extract and test only forest species
DIST <- subset(DIST_cor, subset = (bird.pitch$Habitat == "forest"))
TIME <- subset(bird.pitch$TIME, subset = (bird.pitch$Habitat == "forest"))
GRAD <- subset(bird.pitch$GRAD,
   subset = (bird.pitch$Habitat == "forest"))

#STEP 4: fit the model using optim
res <- optim(par = c(0.1,0.001), fn=sisterContinuous, model = c("BM_linear"),
     DIST=DIST, TIME=TIME, GRAD=GRAD, method="L-BFGS-B",lower=c(0,-5),upper=c(Inf,5))
```

`starting.values` *returns the null starting values used in model.test.sisters*

Description

returns the null starting values used in model.test.sisters

Usage

```
starting.values(MODEL)
```

Arguments

`MODEL`	any one of the evolutionary models implemented by model.test.sisters except BM_2rate and OU_2rate.

Details

because nlm (and other optimization functions in R) often gets stuck on local likelihood optima, model.test.sisters uses a variety of starting parameters in combination with nlm optimization and reports the results for the best set of starting parameters as the maximum likelihood estimate. The null starting parameters have been optimized for rates of Beta and Alpha that are typically less than 1. The user can also provide their own matrix of starting parameters. This function is provided here so the user can determine if the starting parameters are suited to their particular dataset. Alternatively, values of Euclidean distances and of L can also be transformed (i.e. by dividing large values by a constant) so they lie within a range acceptable for the starting parameters.

Value

returns a matrix with the starting values. Each column is a different parameter, and the last column is NA

Author(s)

Jason T. Weir

References

Weir JT, D Wheatcroft, & T Price. 2012. The role of ecological constraint in driving the evolution of avian song frequency across a latitudinal gradient. Evolution 66, 2773-2783.

Weir JT, & D Wheatcroft. 2011. A latitudinal gradient in rates of evolution of avian syllable diversity and song length. Proceedings of the Royal Society of London, B 278, 1713-1720.

See Also

model.test.sisters

Examples

```
starting.values(MODEL = "OU_linear")
```

`TypeI.error` *performs a simulation based analysis of type I error*

Description

performs a simulation based analysis of type I error

Usage

```
TypeI.error(TIME, GRAD, beta, alpha=0, null.model, REP=1, N,
   write.file = "FALSE", wd = "")
```

Arguments

`TIME`	vector of evolutionary ages (i.e. node ages) for sister pair dataset
`GRAD`	vector of gradient values (i.e. any continuous variable) for sister pair dataset
`beta`	Evolutionary rate, beta, to simulate under
`alpha`	value of evolutionary constraint, alpha, when null.model = "OU_null". Should be set to 0 when using "BM_null"
`null.model`	Either "BM_null" or "OU_null"
`REP`	How many replicated datasets of TIME and GRAD to use. Default = 1. Example: REP=3 generates a dataset with each element in TIME and GRAD repeated 3 times. This option will be used primarily for calculating statistical power as a function of increasing number of sister pairs
`N`	The number of simulations to perform
`write.file`	If true, writes output to several files
`wd`	directory to write files to if other than the current working directory. (Windows example, "D:/SIMS/"

Details

Performs an analysis of type I error (e.g. the probability of rejecting a true null hypothesis) when the the model with the lowest AICc is chosen as the best fit. The null hypothesis here tested is that rates of evolution do not vary as a function of gradient (e.g. "BM_null", and "OU_null"). The alternative, is rates do vary as a linear function of a gradient (e.g. "BM_linear", "OU_linear_beta", "OU_linear"). Currently, only "BM_linear", "OU_linear_beta", "OU_linear" are included in the candidate set of gradient models.

Value

Returns a list with the following elements: simulation_parameters The parameters and model under which simulation occurred TypeI_errors Returns the Type I error and the appropriate threshold delta AICc value necessary to reject the null hypothesis while maintaining a type I error of 0.05. model_parameters Also returns the median, 0 percentile and 95 percentile of the distribution of parameter values estimated across the simulations for each model. These can be used to check for bias in the null models.

Author(s)

Jason T. Weir

See Also

power.test

Examples

```
## Not run:

 ###simulate data
  set.seed(seed = 3)
  TIME = runif(n=300, min = 0, max = 10)
  GRAD = runif(n=300, min = 0, max = 60)
  DATA1 <- sim.sisters(TIME = TIME, GRAD=GRAD, parameters = c(2), model=c("BM_null"))

  ###run typeI error test. This should be run for a minimum of N=1000 simulations
  TypeI.error(TIME, GRAD, beta=2, null.model="BM_null", REP=1, N=10,
     write.file = "FALSE", wd = "")

## End(Not run)#end dontrun
```

Index

∗Topic **Brownian Motion**
model.test.sisters, 10
sim.sisters, 22
sisterContinuous, 24
starting.values, 27
∗Topic **Expectation**
expectation.gradient, 6
expectation.time, 8
∗Topic **Ornstein Uhlenbeck**
sim.sisters, 22
sisterContinuous, 24
starting.values, 27
∗Topic **Ornstein Ulhembeck**
model.test.sisters, 10
∗Topic **Simulation**
parameter.reestimation, 16
power.test, 18
sim.sisters, 22
TypeI.error, 28
∗Topic **Statistical power**
power.test, 18
∗Topic **Type I error**
TypeI.error, 28
∗Topic **Type II error**
power.test, 18
∗Topic **bootstrap**
bootstrap.test, 2
∗Topic **confidence interval, plot**
plotGradient.ci, 17
∗Topic **confidence interval**
Profile.like.CI, 20
∗Topic **delta AICc**
TypeI.error, 28
∗Topic **parameter re-estimation**
parameter.reestimation, 16

bird.pitch *(*EvoRAG-data*)*, 5
bird.syllables *(*EvoRAG-data*)*, 5
bootstrap.test, 2

EvoRAG *(*EvoRAG-package*)*, 2
EvoRAG-data, 5
EvoRAG-internal, 6
EvoRAG-package, 2
expectation.gradient, 6
expectation.time, 8

find.mle.sister *(*EvoRAG-internal*)*, 6

model.test.sisters, 10
MScorrection, 14
MScorrection_MSwithin *(*EvoRAG-internal*)*, 6

parameter.reestimation, 16
plotGradient.ci, 17
power.test, 18
Profile.like.CI, 20

sim.sisters, 22
simulation.analysis *(*EvoRAG-internal*)*, 6
sisterContinuous, 24
sisterContinuous_logSpace *(*EvoRAG-internal*)*, 6
sisterContinuous_logSpace_profile_CI *(*EvoRAG-internal*)*, 6
starting.values, 27

TypeI.error, 28