

# Package ‘evolvability’

December 9, 2021

**Type** Package

**Title** Calculation of Evolvability Parameters

**Version** 2.0.0

## Description

Provides tools for calculating evolvability parameters from estimated G-matrices as defined in Hansen and Houle (2008) <[doi:10.1111/j.1420-9101.2008.01573.x](https://doi.org/10.1111/j.1420-9101.2008.01573.x)> and fits phylogenetic comparative models that link the rate of evolution of a trait to the state of another evolving trait (see Hansen et al. 2021 Systematic Biology <[doi:10.1093/sysbio/syab079](https://doi.org/10.1093/sysbio/syab079)>). The package was released with Bolstad et al. (2014) <[doi:10.1098/rstb.2013.0255](https://doi.org/10.1098/rstb.2013.0255)>, which contains some examples of use.

**License** GPL (>= 2)

**Imports** coda, Matrix, ape, lme4, stats

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Geir H. Bolstad [aut, cre]

**Maintainer** Geir H. Bolstad <[geir.bolstad@nina.no](mailto:geir.bolstad@nina.no)>

**Repository** CRAN

**Date/Publication** 2021-12-08 23:00:12 UTC

## R topics documented:

evolvability-package . . . . .	2
Almer . . . . .	3
Almer_boot . . . . .	4
Almer_SE . . . . .	5
Almer_sim . . . . .	6
conditionalG . . . . .	6

evolvabilityBeta . . . . .	8
evolvabilityBetaMCMC . . . . .	9
evolvabilityBetaMCMC2 . . . . .	10
evolvabilityMeans . . . . .	12
evolvabilityMeansMCMC . . . . .	13
GLS . . . . .	14
macro_pred . . . . .	15
meanStdG . . . . .	16
meanStdGMCMC . . . . .	17
phylH . . . . .	18
plot.rate_gls . . . . .	19
plot.simulate_rate . . . . .	20
randomBeta . . . . .	21
rate_gls . . . . .	22
rate_gls_boot . . . . .	25
rate_gls_sim . . . . .	26
round_and_format . . . . .	27
simulate_rate . . . . .	28
summary.evolvabilityBeta . . . . .	30
summary.evolvabilityBetaMCMC . . . . .	31

<b>Index</b>	<b>32</b>
--------------	-----------

---

evolvability-package    *Calculation of Evolvability Parameters*

---

## Description

This package calculates evolvability parameters from estimated G-matrices as defined in Hansen and Houle (2008) <doi:10.1111/j.1420-9101.2008.01573.x>. It can use both point estimates and posterior/bootstrap distributions of the G-matrices. This package was released with Bolstad et al. (2014) <doi:10.1098/rstb.2013.0255>, which contains some examples of use.

## Details

Package:    evolvability  
Type:        Package  
Version:    1.1.0  
Date:        2015-04-13  
License:    GPL (>= 2)

## Author(s)

Geir H. Bolstad <geir.bolstad@nina.no>

## References

- Bolstad G. H., Hansen T. F., Pelabon C. Falahati-Anabaran M., Perez-Barrales R. & Armbruster W. S. 2014. Genetic constraints predict evolutionary divergence in Dalechampia blossoms. *Phil. Trans. R. Soc. B.* 369:20130255. doi:10.1098/rstb.2013.0255
- Hansen T. F. & Houle D. (2008) Measuring and comparing evolvability and constraint in multivariate characters. *J. Evol. Biol.* 21:1201-1219. doi:10.1111/j.1420-9101.2008.01573.x

---

 Almer

---

*Linear mixed model with correlated random effects structure*


---

## Description

Almer fits a univariate linear mixed model incorporating a correlated random effects structure. Can be used to fit phylogenetic mixed models and animal models. The function is based on the [lme4](#) package and is very similar to [lmer](#), apart from the A argument.

## Usage

```
Almer(
  formula,
  data = NULL,
  A = list(),
  REML = TRUE,
  control = lme4::lmerControl(check.nobs.vs.nlev = "ignore", check.nobs.vs.rankZ =
    "ignore", check.nobs.vs.nRE = "ignore"),
  start = NULL,
  verbose = 0L,
  weights = NULL,
  na.action = "na.omit",
  offset = NULL,
  contrasts = NULL,
  devFunOnly = FALSE,
  ...
)
```

## Arguments

formula	as in <a href="#">lmer</a> .
data	as in <a href="#">lmer</a> .
A	an optional named list of sparse matrices. The names must correspond to the names of the random effects in the formula argument. All levels of the random effect should appear as row and column names for the matrices.
REML	as in <a href="#">lmer</a> .
control	as in <a href="#">lmer</a> .
start	as in <a href="#">lmer</a> .

verbose	as in <a href="#">lmer</a> .
weights	as in <a href="#">lmer</a> .
na.action	as in <a href="#">lmer</a> .
offset	as in <a href="#">lmer</a> .
contrasts	as in <a href="#">lmer</a> .
devFunOnly	as in <a href="#">lmer</a> .
...	as in <a href="#">lmer</a> .

**Value**

Almer an object of class [merMod](#).

**Author(s)**

Geir H. Bolstad

**Examples**

```
# See the vignette 'Phylogenetic mixed model'.
```

---

Almer\_boot

*Parametric bootstrap on [Almer](#) model fit*

---

**Description**

Almer\_boot performs a parametric bootstrap from an [Almer](#) model fit

**Usage**

```
Almer_boot(mod, nsim = 1000)
```

**Arguments**

mod	A fitted object from <a href="#">Almer</a>
nsim	The number of simulations.

**Value**

Almer\_boot a list with entries `fixef`, `vcov`, `fixef_distribution` and `vcov_distribution`, where the two first entries includes the means, standard deviations, and quantiles of the fixed effects means and (co)variances, respectively, and the two latter includes the complete bootstrap distribution.

**Author(s)**

Geir H. Bolstad

**Examples**

```
# See the vignette 'Phylogenetic mixed model'.
```

---

`Almer_SE`*Linear mixed model for response variables with uncertainty*

---

## Description

`Almer_SE` Linear mixed model for response variables with uncertainty

## Usage

```
Almer_SE(  
  formula,  
  SE = NULL,  
  maxiter = 100,  
  control = lme4::lmerControl(check.nobs.vs.nlev = "ignore", check.nobs.vs.rankZ =  
    "ignore", check.nobs.vs.nRE = "ignore"),  
  ...  
)
```

## Arguments

<code>formula</code>	as in <a href="#">lmer</a> .
<code>SE</code>	A vector of standard errors associated with the response variable. NB! Must have column name "SE" in the data.
<code>maxiter</code>	The maximum number of iterations.
<code>control</code>	as in <a href="#">lmer</a> .
<code>...</code>	Further optional arguments, see <a href="#">Almer</a> .

## Value

`Almer_SE` returns an object of class [merMod](#).

## Author(s)

Geir H. Bolstad

## Examples

```
# See the vignette 'Phylogenetic mixed model'.
```

Almer\_sim *Simulate responses from Almer fit*

---

**Description**

Almer\_sim Simulate responses from an [Almer](#) model fit.

**Usage**

```
Almer_sim(mod, nsim = 1000)
```

**Arguments**

mod	A fitted object from <a href="#">Almer</a>
nsim	The number of simulations.

**Details**

This function is only included as the [simulate.merMod](#) function did not seem to work properly when the number of random effect levels equal the number of observations.

**Value**

Almer\_sim a matrix of simulated responses, columns correspond to each simulations.

**Author(s)**

Geir H. Bolstad

**Examples**

```
# See the vignette 'Phylogenetic mixed model'.
```

---

conditionalG *Computing a conditional sub-matrix of G*

---

**Description**

conditinoalG calculates a conditional variance matrix.

**Usage**

```
conditionalG(G, condition_on = NULL)
```

**Arguments**

G	A variance matrix (must be symmetric and positive definite).
condition_on	Either an integer with the column number indicating which trait to condition on or a vector with several column numbers (integers).

**Details**

The function calculates a sub-matrix of G conditional on the traits defined by the the condition\_on vector. The function is based on equation 3 in Hansen et al. (2003).

**Value**

A matrix that is a sub-matrix of the input matrix conditional on the non-included traits.

**Author(s)**

Geir H. Bolstad

**References**

Hansen TF, Armbruster WS, Carlsson ML & Pélabon C. 2003. Evolvability and genetic constraint in Dalechampia blossoms: genetic correlations and conditional evolvability. *J. Exp. Zool.* 296B:23-39.

**Examples**

```
# Constructing a G-matrix:
G <- matrix(c(
  1, 1, 0, 1,
  1, 2, 1, 1,
  0, 1, 2, 1,
  1, 1, 1, 3
), ncol = 4)

# Computing a conditional 2x2 sub-matrix by conditioning on
# trait 3 and 4:
G_sub_conditional <- conditionalG(G, condition_on = c(3, 4))
G_sub_conditional

# The average evolvabilities of this matrix can then be
# compared can than be compared to the average evolvabilities
# of the corresponding unconditional sub-matrix of G:
evolvabilityMeans(G_sub_conditional)
evolvabilityMeans(G[-c(3, 4), -c(3, 4)])
```

---

evolvabilityBeta      *Calculate evolvability parameters along a set of selection gradients*

---

### Description

G needs to be symmetric and positive definite.

### Usage

```
evolvabilityBeta(G, Beta, means = 1)
```

### Arguments

G	A variance matrix.
Beta	Either a vector or a matrix of unit length selection gradients stacked column wise.
means	An optional vector of trait means (for internal mean standardization).

### Details

evolvabilityBeta calculates (unconditional) evolvability (e), respondability (r), conditional evolvability (c), autonomy (a) and integration (i) along selection gradients given an additive-genetic variance matrix as described in Hansen and Houle (2008).

### Value

An object of class 'evolvabilityBeta', which is a list with the following components:

Beta	The matrix of selection gradients.
e	The evolvability of each selection gradient.
r	The respondability of each selection gradient.
c	The conditional evolvability of each selection gradient.
a	The autonomy of each selection gradient.
i	The integration of each selection gradient.

### Author(s)

Geir H. Bolstad

### References

Hansen, T. F. & Houle, D. (2008) Measuring and comparing evolvability and constraint in multivariate characters. *J. Evol. Biol.* 21:1201-1219.

### Examples

```
G <- matrix(c(1, 1, 0, 1, 2, 2, 0, 2, 3), ncol = 3) / 10
```



```
Beta <- randomBeta(5, 3)
X <- evolvabilityBeta(G, Beta)
summary(X)
```

---

evolvabilityBetaMCMC *Calculate posterior distribution of evolvability parameters from a set of selection gradients*

---

### Description

evolvabilityBetaMCMC calculates (unconditional) evolvability (e), responsibility (r), conditional evolvability (c), autonomy (a) and integration (i) from selection gradients given the posterior distribution of an additive-genetic variance matrix. These measures and their meanings are described in Hansen and Houle (2008).

### Usage

```
evolvabilityBetaMCMC(G_mcmc, Beta, post.dist = FALSE)
```

### Arguments

G_mcmc	posterior distribution of a variance matrix in the form of a table. Each row in the table must be one iteration of the posterior distribution (or bootstrap distribution). Each iteration of the matrix must be on the form as given by $c(x)$ , where $x$ is a matrix. A posterior distribution of a matrix in the slot VCV of a object of class MCMCg1mm is by default on this form.
Beta	either a vector or a matrix of unit length selection gradients stacked column wise.
post.dist	logical: should the posterior distribution of the evolvability parameters be saved.

### Value

An object of class 'evolvabilityBetaMCMC', which is a list with the following components:

eB	The posterior median and highest posterior density interval of evolvability for each selection gradient.
rB	The posterior median and highest posterior density interval of responsibility for each selection gradient.
cB	The posterior median and highest posterior density interval of conditional evolvability for each selection gradient.
aB	The posterior median and highest posterior density interval of autonomy for each selection gradient.
iB	The posterior median and highest posterior density interval of integration for each selection gradient.
Beta	The matrix of selection gradients.
summary	The means of evolvability parameters across all selection gradients.
post.dist	The full posterior distribution.

### Author(s)

Geir H. Bolstad

## References

Hansen, T. F. & Houle, D. (2008) Measuring and comparing evolvability and constraint in multivariate characters. *J. Evol. Biol.* 21:1201-1219.

## Examples

```
# Simulating a posterior distribution
# (or bootstrap distribution) of a G-matrix:
G <- matrix(c(1, 1, 0, 1, 4, 1, 0, 1, 2), ncol = 3)
G_mcmc <- sapply(c(G), function(x) rnorm(10, x, 0.01))
G_mcmc <- t(apply(G_mcmc, 1, function(x) {
  G <- matrix(x, ncol = sqrt(length(x)))
  G[lower.tri(G)] <- t(G)[lower.tri(G)]
  c(G)
}))

# Simulating a posterior distribution
# (or bootstrap distribution) of trait means:
means <- c(1, 1.4, 2.1)
means_mcmc <- sapply(means, function(x) rnorm(10, x, 0.01))

# Mean standardizing the G-matrix:
G_mcmc <- meanStdGMCMC(G_mcmc, means_mcmc)

# Generating selection gradients in five random directions:
Beta <- randomBeta(5, 3)

# Calculating evolvability parameters:
x <- evolvabilityBetaMCMC2(G_mcmc, Beta, post.dist = TRUE)
summary(x)
```

---

evolvabilityBetaMCMC2 *Calculate posterior distribution of evolvability parameters from a selection gradient estimated with uncertainty*

---

## Description

evolvabilityBetaMCMC2 calculates (unconditional) evolvability (e), responsibility (r), conditional evolvability (c), autonomy (a) and integration (i) along a selection gradient estimate with uncertainty.

## Usage

```
evolvabilityBetaMCMC2(G_mcmc, Beta_mcmc, post.dist = FALSE)
```

**Arguments**

G_mcmc	A posterior distribution of a variance matrix in the form of a table. Each row in the table must be one iteration of the posterior distribution (or bootstrap distribution). Each iteration of the matrix must be on the form as given by <code>c(x)</code> , where <code>x</code> is a matrix. A posterior distribution of a matrix in the slot VCV of a object of class <code>MCMCglmm</code> is by default on this form.
Beta_mcmc	A posterior distribution of a unit length selection gradient where iterations are given row wise.
post.dist	logical: should the posterior distribution of the evolvability parameters be saved.
Beta.median summary	posterior median and highest posterior density interval of the selection gradient. The posterior median and highest posterior density interval of evolvability parameters.
post.dist	The full posterior distributions of the evolvability parameters.

**Author(s)**

Geir H. Bolstad

**References**

Hansen, T. F. & Houle, D. (2008) Measuring and comparing evolvability and constraint in multivariate characters. *J. Evol. Biol.* 21:1201-1219.

**Examples**

```
{
# Simulating a posterior distribution
# (or bootstrap distribution) of a G-matrix:
G <- matrix(c(1, 1, 0, 1, 4, 1, 0, 1, 2), ncol = 3)
G_mcmc <- sapply(c(G), function(x) rnorm(10, x, 0.01))
G_mcmc <- t(apply(G_mcmc, 1, function(x) {
  G <- matrix(x, ncol = sqrt(length(x)))
  G[lower.tri(G)] <- t(G)[lower.tri(G)]
  c(G)
}))

# Simulating a posterior distribution
# (or bootstrap distribution) of trait means:
means <- c(1, 1.4, 2.1)
means_mcmc <- sapply(means, function(x) rnorm(10, x, 0.01))

# Mean standardizing the G-matrix:
G_mcmc <- meanStdGMCMC(G_mcmc, means_mcmc)

# Simulating a posterior distribution (or bootstrap distribution)
# of a unit length selection gradient:
Beta <- randomBeta(1, 3)
Beta.mcmc <- sapply(c(Beta), function(x) rnorm(10, x, 0.01))
Beta.mcmc <- t(apply(Beta.mcmc, 1, function(x) x / sqrt(sum(x^2))))
```

```
# Running the model:
  evolvabilityBetaMCMC2(G_mcmc, Beta_mcmc = Beta.mcmc, post.dist = TRUE)
}
```

---

evolvabilityMeans      *Calculate average evolvability parameters of a G-matrix*

---

### Description

evolvabilityMeans calculates the average (unconditional) evolvability (e), responsibility (r), conditional evolvability (c), autonomy (a) and integration (i) of an additive-genetic variance matrix using the approximation formulas described in Hansen and Houle (2008, 2009).

### Usage

```
evolvabilityMeans(G, means = 1)
```

### Arguments

G	A variance matrix (must be symmetric and positive definite).
means	An optional vector of trait means, for mean standardization.

### Details

The equations for calculating the evolvability parameters are approximations, except for the minimum, maximum and unconditional evolvability which are exact. The bias of the approximations depends on the dimensionality of the G-matrix, with higher bias for few dimensions (see Hansen and Houle 2008). For low dimensional G-matrices, we recommend estimating the averages of the evolvability parameters using `evolvabilityBetaMCMC` over many random selection gradients (`randomBeta`). The maximum and minimum evolvability, which are also the maximum and minimum responsibility and conditional evolvability, equals the largest and smallest eigenvalue of the G-matrix, respectively.

### Value

A vector with the following components:

e_mean	The average (unconditional) evolvability.
e_min	The minimum evolvability.
e_max	The maximum evolvability.
r_mean	The average responsibility.
c_mean	The average conditional evolvability.
a_mean	The average autonomy.
i_mean	The average integration.

### Author(s)

Geir H. Bolstad

## References

- Hansen, T. F. & Houle, D. (2008) Measuring and comparing evolvability and constraint in multivariate characters. *J. Evol. Biol.* 21:1201-1219.  
 Hansen, T. F. & Houle, D. (2009) Corrigendum. *J. Evol. Biol.* 22:913-915.

## Examples

```
G <- matrix(c(1, 1, 0, 1, 2, 1, 0, 1, 2), ncol = 3)
evolvabilityMeans(G)
```

---

evolvabilityMeansMCMC *Calculate posterior distribution of average evolvability parameters of a G-matrix*

---

## Description

evolvabilityMeans calculates the average (unconditional) evolvability (e), responsibility (r), conditional evolvability (c), autonomy (a) and integration (i) given the posterior distribution of an additive-genetic variance matrix using the approximation formulas described in Hansen and Houle (2008, 2009).

## Usage

```
evolvabilityMeansMCMC(G_mcmc)
```

## Arguments

**G\_mcmc** the posterior distribution of a variance matrix in the form of a table. Each row in the table must be one iteration of the posterior distribution (or bootstrap distribution). Each iteration of the matrix must be on the form as given by `c(x)`, where `x` is a matrix. A posterior distribution of a matrix in the slot `VCV` of an object of class `MCMCglmm` is by default on this form.

## Details

The equations for calculating the evolvability parameters are approximations, except for the minimum, maximum and unconditional evolvability which are exact. The bias of the approximations depends on the dimensionality of the G-matrix, with higher bias for few dimensions (see Hansen and Houle 2008). For low dimensional G-matrices, we recommend estimating the averages of the evolvability parameters using `evolvabilityBetaMCMC` over many random selection gradients (`randomBeta`). The maximum and minimum evolvability, which are also the maximum and minimum responsibility and conditional evolvability, equals the largest and smallest eigenvalue of the G-matrix, respectively.

## Value

An object of class `'evolvabilityMeansMCMC'`, which is a list with the following components:

post.dist                   The posterior distribution of the average evolvability parameters.  
 post.medians                The posterior medians and HPD interval of the average evolvability parameters.

**Author(s)**

Geir H. Bolstad

**References**

Hansen, T. F. & Houle, D. (2008) Measuring and comparing evolvability and constraint in multi-variate characters. *J. Evol. Biol.* 21:1201-1219.  
 Hansen, T. F. & Houle, D. (2009) Corrigendum. *J. Evol. Biol.* 22:913-915.

**Examples**

```
# Simulating a posterior distribution
# (or bootstrap distribution) of a G-matrix:
G <- matrix(c(1, 1, 0, 1, 4, 1, 0, 1, 2), ncol = 3)
G_mcmc <- sapply(c(G), function(x) rnorm(10, x, 0.01))
G_mcmc <- t(apply(G_mcmc, 1, function(x) {
  G <- matrix(x, ncol = sqrt(length(x)))
  G[lower.tri(G)] <- t(G)[lower.tri(G)]
  c(G)
}))

# Simulating a posterior distribution
# (or bootstrap distribution) of trait means:
means <- c(1, 1.4, 2.1)
means_mcmc <- sapply(means, function(x) rnorm(10, x, 0.01))

# Mean standardizing the G-matrix:
G_mcmc <- meanStdGMCMC(G_mcmc, means_mcmc)

# Estimating average evolvability parameters:
evolvabilityMeansMCMC(G_mcmc)
```

---

 GLS

*Generalized least square*


---

**Description**

GLS utilizes `lm.fit` and Cholesky decomposition to fit a generalized least squares regression

**Usage**

```
GLS(y, X, R = NULL, L = NULL, coef_only = FALSE)
```

**Arguments**

y	response variable
X	design matrix
R	residual covariance or correlation matrix (can be sparse), ignored if L is provided.
L	lower triangular matrix of the Cholesky decomposition of R (optional).
coef_only	reduces the output of the model to the estimated coefficients (and the generalized residual sums of squares) only.

**Details**

Note that the size of R does not matter (i.e. if R is multiplied by a scalar, the results don't change). Note also that the R-squared is estimated as  $1 - \text{GSSE}/\text{GSST}$ , where GSSE is the generalized residual sum of squares (i.e. the objective function score of the model) and GSST is the generalized total sum of squares (i.e. the objective function score of the model when only the intercept is included in the model)

**Value**

GLS a [list](#) of

- coef: a table of estimates and standard errors
- R2: the R-squared of the model fit
- sigma2: the residual variance
- GSSE: the generalized residual sum of squares (objective function score)
- coef\_vcov: the error variance matrix of the estimates

**Author(s)**

Geir H. Bolstad

---

macro\_pred

*Macroevolutionary predictions*

---

**Description**

macro\_pred Macroevolutionary predictions

**Usage**

macro\_pred(y, V, useLFO = TRUE)

**Arguments**

y	vector of species means
V	phylogenetic variance matrix, must have same order as y
useLFO	excludes the focal species when calculating the corresponding species' mean. The correct way is to use TRUE, but in practice it has little effect and FALSE will speed up the model fit.

**Details**

macro\_pred Gives a vector of macroevolutionary predictions for each species based on the other species given the phylogeny and a phylogenetic variance matrix.

**Value**

macro\_pred returns a of macroevolutionary predictions at the tips.

**Author(s)**

Geir H. Bolstad

**Examples**

```
# Trait values
y <- rnorm(3)

# A variance matrix (the diagonal must be the same order as y).
V <- matrix(c(1.0, 0.5, 0.2, 0.5, 1.0, 0.4, 0.2, 0.4, 1.0), ncol = 3)

# Macroevolutionary predictions (output in the same order as y).
macro_pred(y, V)
```

---

meanStdG

*Mean standardize a variance matrix*

---

**Description**

meanStdG mean standardizes a variance matrix (e.g. a G-matrix).

**Usage**

```
meanStdG(G, means)
```

**Arguments**

G	A variance matrix.
means	A vector of trait means.



**Value**

A mean standardized variance matrix.

**Author(s)**

Geir H. Bolstad

**Examples**

```
G <- matrix(c(1, 1, 0, 1, 4, 1, 0, 1, 2), ncol = 3)
means <- c(1, 1.4, 2.1)
meanStdG(G, means)
```

---

meanStdGMCMC

*Mean standardize the posterior distribution of a G-matrix*

---

**Description**

meanStdGMCMC mean standardizes the posterior distribution of a variance matrix (e.g. a G-matrix)

**Usage**

```
meanStdGMCMC(G_mcmc, means_mcmc)
```

**Arguments**

G_mcmc	A posterior distribution of a variance matrix in the form of a table. Each row in the table must be one iteration of the posterior distribution (or bootstrap distribution). Each iteration of the matrix must be on the form as given by <code>c(x)</code> , where <code>x</code> is a matrix. A posterior distribution of a matrix in the slot VCV of a object of class <code>MCMCglmm</code> is by default on this form.
means_mcmc	A posterior distribution of a vector of means in the form of a table. Each row in the table must be one iteration of the posterior distribution (or bootstrap distribution). A posterior distribution of a mean vector in the slot <code>So1</code> of an object of class <code>MCMCglmm</code> is by default on this form.

**Value**

The posterior distribution of a mean standardized variance matrix.

**Author(s)**

Geir H. Bolstad

**Examples**

```

# Simulating a posterior distribution
# (or bootstrap distribution) of a G-matrix:
G <- matrix(c(1, 1, 0, 1, 4, 1, 0, 1, 2), ncol = 3)
G_mcmc <- sapply(c(G), function(x) rnorm(10, x, 0.01))
G_mcmc <- t(apply(G_mcmc, 1, function(x) {
  G <- matrix(x, ncol = sqrt(length(x)))
  G[lower.tri(G)] <- t(G)[lower.tri(G)]
  c(G)
}))

# Simulating a posterior distribution
# (or bootstrap distribution) of trait means:
means <- c(1, 1.4, 2.1)
means_mcmc <- sapply(means, function(x) rnorm(10, x, 0.01))

# Mean standardizing the G-matrix:
meanStdGMCMC(G_mcmc, means_mcmc)

```

---

 phylH

*Phylogenetic heritability*


---

**Description**

phylH calculates the phylogenetic heritability from an Almer model fit and provides associated uncertainty using parametric bootstrapping.

**Usage**

```
phylH(mod, numerator, residual = "Residual", nsim = 10)
```

**Arguments**

mod	An object of class 'merMod'
numerator	The name of phylogenetic effect level
residual	name of the residual effect level
nsim	number of bootstraps

**Value**

phylH returns a list with the REML estimate, the 95% confidence interval from the parametric bootstrap, and the bootstrap samples.

**Author(s)**

Geir H. Bolstad

**Examples**

```
# See the vignette 'Phylogenetic mixed model'.
```

---

```
plot.rate_gls          Plot of rate_gls object
```

---

**Description**

plot method for class 'rate\_gls'.

**Usage**

```
## S3 method for class 'rate_gls'
plot(
  x,
  scale = "SD",
  print_param = TRUE,
  digits_param = 2,
  digits_rsquared = 1,
  main = "GLS regression",
  xlab = "x",
  ylab = "Response",
  col = "grey",
  cex.legend = 1,
  ...
)
```

**Arguments**

x	An object of class 'rate_gls'.
scale	The scale of the y-axis, either the variance scale ('VAR'), that is $y^2$ , or the standard deviation scale ('SD'), that is $\text{abs}(y)$ .
print_param	logical: if parameter estimates should be printed in the plot or not.
digits_param	The number of significant digits displayed for the parameters in the plots.
digits_rsquared	The number of decimal places displayed for the r-squared.
main	as in <a href="#">plot</a> .
xlab	as in <a href="#">plot</a> .
ylab	as in <a href="#">plot</a> .
col	as in <a href="#">plot</a> .
cex.legend	A character expansion factor relative to current <code>par("cex")</code> for the printed parameters.
...	Additional arguments passed to <a href="#">plot</a> .

**Details**

Plots the gls rate regression fitted by the `rate_gls` function. The regression line gives the expected variance or standard deviation (depending on scale). The regression is linear on the variance scale.

**Value**

plot returns a plot of the gls rate regression

**Author(s)**

Geir H. Bolstad

**Examples**

```
# See the vignette 'Analyzing rates of evolution'.
```

---

```
plot.simulate_rate      Plot of simulate_rate object
```

---

**Description**

plot method for class 'simulate\_rate'.

**Usage**

```
## S3 method for class 'simulate_rate'
plot(
  x,
  response = "rate_y",
  xlab = "Simulation timesteps",
  ylab = "Evolutionary rate of y",
  ...
)
```

**Arguments**

x	An object of class 'simulate_rate'.
response	The variable for the y-axis of the plot, can be 'rate_y', 'y', or 'x'.
xlab	A label for the x axis.
ylab	A label for the y axis.
...	Additional arguments passed to <code>plot</code> .

**Details**

No plot is returned if `model = 'recent_evol'`.

**Value**

`plot` A plot of the evolution of the traits x or y, or the evolution of the evolutionary rate of y (i.e.  $\sqrt{a + bx}$ ) in the simulation.

**Author(s)**

Geir H. Bolstad

**Examples**

```
# See the vignette 'Analyzing rates of evolution'.
```

---

randomBeta

*Generating selection gradients/vectors in random directions.*

---

**Description**

randomBeta generates unit length vectors (selection gradients) uniformly distributed in a k-dimensional hypersphere.

**Usage**

```
randomBeta(n = 1, k = 2)
```

**Arguments**

n	Number of selection gradients/vectors.
k	Number of dimensions.

**Details**

randomBeta exploits the spherical symmetry of a multidimensional Gaussian density function. Each element of each vector is randomly sampled from a univariate Gaussian distribution with zero mean and unit variance. The vector is then divided by its norm to standardize it to unit length.

**Value**

randomBeta returns a matrix where the vectors are stacked column wise.

**Author(s)**

Geir H. Bolstad

**Examples**

```
# Two vectors of dimension 3:  
randomBeta(n = 2, k = 3)
```

rate\_gls

*Generalized least squares rate model***Description**

rate\_gls fits a generalized least squares model to estimate parameters of an evolutionary model of two traits  $x$  and  $y$ , where the evolutionary rate of  $y$  depends on the value of  $x$ . Three models are implemented. In the two first, 'predictor\_BM' and 'predictor\_gBM', the evolution of  $y$  follows a Brownian motion with variance linear in  $x$ , while the evolution of  $x$  either follows a Brownian motion or a geometric Brownian motion, respectively. In the third model, 'recent\_evol', the residuals of the macroevolutionary predictions of  $y$  have variance linear in  $x$ . It is highly recommended to read Hansen et al. (in review) and vignette("Analyzing\_rates\_of\_evolution") before fitting these models.

**Usage**

```
rate_gls(
  x,
  y,
  species,
  tree,
  model = "predictor_BM",
  startv = list(a = NULL, b = NULL),
  maxiter = 100,
  silent = FALSE,
  useLFO = TRUE,
  tol = 0.001
)
```

**Arguments**

<code>x</code>	The explanatory variable, which must be equal to the length of <code>y</code> and tips on the tree.
<code>y</code>	The trait values of response variable. Note that the algorithm mean centers <code>y</code> .
<code>species</code>	A vector with the names of the species, must be equal in length and in the same order as <code>x</code> and <code>y</code> .
<code>tree</code>	An object of class <code>phylo</code> , needs to be ultrametric and with total length of unit, tips must have the same names as in <code>species</code> .
<code>model</code>	The acronym of the evolutionary model to be fitted. There are three options: 'predictor_BM', 'predictor_gBM' or 'recent_evol' (see details).
<code>startv</code>	A vector of optional starting values for the <code>a</code> and <code>b</code> parameters.
<code>maxiter</code>	The maximum number of iterations for updating the GLS.
<code>silent</code>	logical: if the function should not print the generalized sum of squares for each iteration.

useLFO	logical: whether the focal species should be left out when calculating the corresponding species' means. Note that this is only relevant for the 'recent_evol' model. The most correct is to use TRUE, but in practice it has little effect and FALSE will speed up the model fit (particularly useful when bootstrapping). LFO is an acronym for 'Leave the Focal species Out'.
tol	tolerance for convergence. If the change in 'a' and 'b' is below this limit between the two last iteration, convergence is reached. The change is measured in proportion to the standard deviation of the response for 'a' and the ratio of the standard deviation of the response to the standard deviation of the predictor for 'b'.

### Details

rate\_gls is an iterative generalized least squares (GLS) model fitting a regression where the response variable is a vector of squared mean-centered y-values for the 'predictor\_BM' and 'predictor\_gBM' models and squared deviation from the evolutionary predictions (see [macro\\_pred](#)) for the 'recent\_evol' model. Note that the algorithm mean centers x in the 'predictor\_BM' and 'recent\_evol' analyses, while it mean standardized x (i.e. divided x by its mean) in the 'predictor\_gBM'. The evolutionary parameters a and b are inferred from the intercept and the slope of the GLS fit. Again, it is highly recommended to read Hansen et al. (in review) and vignette("Analyzing\_rates\_of\_evolution") before fitting these models. In Hansen et al. (2021) the three models 'predictor\_BM', 'predictor\_gBM' and 'recent\_evol' are referred to as 'Model 1', 'Model 2' and 'Model 3', respectively.

### Value

An object of class 'rate\_gls', which is a list with the following components:

model	The name of the model ('predictor_BM', 'predictor_gBM' or 'recent_evolution').
param	The focal parameter estimates and their standard errors, where 'a' and 'b' are parameters of the model.
Rsquared	The generalized R squared of the GLS model fit.
a_all_iterations	The values for the parameter a in all iterations.
b_all_iterations	The values for the parameter b in all iterations.
R	The residual variance matrix.
Beta	The intercept and slope of GLS regression (response is y2 and explanatory variable is x).
Beta_vcov	The error variance matrix of Beta.
tree	The phylogenetic tree.
data	The data used in the GLS regression.
convergence	Whether the algorithm converged or not.
additional_param	Some additional parameter estimates.
call	The function call.

### Author(s)

Geir H. Bolstad

### References

Hansen TF, Bolstad GH, Tsuboi M. 2021. Analyzing disparity and rates of morphological evolution with model-based phylogenetic comparative methods. *Systematic Biology*. syab079.

doi:10.1093/sysbio/syab079

## Examples

```
# Also see vignette("Analyzing_rates_of_evolution").
## Not run:
# Generating a tree with 500 species
set.seed(102)
tree <- ape::rtree(n = 500)
tree <- ape::chronopl(tree, lambda = 1, age.min = 1)

### model = 'predictor_BM' ###
sim_data <- simulate_rate(tree,
  startv_x = 0, sigma_x = 0.25, a = 1, b = 1, model =
  "predictor_BM"
)
head(sim_data$tips)
gls_mod <- rate_gls(
  x = sim_data$tips$x, y = sim_data$tips$y,
  species = sim_data$tips$species, tree, model = "predictor_BM"
)
gls_mod$param
par(mfrow = c(1, 2))
# Response shown on the standard deviation scale (default):
plot(gls_mod, scale = "SD", cex.legend = 0.8)
# Response shown on the variance scale, where the regression is linear:
plot(gls_mod, scale = "VAR", cex.legend = 0.8)
par(mfrow = c(1, 1))
# Parametric bootstrapping to get the uncertainty of the parameter estimates
# taking the complete process into account.
# (this takes some minutes)
gls_mod_boot <- rate_gls_boot(gls_mod, n = 1000)
gls_mod_boot$summary

### model = 'predictor_gBM' ###
sim_data <- simulate_rate(tree,
  startv_x = 1, sigma_x = 1, a = 1, b = 1,
  model = "predictor_gBM"
)
head(sim_data$tips)
gls_mod <- rate_gls(
  x = sim_data$tips$x, y = sim_data$tips$y, species = sim_data$tips$species,
  tree, model = "predictor_gBM"
)
gls_mod$param
plot(gls_mod)
par(mfrow = c(1, 2))
# Response shown on the standard deviation scale (default):
plot(gls_mod, scale = "SD", cex.legend = 0.8)
# Response shown on the variance scale, where the regression is linear:
plot(gls_mod, scale = "VAR", cex.legend = 0.8)
# is linear.
par(mfrow = c(1, 1))
```



```

# Parametric bootstrapping to get the uncertainty of the parameter estimates
# taking the complete process into account. (This takes some minutes.)
gls_mod_boot <- rate_gls_boot(gls_mod, n = 1000)
gls_mod_boot$summary

### model = 'recent_evol' ###
sim_data <- simulate_rate(tree,
  startv_x = 0, sigma_x = 1, a = 1, b = 1, sigma_y = 1,
  model = "recent_evol"
)
head(sim_data$tips)
gls_mod <- rate_gls(
  x = sim_data$tips$x, y = sim_data$tips$y, species = sim_data$tips$species,
  tree, model = "recent_evol", useLFO = FALSE
)
# useLFO = TRUE is somewhat slower, and although more correct it should give
# very similar estimates in most situations.
gls_mod$param
par(mfrow = c(1, 2))
# Response shown on the standard deviation scale (default):
plot(gls_mod, scale = "SD", cex.legend = 0.8)
# Response shown on the variance scale, where the regression is linear:
plot(gls_mod, scale = "VAR", cex.legend = 0.8)
# linear.
par(mfrow = c(1, 1))

# Parametric bootstrapping to get the uncertainty of the parameter estimates
# taking the complete process into account. Note that x is considered as
# fixed effect. (This takes a long time.)
gls_mod_boot <- rate_gls_boot(gls_mod, n = 1000, useLFO = FALSE)
gls_mod_boot$summary

## End(Not run)

```

---

rate\_gls\_boot

*Bootstrap of the [rate\\_gls](#) model fit*


---

## Description

rate\_gls\_boot performs a parametric bootstrap of a [rate\\_gls](#) model fit.

## Usage

```

rate_gls_boot(
  object,
  n = 10,
  useLFO = TRUE,
  silent = FALSE,
  maxiter = 100,

```

```

    tol = 0.001
  )

```

### Arguments

object	The output from <code>rate_gls</code> .
n	The number of bootstrap samples
useLFO	logical: when calculating the mean vector of the traits in the 'recent_evol' analysis, should the focal species be left out when calculating the corresponding species' mean. The correct way is to use TRUE, but in practice it has little effect and FALSE will speed up the model fit (particularly useful when bootstrapping).
silent	logical: whether or not the bootstrap iterations should be printed.
maxiter	The maximum number of iterations for updating the GLS.
tol	tolerance for convergence. If the change in 'a' and 'b' is below this limit between the two last iteration, convergence is reached. The change is measured in proportion to the standard deviation of the response for 'a' and the ratio of the standard deviation of the response to the standard deviation of the predictor for 'b'.

### Value

A list where the first slot is a table with the original estimates and SE from the GLS fit in the two first columns followed by the bootstrap estimate of the SE and the 2.5%, 50% and 97.5% quantiles of the bootstrap distribution. The second slot contains the complete distribution.

### Author(s)

Geir H. Bolstad

### Examples

```

# See the vignette 'Analyzing rates of evolution' and in the help
# page of rate_gls.

```

---

rate\_gls\_sim

*Simulate responses from `rate_gls_fit`*

---

### Description

rate\_gls\_sim responses from the models defined by an object of class 'rate\_gls'.

### Usage

```
rate_gls_sim(object, nsim = 10)
```

**Arguments**

object            The fitted object from [rate\\_gls](#)  
 nsim             The number of simulations.

**Details**

`rate_gls_sim` simply passes the estimates in an object of class 'rate\_gls' to the function [simulate\\_rate](#) for simulating responses of the evolutionary process. It is mainly intended for internal use in [rate\\_gls\\_boot](#).

**Value**

An object of class 'simulate\_rate', which is a list with the following components:

<code>tips</code>	A data frame of x and y values for the tips.
<code>percent_negative_roots</code>	The percent of iterations with negative roots in the rates of y (not given for model = 'recent_evol').
<code>compl_dynamics</code>	A list with the output of the complete dynamics (not given for model = 'recent_evol').

**Author(s)**

Geir H. Bolstad

**Examples**

```
# See the vignette 'Analyzing rates of evolution'.
```

---

round\_and\_format            *Rounds and formats in the same function*

---

**Description**

`round_and_format` rounds and formats a numeric vector. This is useful for providing output for tables or plots in a standardized format.

**Usage**

```
round_and_format(  
  x,  
  digits = 2,  
  sign_digits = NULL,  
  scientific = FALSE,  
  trim = TRUE  
)
```

**Arguments**

x	A numeric vector.
digits	Number of decimal places.
sign_digits	Number of significant digits (if given this overrides digits).
scientific	logical: whether encoding should be in scientific notation or not.
trim	logical: if leading blanks for justification to common width should be excluded or not.

**Value**

Rounded and formatted values as characters.

**Author(s)**

Geir H. Bolstad

---

simulate\_rate

*Simulating evolutionary rate model*

---

**Description**

simulate\_rate Simulates three different evolutionary rates models. In the two first, 'predictor\_BM' and 'predictor\_gBM', the evolution of y follows a Brownian motion with variance linear in x, while the evolution of x either follows a Brownian motion or a geometric Brownian motion, respectively. In the third model, 'recent\_evol', the residuals of the macroevolutionary predictions of y have variance linear in x.

**Usage**

```
simulate_rate(  
  tree,  
  startv_x = NULL,  
  sigma_x = NULL,  
  a,  
  b,  
  sigma_y = NULL,  
  x = NULL,  
  model = "predictor_BM"  
)
```

**Arguments**

tree	A <a href="#">phylo</a> object. Must be ultrametric and scaled to unit length.
startv_x	The starting value for x (usually 0 for 'predictor_BM' and 'recent_evol', and 1 for 'predictor_gBM').
sigma_x	The evolutionary rate of x.
a	A parameter of the evolutionary rate of y.
b	A parameter of the evolutionary rate of y.
sigma_y	The evolutionary rate for the macroevolution of y (Brownian motion process) used in the 'recent_evolution' model.
x	Optional fixed values of x (only for the 'recent_evol' model), must equal number of tips in the phylogeny, must correspond to the order of the tip labels.
model	Either a Brownian motion model of x 'predictor_BM', geometric Brownian motion model of x 'predictor_gBM', or 'recent_evol'.

**Details**

See the vignette 'Analyzing rates of evolution' for an explanation of the evolutionary models. The data of the tips can be analyzed with the function [rate\\_gls](#). Note that a large part of parameter space will cause negative roots in the rates of y (i.e. negative  $a+bx$ ). In these cases the rates are set to 0. A warning message is given if the number of such instances is larger than 0.1%. For model 1 and 2, it is possible to set 'a=scalame', if this chosen then 'a' will be given the lowest possible value constrained by  $a+bx>0$ .

**Value**

An object of class 'simulate\_rate', which is a list with the following components:

tips	A data frame of x and y values for the tips.
percent_negative_roots	The percent of iterations with negative roots in the rates of y (not given for model = 'recent_evol').
compl_dynamics	A list with the output of the complete dynamics (not given for model = 'recent_evol').

**Author(s)**

Geir H. Bolstad

**Examples**

```
# Also see the vignette 'Analyzing rates of evolution'.
## Not run:
# Generating a tree with 50 species
set.seed(102)
tree <- ape::rtree(n = 50)
tree <- ape::chronopl(tree, lambda = 1, age.min = 1)

### model = 'predictor_BM' ###
sim_data <- simulate_rate(tree, startv_x = 0, sigma_x = 0.25, a = 1, b = 1,
model = "predictor_BM")
```

```

head(sim_data$tips)
par(mfrow = c(1, 3))
plot(sim_data)
plot(sim_data, response = "y")
plot(sim_data, response = "x")
par(mfrow = c(1, 1))

### model = 'predictor_gBM' ###
sim_data <- simulate_rate(tree, startv_x = 1, sigma_x = 1, a = 1, b = 0.1,
model = "predictor_gBM")
head(sim_data$tips)
par(mfrow = c(1, 3))
plot(sim_data)
plot(sim_data, response = "y")
plot(sim_data, response = "x")
par(mfrow = c(1, 1))

### model = 'recent_evol' ###
sim_data <- simulate_rate(tree,
  startv_x = 0, sigma_x = 1, a = 1, b = 1, sigma_y = 1,
  model = "recent_evol"
)
head(sim_data$tips)

## End(Not run)

```

---

summary.evolvabilityBeta

*Summarizing evolvability parameters over a set of selection gradients*


---

## Description

summary method for class 'evolvabilityBeta'.

## Usage

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

## Arguments

object	An object of class 'evolvabilityBeta'.
...	Additional arguments.

## Value

A list with the following components:

Averages	The averages of the evolvability parameters over all selection gradients.
Minimum	The minimum of the evolvability parameters over all selection gradients.
Maximum	The maximum of the evolvability parameters over all selection gradients.

**Author(s)**

Geir H. Bolstad

**See Also**

[evolvabilityBeta](#)

---

summary.evolvabilityBetaMCMC

*Summarizing posterior distribution of evolvability parameters over a set of selection gradients*

---

**Description**

summary method for class 'evolvabilityBetaMCMC'.

**Usage**

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

**Arguments**

object            an object of class 'evolvabilityBetaMCMC'.  
...                additional arguments affecting the summary produced.

**Value**

A list with the following components:

Averages	The averages of the evolvability parameters over all selection gradients.
Minimum	The minimum (given by the posterior median) of the evolvability parameters over all selection gradients.
Maximum	The maximum (given by the posterior median) of the evolvability parameters over all selection gradients.

**Author(s)**

Geir H. Bolstad

**See Also**

[evolvabilityBetaMCMC](#)

# Index

- \* **algebra**
  - conditionalG, 6
  - evolvabilityBeta, 8
  - evolvabilityBetaMCMC, 9
  - evolvabilityBetaMCMC2, 10
  - evolvabilityMeans, 12
  - evolvabilityMeansMCMC, 13
  - meanStdG, 16
  - meanStdGMCMC, 17
  - summary.evolvabilityBeta, 30
  - summary.evolvabilityBetaMCMC, 31
- \* **array**
  - conditionalG, 6
  - evolvabilityBeta, 8
  - evolvabilityBetaMCMC, 9
  - evolvabilityBetaMCMC2, 10
  - evolvabilityMeans, 12
  - evolvabilityMeansMCMC, 13
  - meanStdG, 16
  - meanStdGMCMC, 17
  - summary.evolvabilityBeta, 30
  - summary.evolvabilityBetaMCMC, 31
- \* **multivariate**
  - evolvabilityBetaMCMC, 9
  - evolvabilityBetaMCMC2, 10
  - meanStdGMCMC, 17
- \* **package**
  - evolvability-package, 2
- Almer, 3, 4–6
- Almer\_boot, 4
- Almer\_SE, 5
- Almer\_sim, 6
- conditionalG, 6
- evolvability (evolvability-package), 2
- evolvability-package, 2
- evolvabilityBeta, 8, 31
- evolvabilityBetaMCMC, 9, 31
- evolvabilityBetaMCMC2, 10
- evolvabilityMeans, 12
- evolvabilityMeansMCMC, 13
- GLS, 14
- list, 15
- lme4, 3
- lmer, 3–5
- macro\_pred, 15, 23
- meanStdG, 16
- meanStdGMCMC, 17
- merMod, 4, 5
- phylH, 18
- phylo, 22, 29
- plot, 19, 20
- plot.rate\_gls, 19
- plot.simulate\_rate, 20
- randomBeta, 21
- rate\_gls, 20, 22, 25–27, 29
- rate\_gls\_boot, 25, 27
- rate\_gls\_sim, 26
- round\_and\_format, 27
- simulate.merMod, 6
- simulate\_rate, 27, 28
- summary.evolvabilityBeta, 30
- summary.evolvabilityBetaMCMC, 31