

# Package ‘evolqg’

April 8, 2019

**Type** Package

**Title** Tools for Evolutionary Quantitative Genetics

**Version** 0.2-6

**Date** 2019-04-08

**Author** Ana Paula Assis, Diogo Melo, Edgar Zanella, Fabio Machado, Guilherme Garcia

**Maintainer** Diogo Melo <diogro@usp.br>

**Description** Provides functions for covariance matrix comparisons, estimation of repeatabilities in measurements and matrices, and general evolutionary quantitative genetics tools.

**Depends** R (>= 3.1.0), plyr (>= 1.7.1)

**Imports** Rcpp (>= 0.11), Matrix, reshape2, ggplot2, vegan, ape, expm, matrixcalc, mvtnorm, coda, igraph, MCMCpack, graphics, grDevices, methods, stats, utils

**Suggests** testthat, foreach, magrittr, grid, gridExtra, doParallel

**LinkingTo** Rcpp, RcppArmadillo

**License** MIT + file LICENSE

**BugReports** <https://github.com/lem-usp/evolqg/issues>

**Encoding** UTF-8

**RoxygenNote** 6.1.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-04-08 18:42:54 UTC

## R topics documented:

AlphaRep . . . . .	3
BayesianCalculateMatrix . . . . .	4
BootstrapR2 . . . . .	5
BootstrapRep . . . . .	6

BootstrapStat . . . . .	7
CalcAVG . . . . .	8
CalcEigenSd . . . . .	9
CalcICV . . . . .	10
CalcR2 . . . . .	11
CalcR2CvCorrected . . . . .	12
CalcRepeatability . . . . .	13
CalculateMatrix . . . . .	14
ComparisonMap . . . . .	15
CreateHypotMatrix . . . . .	16
DeltaZCorr . . . . .	16
dentus . . . . .	18
dentus.tree . . . . .	18
DriftTest . . . . .	19
EigenTensorDecomposition . . . . .	20
evolqq . . . . .	22
ExtendMatrix . . . . .	22
KrzCor . . . . .	23
KrzProjection . . . . .	25
KrzSubspace . . . . .	26
LModularity . . . . .	27
MantelCor . . . . .	29
MantelModTest . . . . .	31
MatrixCompare . . . . .	33
MatrixDistance . . . . .	33
MeanMatrix . . . . .	35
MeanMatrixStatistics . . . . .	36
MINT . . . . .	37
MonteCarloR2 . . . . .	38
MonteCarloRep . . . . .	39
MonteCarloStat . . . . .	41
MultiMahalanobis . . . . .	42
MultivDriftTest . . . . .	43
Normalize . . . . .	45
OverlapDist . . . . .	45
PCAsimilarity . . . . .	46
PCScoreCorrelation . . . . .	47
PhyloCompare . . . . .	48
PhyloMantel . . . . .	49
PhyloW . . . . .	51
PlotRarefaction . . . . .	51
PlotTreeDriftTest . . . . .	52
PrintMatrix . . . . .	53
ProjectMatrix . . . . .	54
RandCorr . . . . .	56
RandomMatrix . . . . .	56
RandomSkewers . . . . .	57
Rarefaction . . . . .	59

<i>AlphaRep</i>	3
RarefactionStat . . . . .	61
RemoveSize . . . . .	62
RevertMatrix . . . . .	63
RiemannDist . . . . .	64
RSProjection . . . . .	65
SingleComparisonMap . . . . .	66
SRD . . . . .	67
TestModularity . . . . .	69
TreeDriftTest . . . . .	70
<b>Index</b>	<b>72</b>

---

AlphaRep	<i>Alpha Repetability</i>
----------	---------------------------

---

### Description

Calculates the matrix repeatability using the equation in Cheverud 1996 Quantitative genetic analysis of cranial morphology in the cotton-top (*Saguinus oedipus*) and saddle-back (*S. fuscicollis*) tamarins. *Journal of Evolutionary Biology* 9, 5-42.

### Usage

`AlphaRep(cor.matrix, sample.size)`

### Arguments

<code>cor.matrix</code>	Correlation matrix
<code>sample.size</code>	Sample size used in matrix estimation

### Value

Alpha repeatability for correlation matrix

### Author(s)

Diogo Melo, Guilherme Garcia

### References

Cheverud 1996 Quantitative genetic analysis of cranial morphology in the cotton-top (*Saguinus oedipus*) and saddle-back (*S. fuscicollis*) tamarins. *Journal of Evolutionary Biology* 9, 5-42.

### See Also

[MonteCarloStat](#), [BootstrapRep](#)

**Examples**

```
#For single matrices
cor.matrix <- RandomMatrix(10)
AlphaRep(cor.matrix, 10)
AlphaRep(cor.matrix, 100)
#For many matrices
mat.list <- RandomMatrix(10, 100)
sample.sizes <- floor(runif(100, 20, 50))
unlist(Map(AlphaRep, mat.list, sample.sizes))
```

---

**BayesianCalculateMatrix**

*Calculate Covariance Matrix from a linear model fitted with lm() using different estimators*

---

**Description**

Calculates covariance matrix using the maximum likelihood estimator, the maximum a posteriori (MAP) estimator under a regularized Wishart prior, and if the sample is large enough can give samples from the posterior and the median posterior estimator.

**Usage**

```
BayesianCalculateMatrix(linear.m, samples = NULL, ..., nu = NULL,
  S_0 = NULL)
```

**Arguments**

linear.m	Linear model adjusted for original data
samples	number os samples to be generated from the posterior. Requires sample size to be at least as large as the number of dimensions
...	aditional arguments, currently ignored
nu	degrees of freedom in prior distribution, defaults to the number of traits (this can be a too strong prior)
S_0	cross product matrix of the prior. Default is to use the observed voriaces and zero covariances

**Value**

Estimated covariance matrices and posterior samples

**Author(s)**

Diogo Melo, Fabio Machado

## References

- Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.
- Schafer, J., e Strimmer, K. (2005). A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statistical applications in genetics and molecular biology, 4(1).

## Examples

```
data(iris)
iris.lm = lm(as.matrix(iris[,1:4])~iris[,5])
matrices <- BayesianCalculateMatrix(iris.lm, nu = 0.1, samples = 100)
```

---

BootstrapR2

*R2 confidence intervals by bootstrap resampling*

---

## Description

Random populations are generated by resampling the supplied data or residuals. R2 is calculated on all the random population's correlation matrices, providing a distribution based on the original data.

## Usage

```
BootstrapR2(ind.data, iterations = 1000, parallel = FALSE)
```

## Arguments

<code>ind.data</code>	Matrix of residuals or individual measurements
<code>iterations</code>	Number of resamples to take
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

## Value

returns a vector with the R2 for all populations

## Author(s)

Diogo Melo Guilherme Garcia

## See Also

[BootstrapRep](#), [AlphaRep](#)

## Examples

```
r2.dist <- BootstrapR2(iris[,1:4], 30)
quantile(r2.dist)
```

---

BootstrapRep

*Bootstrap analysis via resampling*

---

### Description

Calculates the repeatability of the covariance matrix of the supplied data via bootstrap resampling

### Usage

```
BootstrapRep(ind.data, ComparisonFunc, iterations = 1000,  
             sample.size = dim(ind.data)[1], correlation = FALSE,  
             parallel = FALSE)
```

### Arguments

<code>ind.data</code>	Matrix of residuals or individual measurements
<code>ComparisonFunc</code>	comparison function
<code>iterations</code>	Number of resamples to take
<code>sample.size</code>	Size of resamples, default is the same size as <code>ind.data</code>
<code>correlation</code>	If TRUE, correlation matrix is used, else covariance matrix.
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

### Details

Samples with replacement are taken from the full population, a statistic calculated and compared to the full population statistic.

### Value

returns the mean repeatability, that is, the mean value of comparisons from samples to original statistic.

### Author(s)

Diogo Melo, Guilherme Garcia

### See Also

[MonteCarloStat](#), [AlphaRep](#)

**Examples**

```

BootstrapRep(iris[,1:4], MantelCor, iterations = 5, correlation = TRUE)

BootstrapRep(iris[,1:4], RandomSkewers, iterations = 50)

BootstrapRep(iris[,1:4], KrzCor, iterations = 50, correlation = TRUE)

BootstrapRep(iris[,1:4], PCAsimilarity, iterations = 50)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#BootstrapRep(iris[,1:4], PCAsimilarity,
#             iterations = 5,
#             parallel = TRUE)

```

---

 BootstrapStat

*Non-Parametric population samples and statistic comparison*


---

**Description**

Random populations are generated via resampling using the supplied population. A statistic is calculated on the random population and compared to the statistic calculated on the original population.

**Usage**

```

BootstrapStat(ind.data, iterations, ComparisonFunc, StatFunc,
             sample.size = dim(ind.data)[1], parallel = FALSE)

```

**Arguments**

<code>ind.data</code>	Matrix of residuals or individual measurements
<code>iterations</code>	Number of resamples to take
<code>ComparisonFunc</code>	comparison function
<code>StatFunc</code>	Function for calculating the statistic
<code>sample.size</code>	Size of resamples, default is the same size as <code>ind.data</code>
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

**Value**

returns the mean repeatability, that is, the mean value of comparisons from samples to original statistic.

**Author(s)**

Diogo Melo, Guilherme Garcia

**See Also**

[BootstrapRep](#), [AlphaRep](#)

**Examples**

```
cov.matrix <- RandomMatrix(5, 1, 1, 10)

BootstrapStat(iris[,1:4], iterations = 50,
              ComparisonFunc = function(x, y) PCAsimilarity(x, y)[1],
              StatFunc = cov)

#Calculating R2 confidence intervals
r2.dist <- BootstrapR2(iris[,1:4], 30)
quantile(r2.dist)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#BootstrapStat(iris[,1:4], iterations = 100,
#              ComparisonFunc = function(x, y) KrzCor(x, y)[1],
#              StatFunc = cov,
#              parallel = TRUE)
```

---

CalcAVG

*Calculates mean correlations within- and between-modules*

---

**Description**

Uses a binary correlation matrix as a mask to calculate average within- and between-module correlations. Also calculates the ratio between them and the Modularity Hypothesis Index.

**Usage**

```
CalcAVG(cor.hypothesis, cor.matrix, MHI = TRUE, landmark.dim = NULL)
```

**Arguments**

`cor.hypothesis` Hypothetical correlation matrix, with 1s within-modules and 0s between modules

`cor.matrix` Observed empirical correlation matrix.



MHI	Indicates if Modularity Hypothesis Index should be calculated instead of AVG Ratio.
landmark.dim	Used if within-landmark correlations are to be excluded in geometric morphometric data. Either 2 for 2d data or 3 for 3d data. Default is NULL for non geometric morphometric data.

**Value**

a named vector with the mean correlations and derived statistics

**Examples**

```
# Module vectors
modules = matrix(c(rep(c(1, 0, 0), each = 5),
                  rep(c(0, 1, 0), each = 5),
                  rep(c(0, 0, 1), each = 5)), 15)

# Binary modular matrix
cor.hypot = CreateHypotMatrix(modules)[[4]]

# Modular correlation matrix
hypot.mask = matrix(as.logical(cor.hypot), 15, 15)
mod.cor = matrix(NA, 15, 15)
mod.cor[ hypot.mask] = runif(length(mod.cor[ hypot.mask]), 0.8, 0.9) # within-modules
mod.cor[!hypot.mask] = runif(length(mod.cor[!hypot.mask]), 0.3, 0.4) # between-modules
diag(mod.cor) = 1
mod.cor = (mod.cor + t(mod.cor))/2 # correlation matrices should be symmetric

CalcAVG(cor.hypot, mod.cor)
CalcAVG(cor.hypot, mod.cor, MHI = TRUE)
```

---

CalcEigenSd

*Integration*

---

**Description**

Calculates the standardized variance of the eigenvalues of a correlation matrix. This is a measure of overall integration between traits.

**Usage**

```
CalcEigenSd(c.matrix)
```

**Arguments**

`c.matrix` Covariance or correlation matrix.

**Value**

Mean squared value of off diagonal elements of correlation matrix

**Author(s)**

Diogo Melo

**References**

Pavlicev, Mihaela, James M. Cheverud, and Gunter P. Wagner. 2009. "Measuring Morphological Integration Using Eigenvalue Variance." *Evolutionary Biology* 36 (1): 157-70.

**See Also**

[CalcR2](#), [CalcICV](#), [Flexibility](#)

**Examples**

```
cov.matrix <- RandomMatrix(10, 1, 1, 10)
# both of the following calls are equivalent,
# CalcEigenSd() converts covariance matrices to correlation matrices internally
CalcEigenSd(cov.matrix)
CalcEigenSd(cov2cor(cov.matrix))
```

---

CalcICV

*Calculates the ICV of a covariance matrix.*

---

**Description**

Calculates the coefficient of variation of the eigenvalues of a covariance matrix, a measure of integration comparable to the  $R^2$  in correlation matrices.

**Usage**

```
CalcICV(cov.matrix)
```

**Arguments**

`cov.matrix`      Covariance matrix.

**Value**

coefficient of variation of the eigenvalues of a covariance matrix

**Author(s)**

Diogo Melo

## References

Shirai, Leila T, and Gabriel Marroig. 2010. "Skull Modularity in Neotropical Marsupials and Monkeys: Size Variation and Evolutionary Constraint and Flexibility." *Journal of Experimental Zoology Part B: Molecular and Developmental Evolution* 314 B (8): 663-83. doi:10.1002/jez.b.21367.

Porto, Arthur, Leila Teruko Shirai, Felipe Bandoni de Oliveira, and Gabriel Marroig. 2013. "Size Variation, Growth Strategies, and the Evolution of Modularity in the Mammalian Skull." *Evolution* 67 (July): 3305-22. doi:10.1111/evo.12177.

## See Also

[CalcR2](#)

## Examples

```
cov.matrix <- RandomMatrix(10, 1, 1, 10)
CalcICV(cov.matrix)
```

---

CalcR2	<i>Mean Squared Correlations</i>
--------	----------------------------------

---

## Description

Calculates the mean squared correlation of a covariance or correlation matrix. Measures integration.

## Usage

```
CalcR2(c.matrix)
```

## Arguments

`c.matrix`      Covariance or correlation matrix.

## Value

Mean squared value of off diagonal elements of correlation matrix

## Author(s)

Diogo Melo, Guilherme Garcia

## References

Porto, Arthur, Felipe B. de Oliveira, Leila T. Shirai, Valderes de Conto, and Gabriel Marroig. 2009. "The Evolution of Modularity in the Mammalian Skull I: Morphological Integration Patterns and Magnitudes." *Evolutionary Biology* 36 (1): 118-35. doi:10.1007/s11692-008-9038-3.

Porto, Arthur, Leila Teruko Shirai, Felipe Bandoni de Oliveira, and Gabriel Marroig. 2013. "Size Variation, Growth Strategies, and the Evolution of Modularity in the Mammalian Skull." *Evolution* 67 (July): 3305-22. doi:10.1111/evo.12177.

**See Also**[Flexibility](#)**Examples**

```
cov.matrix <- RandomMatrix(10, 1, 1, 10)
# both of the following calls are equivalent,
# CalcR2() converts covariance matrices to correlation matrices internally
CalcR2(cov.matrix)
CalcR2(cov2cor(cov.matrix))
```

---

CalcR2CvCorrected	<i>Corrected integration value</i>
-------------------	------------------------------------

---

**Description**

Calculates the Young correction for integration, using bootstrap resampling

**Usage**

```
CalcR2CvCorrected(ind.data, ...)

## Default S3 method:
CalcR2CvCorrected(ind.data, cv.level = 0.06,
  iterations = 1000, parallel = FALSE, ...)

## S3 method for class 'lm'
CalcR2CvCorrected(ind.data, cv.level = 0.06,
  iterations = 1000, ...)
```

**Arguments**

<code>ind.data</code>	Matrix of individual measurements, or adjusted linear model
<code>...</code>	additional arguments passed to other methods
<code>cv.level</code>	Coefficient of variation level chosen for integration index adjustment in linear model. Defaults to 0.06.
<code>iterations</code>	Number of resamples to take
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

**Value**

List with adjusted integration indexes, fitted models and simulated distributions of integration indexes and mean coefficient of variation.

**Author(s)**

Diogo Melo, Guilherme Garcia

**References**

Young, N. M., Wagner, G. P., and Hallgrímsson, B. (2010). Development and the evolvability of human limbs. *Proceedings of the National Academy of Sciences of the United States of America*, 107(8), 3400-5. doi:10.1073/pnas.0911856107

**See Also**

[MeanMatrixStatistics](#), [CalcR2](#)

**Examples**

```
integration.dist = CalcR2CvCorrected(iris[,1:4])

#adjusted values
integration.dist[[1]]

#ploting models
library(ggplot2)
ggplot(integration.dist$dist, aes(r2, mean_cv)) + geom_point() +
  geom_smooth(method = 'lm', color= 'black') + theme_bw()

ggplot(integration.dist$dist, aes(eVals_cv, mean_cv)) + geom_point() +
  geom_smooth(method = 'lm', color= 'black') + theme_bw()
```

---

CalcRepeatability

*Parametric per trait Repeatabilities*

---

**Description**

Estimates the variance in the sample not due to measurement error

**Usage**

```
CalcRepeatability(ID, ind.data)
```

**Arguments**

ID	identity of individuals
ind.data	individual measurements

**Value**

vector of repeatabilities

**Note**

Requires at least two observations per individual

**Author(s)**

Guilherme Garcia

**References**

Lessels, C. M., and Boag, P. T. (1987). Unrepeatable repeatabilities: a common mistake. *The Auk*, 2(January), 116-121.

**Examples**

```
num.ind = length(iris[,1])
ID = rep(1:num.ind, 2)
ind.data = rbind(iris[,1:4], iris[,1:4]+array(rnorm(num.ind*4, 0, 0.1), dim(iris[,1:4])))
CalcRepeatability(ID, ind.data)
```

---

CalculateMatrix

*Calculate Covariance Matrix from a linear model fitted with lm()*

---

**Description**

Calculates covariance matrix using the maximum likelihood estimator and the model residuals.

**Usage**

```
CalculateMatrix(linear.m)
```

**Arguments**

linear.m      Linear model adjusted for original data.

**Value**

Estimated covariance matrix.

**Author(s)**

Diogo Melo, Fabio Machado

**References**

<https://github.com/lem-usp/evolqg/wiki/>

## Examples

```
data(iris)
options(contrasts=c("contr.sum", "contr.poly"))
iris.lm = lm(as.matrix(iris[,1:4])~iris[,5])
cov.matrix <- CalculateMatrix(iris.lm)

#To obtain a correlation matrix, use:
cor.matrix <- cov2cor(cov.matrix)
```

---

ComparisonMap

*Generic Comparison Map functions for creating parallel list methods  
Internal functions for making efficient comparisons.*

---

## Description

Generic Comparison Map functions for creating parallel list methods Internal functions for making efficient comparisons.

## Usage

```
ComparisonMap(matrix.list, MatrixCompFunc, ..., repeat.vector = NULL,
              parallel = FALSE)
```

## Arguments

<code>matrix.list</code>	list of matrices being compared
<code>MatrixCompFunc</code>	Function used to compare pair of matrices, must output a vector: comparisons and probabilities
<code>...</code>	Additional arguments to <code>MatrixCompFunc</code>
<code>repeat.vector</code>	Vector of repeatabilities for correlation correction.
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

## Value

Matrix of comparisons, matrix of probabilities.

## Author(s)

Diogo Melo

## See Also

[MantelCor](#), [KrzCor](#), [RandomSkewers](#)

---

CreateHypotMatrix      *Creates binary correlation matrices*

---

### Description

Takes a binary vector or column matrix and generates list of binary correlation matrices representing the partition in the vectors.

### Usage

```
CreateHypotMatrix(modularity.hypot)
```

### Arguments

modularity.hypot  
 Matrix of hypothesis. Each line represents a trait and each column a module. if modularity.hypot[i,j] == 1, trait i is in module j.

### Value

binary matrix or list of binary matrices. If a matrix is passed, all the vectors are combined in the last binary matrix (total hypothesis of full integration hypothesis).

### Examples

```
rand.hypots <- matrix(sample(c(1, 0), 30, replace=TRUE), 10, 3)
CreateHypotMatrix(rand.hypots)
```

---

DeltaZCorr      *Compare matrices via the correlation between response vectors*

---

### Description

Compares the expected response to selection for two matrices for a specific set of selection gradients (not random gradients like in the RandomSkewers method)

### Usage

```
DeltaZCorr(cov.x, cov.y, skewers, ...)

## Default S3 method:
DeltaZCorr(cov.x, cov.y, skewers, ...)

## S3 method for class 'list'
DeltaZCorr(cov.x, cov.y = NULL, skewers,
           parallel = FALSE, ...)
```



**Arguments**

<code>cov.x</code>	Single covariance matrix or list of covariance matrices. If single matrix is supplied, it is compared to <code>cov.y</code> . If list is supplied and no <code>cov.y</code> is supplied, all matrices are compared. If <code>cov.y</code> is supplied, all matrices in list are compared to it.
<code>cov.y</code>	First argument is compared to <code>cov.y</code> . Optional if <code>cov.x</code> is a list.
<code>skewers</code>	matrix of column vectors to be used as gradients
<code>...</code>	additional arguments passed to other methods.
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

**Value**

vector of vector correlations between the expected responses for the two matrices for each supplied vector

**Author(s)**

Diogo Melo, Guilherme Garcia

**References**

Cheverud, J. M., and Marroig, G. (2007). Comparing covariance matrices: Random skewers method compared to the common principal components model. *Genetics and Molecular Biology*, 30, 461-469.

**See Also**

[KrzCor](#), [MantelCor](#)

**Examples**

```
x <- RandomMatrix(10, 1, 1, 10)
y <- RandomMatrix(10, 1, 1, 10)

n_skewers = 10
skewers = matrix(rnorm(10*n_skewers), 10, n_skewers)
DeltaZCorr(x, y, skewers)
```

---

dentus

*Example multivariate data set*

---

### **Description**

Simulated example of 4 continuous bone lengths from 5 species.

### **Usage**

```
data(dentus)
```

### **Format**

A data frame with 300 rows and 5 variables

### **Details**

- humerus
- ulna
- femur
- tibia
- species

---

dentus.tree

*Tree for dentus example species*

---

### **Description**

Hypothetical tree for the species in the dentus data set.

### **Usage**

```
data(dentus.tree)
```

### **Format**

ape tree object

---

DriftTest	<i>Test drift hypothesis</i>
-----------	------------------------------

---

**Description**

Given a set of covariance matrices and means for terminals, test the hypothesis that observed divergence is larger/smaller than expected by drift alone using a regression of the between-group variances on the within-group eigenvalues.

**Usage**

```
DriftTest(means, cov.matrix, show.plot = TRUE)
```

**Arguments**

means	list or array of species means being compared. array must have means in the rows.
cov.matrix	ancestral covariance matrix for all populations
show.plot	boolean. If TRUE, plot of eigenvalues of ancestral matrix by between group variance is showed.

**Value**

list of results containing:

regression: the linear regression between the log of the eigenvalues of the ancestral matrix and the log of the between group variance (projected on the eigenvectors of the ancestral matrix)

coefficient\_CI\_95: confidence intervals for the regression coefficients

log.between\_group\_variance: log of the between group variance (projected on the ancestral matrix eigenvectors)

log.W\_eVals: log of the ancestral matrix eigenvalues

plot: plot of the regression using ggplot2

**Note**

If the regression coefficient is significantly different to one, the null hypothesis of drift is rejected.

**Author(s)**

Ana Paula Assis, Diogo Melo

**References**

Marroig, G., and Cheverud, J. M. (2004). Did natural selection or genetic drift produce the cranial diversification of neotropical monkeys? *The American Naturalist*, 163(3), 417-428. doi:10.1086/381693

Proa, M., O'Higgins, P. and Monteiro, L. R. (2013), Type I error rates for testing genetic drift with phenotypic covariance matrices: A simulation study. *Evolution*, 67: 185-195. doi: 10.1111/j.1558-5646.2012.01746.x

## Examples

```
#Input can be an array with means in each row or a list of mean vectors
means = array(rnorm(40*10), c(10, 40))
cov.matrix = RandomMatrix(40, 1, 1, 10)
DriftTest(means, cov.matrix)
```

---

EigenTensorDecomposition

*Eigentensor Decomposition*

---

## Description

This function performs eigentensor decomposition on a set of covariance matrices.

## Usage

```
EigenTensorDecomposition(matrices, return.projection = TRUE, ...)
```

```
## S3 method for class 'list'
EigenTensorDecomposition(matrices,
  return.projection = TRUE, ...)
```

```
## Default S3 method:
EigenTensorDecomposition(matrices,
  return.projection = TRUE, ...)
```

## Arguments

matrices	k x k x m array of m covariance matrices with k traits;
return.projection	Should we project covariance matrices into estimated eigentensors? Defaults to TRUE
...	additional arguments for methods

## Details

The number of estimated eigentensors is the minimum between the number of data points (m) and the number of independent variables ( $k(k + 1)/2$ ) minus one, in a similar manner to the usual principal component analysis.

**Value**

List with the following components:

mean mean covariance matrices used to center the sample (obtained from [MeanMatrix](#))

mean.sqrt square root of mean matrix (saved for use in other functions, such as [ProjectMatrix](#) and [RevertMatrix](#))

values vector of ordered eigenvalues associated with eigentensors;

matrices array of eigentensor in matrix form;

projection matrix of unstandardized projected covariance matrices over eigentensors.

**Author(s)**

Guilherme Garcia, Diogo Melo

**References**

Basser P. J., Pajevic S. 2007. Spectral decomposition of a 4th-order covariance tensor: Applications to diffusion tensor MRI. *Signal Processing*. 87:220-236.

Hine E., Chenoweth S. F., Rundle H. D., Blows M. W. 2009. Characterizing the evolution of genetic variance using genetic covariance tensors. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*. 364:1567-78.

**See Also**

[ProjectMatrix](#), [RevertMatrix](#)

**Examples**

```
## Not run:
data(dentus)

dentus.vcv <- daply (dentus, .(species), function(x) cov(x[,-5]))

dentus.vcv <- aperm(dentus.vcv, c(2, 3, 1))

dentus.etd <- EigenTensorDecomposition(dentus.vcv, TRUE)

# Plot some results
par(mfrow = c(1, 2))
plot(dentus.etd $ values, pch = 20, type = 'b', ylab = 'Eigenvalue')
plot(dentus.etd $ projection [, 1:2], pch = 20,
      xlab = 'Eigentensor 1', ylab = 'Eigentensor 2')
text(dentus.etd $ projection [, 1:2],
      labels = rownames (dentus.etd $ projection), pos = 2)

# we can also deal with posterior samples of covariance matrices using plyr

dentus.models <- dlply(dentus, .(species),
                      lm, formula = cbind(humerus, ulna, femur, tibia) ~ 1)
```

```

dentus.matrices <- llply(dentus.models, BayesianCalculateMatrix, samples = 100)

dentus.post.vcv <- laply(dentus.matrices, function (L) L $ Ps)

dentus.post.vcv <- aperm(dentus.post.vcv, c(3, 4, 1, 2))

# this will perform one eigentensor decomposition for each set of posterior samples
dentus.post.etd <- alply(dentus.post.vcv, 4, EigenTensorDecomposition)

# which would allow us to observe the posterior
# distribution of associated eigenvalues, for instance
dentus.post.eval <- laply (dentus.post.etd, function (L) L $ values)

boxplot(dentus.post.eval, xlab = 'Index', ylab = 'Value',
        main = 'Posterior Eigenvalue Distribution')

## End(Not run)

```

---

evolqg	<i>EvolQG</i>
--------	---------------

---

### Description

EvolQG

---

ExtendMatrix	<i>Control Inverse matrix noise with Extension</i>
--------------	--

---

### Description

Calculates the extended covariance matrix estimation as described in Marroig et al. 2012

### Usage

```
ExtendMatrix(cov.matrix, var.cut.off = 1e-04, ret.dim = NULL)
```

### Arguments

cov.matrix	Covariance matrix
var.cut.off	Cut off for second derivative variance. Ignored if ret.dim is passed.
ret.dim	Number of retained eigen values

### Value

Extended covariance matrix and second derivative variance

**Note**

Covariance matrix being extended should be larger then 10x10

**Author(s)**

Diogo Melo

**References**

Marroig, G., Melo, D. A. R., and Garcia, G. (2012). Modularity, noise, and natural selection. *Evolution; international journal of organic evolution*, 66(5), 1506-24. doi:10.1111/j.1558-5646.2011.01555.x

**Examples**

```
cov.matrix = RandomMatrix(11, 1, 1, 100)
ext.matrix = ExtendMatrix(cov.matrix, var.cut.off = 1e-6)
ext.matrix = ExtendMatrix(cov.matrix, ret.dim = 6)
```

---

 KrzCor

---

*Compare matrices via Krzanowski Correlation*


---

**Description**

Calculates covariance matrix correlation via Krzanowski Correlation

**Usage**

```
KrzCor(cov.x, cov.y, ...)

## Default S3 method:
KrzCor(cov.x, cov.y, ret.dim = NULL, ...)

## S3 method for class 'list'
KrzCor(cov.x, cov.y = NULL, ret.dim = NULL,
       repeat.vector = NULL, parallel = FALSE, ...)

## S3 method for class 'mcmc_sample'
KrzCor(cov.x, cov.y, ret.dim = NULL,
       parallel = FALSE, ...)
```

**Arguments**

cov.x	Single covariance matrix or list of covariance matrices. If single matrix is supplied, it is compared to cov.y. If list is supplied and no cov.y is supplied, all matrices are compared to each other. If cov.y is supplied, all matrices in list are compared to it.
cov.y	First argument is compared to cov.y. Optional if cov.x is a list.

... additional arguments passed to other methods  
 ret.dim number of retained dimensions in the comparison, default for nxn matrix is n/2-1  
 repeat.vector Vector of repeatabilities for correlation correction.  
 parallel if TRUE and a list is passed, computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

### Value

If cov.x and cov.y are passed, returns Kzranowski correlation

If cov.x is a list and cov.y is passed, same as above, but for all matrices in cov.x.

If only a list is passed to cov.x, a matrix of Kzranowski correlation values. If repeat.vector is passed, comparison matrix is corrected above diagonal and repeatabilities returned in diagonal.

### Author(s)

Diogo Melo, Guilherme Garcia

### References

Kzranowski, W. J. (1979). Between-Groups Comparison of Principal Components. *Journal of the American Statistical Association*, 74(367), 703. doi:10.2307/2286995

### See Also

[RandomSkewers](#), [KrzProjection](#), [MantelCor](#)

### Examples

```
c1 <- RandomMatrix(10, 1, 1, 10)
c2 <- RandomMatrix(10, 1, 1, 10)
c3 <- RandomMatrix(10, 1, 1, 10)
KrzCor(c1, c2)

KrzCor(list(c1, c2, c3))

reps <- unlist(lapply(list(c1, c2, c3), MonteCarloRep, 10, KrzCor, iterations = 10))
KrzCor(list(c1, c2, c3), repeat.vector = reps)

c4 <- RandomMatrix(10)
KrzCor(list(c1, c2, c3), c4)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#KrzCor(list(c1, c2, c3), parallel = TRUE)
```



KrzProjection

*Compare matrices via Modified Krzanowski Correlation***Description**

Calculates the modified Krzanowski correlation between matrices, projecting the variance in each principal components of the first matrix in to the ret.dim.2 components of the second matrix.

**Usage**

```
KrzProjection(cov.x, cov.y, ...)
```

```
## Default S3 method:
```

```
KrzProjection(cov.x, cov.y, ret.dim.1 = NULL,
  ret.dim.2 = NULL, ...)
```

```
## S3 method for class 'list'
```

```
KrzProjection(cov.x, cov.y = NULL, ret.dim.1 = NULL,
  ret.dim.2 = NULL, parallel = FALSE, full.results = FALSE, ...)
```

**Arguments**

<code>cov.x</code>	Single covariance matrix ou list of covariance matrices. If <code>cov.x</code> is a single matrix is supplied, it is compared to <code>cov.y</code> . If <code>cov.x</code> is a list of matrices is supplied and no <code>cov.y</code> is supplied, all matrices are compared between each other. If <code>cov.x</code> is a list of matrices and a single <code>cov.y</code> matrix is supplied, all matrices in list are compared to it.
<code>cov.y</code>	First argument is compared to <code>cov.y</code> . If <code>cov.x</code> is a list, every element in <code>cov.x</code> is projected in <code>cov.y</code> .
<code>...</code>	additional arguments passed to other methods
<code>ret.dim.1</code>	number of retained dimensions for first matrix in comparison, default for nxn matrix is $n/2-1$
<code>ret.dim.2</code>	number of retained dimensions for second matrix in comparison, default for nxn matrix is $n/2-1$
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .
<code>full.results</code>	if FALSE returns only total variance, if TRUE also per PC variance.

**Value**

Ratio of projected variance to total variance, and ratio of projected total in each PC

**Author(s)**

Diogo Melo, Guilherme Garcia

**References**

Krzanowski, W. J. (1979). Between-Groups Comparison of Principal Components. *Journal of the American Statistical Association*, 74(367), 703. doi:10.2307/2286995

**See Also**

[RandomSkewers](#), [MantelCor](#)

**Examples**

```
c1 <- RandomMatrix(10)
c2 <- RandomMatrix(10)
KrzProjection(c1, c2)

m.list <- RandomMatrix(10, 3)
KrzProjection(m.list)
KrzProjection(m.list, full.results = TRUE)
KrzProjection(m.list, ret.dim.1 = 5, ret.dim.2 = 4)
KrzProjection(m.list, ret.dim.1 = 4, ret.dim.2 = 5)

KrzProjection(m.list, c1)
KrzProjection(m.list, c1, full.results = TRUE)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#KrzProjection(m.list, parallel = TRUE)
```

---

KrzSubspace

*Krzanowski common subspaces analysis*

---

**Description**

Calculates the subspace most similar across a set of covariance matrices.

**Usage**

```
KrzSubspace(cov.matrices, k = NULL)
```

**Arguments**

cov.matrices	list of covariance matrices
k	number of dimention to be retained in calculating the subspace

**Value**

H shared space matrix

k\_eVals\_H eigen values for shared space matrix, maximum value for each is the number of matrices, representing a fully shared direction

k\_eVecs\_H eigen vectors of shared space matrix

angles between each population subspace and each eigen vector of shared space matrix

**Note**

can be used to implement the Baysean comparision from Aguirre et al. 2014

**References**

Aguirre, J. D., E. Hine, K. McGuigan, and M. W. Blows. "Comparing G: multivariate analysis of genetic variation in multiple populations." *Heredity* 112, no. 1 (2014): 21-29.

**Examples**

```
data(dentus)
dentus.matrices = dlply(dentus, .(species), function(x) cov(x[-5]))
KrzSubspace(dentus.matrices, k = 2)

# The method in Aguirre et al. 2014 can de implemented usign this function as follows:

#Random input data with dimensions traits x traits x populations x MCMCsamples:
cov.matrices = aperm(aapply(1:10, 1, function(x) laply(RandomMatrix(6, 40,
    variance = runif(6,1, 10)),
    identity)),
    c(3, 4, 1, 2))

library(magrittr)
Hs = alply(cov.matrices, 4, function(x) alply(x, 3)) %>% llply(function(x) KrzSubspace(x, 3)$H)
avgH = Reduce("+", Hs)/length(Hs)
avgH.vec <- eigen(avgH)$vectors
MCMC.H.val = laply(Hs, function(mat) diag(t(avgH.vec) %**% mat %**% avgH.vec))

# confidence intervals for variation in shared subspace directions
library(coda)
HPDinterval(as.mcmc(MCMC.H.val))
```

---

LModularity

*L Modularity*


---

**Description**

Calculates the L-Modularity (Newman-type modularity) and the partition of traits that minimizes L-Modularity. Wrapper for using correlations matrices in community detections algorithms from igraph.

**Usage**

```
LModularity(cor.matrix, method = optimal.community, ...)
```

**Arguments**

```
cor.matrix    correlation matrix
method        community detection function
...           Additional arguments to igraph community detection function
```

**Value**

List with L-Modularity value and trait partition

**Note**

Community detection is done by transforming the correlation matrix into a weighted graph and using community detection algorithms on this graph. Default method is optimal but slow. See igraph documentation for other options.

If negative correlations are present, the square of the correlation matrix is used as weights.

**References**

Modularity and community structure in networks (2006) M. E. J. Newman, 8577-8582, doi: 10.1073/pnas.0601602103

**Examples**

```
## Not run:
# A modular matrix:
modules = matrix(c(rep(c(1, 0, 0), each = 5),
  rep(c(0, 1, 0), each = 5),
  rep(c(0, 0, 1), each = 5)), 15)
cor.hypot = CreateHypotMatrix(modules)[[4]]
hypot.mask = matrix(as.logical(cor.hypot), 15, 15)
mod.cor = matrix(NA, 15, 15)
mod.cor[ hypot.mask] = runif(length(mod.cor[ hypot.mask]), 0.8, 0.9) # within-modules
mod.cor[!hypot.mask] = runif(length(mod.cor[!hypot.mask]), 0.3, 0.4) # between-modules
diag(mod.cor) = 1
mod.cor = (mod.cor + t(mod.cor))/2 # correlation matrices should be symmetric

# requires a custom igraph installation with GLPK installed in the system
LModularity(mod.cor)
## End(Not run)
```

---

MantelCor *Compare matrices via Mantel Correlation*

---

### Description

Calculates correlation matrix correlation and significance via Mantel test.

### Usage

```
MantelCor(cor.x, cor.y, ...)

## Default S3 method:
MantelCor(cor.x, cor.y, permutations = 1000, ...,
  landmark.dim = NULL, withinLandmark = FALSE, mod = FALSE)

## S3 method for class 'list'
MantelCor(cor.x, cor.y = NULL, permutations = 1000,
  repeat.vector = NULL, parallel = FALSE, ...)

## S3 method for class 'mcmc_sample'
MantelCor(cor.x, cor.y, ..., parallel = FALSE)

MatrixCor(cor.x, cor.y, ...)

## Default S3 method:
MatrixCor(cor.x, cor.y, ...)

## S3 method for class 'list'
MatrixCor(cor.x, cor.y = NULL, permutations = 1000,
  repeat.vector = NULL, parallel = FALSE, ...)

## S3 method for class 'mcmc_sample'
MatrixCor(cor.x, cor.y, ..., parallel = FALSE)
```

### Arguments

cor.x	Single correlation matrix or list of correlation matrices. If single matrix is supplied, it is compared to cor.y. If list is supplied and no cor.y is supplied, all matrices are compared. If cor.y is supplied, all matrices in list are compared to it.
cor.y	First argument is compared to cor.y. Optional if cor.x is a list.
...	additional arguments passed to other methods
permutations	Number of permutations used in significance calculation.
landmark.dim	Used if permutations should be performed maintaining landmark structure in geometric morphometric data. Either 2 for 2d data or 3 for 3d data. Default is NULL for non geometric morphometric data.

<code>withinLandmark</code>	Logical. If TRUE within-landmark correlations are used in the calculation of matrix correlation. Only used if <code>landmark.dim</code> is passed, default is FALSE.
<code>mod</code>	Set TRUE to use <code>mantel</code> in testing modularity hypothesis. Should only be used in <code>MantelModTest</code> .
<code>repeat.vector</code>	Vector of repeatabilities for correlation correction.
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

### Value

If `cor.x` and `cor.y` are passed, returns matrix pearson correlation and significance via Mantel permutations.

If `cor.x` is a list of matrices and `cor.y` is passed, same as above, but for all matrices in `cor.x`.

If only `cor.x` is passed, a matrix of MantelCor average values and probabilities of all comparisons. If `repeat.vector` is passed, comparison matrix is corrected above diagonal and repeatabilities returned in diagonal.

### Note

If the significance is not needed, `MatrixCor` provides the correlation and skips the permutations, so it is much faster.

### Author(s)

Diogo Melo, Guilherme Garcia

### References

[http://en.wikipedia.org/wiki/Mantel\\_test](http://en.wikipedia.org/wiki/Mantel_test)

### See Also

[KrzCor](#), [RandomSkewers](#), [mantel](#), [RandomSkewers](#), [TestModularity](#), [MantelModTest](#)

### Examples

```
c1 <- RandomMatrix(10, 1, 1, 10)
c2 <- RandomMatrix(10, 1, 1, 10)
c3 <- RandomMatrix(10, 1, 1, 10)
MantelCor(cov2cor(c1), cov2cor(c2))

cov.list <- list(c1, c2, c3)
cor.list <- llply(list(c1, c2, c3), cov2cor)

MantelCor(cor.list)

# For repeatabilities we can use MatrixCor, which skips the significance calculation
reps <- unlist(lapply(cov.list, MonteCarloRep, 10, MatrixCor, correlation = TRUE))
MantelCor(cor.list, repeat.vector = reps)
```

```

c4 <- RandomMatrix(10)
MantelCor(cor.list, c4)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#MantelCor(cor.list, parallel = TRUE)

```

---

MantelModTest

*Test single modularity hypothesis using Mantel correlation*


---

### Description

Calculates the correlation and Mantel significance test between a hypothetical binary modularity matrix and a correlation matrix. Also gives mean correlation within- and between-modules. This function is usually only called by TestModularity.

### Usage

```

MantelModTest(cor.hypothesis, cor.matrix, ...)

## Default S3 method:
MantelModTest(cor.hypothesis, cor.matrix,
  permutations = 1000, MHI = FALSE, ..., landmark.dim = NULL,
  withinLandmark = FALSE)

## S3 method for class 'list'
MantelModTest(cor.hypothesis, cor.matrix,
  permutations = 1000, MHI = FALSE, landmark.dim = NULL,
  withinLandmark = FALSE, ..., parallel = FALSE)

```

### Arguments

<code>cor.hypothesis</code>	Hypothetical correlation matrix, with 1s within-modules and 0s between modules.
<code>cor.matrix</code>	Observed empirical correlation matrix.
<code>...</code>	additional arguments passed to MantelCor
<code>permutations</code>	Number of permutations used in significance calculation.
<code>MHI</code>	Indicates if Modularity Hypothesis Index should be calculated instead of AVG Ratio.

landmark.dim	Used if permutations should be performed mantaining landmark structure in geometric morphometric data. Either 2 for 2d data or 3 for 3d data. Default is NULL for non geometric morphometric data.
withinLandmark	Logical. If TRUE within-landmark correlation are used in calculation of correlation. Only used if landmark.dim is passed, default is FALSE.
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

### Details

CalcAVG can be used when a significance test is not required.

### Value

Returns a vector with the matrix correlation, significance via Mantel, within- and between module correlation.

### Author(s)

Diogo Melo, Guilherme Garcia

### References

Porto, Arthur, Felipe B. Oliveira, Leila T. Shirai, Valderes Conto, and Gabriel Marroig. 2009. "The Evolution of Modularity in the Mammalian Skull I: Morphological Integration Patterns and Magnitudes." *Evolutionary Biology* 36 (1): 118-35. doi:10.1007/s11692-008-9038-3.

Modularity and Morphometrics: Error Rates in Hypothesis Testing Guilherme Garcia, Felipe Bandoni de Oliveira, Gabriel Marroig bioRxiv 030874; doi: <http://dx.doi.org/10.1101/030874>

### See Also

[mantel](#), [MantelCor](#), [CalcAVG](#), [TestModularity](#)

### Examples

```
# Create a single modularity hypothesis:
hypot = rep(c(1, 0), each = 6)
cor.hypot = CreateHypotMatrix(hypot)

# First with an unstructured matrix:
un.cor = RandomMatrix(12)
MantelModTest(cor.hypot, un.cor)

# Now with a modular matrix:
hypot.mask = matrix(as.logical(cor.hypot), 12, 12)
mod.cor = matrix(NA, 12, 12)
mod.cor[ hypot.mask] = runif(length(mod.cor[ hypot.mask]), 0.8, 0.9) # within-modules
mod.cor[!hypot.mask] = runif(length(mod.cor[!hypot.mask]), 0.3, 0.4) # between-modules
diag(mod.cor) = 1
mod.cor = (mod.cor + t(mod.cor))/2 # correlation matrices should be symmetric
```



```
MantelModTest(cov.hypot, mod.cor)
```

---

MatrixCompare

*Matrix Compare*

---

### Description

Compare two matrices using all available methods. Currently RandomSkewers, MantelCor, KrzCor and PCASimilarity

### Usage

```
MatrixCompare(cov.x, cov.y, id = ".id")
```

### Arguments

cov.x	covariance or correlation matrix
cov.y	covariance or correlation matrix
id	name of the comparison column

### Value

data.frame of comparisons

### Examples

```
cov.x = RandomMatrix(10, 1, 1, 10)
cov.y = RandomMatrix(10, 1, 10, 20)
MatrixCompare(cov.x, cov.y)
```

---

MatrixDistance

*Matrix distance*

---

### Description

Calculates Distances between covariance matrices.

**Usage**

```
MatrixDistance(cov.x, cov.y, distance, ...)

## Default S3 method:
MatrixDistance(cov.x, cov.y,
  distance = c("OverlapDist", "RiemannDist"), ...)

## S3 method for class 'list'
MatrixDistance(cov.x, cov.y = NULL,
  distance = c("OverlapDist", "RiemannDist"), ..., parallel = FALSE)
```

**Arguments**

<code>cov.x</code>	Single covariance matrix or list of covariance matrices. If single matrix is supplied, it is compared to <code>cov.y</code> . If list is supplied and no <code>cov.y</code> is supplied, all matrices are compared. If <code>cov.y</code> is supplied, all matrices in list are compared to it.
<code>cov.y</code>	First argument is compared to <code>cov.y</code> . Optional if <code>cov.x</code> is a list.
<code>distance</code>	distance function for use in calculation. Currently supports "Riemann" and "Overlap".
<code>...</code>	additional arguments passed to other methods
<code>parallel</code>	if TRUE and a list is passed, computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

**Value**

If `cov.x` and `cov.y` are passed, returns distance between them.

If is a list `cov.x` and `cov.y` are passed, same as above, but for all matrices in `cov.x`.

If only a list is passed to `cov.x`, a matrix of Distances is returned

**Author(s)**

Diogo Melo

**See Also**

[RiemannDist,OverlapDist](#)

**Examples**

```
c1 <- RandomMatrix(10)
c2 <- RandomMatrix(10)
c3 <- RandomMatrix(10)
MatrixDistance(c1, c2, "OverlapDist")
MatrixDistance(c1, c2, "RiemannDist")

MatrixDistance(list(c1, c2, c3), distance = "OverlapDist")
```

```
c4 <- RandomMatrix(10)
MatrixDistance(list(c1, c2, c3), c4)
```

---

MeanMatrix

*Mean Covariance Matrix*

---

### Description

Estimate geometric mean for a set of covariance matrices

### Usage

```
MeanMatrix(matrix.array, tol = 1e-10)
```

### Arguments

`matrix.array`     $k \times k \times m$  array of covariance matrices, with  $k$  traits and  $m$  matrices  
`tol`                minimum riemannian distance between sequential iterated means for accepting an estimated matrix

### Value

geometric mean covariance matrix

### Author(s)

Guilherme Garcia, Diogo Melo

### References

Bini, D. A., Iannazzo, B. 2013. Computing the Karcher Mean of Symmetric Positive Definite Matrices. *Linear Algebra and Its Applications*, 16th ILAS Conference Proceedings, Pisa 2010, 438 (4): 1700-1710. doi:10.1016/j.laa.2011.08.052.

### See Also

[EigenTensorDecomposition](#), [RiemannDist](#)

---

MeanMatrixStatistics *Calculate mean values for various matrix statistics*

---

### Description

Calculates: Mean Squared Correlation, ICV, Autonomy, ConditionalEvolvability, Constraints, Evolvability, Flexibility, Pc1Percent, Respondability.

### Usage

```
MeanMatrixStatistics(cov.matrix, iterations = 1000, full.results = F,
  parallel = FALSE)
```

### Arguments

<code>cov.matrix</code>	A covariance matrix
<code>iterations</code>	Number of random vectors to be used in calculating the stochastic statistics
<code>full.results</code>	If TRUE, full distribution of statistics will be returned.
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

### Value

`dist` Full distribution of stochastic statistics, only if `full.results == TRUE`  
`mean` Mean value for all statistics

### Author(s)

Diogo Melo Guilherme Garcia

### References

Hansen, T. F., and Houle, D. (2008). Measuring and comparing evolvability and constraint in multivariate characters. *Journal of evolutionary biology*, 21(5), 1201-19. doi:10.1111/j.1420-9101.2008.01573.x

### Examples

```
cov.matrix <- cov(iris[,1:4])
MeanMatrixStatistics(cov.matrix)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##Mac and Linux:
#registerDoParallel(cores = 2)
#MeanMatrixStatistics(cov.matrix, parallel = TRUE)
```

**Description**

Combines and compares many modularity hypothesis to a covariance matrix. Comparison values are adjusted to the number of zeros in the hypothesis using a linear regression. Best hypothesis can be assessed using a jackknife procedure.

**Usage**

```
MINT(c.matrix, modularity.hypot, significance = FALSE,
      sample.size = NULL, iterations = 1000)
```

```
JackknifeMINT(ind.data, modularity.hypot, n = 1000,
               leave.out = floor(dim(ind.data)[1]/10), ...)
```

**Arguments**

c.matrix	Correlation or covariance matrix
modularity.hypot	Matrix of hypothesis. Each line represents a trait and each column a module. if modularity.hypot[i,j] == 1, trait i is in module j.
significance	Logical. Indicates if goodness of fit test should be performed.
sample.size	sample size in goodness of fit simulations via MonteCarlo
iterations	number of goodness of fit simulations
ind.data	Matrix of residuals or individual measurements
n	number of jackknife resamplings
leave.out	number of individuals to be left out of each jackknife, default is 10%
...	additional arguments to be passed to rapply for the jackknife

**Note**

Hypothesis can be named as column names, and these will be used to make labels in the output.

Jackknife will return the best hypothesis for each sample.

**References**

Marquez, E.J. 2008. A statistical framework for testing modularity in multidimensional data. *Evolution* 62:2688-2708.

Parsons, K.J., Marquez, E.J., Albertson, R.C. 2012. Constraint and opportunity: the genetic basis and evolution of modularity in the cichlid mandible. *The American Naturalist* 179:64-78.

[http://www-personal.umich.edu/~emarquez/morph/doc/mint\\_man.pdf](http://www-personal.umich.edu/~emarquez/morph/doc/mint_man.pdf)

**Examples**

```

# Creating a modular matrix:
modules = matrix(c(rep(c(1, 0, 0), each = 5),
                    rep(c(0, 1, 0), each = 5),
                    rep(c(0, 0, 1), each = 5)), 15)

cor.hypot = CreateHypotMatrix(modules)[[4]]
hypot.mask = matrix(as.logical(cor.hypot), 15, 15)
mod.cor = matrix(NA, 15, 15)
mod.cor[ hypot.mask] = runif(length(mod.cor[ hypot.mask]), 0.8, 0.9) # within-modules
mod.cor[!hypot.mask] = runif(length(mod.cor[!hypot.mask]), 0.1, 0.2) # between-modules
diag(mod.cor) = 1
mod.cor = (mod.cor + t(mod.cor))/2 # correlation matrices should be symmetric

# True hypothesis and a bunch of random ones.
hypothetical.modules = cbind(modules, matrix(sample(c(1, 0), 4*15, replace=TRUE), 15, 4))

# if hypothesis columns are not named they are assigned numbers
colnames(hypothetical.modules) <- letters[1:7]

MINT(mod.cor, hypothetical.modules)

random_var = runif(15, 1, 10)
mod.data = rmvnorm::rmvnorm(100, sigma = sqrt(outer(random_var, random_var)) * mod.cor)
out_jack = JackKnifeMINT(mod.data, hypothetical.modules, n = 50)

library(ggplot2)

ggplot(out_jack, aes(rank, corrected.gamma)) + geom_point() +
  geom_errorbar(aes(ymin = lower.corrected, ymax = upper.corrected))

```

---

MonteCarloR2

*R2 confidence intervals by parametric sampling*


---

**Description**

Using a multivariate normal model, random populations are generated using the supplied covariance matrix. R2 is calculated on all the random population, providing a distribution based on the original matrix.

**Usage**

```

MonteCarloR2(cov.matrix, sample.size, iterations = 1000,
             parallel = FALSE)

```

**Arguments**

cov.matrix	Covariance matrix.
sample.size	Size of the random populations

<code>iterations</code>	Number of random populations
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

**Details**

Since this function uses multivariate normal model to generate populations, only covariance matrices should be used.

**Value**

returns a vector with the R2 for all populations

**Author(s)**

Diogo Melo Guilherme Garcia

**See Also**

[BootstrapRep](#), [AlphaRep](#)

**Examples**

```
r2.dist <- MonteCarloR2(RandomMatrix(10, 1, 1, 10), 30)
quantile(r2.dist)
```

---

MonteCarloRep

*Parametric repeatabilities with covariance or correlation matrices*

---

**Description**

Using a multivariate normal model, random populations are generated using the supplied covariance matrix. A statistic is calculated on the random population and compared to the statistic calculated on the original matrix.

**Usage**

```
MonteCarloRep(cov.matrix, sample.size, ComparisonFunc, ...,
              iterations = 1000, correlation = FALSE, parallel = FALSE)
```

**Arguments**

<code>cov.matrix</code>	Covariance matrix.
<code>sample.size</code>	Size of the random populations.
<code>ComparisonFunc</code>	comparison function.
<code>...</code>	Additional arguments passed to <code>ComparisonFunc</code> .
<code>iterations</code>	Number of random populations.

correlation	If TRUE, correlation matrix is used, else covariance matrix. MantelCor and MatrixCor should always uses correlation matrix.
parallel	If is TRUE and list is passed, computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

### Details

Since this function uses multivariate normal model to generate populations, only covariance matrices should be used, even when computing repeatabilities for covariances matrices.

### Value

returns the mean repeatability, or mean value of comparisons from samples to original statistic.

### Author(s)

Diogo Melo Guilherme Garcia

### See Also

[BootstrapRep](#), [AlphaRep](#)

### Examples

```

cov.matrix <- RandomMatrix(5, 1, 1, 10)

MonteCarloRep(cov.matrix, sample.size = 30, RandomSkewers, iterations = 20)
MonteCarloRep(cov.matrix, sample.size = 30, RandomSkewers, num.vectors = 100,
              iterations = 20, correlation = TRUE)
MonteCarloRep(cov.matrix, sample.size = 30, MatrixCor, correlation = TRUE)
MonteCarloRep(cov.matrix, sample.size = 30, KrzCor, iterations = 20)
MonteCarloRep(cov.matrix, sample.size = 30, KrzCor, correlation = TRUE)

#Creating repeatability vector for a list of matrices
mat.list <- RandomMatrix(5, 3, 1, 10)
laply(mat.list, MonteCarloRep, 30, KrzCor, correlation = TRUE)

##Multiple threads can be used with doMC library
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#MonteCarloRep(cov.matrix, 30, RandomSkewers, iterations = 100, parallel = TRUE)

```



---

MonteCarloStat	<i>Parametric population samples with covariance or correlation matrices</i>
----------------	--

---

### Description

Using a multivariate normal model, random populations are generated using the supplied covariance matrix. A statistic is calculated on the random population and compared to the statistic calculated on the original matrix.

### Usage

```
MonteCarloStat(cov.matrix, sample.size, iterations, ComparisonFunc,  
StatFunc, parallel = FALSE)
```

### Arguments

<code>cov.matrix</code>	Covariance matrix.
<code>sample.size</code>	Size of the random populations
<code>iterations</code>	Number of random populations
<code>ComparisonFunc</code>	Comparison functions for the calculated statistic
<code>StatFunc</code>	Function for calculating the statistic
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

### Details

Since this function uses multivariate normal model to generate populations, only covariance matrices should be used.

### Value

returns the mean repeatability, or mean value of comparisons from samples to original statistic.

### Author(s)

Diogo Melo, Guilherme Garcia

### See Also

[BootstrapRep](#), [AlphaRep](#)

**Examples**

```

cov.matrix <- RandomMatrix(5, 1, 1, 10)

MonteCarloStat(cov.matrix, sample.size = 30, iterations = 50,
               ComparisonFunc = function(x, y) PCAsimilarity(x, y)[1],
               StatFunc = cov)

#Calculating R2 confidence intervals
r2.dist <- MonteCarloR2(RandomMatrix(10, 1, 1, 10), 30)
quantile(r2.dist)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##Mac and Linux:
#registerDoParallel(cores = 2)
#MonteCarloStat(cov.matrix, sample.size = 30, iterations = 100,
#               ComparisonFunc = function(x, y) KrzCor(x, y)[1],
#               StatFunc = cov,
#               parallel = TRUE)

```

---

MultiMahalanobis

*Calculate Mahalanobis distance for many vectors*


---

**Description**

Calculates the Mahalanobis distance between a list of species mean, using a global covariance matrix

**Usage**

```
MultiMahalanobis(means, cov.matrix, parallel = FALSE)
```

**Arguments**

means	list or array of species means being compared. array must have means in the rows.
cov.matrix	a single covariance matrix defining the scale (or metric tensor) to be used in the distance calculation.
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

**Value**

returns a matrix of species-species distances.

**Author(s)**

Diogo Melo

**References**[http://en.wikipedia.org/wiki/Mahalanobis\\_distance](http://en.wikipedia.org/wiki/Mahalanobis_distance)**See Also**[mahalanobis](#)**Examples**

```

mean.1 <- colMeans(matrix(rnorm(30*10), 30, 10))
mean.2 <- colMeans(matrix(rnorm(30*10), 30, 10))
mean.3 <- colMeans(matrix(rnorm(30*10), 30, 10))
mean.list <- list(mean.1, mean.2, mean.3)

# If cov.matrix is the identity, calculated distance is euclidian:
euclidian <- MultiMahalanobis(mean.list, diag(10))
# Using a matrix with half the variance will give twice the distance between each mean:
half.euclidian <- MultiMahalanobis(mean.list, diag(10)/2)

# Other covariance matrices will give different distances, measured in the scale of the matrix
non.euclidian <- MultiMahalanobis(mean.list, RandomMatrix(10))

#Input can be an array with means in each row
mean.array = array(1:36, c(9, 4))
mat = RandomMatrix(4)
MultiMahalanobis(mean.array, mat)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##Mac and Linux:
#registerDoParallel(cores = 2)
#MultiMahalanobis(mean.list, RandomMatrix(10), parallel = TRUE)

```

---

MultivDriftTest

*Multivariate genetic drift test for 2 populations*


---

**Description**

This function estimates populations evolving through drift from an ancestral population, given an effective population size, number of generations separating them and the ancestral G-matrix. It calculates the magnitude of morphological divergence expected and compare it to the observed magnitude of morphological change.

**Usage**

```
MultivDriftTest(population1, population2, G, Ne, generations,  
iterations = 1000)
```

**Arguments**

population1	dataframe with original measurements for the ancestral population
population2	dataframe with original measurements for the derived population
G	ancestral G matrix
Ne	effective population size estimated for the populations
generations	time in generations separating both populations
iterations	number of simulations to perform

**Value**

list with the 95 drift and the range of the observed magnitude of morphological change

**Note**

Each trait is estimated independently.

**Author(s)**

Ana Paula Assis

**References**

Hohenlohe, P.A ; Arnold, S.J. (2008). MIPod: a hypothesis testing framework for microevolutionary inference from patterns of divergence. *American Naturalist*, 171(3), 366-385. doi: 10.1086/527498

**Examples**

```
data(dentus)  
A <- dentus[dentus$species=="A",-5]  
B <- dentus[dentus$species=="B",-5]  
G <- cov(A)  
MultivDriftTest(A, B, G, Ne = 1000, generations = 250)
```

---

Normalize	<i>Normalize and Norm</i>
-----------	---------------------------

---

**Description**

Norm returns the euclidian norm of a vector, Normalize returns a vector with unit norm.

**Usage**

```
Normalize(x)
```

```
Norm(x)
```

**Arguments**

x                    Numeric vector

**Value**

Normalized vector or input vector norm.

**Author(s)**

Diogo Melo, Guilherme Garcia

**Examples**

```
x <- rnorm(10)
n.x <- Normalize(x)
Norm(x)
Norm(n.x)
```

---

OverlapDist	<i>Distribution overlap distance</i>
-------------	--------------------------------------

---

**Description**

Calculates the overlap between two normal distributions, defined as the probability that a draw from one distribution comes from the other

**Usage**

```
OverlapDist(cov.x, cov.y, iterations = 10000)
```

**Arguments**

cov.x	covariance or correlation matrix
cov.y	covariance or correlation matrix
iterations	number of drows

**Value**

Overlap distance between cov.x and cov.y

**References**

Ovaskainen, O. (2008). A Bayesian framework for comparative quantitative genetics. *Proceedings of the Royal Society B*, 669-678. doi:10.1098/rspb.2007.0949

---

PCAsimilarity

*Compare matrices using PCA similarity factor*

---

**Description**

Compare matrices using PCA similarity factor

**Usage**

```
PCAsimilarity(cov.x, cov.y, ...)

## Default S3 method:
PCAsimilarity(cov.x, cov.y, ret.dim = NULL, ...)

## S3 method for class 'list'
PCAsimilarity(cov.x, cov.y = NULL, ...,
  repeat.vector = NULL, parallel = FALSE)

## S3 method for class 'mcmc_sample'
PCAsimilarity(cov.x, cov.y, ..., parallel = FALSE)
```

**Arguments**

cov.x	Single covariance matrix ou list of covariance matrices. If cov.x is a single matrix, it is compared to cov.y. If cov.x is a list and no cov.y is supplied, all matrices are compared to each other. If cov.x is a list and cov.y is supplied, all matrices in cov.x are compared to cov.y.
cov.y	First argument is compared to cov.y.
...	additional arguments passed to other methods
ret.dim	number of retained dimensions in the comparison. Defaults to all.
repeat.vector	Vector of repeatabilities for correlation correction.
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

**Value**

Ratio of projected variance to total variance

**Author(s)**

Edgar Zanella Alvarenga

**References**

Singhal, A. and Seborg, D. E. (2005), Clustering multivariate time-series data. *J. Chemometrics*, 19: 427-438. doi: 10.1002/cem.945

**See Also**

[KrzProjection](#), [KrzCor](#), [RandomSkewers](#), [MantelCor](#)

**Examples**

```
c1 <- RandomMatrix(10)
c2 <- RandomMatrix(10)
PCAsimilarity(c1, c2)

m.list <- RandomMatrix(10, 3)
PCAsimilarity(m.list)

PCAsimilarity(m.list, c1)
```

---

PCScoreCorrelation      *PC Score Correlation Test*

---

**Description**

Given a set of covariance matrices and means for terminals, test the hypothesis that observed divergence is larger/smaller than expected by drift alone using the correlation on principal component scores.

**Usage**

```
PCScoreCorrelation(means, cov.matrix, taxons = names(means),
  show.plots = FALSE)
```

**Arguments**

means	list or array of species means being compared. array must have means in the rows.
cov.matrix	ancestral covariance matrix for all populations
taxons	names of taxons being compared. Must be in the same order of the means.
show.plots	boolean. If TRUE, plot of eigenvalues of ancetral matrix by between group variance is showed.

**Value**

list of results containing:

correlation matrix of principal component scores and p.values for each correlation. Lower triangle of output are correlations, and upper triangle are p.values.

if show.plots is TRUE, also returns a list of plots of all projections of the nth PCs, where n is the number of taxons.

**Author(s)**

Ana Paula Assis, Diogo Melo

**References**

Marroig, G., and Cheverud, J. M. (2004). Did natural selection or genetic drift produce the cranial diversification of neotropical monkeys? *The American Naturalist*, 163(3), 417-428. doi:10.1086/381693

**Examples**

```
#Input can be an array with means in each row or a list of mean vectors
means = array(rnorm(40*10), c(10, 40))
cov.matrix = RandomMatrix(40, 1, 1, 10)
taxons = LETTERS[1:10]
PCScoreCorrelation(means, cov.matrix, taxons)

##Plots list can be displayed using grid.arrange()
#library(gridExtra)
#pc.score.output <- PCScoreCorrelation(means, cov.matrix, taxons, TRUE)
#do.call(grid.arrange, c(pc.score.output$plots, list(nrow=4, ncol=6)))
##Or we can print to file:
#ggsave("multipage.pdf", do.call(marrangeGrob, c(pc.score.output$plots, list(nrow=2, ncol=2))))
```

---

PhyloCompare

*Compares sister groups*

---

**Description**

Calculates the comparison of some statistic between sister groups along a phylogeny

**Usage**

```
PhyloCompare(tree, node.data, ComparisonFunc = PCAsimilarity, ...)
```

**Arguments**

tree	phylogenetic tree
node.data	list of node data
ComparisonFunc	comparison function, default is PCAsimilarity
...	Additional arguments passed to ComparisonFunc



**Value**

list with a data.frame of calculated comparisons for each node, using labels or numbers from tree; and a list of comparisons for plotting using phytools (see examples)

**Note**

Phylogeny must be fully resolved

**Author(s)**

Diogo Melo

**Examples**

```
library(ape)
data(bird.orders)
tree <- bird.orders
mat.list <- RandomMatrix(5, length(tree$tip.label))
names(mat.list) <- tree$tip.label
sample.sizes <- runif(length(tree$tip.label), 15, 20)
phylo.state <- PhyloW(tree, mat.list, sample.sizes)

phylo.comparisons <- PhyloCompare(tree, phylo.state)

# plotting results on a phylogeny:
## Not run:
library(phytools)
plotBranchbyTrait(tree, phylo.comparisons[[2]])

## End(Not run)
```

---

PhyloMantel

*Mantel test with phylogenetic permutations*

---

**Description**

Performs a matrix correlation with significance given by a phylogenetic Mantel Test. Pairs of rows and columns are permuted with probability proportional to their phylogenetic distance.

**Usage**

```
PhyloMantel(tree, matrix.1, matrix.2, ..., permutations = 1000,
  ComparisonFunc = function(x, y) cor(x[lower.tri(x)], y[lower.tri(y)]),
  k = 1)
```

**Arguments**

tree	phylogenetic tree. Tip labels must match names in input matrices
matrix.1	pair-wise comparison/distance matrix
matrix.2	pair-wise comparison/distance matrix
...	additional parameters, currently ignored
permutations	Number of permutations used in significance calculation
ComparisonFunc	comparison function, default is MatrixCor
k	determines the influence of the phylogeny. 1 is strong influence, and larger values converge to a traditional mantel test.

**Value**

returns a vector with the comparison value and the proportion of times the observed comparison is smaller than the correlations from the permutations.

**Note**

This method should only be used when there is no option other than representing data as pair-wise. It suffers from low power, and alternatives should be used when available.

**Author(s)**

Diogo Melo, adapted from Harmon & Glor 2010

**References**

Harmon, L. J., & Glor, R. E. (2010). Poor statistical performance of the Mantel test in phylogenetic comparative analyses. *Evolution*, 64(7), 2173-2178.

Lapointe, F. J., & Garland, Jr, T. (2001). A generalized permutation model for the analysis of cross-species data. *Journal of Classification*, 18(1), 109-127.

**Examples**

```
data(dentus)
data(dentus.tree)
tree = dentus.tree
cor.matrices = dply(dentus, ,(species), function(x) cor(x[-5]))
comparisons = MatrixCor(cor.matrices)

sp.means = dply(dentus, ,(species), function(x) colMeans(x[-5]))
mh.dist = MultiMahalanobis(means = sp.means, cov.matrix = PhyloW(dentus.tree, cor.matrices)$'6')
PhyloMantel(dentus.tree, comparisons, mh.dist, k = 10000)

#similar to MantelCor for large k:
## Not run:
PhyloMantel(dentus.tree, comparisons, mh.dist, k = 10000)
MantelCor(comparisons, mh.dist)

## End(Not run)
```

---

PhyloW	<i>Calculates ancestral states of some statistic</i>
--------	--

---

**Description**

Calculates weighted average of covariances matrices along a phylogeny, returning a within-group covariance matrix for each node.

**Usage**

```
PhyloW(tree, tip.data, tip.sample.size = NULL)
```

**Arguments**

tree	phylogenetic tree
tip.data	list of tip nodes covariance matrices
tip.sample.size	vector of tip nodes sample sizes

**Value**

list with calculated within-group matrices, using labels or numbers from tree

**Examples**

```
library(ape)
data(dentus)
data(dentus.tree)
tree <- dentus.tree
mat.list <- dply(dentus, 'species', function(x) cov(x[,1:4]))
sample.sizes <- runif(length(tree$tip.label), 15, 20)
PhyloW(tree, mat.list, sample.sizes)
```

---

PlotRarefaction	<i>Plot Rarefaction analysis</i>
-----------------	----------------------------------

---

**Description**

A specialized plotting function displays the results from Rarefaction functions in publication quality.

**Usage**

```
PlotRarefaction(comparison.list, y.axis = "Statistic",
  x.axis = "Number of sampled specimens")
```

**Arguments**

comparison.list      output from rarefaction functions can be used in plotting  
y.axis                Y axis lable in plot  
x.axis                Y axis lable in plot

**Author(s)**

Diogo Melo, Guilherme Garcia

**See Also**

[BootstrapRep](#)

**Examples**

```
ind.data <- iris[1:50,1:4]

results.RS <- Rarefaction(ind.data, PCAsimilarity, num.reps = 5)
results.Mantel <- Rarefaction(ind.data, MatrixCor, correlation = TRUE, num.reps = 5)
results.KrzCov <- Rarefaction(ind.data, KrzCor, num.reps = 5)
results.PCA <- Rarefaction(ind.data, PCAsimilarity, num.reps = 5)

#Plotting using ggplot2
a <- PlotRarefaction(results.RS, "Random Skewers")
b <- PlotRarefaction(results.Mantel, "Mantel")
c <- PlotRarefaction(results.KrzCov, "KrzCor")
d <- PlotRarefaction(results.PCA, "PCAsimilarity")

library(grid)
grid.newpage()
pushViewport(viewport(layout = grid.layout(2, 2)))
vplayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
print(a, vp = vplayout(1, 1))
print(b, vp = vplayout(1, 2))
print(c, vp = vplayout(2, 1))
print(d, vp = vplayout(2, 2))
```

---

PlotTreeDriftTest      *Plot results from TreeDriftTest*

---

**Description**

Plot which labels reject drift hypothesis.

**Usage**

```
PlotTreeDriftTest(test.list, tree, ...)
```

**Arguments**

test.list	Output from TreeDriftTest
tree	phylogenetic tree
...	adition arguments to plot

**Author(s)**

Diogo Melo

**See Also**

DriftTest TreeDriftTest

**Examples**

```
library(ape)
data(bird.orders)

tree <- bird.orders
mean.list <- llply(tree$tip.label, function(x) rnorm(5))
names(mean.list) <- tree$tip.label
cov.matrix.list <- RandomMatrix(5, length(tree$tip.label))
names(cov.matrix.list) <- tree$tip.label
sample.sizes <- runif(length(tree$tip.label), 15, 20)

test.list <- TreeDriftTest(tree, mean.list, cov.matrix.list, sample.sizes)
PlotTreeDriftTest(test.list, tree)
```

---

PrintMatrix

*Print Matrix to file*

---

**Description**

Print a matrix or a list of matrices to file

**Usage**

```
PrintMatrix(x, ...)
```

## Default S3 method:

```
PrintMatrix(x, output.file = "./matrix.csv", ...)
```

## S3 method for class 'list'

```
PrintMatrix(x, output.file = "./matrix.csv", ...)
```

**Arguments**

x	Matrix or list of matrices
...	Additional parameters
output.file	Output file

**Value**

Prints coma separated matrices, with labels

**Author(s)**

Diogo Melo

**Examples**

```
m.list <- RandomMatrix(10, 4)
PrintMatrix(m.list)
```

---

ProjectMatrix

*Project Covariance Matrix*

---

**Description**

This function projects a given covariance matrix into the basis provided by an eigentensor decomposition.

**Usage**

```
ProjectMatrix(matrix, etd)
```

**Arguments**

matrix	A symmetric covariance matrix for k traits
etd	Eigentensor decomposition of m covariance matrices for k traits (obtained from <a href="#">EigenTensorDecomposition</a> )

**Value**

Vector of scores of given covariance matrix onto eigentensor basis.

**Author(s)**

Guilherme Garcia, Diogo Melo

## References

Basser P. J., Pajevic S. 2007. Spectral decomposition of a 4th-order covariance tensor: Applications to diffusion tensor MRI. *Signal Processing*. 87:220-236.

Hine E., Chenoweth S. F., Rundle H. D., Blows M. W. 2009. Characterizing the evolution of genetic variance using genetic covariance tensors. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*. 364:1567-78.

## See Also

[EigenTensorDecomposition](#), [RevertMatrix](#)

## Examples

```
# this function is useful for projecting posterior samples for a set of
# covariance matrices onto the eigentensor decomposition done
# on their estimated means
## Not run:
data(dentus)

dentus.models <- dlply(dentus, .(species), lm,
                      formula = cbind(humerus, ulna, femur, tibia) ~ 1)

dentus.matrices <- llply(dentus.models, BayesianCalculateMatrix, samples = 100)

dentus.post.vcv <- laply(dentus.matrices, function(L) L $ Ps)
dentus.post.vcv <- aperm(dentus.post.vcv, c(3, 4, 1, 2))

dentus.mean.vcv <- aapply(dentus.post.vcv, 3, MeanMatrix)
dentus.mean.vcv <- aperm(dentus.mean.vcv, c(2, 3, 1))

dentus.mean.etc <- EigenTensorDecomposition(dentus.mean.vcv)
dentus.mean.proj <- data.frame('species' = LETTERS[1:5], dentus.mean.etc $ projection)

dentus.post.proj <- adply(dentus.post.vcv, c(3, 4), ProjectMatrix, etc = dentus.mean.etc)
colnames(dentus.post.proj)[1:2] <- c('species', 'sample')
levels(dentus.post.proj $ species) <- LETTERS[1:5]

require(ggplot2)
ggplot() +
  geom_point(aes(x = ET1, y = ET2, color = species),
            data = dentus.mean.proj, shape = '+', size = 8) +
  geom_point(aes(x = ET1, y = ET2, color = species),
            data = dentus.post.proj, shape = '+', size = 3) +
  theme_bw()

## End(Not run)
```

---

RandCorr	<i>Random correlation matrix</i>
----------	----------------------------------

---

**Description**

Internal function for generating random correlation matrices. Use RandomMatrix() instead.

**Usage**

```
RandCorr(num.traits, ke = 10^-3)
```

**Arguments**

num.traits	Number of traits in random matrix
ke	Parameter for correlation matrix generation. Involves check for positive definiteness

**Value**

Random Matrix

**Author(s)**

Diogo Melo Edgar Zanella

---

RandomMatrix	<i>Random matrices for tests</i>
--------------	----------------------------------

---

**Description**

Provides random covariance/correlation matrices for quick tests. Should not be used for statistics or hypothesis testing.

**Usage**

```
RandomMatrix(num.traits, num.matrices = 1, min.var = 1, max.var = 1,  
variance = NULL, ke = 10^-3, LKJ = TRUE, shape = 2)
```



**Arguments**

num.traits	Number of traits in random matrix
num.matrices	Number of matrices to be generated. If greater than 1, a list is returned.
min.var	Lower value for random variance in covariance matrices
max.var	Upper value for random variance in covariance matrices
variance	Variance vector. If present will be used in all matrices
ke	Parameter for correlation matrix generation. Involves check for positive definiteness
LKJ	logical. Use LKJ distribution for generating correlation matrices.
shape	Shape parameter for the LKJ distribution. Values closer to zero leads to a more uniform distribution correlations. Higher values lead to correlations closer to zero.

**Value**

Returns either a single matrix, or a list of matrices of equal dimension

**Author(s)**

Diogo Melo Edgar Zanella

**Examples**

```
#single 10x10 correlation matrix
RandomMatrix(10)

#single 5x5 covariance matrix, variances between 3 and 4
RandomMatrix(5, 1, 3, 4)

#two 3x3 covariance matrices, with shared variances
RandomMatrix(3, 2, variance= c(3, 4, 5))

#large 10x10 matrix list, with wide range of variances
RandomMatrix(10, 100, 1, 300)
```

---

RandomSkewers

*Compare matrices via RandomSkewers*

---

**Description**

Calculates covariance matrix correlation via random skewers

**Usage**

```

RandomSkewers(cov.x, cov.y, ...)

## Default S3 method:
RandomSkewers(cov.x, cov.y, num.vectors = 1000, ...)

## S3 method for class 'list'
RandomSkewers(cov.x, cov.y = NULL, num.vectors = 1000,
  repeat.vector = NULL, parallel = FALSE, ...)

## S3 method for class 'mcmc_sample'
RandomSkewers(cov.x, cov.y, num.vectors = 1000,
  parallel = FALSE, ...)

```

**Arguments**

<code>cov.x</code>	Single covariance matrix or list of covariance matrices. If single matrix is supplied, it is compared to <code>cov.y</code> . If list is supplied and no <code>cov.y</code> is supplied, all matrices are compared. If <code>cov.y</code> is supplied, all matrices in list are compared to it.
<code>cov.y</code>	First argument is compared to <code>cov.y</code> . Optional if <code>cov.x</code> is a list.
<code>...</code>	additional arguments passed to other methods.
<code>num.vectors</code>	Number of random vectors used in comparison.
<code>repeat.vector</code>	Vector of repeatabilities for correlation correction.
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

**Value**

If `cov.x` and `cov.y` are passed, returns average value of response vectors correlation (`'correlation'`), significance (`'probability'`) and standard deviation of response vectors correlation (`'correlation_sd'`)

If `cov.x` and `cov.y` are passed, same as above, but for all matrices in `cov.x`.

If only a list is passed to `cov.x`, a matrix of RandomSkewers average values and probabilities of all comparisons. If `repeat.vector` is passed, comparison matrix is corrected above diagonal and repeatabilities returned in diagonal.

**Author(s)**

Diogo Melo, Guilherme Garcia

**References**

Cheverud, J. M., and Marroig, G. (2007). Comparing covariance matrices: Random skewers method compared to the common principal components model. *Genetics and Molecular Biology*, 30, 461-469.

**See Also**[KrzCor](#), [MantelCor](#)**Examples**

```

c1 <- RandomMatrix(10, 1, 1, 10)
c2 <- RandomMatrix(10, 1, 1, 10)
c3 <- RandomMatrix(10, 1, 1, 10)
RandomSkewers(c1, c2)

RandomSkewers(list(c1, c2, c3))

reps <- unlist(lapply(list(c1, c2, c3), MonteCarloRep, sample.size = 10,
                      RandomSkewers, num.vectors = 100,
                      iterations = 10))
RandomSkewers(list(c1, c2, c3), repeat.vector = reps)

c4 <- RandomMatrix(10)
RandomSkewers(list(c1, c2, c3), c4)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#RandomSkewers(list(c1, c2, c3), parallel = TRUE)

```

---

Rarefaction

*Rarefaction analysis via resampling*


---

**Description**

Calculates the repeatability of a statistic of the data, such as correlation or covariance matrix, via bootstrap resampling with varying sample sizes, from 2 to the size of the original data.

**Usage**

```

Rarefaction(ind.data, ComparisonFunc, ..., num.reps = 10,
           correlation = FALSE, replace = FALSE, parallel = FALSE)

```

**Arguments**

<code>ind.data</code>	Matrix of residuals or individual measurements
<code>ComparisonFunc</code>	comparison function
<code>...</code>	Additional arguments passed to <code>ComparisonFunc</code>

num.reps	number of populations sampled per sample size
correlation	If TRUE, correlation matrix is used, else covariance matrix. MantelCor always uses correlation matrix.
replace	If true, samples are taken with replacement
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

### Details

Samples of various sizes, with replacement, are taken from the full population, a statistic calculated and compared to the full population statistic.

A specialized plotting function displays the results in publication quality.

Bootstrapping may be misleading with very small sample sizes. Use with caution if original sample sizes are small.

### Value

returns the mean value of comparisons from samples to original statistic, for all sample sizes.

### Author(s)

Diogo Melo, Guilherme Garcia

### See Also

[BootstrapRep](#)

### Examples

```
ind.data <- iris[1:50,1:4]

results.RS <- Rarefaction(ind.data, PCAsimilarity, num.reps = 5)
results.Mantel <- Rarefaction(ind.data, MatrixCor, correlation = TRUE, num.reps = 5)
results.KrzCov <- Rarefaction(ind.data, KrzCor, num.reps = 5)
results.PCA <- Rarefaction(ind.data, PCAsimilarity, num.reps = 5)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##Mac and Linux:
#registerDoParallel(cores = 2)
#results.KrzCov <- Rarefaction(ind.data, KrzCor, num.reps = 5, parallel = TRUE)

#Easy access
library(reshape2)
melt(results.RS)
```

---

RarefactionStat	<i>Non-Parametric rarefacted population samples and statistic comparison</i>
-----------------	--

---

### Description

Calculates the repeatability of a statistic of the data, such as correlation or covariance matrix, via resampling with varying sample sizes, from 2 to the size of the original data.

### Usage

```
RarefactionStat(ind.data, StatFunc, ComparisonFunc, ..., num.reps = 10,  
  replace = FALSE, parallel = FALSE)
```

### Arguments

<code>ind.data</code>	Matrix of residuals or individual measurements
<code>StatFunc</code>	Function for calculating the statistic
<code>ComparisonFunc</code>	comparison function
<code>...</code>	Additional arguments passed to <code>ComparisonFunc</code>
<code>num.reps</code>	number of populations sampled per sample size
<code>replace</code>	If true, samples are taken with replacement
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .

### Details

Samples of various sizes, without replacement, are taken from the full population, a statistic calculated and compared to the full population statistic.

A specialized plotting function displays the results in publication quality.

Bootstrapping may be misleading with very small sample sizes. Use with caution.

### Value

returns the mean value of comparisons from samples to original statistic, for all sample sizes.

### Author(s)

Diogo Melo, Guilherme Garcia

### See Also

[BootstrapRep](#)

**Examples**

```
ind.data <- iris[1:50,1:4]

#Can be used to calculate any statistic via Rarefaction, not just comparisons
#Integration, for instanse:
results.R2 <- RarefactionStat(ind.data, cor, function(x, y) CalcR2(y), num.reps = 5)

#Easy access
library(reshape2)
melt(results.R2)

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##Mac and Linux:
#registerDoParallel(cores = 2)
#results.R2 <- RarefactionStat(ind.data, cor, function(x, y) CalcR2(y), parallel = TRUE)
```

---

RemoveSize

*Remove Size Variation*

---

**Description**

Removes first principal component effect in a covariance matrix.

**Usage**

```
RemoveSize(cov.matrix)
```

**Arguments**

cov.matrix      Covariance matrix

**Details**

Function sets the first eigen value to zero.

**Value**

Altered covariance matrix with no variation on former first principal component

**Author(s)**

Diogo Melo, Guilherme Garcia

**Examples**

```
cov.matrix <- RandomMatrix(10, 1, 1, 10)
no.size.cov.matrix <- RemoveSize(cov.matrix)
eigen(cov.matrix)
eigen(no.size.cov.matrix)
```

---

RevertMatrix

*Revert Matrix*


---

**Description**

Constructs a covariance matrix based on scores over covariance matrix eigentensors.

**Usage**

```
RevertMatrix(values, etd, scaled = TRUE)
```

**Arguments**

values	vector of values to build matrix, each value corresponding to a score on the ordered set of eigentensors (up to the maximum number of eigentensors on the target decomposition); if there are less values than eigentensors provided in etd (see below), the function will assume zero as the value for the score in remaining eigentensors
etd	Eigentensor decomposition of m covariance matrices for k traits (obtained from <a href="#">EigenTensorDecomposition</a> )
scaled	should we treat each score as a value given in standard deviations for each eigentensor? Defaults to TRUE

**Value**

A symmetric covariance matrix with k traits

**References**

Basser P. J., Pajevic S. 2007. Spectral decomposition of a 4th-order covariance tensor: Applications to diffusion tensor MRI. *Signal Processing*. 87:220-236.

Hine E., Chenoweth S. F., Rundle H. D., Blows M. W. 2009. Characterizing the evolution of genetic variance using genetic covariance tensors. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*. 364:1567-78.

## Examples

```
## we can use RevertMatrix to represent eigentensors using SRD to compare two matrices
## which differ with respect to their projections on a single directions

data(dentus)

dentus.vcv <- dapply (dentus, ,(species), function(x) cov(x[,-5]))

dentus.vcv <- aperm(dentus.vcv, c(2, 3, 1))

dentus.etd <- EigenTensorDecomposition(dentus.vcv, TRUE)

## calling RevertMatrix with a single value will use this value as the score
## on the first eigentensor and use zero as the value of remaining scores

low.et1 <- RevertMatrix(-1.96, dentus.etd, TRUE)
upp.et1 <- RevertMatrix(1.96, dentus.etd, TRUE)

srd.et1 <- SRD(low.et1, upp.et1)

plot(srd.et1)

## we can also look at the second eigentensor, by providing each call
## of RevertMatrix with a vector of two values, the first being zero

low.et2 <- RevertMatrix(c(0, -1.96), dentus.etd, TRUE)
upp.et2 <- RevertMatrix(c(0, 1.96), dentus.etd, TRUE)

srd.et2 <- SRD(low.et2, upp.et2)

plot(srd.et2)
```

---

RiemannDist

*Matrix Riemann distance*

---

## Description

Return distance between two covariance matrices

## Usage

```
RiemannDist(cov.x, cov.y)
```

## Arguments

cov.x	covariance or correlation matrix
cov.y	covariance or correlation matrix



**Value**

Riemann distance between cov.x and cov.y

**Author(s)**

Edgar Zanella

**References**

Mitteroecker, P., & Bookstein, F. (2009). The ontogenetic trajectory of the phenotypic covariance matrix, with examples from craniofacial shape in rats and humans. *Evolution*, 63(3), 727-737. doi:10.1111/j.1558-5646.2008.00587.x

---

RSProjection

*Random Skewers projection*

---

**Description**

Uses Bayesian posterior samples of a set of covariance matrices to identify directions of the morphospace in which these matrices differ in their amount of genetic variance.

**Usage**

```
RSProjection(cov.matrix.array, p = 0.95, num.vectors = 1000)
```

```
PlotRSprojection(rs_proj, cov.matrix.array, p = 0.95, ncols = 5)
```

**Arguments**

cov.matrix.array	Array with dimentions traits x traits x populations x MCMCsamples
p	significance treashhold for comparison of variation in each random direction
num.vectors	number of random vectors
rs_proj	output from RSProjection
ncols	number of columns in plot

**Value**

projection of all matrices in all random vectors

set of random vectors and confidence intervals for the projections

eigen decomposition of the random vectors in directions with significant differences of variations

**References**

Aguirre, J. D., E. Hine, K. McGuigan, and M. W. Blows. "Comparing G: multivariate analysis of genetic variation in multiple populations." *Heredity* 112, no. 1 (2014): 21-29.

**Examples**

```

library(magrittr)
# small MCMCsample to reduce run time, actual sample should be larger
data(dentus)
cov.matrices = dplyr::dply(dentus, ~(species), function(x) lm(as.matrix(x[,1:4])~1)) %>%
  laply(function(x) BayesianCalculateMatrix(x, samples = 50)$Ps)
cov.matrices = aperm(cov.matrices, c(3, 4, 1, 2))
rs_proj = RSProjection(cov.matrices, p = 0.8)
PlotRSprojection(rs_proj, cov.matrices, ncol = 5)

```

---

SingleComparisonMap     *Generic Single Comparison Map functions for creating parallel list methods Internal functions for making efficient comparisons.*

---

**Description**

Generic Single Comparison Map functions for creating parallel list methods Internal functions for making efficient comparisons.

**Usage**

```

SingleComparisonMap(matrix.list, y.mat, MatrixCompFunc, ...,
  parallel = FALSE)

```

**Arguments**

matrix.list	list of matrices being compared
y.mat	single matrix being compared to list
MatrixCompFunc	Function used to compare pair of matrices, must output a vector: comparisons and probabilities
...	Additional arguments to MatrixCompFunc
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

**Value**

Matrix of comparisons, matrix of probabilities.

**Author(s)**

Diogo Melo

**See Also**

[MantelCor](#), [KrzCor](#), [RandomSkewers](#)

**Description**

Based on Random Skewers technique, selection response vectors are expanded in direct and indirect component by trait and compared via vector correlations.

**Usage**

```
SRD(cov.x, cov.y, ...)

## Default S3 method:
SRD(cov.x, cov.y, iterations = 1000, ...)

## S3 method for class 'list'
SRD(cov.x, cov.y = NULL, iterations = 1000,
     parallel = FALSE, ...)

## S3 method for class 'SRD'
plot(x, matrix.label = "", ...)
```

**Arguments**

<code>cov.x</code>	Covariance matrix being compared. <code>cov.x</code> can be a matrix or a list.
<code>cov.y</code>	Covariance matrix being compared. Ignored if <code>cov.x</code> is a list.
<code>...</code>	additional parameters passed to other methods
<code>iterations</code>	Number of random vectors used in comparison
<code>parallel</code>	if TRUE computations are done in parallel. Some foreach backend must be registered, like <code>doParallel</code> or <code>doMC</code> .
<code>x</code>	Output from SRD function, used in plotting
<code>matrix.label</code>	Plot label

**Details**

Output can be plotted using `PlotSRD` function

**Value**

List of SRD scores means, confidence intervals, standard deviations, centered means e centered standard deviations

`pc1` scored along the `pc1` of the mean/SD correlation matrix

`model` List of linear model results from mean/SD correlation. Quantiles, interval and divergent traits

**Note**

If input is a list, output is a symmetric list array with pairwise comparisons.

**Author(s)**

Diogo Melo, Guilherme Garcia

**References**

Marroig, G., Melo, D., Porto, A., Sebastiao, H., and Garcia, G. (2011). Selection Response Decomposition (SRD): A New Tool for Dissecting Differences and Similarities Between Matrices. *Evolutionary Biology*, 38(2), 225-241. doi:10.1007/s11692-010-9107-2

**See Also**

[RandomSkewers](#)

**Examples**

```
cov.matrix.1 <- cov(matrix(rnorm(30*10), 30, 10))
cov.matrix.2 <- cov(matrix(rnorm(30*10), 30, 10))
colnames(cov.matrix.1) <- colnames(cov.matrix.2) <- sample(letters, 10)
rownames(cov.matrix.1) <- rownames(cov.matrix.2) <- colnames(cov.matrix.1)
srd.output <- SRD(cov.matrix.1, cov.matrix.2)

#lists
m.list <- RandomMatrix(10, 4)
srd.array.result = SRD(m.list)

#divergent traits
colnames(cov.matrix.1)[as.logical(srd.output$model$code)]

#Plot
plot(srd.output)

## For the array generated by SRD(m.list) you must index the individual positions for plotting:
plot(srd.array.result[1,2][[1]])
plot(srd.array.result[3,4][[1]])

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#SRD(m.list, parallel = TRUE)
```

---

TestModularity	<i>Test modularity hypothesis</i>
----------------	-----------------------------------

---

### Description

Tests modularity hypothesis using cor.matrix matrix and trait groupings

### Usage

```
TestModularity(cor.matrix, modularity.hypot, permutations = 1000,
  MHI = FALSE, ..., landmark.dim = NULL, withinLandmark = FALSE)
```

### Arguments

cor.matrix	Correlation matrix
modularity.hypot	Matrix of hypothesis. Each line represents a trait and each column a module. if modularity.hypot[i,j] == 1, trait i is in module j.
permutations	Number of permutations, to be passed to MantelModTest
MHI	Indicates if test should use Modularity Hypothesis Index instead of AVG Ratio
...	additional arguments passed to MantelModTest
landmark.dim	Used if permutations should be performed mantaining landmark structure in geometric morphometric data. Either 2 for 2d data or 3 for 3d data. Default is NULL for non geometric morphometric data.
withinLandmark	Logical. If TRUE within-landmark correlations are used in the calculation of matrix correlation. Only used if landmark.dim is passed, default is FALSE.

### Value

Returns mantel correlation and associated probability for each modularity hypothesis, along with AVG+, AVG-, AVG Ratio for each module. A total hypothesis combining all hypotesis is also tested.

### Author(s)

Diogo Melo, Guilherme Garcia

### References

Porto, Arthur, Felipe B. Oliveira, Leila T. Shirai, Valderes Conto, and Gabriel Marroig. 2009. "The Evolution of Modularity in the Mammalian Skull I: Morphological Integration Patterns and Magnitudes." *Evolutionary Biology* 36 (1): 118-35. doi:10.1007/s11692-008-9038-3.

### See Also

[MantelModTest](#)

## Examples

```
cor.matrix <- RandomMatrix(10)
rand.hypots <- matrix(sample(c(1, 0), 30, replace=TRUE), 10, 3)
mod.test <- TestModularity(cor.matrix, rand.hypots)

cov.matrix <- RandomMatrix(10, 1, 1, 10)
cov.mod.test <- TestModularity(cov.matrix, rand.hypots, MHI = TRUE)
nosize.cov.mod.test <- TestModularity(RemoveSize(cov.matrix), rand.hypots, MHI = TRUE)
```

---

TreeDriftTest	<i>Drift test along phylogeny</i>
---------------	-----------------------------------

---

## Description

Performs a regression drift test along a phylogeny using DriftTest function.

## Usage

```
TreeDriftTest(tree, mean.list, cov.matrix.list, sample.sizes = NULL)
```

## Arguments

tree	phylogenetic tree
mean.list	list of tip node means. Names must match tip node labels.
cov.matrix.list	list of tip node covariance matrices. Names must match tip node labels.
sample.sizes	vector of tip nodes sample sizes

## Value

A list of regression drift tests performed in nodes with over 4 descendant tips.

## Author(s)

Diogo Melo

## See Also

DriftTest PlotTreeDriftTest

**Examples**

```
library(ape)
data(bird.orders)

tree <- bird.orders
mean.list <- llply(tree$tip.label, function(x) rnorm(5))
names(mean.list) <- tree$tip.label
cov.matrix.list <- RandomMatrix(5, length(tree$tip.label))
names(cov.matrix.list) <- tree$tip.label
sample.sizes <- runif(length(tree$tip.label), 15, 20)

test.list <- TreeDriftTest(tree, mean.list, cov.matrix.list, sample.sizes)

#Ancestral node plot:
test.list[[length(test.list)]]$plot
```

# Index

- \*Topic **Autonomy**
  - MeanMatrixStatistics, 36
- \*Topic **ConditionalEvolvability**
  - MeanMatrixStatistics, 36
- \*Topic **Constraints**
  - MeanMatrixStatistics, 36
- \*Topic **Evolvability**
  - MeanMatrixStatistics, 36
- \*Topic **Flexibility**
  - MeanMatrixStatistics, 36
- \*Topic **Krzanowski**
  - KrzCor, 23
  - KrzProjection, 25
  - PCAsimilarity, 46
- \*Topic **PCA**
  - PCAsimilarity, 46
- \*Topic **Pc1Percent**
  - MeanMatrixStatistics, 36
- \*Topic **RandomSkewers**
  - SRD, 67
- \*Topic **Responsability**
  - MeanMatrixStatistics, 36
- \*Topic **SRD**
  - SRD, 67
- \*Topic **bootstap**
  - PlotRarefaction, 51
  - Rarefaction, 59
  - RarefactionStat, 61
- \*Topic **bootstrap**
  - BootstrapR2, 5
  - BootstrapRep, 6
- \*Topic **correlation**
  - CalcEigenSd, 9
  - CalcR2, 11
  - CalcR2CvCorrected, 12
- \*Topic **covariancematrix**
  - BayesianCalculateMatrix, 4
  - CalculateMatrix, 14
  - ExtendMatrix, 22
- \*Topic **covariance**
  - CalcICV, 10
- \*Topic **datasets**
  - dentus, 18
  - dentus.tree, 18
- \*Topic **extension**
  - ExtendMatrix, 22
- \*Topic **integration**
  - BootstrapR2, 5
  - CalcEigenSd, 9
  - CalcICV, 10
  - CalcR2, 11
  - CalcR2CvCorrected, 12
- \*Topic **manteltest**
  - MantelModTest, 31
- \*Topic **mantel**
  - TestModularity, 69
- \*Topic **matrixDistance**
  - MatrixDistance, 33
  - RiemannDist, 64
- \*Topic **matrixcomparison**
  - DeltaZCorr, 16
  - KrzCor, 23
  - KrzProjection, 25
  - MantelCor, 29
  - MantelModTest, 31
  - MatrixDistance, 33
  - PCAsimilarity, 46
  - RandomSkewers, 57
  - RiemannDist, 64
- \*Topic **matrixcorrelation**
  - DeltaZCorr, 16
  - KrzCor, 23
  - KrzProjection, 25
  - MantelCor, 29
  - MantelModTest, 31
  - PCAsimilarity, 46
  - RandomSkewers, 57
- \*Topic **modularity**



- TestModularity, 69
- \*Topic **montecarlo**
  - BootstrapStat, 7
  - MonteCarloR2, 38
  - MonteCarloRep, 39
  - MonteCarloStat, 41
- \*Topic **parametricssampling**
  - BootstrapStat, 7
  - MonteCarloR2, 38
  - MonteCarloRep, 39
  - MonteCarloStat, 41
- \*Topic **randommatrices**
  - RandCorr, 56
  - RandomMatrix, 56
- \*Topic **randomskewers**
  - DeltaZCorr, 16
  - MantelCor, 29
  - RandomSkewers, 57
- \*Topic **rarefaction**
  - PlotRarefaction, 51
  - Rarefaction, 59
  - RarefactionStat, 61
- \*Topic **repeatability**
  - BootstrapR2, 5
  - BootstrapStat, 7
  - MonteCarloR2, 38
  - MonteCarloRep, 39
  - MonteCarloStat, 41
  - PlotRarefaction, 51
  - Rarefaction, 59
  - RarefactionStat, 61
- \*Topic **repeatatability**
  - AlphaRep, 3
- \*Topic **repetabilities**
  - BootstrapRep, 6
- \*Topic **selectionresponsedecomposition**
  - SRD, 67
- \*Topic **size**
  - RemoveSize, 62
- AlphaRep, 3, 5, 6, 8, 39–41
- Autonomy (MeanMatrixStatistics), 36
- BayesianCalculateMatrix, 4
- BootstrapR2, 5
- BootstrapRep, 3, 5, 6, 8, 39–41, 52, 60, 61
- BootstrapStat, 7
- CalcAVG, 8, 32
- CalcEigenSd, 9
- CalcICV, 10, 10
- CalcR2, 10, 11, 11, 13
- CalcR2CvCorrected, 12
- CalcRepeatability, 13
- CalculateMatrix, 14
- ComparisonMap, 15
- ConditionalEvolvability
  - (MeanMatrixStatistics), 36
- Constraints (MeanMatrixStatistics), 36
- CreateHypotMatrix, 16
- DeltaZCorr, 16
- dentus, 18
- dentus.tree, 18
- DriftTest, 19
- EigenTensorDecomposition, 20, 35, 54, 55, 63
- evolqg, 22
- evolqg-package (evolqg), 22
- Evolvability (MeanMatrixStatistics), 36
- ExtendMatrix, 22
- Flexibility, 10, 12
- Flexibility (MeanMatrixStatistics), 36
- JackKnifeMINT (MINT), 37
- KrzCor, 15, 17, 23, 30, 47, 59, 66
- KrzProjection, 24, 25, 47
- KrzSubspace, 26
- LModularity, 27
- mahalanobis, 43
- mantel, 30, 32
- MantelCor, 15, 17, 24, 26, 29, 32, 47, 59, 66
- MantelModTest, 30, 31, 69
- MatrixCompare, 33
- MatrixCor (MantelCor), 29
- MatrixDistance, 33
- MeanMatrix, 21, 35
- MeanMatrixStatistics, 13, 36
- MINT, 37
- MonteCarloR2, 38
- MonteCarloRep, 39
- MonteCarloStat, 3, 6, 41
- MultiMahalanobis, 42

MultivDriftTest, 43

Norm (Normalize), 45  
Normalize, 45

OverlapDist, 34, 45

Pc1Percent (MeanMatrixStatistics), 36  
PCAsimilarity, 46  
PCScoreCorrelation, 47  
PhyloCompare, 48  
PhyloMantel, 49  
PhyloW, 51  
plot.SRD (SRD), 67  
PlotRarefaction, 51  
PlotRSProjection (RSProjection), 65  
PlotTreeDriftTest, 52  
PrintMatrix, 53  
ProjectMatrix, 21, 54

RandCorr, 56  
RandomMatrix, 56  
RandomSkewers, 15, 24, 26, 30, 47, 57, 66, 68  
Rarefaction, 59  
RarefactionStat, 61  
RemoveSize, 62  
Responsability (MeanMatrixStatistics),  
36  
RevertMatrix, 21, 55, 63  
RiemannDist, 34, 35, 64  
RSProjection, 65

SingleComparisonMap, 66  
SRD, 67

TestModularity, 30, 32, 69  
TreeDriftTest, 70