Package: evolqg (via r-universe)

October 24, 2026

Type Package

Title Evolutionary Quantitative Genetics

Version 0.4-3

Date 2026-09-24

Description Provides functions for covariance matrix comparisons, estimation of repeatabilities in measurements and matrices, and general evolutionary quantitative genetics tools. Melo D, Garcia G, Hubbe A, Assis A P, Marroig G. (2016) <doi:10.12688/f1000research.7082.3>.

Depends R (>= 4.1.0), plyr (>= 1.7.1)

Imports Matrix, reshape2, ggplot2, vegan, ape, expm, numDeriv, Morpho, mvtnorm, coda, igraph, MCMCpack, graphics, grDevices, methods, stats, utils

Suggests dplyr, testthat, foreach, grid, gridExtra, doParallel, cowplot

LinkingTo Rcpp, RcppArmadillo

License MIT + file LICENSE

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

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Diogo Melo [aut, cre] (ORCID: <https://orcid.org/0000-0002-7603-0092>), Ana Paula Assis [aut] (ORCID: <https://orcid.org/0000-0001-8170-6352>), Edgar Zanella [ctb], Fabio Andrade Machado [aut] (ORCID: <https://orcid.org/0000-0002-0215-9926>), Guilherme Garcia [aut], Alex Hubbe [rev] (ORCID: <https://orcid.org/0000-0002-3226-0144>), Gabriel Marroig [ths] (ORCID: <https://orcid.org/0000-0003-2542-2034>)

Maintainer Diogo Melo <diogro@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-09-24 15:20:09 UTC

RemoteUrl https://github.com/cran/evolqg

RemoteRef HEAD

RemoteSha 47daf5b89a59e9a7d76cd211145b75ab85aa363a

Contents

AlphaRep . . . 3
BayesianCalculateMatrix . . . 4
BootstrapR2 . . . 5
BootstrapRep . . . 6
BootstrapStat . . . 8
CalcAVG . . . 9
CalcEigenVar . . . 10
CalcICV . . . 12
CalcR2 . . . 13
CalcR2CvCorrected . . . 14
CalcRepeatability . . . 16
CalculateMatrix . . . 17
Center2MeanJacobianFast . . . 18
ComparisonMap . . . 18
CreateHypotMatrix . . . 19
DeltaZCorr . . . 20
dentus . . . 21
dentus.tree . . . 22
DriftTest . . . 22
EigenTensorDecomposition . . . 23
evolqg . . . 25
ExtendMatrix . . . 26
JacobianArray . . . 27
KrzCor . . . 27
KrzProjection . . . 29
KrzSubspace . . . 31
KrzSubspaceBootstrap . . . 32
KrzSubspaceDataFrame . . . 33
LModularity . . . 34
LocalShapeVariables . . . 35
MantelCor . . . 36
MantelModTest . . . 39
MatrixCompare . . . 41
MatrixDistance . . . 41
MeanMatrix . . . 43
MeanMatrixStatistics . . . 44
MINT . . . 45
MonteCarloR2 . . . 47
MonteCarloRep . . . 48
MonteCarloStat . . . 49

MultiMahalanobis . . . 51
MultivDriftTest . . . 52
Normalize . . . 54
OverlapDist . . . 55
Partition2HypotMatrix . . . 55
PCAsimilarity . . . 56
PCScoreCorrelation . . . 57
PhyloCompare . . . 58
PhyloMantel . . . 59
PhyloW . . . 61
PlotKrzSubspace . . . 61
PlotRarefaction . . . 62
PlotTreeDriftTest . . . 63
PrintMatrix . . . 64
ProjectMatrix . . . 65
RandCorr . . . 66
RandomMatrix . . . 67
RandomSkewers . . . 68
Rarefaction . . . 70
RarefactionStat . . . 72
ratones . . . 73
RelativeEigenanalysis . . . 74
RemoveSize . . . 75
RevertMatrix . . . 76
RiemannDist . . . 77
Rotate2MidlineMatrix . . . 78
RSProjection . . . 78
SingleComparisonMap . . . 79
SRD . . . 80
TestModularity . . . 82
TPS . . . 83
TreeDriftTest . . . 84

Index **85**

`AlphaRep` *Alpha repeatability*

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

`cor.matrix`	Correlation matrix
`sample.size`	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
...	additional 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 variances and zero covariance

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 suplied data or residuals. R2 is calculated on all the random population's correlation matrices, provinding a distribution based on the original data.

Usage

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

Arguments

ind.data	Matrix of residuals or indiviual measurments
iterations	Number of resamples to take
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

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

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

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:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##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 ressampling using the suplied 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

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

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:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##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 morphomotric 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)
```

CalcEigenVar *Integration measure based on eigenvalue dispersion*

Description

Calculates integration indexes based on eigenvalue dispersion of covariance or correlation matrices.

Usage

```
CalcEigenVar(
  matrix,
  sd = FALSE,
  rel = TRUE,
  sample = NULL,
  keep.positive = TRUE
)
```

Arguments

matrix	Covariance/correlation matrix
sd	Logical. Default is FALSE. If TRUE, estimates eigenvalue standard deviation. If FALSE, estimate the eigenvalue variance.
rel	Logical. If TRUE, scales eigenvalue dispersion value by the theoretical maximum.
sample	Default is NULL. If a integer is provided, function calculates the expected integration value for that particular sample size and returns value as a deviation from the expected.
keep.positive	Logical. If TRUE, non-positive eigenvalues are removed from calculation

Details

This function quantifies morphological integration as the dispersion of eigenvalues in a matrix. It takes either a covariance or a correlation matrix as input, and there is no need to discern between them.The output will depend on the combination of parameters specified during input.

As default, the function calculates the relative eigenvalue variance of the matrix, which expresses the eigenvalue variance as a ratio between the actual variance and the theoretical maximum for a matrix of the same size and same amount of variance (same trace), following Machado et al. (2019). If sd=TRUE, the dispersion is measured with the standard deviation of eigenvalues instead of the variance (Pavlicev, 2009). If the sample size is provided, the function automatically calculates the expected integration value for a matrix of the same size but with no integration (e.g. a matrix with all eigenvalues equal). In that case, the result is given as a deviation from the expected and is invariant to sample size (Wagner, 1984).

Value

Integration index based on eigenvalue dispersion.

Author(s)

Fabio Andrade Machado

Diogo Melo

References

Machado, Fabio A., Alex Hubbe, Diogo Melo, Arthur Porto, and Gabriel Marroig. 2019. "Measuring the magnitude of morphological integration: The effect of differences in morphometric representations and the inclusion of size." Evolution 33:402–411.

Pavlicev, Mihaela, James M. Cheverud, and Gunter P. Wagner. 2009. "Measuring Morphological Integration Using Eigenvalue Variance." Evolutionary Biology 36(1):157-170.

Wagner, Gunther P. 1984. "On the eigenvalue distribution of genetic and phenotypic dispersion matrices: evidence for a nonrandom organization of quantitative character variation." Journal of Mathematical Biology 21(1):77–95.

See Also

CalcR2, CalcICV

Examples

```
cov.matrix <- RandomMatrix(10, 1, 1, 10)
# calculates the relative eigenvalue variance of a covariance matrix
CalcEigenVar(cov.matrix)

# calculates the relative eigenvalue variance of a correlation matrix
CalcEigenVar(cov2cor(cov.matrix))

# calculates the relative eigenvalue standard deviation of a covariance
# matrix
CalcEigenVar(cov.matrix, sd=TRUE)
```

```
# calculates the absolute eigenvalue variance of a covariance matrix
CalcEigenVar(cov.matrix, rel=FALSE)

# to evaluate the effect of sampling error on integration
x<-mvtnorm::rmvnorm(10, sigma=cov.matrix)
sample_cov.matrix<-var(x)

# to contrast values of integration obtained from population covariance
# matrix
CalcEigenVar(cov.matrix)
# with the sample integration
CalcEigenVar(sample_cov.matrix)
# and with the integration measured corrected for sampling error
CalcEigenVar(sample_cov.matrix,sample=10)
```

`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.

Details

Warning: CalcEigenVar is strongly preferred and should probably be used in place of this function.

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 *Mean Squared Correlations*

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.

Details

Warning: CalcEigenVar is strongly preferred and should probably be used in place of this function.

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 *Corrected integration value*

Description

Calculates the Young correction for integration, using bootstrap resampling Warning: CalcEigenVar is strongly preferred and should probably be used in place of this function..

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

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

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 Hallgrimsson, 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

```
## Not run:
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()

## End(Not run)
```

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

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

`Center2MeanJacobianFast` *Centered jacobian residuals*

Description

Calculates mean jacobian matrix for a set of jacobian matrices describing a local aspect of shape deformation for a given set of volumes, returning log determinants of deviations from mean jacobian (Woods, 2003).

Usage

```
Center2MeanJacobianFast(jacobArray)
```

Arguments

`jacobArray`	Arrays of Jacobian calculated in the JacobianArray function

Value

array of centered residual jacobians

Author(s)

Guilherme Garcia

Diogo Melo

References

Woods, Roger P. 2003. “Characterizing Volume and Surface Deformations in an Atlas Framework: Theory, Applications, and Implementation.” NeuroImage 18 (3):769-88.

`ComparisonMap` *Generic Comparison Map functions for creating parallel list methods Internal functions for making eficient comparisons.*

Description

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

Usage

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

Arguments

matrix.list	list of matrices being compared
MatrixCompFunc	Function used to compare pair of matrices, must output a vector: comparisons and probabilities
...	Aditional arguments to MatrixCompFunc
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

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

`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. 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.
`skewers`	matrix of column vectors to be used as gradients
`...`	additional arguments passed to other methods.
`parallel`	if TRUE computations are done in parallel. Some foreach back-end must be registered, like doParallel or doMC.

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,,RandomSkewers

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 *Test drift hypothesis*

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	Logical. 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

```
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
oldpar <- 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)
```

```
par(oldpar)


# 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 example
dentus.post.eval <- laply (dentus.post.etd, function (L) L $ values)

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

`evolqg` *EvolQG*

Description

The package for evolutionary quantitative genetics.

Author(s)

Maintainer: Diogo Melo <diogro@gmail.com> (ORCID)

Authors:

- Diogo Melo <diogro@gmail.com> (ORCID)
- Ana Paula Assis (ORCID)
- Fabio Andrade Machado (ORCID)
- Guilherme Garcia

Other contributors:

- Edgar Zanella [contributor]
- Alex Hubbe (ORCID) [reviewer]
- Gabriel Marroig (ORCID) [thesis advisor]

See Also

Useful links:

- Report bugs at https://github.com/lem-usp/evolqg/issues

`ExtendMatrix` *Control Inverse matrix noise with Extension*

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 eigenvalues

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)
```

JacobianArray *Local Jacobian calculation*

Description

Calculates jacobians for a given interpolation in a set of points determined from tesselation (as centroids of each tetrahedron defined, for now...)

Usage

```
JacobianArray(spline, tesselation, ...)
```

Arguments

spline	Thin plate spline calculated by the TPS function
tesselation	matrix of landmarks.
...	Additional arguments to some function

Value

array of jacobians calculated at the centroids

Note

Jacobians are calculated on the row centroids of the tesselation matrix.

Author(s)

Guilherme Garcia

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 suplied, 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 back-end must be registered, like doParallel or doMC.

Value

If cov.x and cov.y are passed, returns Krzanowski 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 Krzanowski 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

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,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)


## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
KrzCor(list(c1, c2, c3), parallel = TRUE)

## End(Not run)
```

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

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

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
KrzProjection(m.list, parallel = TRUE)

## End(Not run)
```

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 dimensions 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 Bayesian comparison 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)

## Not run:
# The method in Aguirre et al. 2014 can de implemented using this function as follows:

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

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))

## End(Not run)
```

`KrzSubspaceBootstrap` *Quasi-Bayesian Krzanowski subspace comparison*

Description

Calculates the usual Krzanowski subspace comparison using a posterior samples for a set of phenotypic covariance matrices. Then, this observed comparison is contrasted to the subspace comparison across a permutation of the original data. Residuals, which are used to calculate the observed P-matrices, are shuffled across groups. This process is repeated, creating a null distribution of subspace comparisons under the hypothesis that all P-matrices come from the same population. This method is a modification on the fully Bayesian method proposed in Aguirre et. al 2013 and improved in Morrisey et al 2019.

Usage

```
KrzSubspaceBootstrap(
  x,
  rep = 1,
  MCMCsamples = 1000,
  parallel = FALSE,
  k = NULL
)
```

Arguments

x	list of linear models from which P-matrices should be calculated
rep	number of bootstrap samples to be made
MCMCsamples	number of MCMCsamples for each P-matrix posterior distribution.
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.
k	number of dimensions to be retained in calculating the subspace.

Value

A list with the observed and randomized eigenvalue distributions for the posterior Krz Subspace comparisons.

References

Aguirre, J. D., E. Hine, K. McGuigan, and M. W. Blows. 2013. "Comparing G: multivariate analysis of genetic variation in multiple populations." Heredity 112 (February): 21–29.

Morrissey, Michael B., Sandra Hangartner, and Keyne Monro. 2019. "A Note on Simulating Null Distributions for G Matrix Comparisons." Evolution; International Journal of Organic Evolution 73 (12): 2512–17.

See Also

KrzSubspaceDataFrame, PlotKrzSubspace

Examples

```
library(plyr)
data(ratones)

model_formula = paste("cbind(", paste(names(ratones)[13:20], collapse = ", "), ") ~ SEX")
lm_models = dlply(ratones, .(LIN), function(df) lm(as.formula(model_formula), data = df))
krz_comparsion = KrzSubspaceBootstrap(lm_models, rep = 100, MCMCsamples = 1000)
krz_df = KrzSubspaceDataFrame(krz_comparsion)
PlotKrzSubspace(krz_df)
```

KrzSubspaceDataFrame *Extract confidence intervals from KrzSubspaceBootstrap*

Description

Returns posterior means and confidence intervals from the object produced by the KrzSubspaceBootstrap() function. Mainly used for ploting using PlotKrzSubspace. See example in the KrzSubspaceBootstrap function.

Usage

```
KrzSubspaceDataFrame(x, n = ncol(observed), prob = 0.95)
```

Arguments

x	output from KrzSubspaceBootstrap function.
n	number of eigenvalues to use
prob	Posterior probability interval. Default is 95%.

Value

Posterior intervals for the eigenvalues of the H matrix in the KrzSubspace comparison.

See Also

KrzSubspaceBootstrap, PlotKrzSubspace

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 detection 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

Details

Warning: Using modularity maximization is almost always a terrible idea. See: https://skewed.de/tiago/blog/modularity-harmful

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)
```

LocalShapeVariables *Local Shape Variables*

Description

Calculates the local shape variables of a set of landmarks using the sequence: - TPS transform between all shapes and the mean shape - Jacobian of the TPS transforms at the centroid of rows of the landmarks in the tesselation argument - Mean center the Jacobians using the Karcher Mean - Take the determinant of the centered jacobians

Usage

```
LocalShapeVariables(
  gpa = NULL,
  cs = NULL,
  landmarks = NULL,
  tesselation,
  run_parallel = FALSE
)
```

Arguments

gpa	Procustes aligned landmarks.
cs	Centoid sizes
landmarks	unaligned landmarks. Ignored if both gpa and cs are passed.
tesselation	matrix of rows of the landmarks. The centroid of each row is used to mark the position of the jacobians
run_parallel	Logical. If computation should be paralleled. Use with caution, can make things worse. Requires that at parallel back-end like doMC be registered

Value

List with TPS functions, jacobian matrices, local shape variables, mean shape, centroid sizes and individual IDs

Author(s)

Guilherme Garcia

Diogo Melo

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 morphomotric 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.
mod	Set TRUE to use mantel in testing modularity hypothesis. Should only be used in MantelModTest.
repeat.vector	Vector of repeatabilities for correlation correction.
parallel	if TRUE computations are done in parallel. Some foreach back-end must be registered, like doParallel or doMC.

Value

If cor.x and cor.y are passed, returns matrix Pearson correlation coefficient 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

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)

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
MantelCor(cor.list, parallel = TRUE)

## End(Not run)
```

`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

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

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.
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 back-end 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(cor.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

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. 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.
distance	distance function for use in calculation. Currently supports "Riemann" and "Overlap".
...	additional arguments passed to other methods
parallel	if TRUE and a list is passed, computations are done in parallel. Some foreach back-end must be registered, like doParallel or doMC.

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")

# Compare multiple matrices
MatrixDistance(list(c1, c2, c3), distance = "OverlapDist")

# Compare multiple matrices to a target matrix
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 x k x 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 = FALSE,
  parallel = FALSE,
  beta.mat = NULL,
  include.R2 = FALSE,
  include.ICV = FALSE
)

Autonomy(cov.matrix, beta.mat = NULL, iterations = 1000)

ConditionalEvolvability(cov.matrix, beta.mat = NULL, iterations = 1000)

Constraints(cov.matrix, beta.mat = NULL, iterations = 1000)

Evolvability(cov.matrix, beta.mat = NULL, iterations = 1000)

Flexibility(cov.matrix, beta.mat = NULL, iterations = 1000)

Pc1Percent(cov.matrix)

Respondability(cov.matrix, beta.mat = NULL, iterations = 1000)
```

Arguments

cov.matrix	A covariance matrix
iterations	Number of random vectors to be used in calculating the stochastic statistics when beta.mat is NULL.
full.results	If TRUE, full distribution of statistics will be returned.
parallel	if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.
beta.mat	Matrix of direction vectors used to calculate the stochastic statistics. If NULL, a random matrix of normalized vectors is generated.
include.R2	Logical. If TRUE, include the legacy mean squared correlation metric in the returned output.
include.ICV	Logical. If TRUE, include the legacy ICV metric in the returned output.

Value

dist Full distribution of stochastic statistics, only if full.resuts == 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)

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
MeanMatrixStatistics(cov.matrix, parallel = TRUE)

## End(Not run)
```

MINT *Modularity and integration analysis tool*

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 jack-knife 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 os goodness of fit simulations
ind.data	Matrix of residuals or individual measurements
n	number of jackknife samples
leave.out	number of individuals to be left out of each jackknife, default is 10%
...	additional arguments to be passed to raply for the jackknife

Value

Dataframe with ranked hypothesis, ordered by the corrected gamma value Jackknife will return the best hypothesis for each sample.

Note

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

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

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 = mvtnorm::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 suplied covariance matrix. R2 is calculated on all the random population, provinding 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
`iterations`	Number of random populations
`parallel`	if TRUE 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.

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 suplied 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

cov.matrix	Covariance matrix.
sample.size	Size of the random populations.
ComparisonFunc	comparison function.
...	Aditional arguments passed to ComparisonFunc.
iterations	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)


## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
MonteCarloRep(cov.matrix, 30, RandomSkewers, iterations = 100, parallel = TRUE)

## End(Not run)
```

MonteCarloStat *Parametric population samples 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

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

Arguments

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

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)

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
##Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)

##Mac and Linux:
library(doParallel)
registerDoParallel(cores = 2)

MonteCarloStat(cov.matrix, sample.size = 30, iterations = 100,
               ComparisonFunc = function(x, y) KrzCor(x, y)[1],
               StatFunc = cov,
               parallel = TRUE)

## End(Not run)
```

`MultiMahalanobis` *Calculate Mahalonabis 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

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)

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
MultiMahalanobis(mean.list, RandomMatrix(10), parallel = TRUE)

## End(Not run)
```

`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`	data.frame with original measurements for the ancestral population
`population2`	data.frame 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` *Normalize and Norm*

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 inpout vector norm.

Author(s)

Diogo Melo, Guilherme Garcia

Examples

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

OverlapDist *Distribution overlap distance*

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

Partition2HypotMatrix *Create binary hypothesis*

Description

Takes a vetor describing a trait partition and returns a binary matrix of the partitions where each line represents a trait and each column a module. In the output matrix, if modularity.hypot[i,j] == 1, trait i is in module j.

Usage

```
Partition2HypotMatrix(x)
```

Arguments

x	vector of trait partition. Each partition receive the same symbol.

Value

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.

Examples

```
x = sample(c(1, 2, 3), 10, replace = TRUE)
Partition2HypotMatrix(x)
```

`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 or 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 back-end 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	Logical. If TRUE, plot of eigenvalues of ancestral 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)

## Not run:
##Plots list can be displayed using plot_grid()
library(cowplot)
pc.score.output <- PCScoreCorrelation(means, cov.matrix, taxons, TRUE)
plot_grid(plotlist = pc.score.output$plots)

## End(Not run)
```

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 = dlply(dentus, .(species), function(x) cor(x[-5]))
comparisons = MatrixCor(cor.matrices)

sp.means = dlply(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 *Calculates ancestral states of some statistic*

Description

Calculates weighted average of covariances matrices along a phylogeny, returning a withing-group covariance matrice 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 <- dlply(dentus, 'species', function(x) cov(x[,1:4]))
sample.sizes <- runif(length(tree$tip.label), 15, 20)
PhyloW(tree, mat.list, sample.sizes)
```

PlotKrzSubspace *Plot KrzSubspace boostrap comparison*

Description

Shows the null and observed distribution of eigenvalues from the Krzanowski subspace comparison

Usage

```
PlotKrzSubspace(x)
```

Arguments

x	output from KrzSubspaceDataFrame() function.

Value

ggplot2 object with the observed vs. random eigenvalues mean and posterior confidence intervals

---

`PlotRarefaction` *Plot Rarefaction analysis*

Description

A specialized ploting 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 ploting
`y.axis`	Y axis lable in plot
`x.axis`	Y axis lable in plot

Value

ggplot2 object with rarefaction 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(cowplot)
plot_grid(a, b, c, d, labels = c("RS",
                                 "Mantel Correlation",
                                 "Krzanowski Correlation",
                                 "PCA Similarity"),
                      scale = 0.9)
```

`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

Value

No return value, called for plot side effects

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, ...)

## S3 method for class 'list'
PrintMatrix(x, output.file, ...)
```

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)
tmp = file.path(tempdir(), "matrix.csv")
PrintMatrix(m.list, output.file = tmp )
```

`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 `EigenTensorDecomposition`)

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

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

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

dentus.post.proj <- adply(dentus.post.vcv, c(3, 4), ProjectMatrix, etd = dentus.mean.etd)
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()
```

RandCorr *Random correlation matrix*

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 defitness

Value

Random Matrix

Author(s)

Diogo Melo Edgar Zanella

`RandomMatrix` *Random matrices for tests*

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 = FALSE,
  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 = 10000, ...)

## S3 method for class 'list'
RandomSkewers(
  cov.x,
  cov.y = NULL,
  num.vectors = 10000,
  repeat.vector = NULL,
  parallel = FALSE,
  ...
)

## S3 method for class 'mcmc_sample'
RandomSkewers(cov.x, cov.y, num.vectors = 10000, 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. 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.
num.vectors	Number of random vectors used in comparison.
repeat.vector	Vector of repeatabilities for correlation correction.
parallel	if TRUE computations are done in parallel. Some foreach back-end must be registered, like doParallel or doMC.

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,DeltaZCorr

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)

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
RandomSkewers(list(c1, c2, c3), parallel = TRUE)

## End(Not run)
```

`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

`ind.data`	Matrix of residuals or individual measurments
`ComparisonFunc`	comparison function
`...`	Additional arguments passed to ComparisonFunc
`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 back-end 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.

Bootstraping 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, RandomSkewers, num.reps = 5)
#' #Easy parsing of results
library(reshape2)
melt(results.RS)

# or :

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)


## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
results.KrzCov <- Rarefaction(ind.data, KrzCor, num.reps = 5, parallel = TRUE)

## End(Not run)
```

`RarefactionStat` *Non-Parametric rarefacted population samples and statistic comparison*

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

`ind.data`	Matrix of residuals or indiviual measurments
`StatFunc`	Function for calculating the statistic
`ComparisonFunc`	comparison function
`...`	Aditional arguments passed to ComparisonFunc
`num.reps`	number of populations sampled per sample size
`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, without replacement, are taken from the full population, a statistic calculated and compared to the full population statistic.

A specialized ploting function displays the results in publication quality.

Bootstraping 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 example:
results.R2 <- RarefactionStat(ind.data, cor, function(x, y) CalcR2(y), num.reps = 5)

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

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
registerDoMC(cores = 2)
results.R2 <- RarefactionStat(ind.data, cor, function(x, y) CalcR2(y), parallel = TRUE)

## End(Not run)
```

---

ratones *Linear distances for five mouse lines*

Description

Skull distances measured from landmarks in 5 mice lines: 4 body weight selection lines and 1 control line. Originally published in Penna, A., Melo, D. et. al (2017) doi:10.1111/evo.13304

Usage

```
data(ratones)
```

Format

data.frame

Source

Dryad Archive doi:10.5061/dryad.5gr8r

References

Penna, A., Melo, D., Bernardi, S., Oyarzabal, M.I. and Marroig, G. (2017), The evolution of phenotypic integration: How directional selection reshapes covariation in mice. Evolution, 71: 2370-2380. doi:10.1111/evo.13304

Examples

```
data(ratones)

# Estimating a W matrix, controlling for line and sex
model_formula = paste0("cbind(",
                       paste(names(ratones)[13:47], collapse = ", "),
                       ") ~ SEX + LIN")
ratones_W_model = lm(model_formula, data = ratones)
W_matrix = CalculateMatrix(ratones_W_model)

# Estimating the divergence between the two direction of selection
delta_Z = colMeans(ratones[ratones$selection == "upwards", 13:47]) -
          colMeans(ratones[ratones$selection == "downwards", 13:47])

 # Reconstructing selection gradients with and without noise control
Beta = solve(W_matrix, delta_Z)
Beta_non_noise = solve(ExtendMatrix(W_matrix, ret.dim = 10)$ExtMat, delta_Z)

# Comparing the selection gradients to the observed divergence
Beta %*% delta_Z /(Norm(Beta) * Norm(delta_Z))
Beta_non_noise %*% delta_Z /(Norm(Beta_non_noise) * Norm(delta_Z))
```

`RelativeEigenanalysis` *Relative Eigenanalysis*

Description

Computes relative eigenvalues and eigenvectors between a pair of covariance matrices.

Usage

```
RelativeEigenanalysis(cov.x, cov.y, symmetric = FALSE)
```

Arguments

`cov.x`	covariance matrix
`cov.y`	covariance matrix
`symmetric`	Logical. If TRUE, computes symmetric eigenanalysis.

Value

list with two objects: eigenvalues and eigenvectors

Author(s)

Guilherme Garcia, Diogo Melo

References

Bookstein, F. L., and P. Mitteroecker, P. "Comparing Covariance Matrices by Relative Eigenanalysis, with Applications to Organismal Biology." Evolutionary Biology 41, no. 2 (June 1, 2014): 336-350. doi:10.1007/s11692-013-9260-5.

Examples

```
data(dentus)
dentus.vcv <- dlply(dentus, .(species), function(df) var(df[, -5]))

dentus.eigrel <- RelativeEigenanalysis(dentus.vcv [[1]], dentus.vcv[[5]])
```

`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 eigenvalue 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 EigenTensorDecomposition)
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 <- daply (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

Rotate2MidlineMatrix *Midline rotate*

Description

Returns the rotation matrix that aligns a specimen sagital line to plane y = 0 (2D) or z = 0 (3D)

Usage

```
Rotate2MidlineMatrix(X, midline)
```

Arguments

X	shape array
midline	rows for the midline landmarks

Value

Rotation matrix

Author(s)

Guilherme Garcia

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 dimensions traits x traits x populations x MCMCsamples
p	significance threshold 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

```
# small MCMCsample to reduce run time, acctual sample should be larger
data(dentus)
cov.matrices = dlply(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 back-end must be registered, like doParallel or doMC.

Value

Matrix of comparisons, matrix of probabilities.

Author(s)

Diogo Melo

See Also

MantelCor, KrzCor,RandomSkewers

SRD *Compare matrices via Selection Response Decomposition*

Description

Based on Random Skewers technique, selection response vectors are expanded in direct and indirect components 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

cov.x	Covariance matrix being compared. cov.x can be a matrix or a list.
cov.y	Covariance matrix being compared. Ignored if cov.x is a list.
...	additional parameters passed to other methods
iterations	Number of random vectors used in comparison
parallel	if TRUE computations are done in parallel. Some foreach back-end must be registered, like doParallel or doMC.
x	Output from SRD function, used in plotting
matrix.label	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 idividual positions for plotting:
plot(srd.array.result[1,2][[1]])
plot(srd.array.result[3,4][[1]])

## Not run:
#Multiple threads can be used with some foreach backend library, like doMC or doParallel
library(doMC)
```

```
registerDoMC(cores = 2)
SRD(m.list, parallel = TRUE)

## End(Not run)
```

`TestModularity` *Test modularity hypothesis*

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 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.
`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 hypothesis 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)
```

TPS *TPS transform*

Description

Calculates the Thin Plate Spline transform between a reference shape and a target shape

Usage

```
TPS(target.shape, reference.shape)
```

Arguments

target.shape	Target shape
reference.shape	Reference shape

Value

A list with the transformation parameters and a function that gives the value of the TPS function at each point for numerical differentiation

Author(s)

Guilherme Garcia

`TreeDriftTest` *Drift test along phylogeny*

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

∗ **Autonomy**
MeanMatrixStatistics, 44
∗ **ConditionalEvolvability**
MeanMatrixStatistics, 44
∗ **Constraints**
MeanMatrixStatistics, 44
∗ **Evolvability**
MeanMatrixStatistics, 44
∗ **Flexibility**
MeanMatrixStatistics, 44
∗ **Krzanowski**
KrzCor, 27
KrzProjection, 29
PCAsimilarity, 56
∗ **PCA**
PCAsimilarity, 56
∗ **Pc1Percent**
MeanMatrixStatistics, 44
∗ **RandomSkewers**
SRD, 80
∗ **Respondability**
MeanMatrixStatistics, 44
∗ **SRD**
SRD, 80
∗ **bootstap**
PlotRarefaction, 62
Rarefaction, 70
RarefactionStat, 72
∗ **bootstrap**
BootstrapR2, 5
BootstrapRep, 6
∗ **correlation**
CalcEigenVar, 10
CalcR2, 13
CalcR2CvCorrected, 14
∗ **covariancematrix**
BayesianCalculateMatrix, 4
CalculateMatrix, 17
ExtendMatrix, 26
∗ **covariance**
CalcICV, 12
∗ **datasets**
dentus, 21
dentus.tree, 22
ratones, 73
∗ **eigenvalues**
CalcEigenVar, 10
∗ **extension**
ExtendMatrix, 26
∗ **integration**
BootstrapR2, 5
CalcEigenVar, 10
CalcICV, 12
CalcR2, 13
CalcR2CvCorrected, 14
∗ **manteltest**
MantelModTest, 39
∗ **mantel**
TestModularity, 82
∗ **matrixDistance**
MatrixDistance, 41
RiemannDist, 77
∗ **matrixcomparison**
DeltaZCorr, 20
KrzCor, 27
KrzProjection, 29
MantelCor, 36
MantelModTest, 39
MatrixDistance, 41
PCAsimilarity, 56
RandomSkewers, 68
RiemannDist, 77
∗ **matrixcorrelation**
DeltaZCorr, 20
KrzCor, 27
KrzProjection, 29
MantelCor, 36
MantelModTest, 39

PCAsimilarity, 56
RandomSkewers, 68
∗ **modularity**
TestModularity, 82
∗ **montecarlo**
BootstrapStat, 8
MonteCarloR2, 47
MonteCarloRep, 48
MonteCarloStat, 49
∗ **parametricsampling**
BootstrapStat, 8
MonteCarloR2, 47
MonteCarloRep, 48
MonteCarloStat, 49
∗ **randommatrices**
RandCorr, 66
RandomMatrix, 67
∗ **randomskewers**
DeltaZCorr, 20
MantelCor, 36
RandomSkewers, 68
∗ **rarefaction**
PlotRarefaction, 62
Rarefaction, 70
RarefactionStat, 72
∗ **repeatability**
AlphaRep, 3
BootstrapR2, 5
BootstrapStat, 8
MonteCarloR2, 47
MonteCarloRep, 48
MonteCarloStat, 49
PlotRarefaction, 62
Rarefaction, 70
RarefactionStat, 72
∗ **repetabilities**
BootstrapRep, 6
∗ **selectionresponsedecomposition**
SRD, 80
∗ **size**
RemoveSize, 75

AlphaRep, 3, *6–8*, *48–50*
Autonomy *(*MeanMatrixStatistics*)*, 44

BayesianCalculateMatrix, 4
BootstrapR2, 5
BootstrapRep, *4*, 6, *6*, *8*, *48–50*, *62*, *71*, *73*
BootstrapStat, 8

CalcAVG, 9, *40*
CalcEigenVar, 10
CalcICV, *11*, 12
CalcR2, *11*, 13, *13*, *15*
CalcR2CvCorrected, 14
CalcRepeatability, 16
CalculateMatrix, 17
Center2MeanJacobianFast, 18
ComparisonMap, 18
ConditionalEvolvability *(*MeanMatrixStatistics*)*, 44
Constraints *(*MeanMatrixStatistics*)*, 44
CreateHypotMatrix, 19

DeltaZCorr, 20, *69*
dentus, 21
dentus.tree, 22
DriftTest, 22

EigenTensorDecomposition, 23, *43*, *65*, *76*
evolqg, 25
evolqg-package *(*evolqg*)*, 25
Evolvability *(*MeanMatrixStatistics*)*, 44
ExtendMatrix, 26

Flexibility, *14*
Flexibility *(*MeanMatrixStatistics*)*, 44

JackKnifeMINT *(*MINT*)*, 45
JacobianArray, 27

KrzCor, *19*, *21*, 27, *38*, *57*, *69*, *80*
KrzProjection, 29, *29*, *57*
KrzSubspace, 31
KrzSubspaceBootstrap, 32, *34*
KrzSubspaceDataFrame, 33, *33*

LModularity, 34
LocalShapeVariables, 35

mahalanobis, *52*
mantel, *38*, *40*
MantelCor, *19*, *21*, *29*, *30*, 36, *40*, *57*, *69*, *80*
MantelModTest, *38*, 39, *83*
MatrixCompare, 41
MatrixCor *(*MantelCor*)*, 36
MatrixDistance, 41
MeanMatrix, *24*, 43
MeanMatrixStatistics, *15*, 44
MINT, 45

MonteCarloR2, 47
MonteCarloRep, 48
MonteCarloStat, *4*, *7*, 49
MultiMahalanobis, 51
MultivDriftTest, 52

Norm *(*Normalize*)*, 54
Normalize, 54

OverlapDist, *42*, 55

Partition2HypotMatrix, 55
Pc1Percent *(*MeanMatrixStatistics*)*, 44
PCAsimilarity, 56
PCScoreCorrelation, 57
PhyloCompare, 58
PhyloMantel, 59
PhyloW, 61
plot.SRD *(*SRD*)*, 80
PlotKrzSubspace, *33, 34*, 61
PlotRarefaction, 62
PlotRSprojection *(*RSProjection*)*, 78
PlotTreeDriftTest, 63
PrintMatrix, 64
ProjectMatrix, *24*, 65

RandCorr, 66
RandomMatrix, 67
RandomSkewers, *19*, *21*, *29, 30*, *38*, *57*, 68, *80, 81*
Rarefaction, 70
RarefactionStat, 72
ratones, 73
RelativeEigenanalysis, 74
RemoveSize, 75
Respondability *(*MeanMatrixStatistics*)*, 44
RevertMatrix, *24*, *65*, 76
RiemannDist, *42, 43*, 77
Rotate2MidlineMatrix, 78
RSProjection, 78

SingleComparisonMap, 79
SRD, 80

TestModularity, *38*, *40*, 82
TPS, 83
TreeDriftTest, 84