Package ‘evolqg’

February 6, 2020

Type Package
Title Tools for Evolutionary Quantitative Genetics
Version 0.2-7
Date 2020-02-05
Author Ana Paula Assis, Diogo Melo, Edgar Zanella, Fabio Andrade Machado, Guilherme Garcia
Maintainer Diogo Melo <diogro@usp.br>
Description Provides functions for covariance matrix comparisons, estimation of repeatabilities in measurements and matrices, and general evolutionary quantitative genetics tools.
Depends R (>= 3.6.0), plyr (>= 1.7.1)
Imports Rcpp (>= 0.11), Matrix, reshape2, ggplot2, vegan, ape, expm, matrixcalc, mvtnorm, coda, igraph, MCMCpack, graphics, grDevices, methods, stats, utils
Suggests testthat, foreach, magrittr, grid, gridExtra, doParallel, PHENIX, cowplot
LinkingTo Rcpp, RcppArmadillo
License MIT + file LICENSE
BugReports https://github.com/lem-usp/evolqg/issues
Encoding UTF-8
RoxygenNote 6.1.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-02-06 14:00:02 UTC

R topics documented:

  AlphaRep ................................................................. 3
  BayesianCalculateMatrix ............................................. 4
  BootstrapR2 .............................................................. 5
### R topics documented:

- BootstrapRep ........................................... 6
- BootstrapStat .......................................... 7
- CalcAVG .................................................. 8
- CalcEigenVar ........................................... 9
- CalcICV .................................................. 11
- CalcR2 ................................................... 12
- CalcR2CvCorrected .................................... 13
- CalcRepeatability ...................................... 14
- CalculateMatrix ........................................ 15
- ComparisonMap ......................................... 16
- CreateHypotMatrix ..................................... 17
- DeltaZCorr .............................................. 17
- dentus ................................................... 19
- dentus.tree .............................................. 19
- DriftTest ................................................ 20
- EigenTensorDecomposition ............................ 21
- evolqg ................................................... 23
- ExtendMatrix .......................................... 23
- KrzCor ................................................... 24
- KrzProjection ......................................... 26
- KrzSubspace ............................................ 27
- LModularity ............................................ 29
- MantelCor .............................................. 30
- MantelModTest ......................................... 32
- MatrixCompare ......................................... 34
- MatrixDistance ........................................ 35
- MeanMatrix ............................................. 36
- MeanMatrixStatistics .................................. 37
- MINT ..................................................... 38
- MonteCarloR2 .......................................... 39
- MonteCarloRep ......................................... 40
- MonteCarloStat ........................................ 42
- MultiMahalanobis ...................................... 43
- MultivDriftTest ........................................ 45
- Normalize ............................................... 46
- OverlapDist ............................................ 47
- Partition2HypotMatrix ................................ 47
- PCAsimilarity .......................................... 48
- PCScoreCorrelation .................................... 49
- PhyloCompare .......................................... 50
- PhyloMantel ............................................. 51
- PhyloW ................................................... 53
- PlotRarefaction ........................................ 53
- PlotTreeDriftTest ...................................... 54
- PrintMatrix ............................................. 55
- ProjectMatrix .......................................... 56
- RandCorr ............................................... 58
- RandomMatrix .......................................... 58
**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**

```r
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

```r
# 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

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

Arguments

- `linear.m`: Linear model adjusted for original data
- `samples`: number of 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 covariances

Value

Estimated covariance matrices and posterior samples

Author(s)

Diogo Melo, Fabio Machado
**Bootstrapped R² (BootstrapR2)**

R² confidence intervals by bootstrap resampling

**Description**

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

**Usage**

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

**Arguments**

- `ind.data`: Matrix of residuals or individual measurements
- `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**

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

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
**BootstrapStat**

**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 re-sampling using the supplied population. A statistic is calculated on the random population and compared to the statistic calculated on the original population.

**Usage**

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

**Arguments**

- **ind.data**: Matrix of residuals or individual measurements
- **iterations**: Number of resamples to take
- **ComparisonFunc**: Comparison function
- **StatFunc**: Function for calculating the statistic
- **sample.size**: Size of resamples, 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.
**CalcEigenVar**

**MHI**
Indicates if Modularity Hypothesis Index should be calculated instead of AVG Ratio.

**landmark.dim**
Used if within-landmark correlations are to be excluded in geometric morphometric data. Either 2 for 2d data or 3 for 3d data. Default is NULL for non geometric morphometric data.

**Value**
a named vector with the mean correlations and derived statistics

**Examples**

```r
# 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)
```

**Arguments**

- **matrix**: Covariance/correlation matrix
- **sd**: Logical. Default is FALSE. If TRUE, estimates eigenvalue standard deviation. If FALSE, estimate the eigenvalue variance.
calc.eigen.var

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.

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

References

See Also
CalcR2, CalcICV, pint

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.

Value

coefficient of variation of the eigenvalues of a covariance matrix

Author(s)

Diogo Melo
References


See Also

CalcR2

Examples

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

Description

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

Usage

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

Arguments

- `c.matrix`: Covariance or correlation matrix.

Value

Mean squared value of off diagonal elements of correlation matrix

Author(s)

Diogo Melo, Guilherme Garcia

References


CalcR2CvCorrected

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

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 indiviual measurments, or adjusted linear model
...
  additional arguments passed to other methods
cv.level  Coefficient of variation level choosen for integration index ajustment 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 coeficient of variation.
Author(s)

Diogo Melo, Guilherme Garcia

References


See Also

MeanMatrixStatistics, CalcR2

Examples

```r
## Not run:
integration.dist = CalcR2CvCorrected(iris[,1:4])

# adjusted values
integration.dist[[1]]

# plotting 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
**CalculateMatrix**

**Value**

vector of repeatabilities

**Note**

Requires at least two observations per individual

**Author(s)**

Guilherme Garcia

**References**


**Examples**

```r
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**

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

**Arguments**

- `linear.m` Linear model adjusted for original data.

**Value**

Estimated covariance matrix.

**Author(s)**

Diogo Melo, Fabio Machado

**References**

https://github.com/lem-usp/evolqg/wiki/
Examples

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

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

Description

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

Usage

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

Arguments

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

`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**

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

---

DeltaZCorr

`DeltaZCorr` compares 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**

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

**Examples**

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


See Also

- KrzCor
- MantelCor

Examples

```r
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**: boolean. If TRUE, plot of eigenvalues of ancestral matrix by between group variance is showed.

Value

- list of results containing:
  - regression: the linear regression between the log of the eigenvalues of the ancestral matrix and the log of the between group variance (projected on the eigenvectors of the ancestral matrix)
  - coefficient_CI_95: confidence intervals for the regression coefficients
  - log.between_group_variance: log of the between group variance (projected on the ancestral matrix eigenvectors)
  - log.W_eVals: log of the ancestral matrix eigenvalues
  - plot: plot of the regression using ggplot2

Note

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

Author(s)

Ana Paula Assis, Diogo Melo

References


**Examples**

```r
#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)
```

**Description**

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

**Usage**

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

### S3 method for class 'list'

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

### Default S3 method:

```r
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

See Also
ProjectMatrix, RevertMatrix

Examples
```r
## Not run:
data(dentus)
dentus.vcv <- daply (dentus, .(species), function(x) cov(x[-5]))
dentus.vcv <- aperm(dentus.vcv, c(2, 3, 1))
dentus.etd <- EigenTensorDecomposition(dentus.vcv, TRUE)

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

# we can also deal with posterior samples of covariance matrices using plyr
dentus.models <- dlply(dentus, .(species),
    lm, formula = cbind(humerus, ulna, femur, tibia) ~ 1)
dentus.matrices <- llply(dentus.models, BayesianCalculateMatrix, samples = 100)
dentus.post.vcv <- laply(dentus.matrices, function (L) L$Ps)
dentus.post.vcv <- aperm(dentus.post.vcv, c(3, 4, 1, 2))

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

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

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

## End(Not run)

---

**evolqg**

**EvolQG**

---

**Description**

EvolQG

---

**ExtendMatrix**

*Control Inverse matrix noise with Extension*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

Extended covariance matrix and second derivative variance

**Note**

Covariance matrix being extended should be larger then 10x10

**Author(s)**

Diogo Melo
References


Examples

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

KrzCor

Compare matrices via Krzanowski Correlation

Description

Calculates covariance matrix correlation via Krzanowski Correlation

Usage

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

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

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

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

Arguments

cov.x

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

cov.y

First argument is compared to cov.y. Optional if cov.x is a list.

...

Additional arguments passed to other methods

ret.dim

number of retained dimensions in the comparison, default for nxn matrix is n/2-1

repeat.vector

Vector of repeatabilities for correlation correction.

parallel

if TRUE and a list is passed, computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.
Value

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


See Also

RandomSkewers.KrzProjection,MantelCor

Examples

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

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

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

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

Compare matrices via Modified Krzanowski Correlation

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

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

## Default S3 method:
KrzProjection(cov.x, cov.y, ret.dim.1 = NULL, ret.dim.2 = NULL, ...)

## S3 method for class 'list'
KrzProjection(cov.x, cov.y = NULL, ret.dim.1 = NULL, ret.dim.2 = NULL, parallel = FALSE, full.results = FALSE, ...)

Arguments
cov.x Single covariance matrix ou list of covariance matrices. If cov.x is a single matrix is suplied, it is compared to cov.y. If cov.x is a list of matrices is suplied and no cov.y is suplied, all matrices are compared between each other. If cov.x is a list of matrices and a single cov.y matrix is suplied, 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.
... aditional 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 backend 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
KrzSubspace

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

References


See Also

RandomSkewers, MantelCor

Examples

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

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

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

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

H shared space matrix

$k_eV_{als \_H}$ eigen values for shared space matrix, maximum value for each is the number of matrices, representing a fully shared direction

$k_eV_{ecs \_H}$ 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**


**Examples**

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

```r
# The method in Aguirre et al. 2014 can be implemented using this function as follows:

# Random input data with dimensions traits x traits x populations x MCMC samples:
cov.matrices = aperm(aapply(1:10, 1, function(x) laply(RandomMatrix(6, 40,
  variance = runif(6,1, 10)),
  identity)),
c(3, 4, 1, 2))
library(magrittr)
Hs = alply(cov.matrices, 4, function(x) alply(x, 3)) %>% llply(function(x) KrzSubspace(x, 3)$H)
avgH = Reduce("+", Hs)/length(Hs)
avgH.vec <- eigen(avgH)$vectors
MCMC.H.val = laply(Hs, function(mat) diag(t(avgH.vec) %*% mat %*% avgH.vec))

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

```r
# End(Not run)
```
Description
Calculates the L-Modularity (Newman-type modularity) and the partition of traits that minimizes L-Modularity. Wrapper for using correlations matrices in community detections algorithms from igraph.

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

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

Value
List with L-Modularity value and trait partition

Note
Community detection is done by transforming the correlation matrix into a weighted graph and using community detections 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

Examples
## Not run:
# A modular matrix:
modules = matrix(c(rep(c(1, 0, 0), each = 5),
rep(c(0, 1, 0), each = 5),
rep(c(0, 0, 1), each = 5)), 15)
cor.hypot = CreateHypotMatrix(modules)[4]
hypot.mask = matrix(as.logical(cor.hypot), 15, 15)
mod.cor = matrix(NA, 15, 15)
mod.cor[ hypot.mask] = runif(length(mod.cor[ hypot.mask]), 0.8, 0.9) # within-modules
mod.cor[[hypot.mask]] = runif(length(mod.cor[[hypot.mask]], 0.3, 0.4) # between-modules
diag(mod.cor) = 1
mod.cor = (mod.cor + t(mod.cor))/2 # correlation matrices should be symmetric
# requires a custom igraph installation with GLPK installed in the system
LModularity(mod.cor)
## End(Not run)

MantelCor

## Compare matrices via Mantel Correlation

### Description

Calculates correlation matrix correlation and significance via Mantel test.

### Usage

```r
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)
```

### 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.
MantelCor

ADITIONAL ARGUMENTS PASSED TO OTHER METHODS

permutations Number of permutations used in significance calculation.

landmark.dim Used if permutations should be performed maintaining landmark structure in geometric morphometric data. Either 2 for 2d data or 3 for 3d data. Default is NULL for non geometric morphometric data.

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 backend must be registered, like doParallel or doMC.

Value

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

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

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

Note

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

Author(s)

Diogo Melo, Guilherme Garcia

References

http://en.wikipedia.org/wiki/Mantel_test

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)
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.
...  aditional arguments passed to MantelCor
MantelModTest

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 morphomotric data. Either 2 for 2d data or 3 for 3d data. Default is NULL for non geometric morphometric data.

withinLandmark Logical. If TRUE within-landmark correlation are used in calculation of correlation. Only used if landmark.dim is passed, default is FALSE.

parallel if TRUE computations are done in parallel. Some foreach backend must be registered, like doParallel or doMC.

Details

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

Value

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

Author(s)

Diogo Melo, Guilherme Garcia

References


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

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

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 suplied, it is compared to cov.y. If list is suplied and no cov.y is suplied, all matrices are compared. If cov.y is suplied, 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 argument is used 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 backend 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")

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

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

MeanMatrix

Mean Covariance Matrix

Description

Estimate geometric mean for a set of covariance matrices

Usage

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

Arguments

matrix.array      k 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

See Also

EigenTensorDecomposition, RiemannDist
MeanMatrixStatistics  

Calculate mean values for various matrix statistics

Description

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

Usage

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

Arguments

cov.matrix  A covariance matrix
iterations  Number of random vectors to be used in calculating the stochastic statistics
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.

Value

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

Author(s)

Diogo Melo Guilherme Garcia

References


Examples

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

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
#Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)
#Mac and Linux:
#registerDoParallel(cores = 2)
#MeanMatrixStatistics(cov.matrix, parallel = TRUE)
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 of goodness of fit simulations
ind.data Matrix of residuals or indiviual measurements
n number of jackknife resamplings

leave.out number of individuals to be left out of each jackknife, default is 10%

... aditional arguments to be passed to raply for the jackknife

Note

Hypothesis can be named as column names, and these will be used to make labels in the output. Jackknife will return the best hypothesis for each sample.

References


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 supplied covariance matrix. R2 is calculated on all the random population, providing a distribution based on the original matrix.

Usage

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

Arguments

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

Description

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

Usage

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

Arguments

cov.matrix Covariance matrix.
sample.size Size of the random populations.
ComparisonFunc comparison function.
... Additional arguments passed to ComparisonFunc.
iterations Number of random populations.
correlation
If TRUE, correlation matrix is used, else covariance matrix. MantelCor and MatrixCor should always use 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)

## Not run:
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)
lapply(mat.list, MonteCarloRep, 30, KrzCor, correlation = TRUE)

##Multiple threads can be used with doMC library

##Windows:
#cl <- makeCluster(2)
#registerDoParallel(cl)
##Mac and Linux:
library(doParallel)
registerDoParallel(cores = 2)
MonteCarloRep(cov.matrix, 30, RandomSkewers, iterations = 100, parallel = TRUE)
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 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 the mean repeatability, or mean value of comparisons from samples to original statistic.

Author(s)

Diogo Melo, Guilherme Garcia

See Also

BootstrapRep, AlphaRep
MultiMahalanobis

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 Mahalanobis distance for many vectors*

**Description**

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

**Usage**

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

**Arguments**

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


Value

returns a matrix of species-species distances.

Author(s)

Diogo Melo

References

http://en.wikipedia.org/wiki/Mahalanobis_distance

See Also

mahalanobis

Examples

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

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

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

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

#Multiple threads can be used with some foreach backend library, like doMC or doParallel
#library(doParallel)
##Windows:
#c1 <- makeCluster(2)
#registerDoParallel(c1)
##Mac and Linux:
#registerDoParallel(cores = 2)
#MultiMahalanobis(mean.list, RandomMatrix(10), parallel = TRUE)
```
MultivDriftTest

Multivariate genetic drift test for 2 populations

Description

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

Usage

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

Arguments

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

Value

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

Note

Each trait is estimated independently.

Author(s)

Ana Paula Assis

References

Normalize

Examples

```r
## Not run:
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)

## End(Not run)
```

---

Normalize and Norm

Description

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

Usage

```r
Normalize(x)

Norm(x)
```

Arguments

- `x` Numeric vector

Value

Normalized vector or input vector norm.

Author(s)

Diogo Melo, Guilherme Garcia

Examples

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

*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 draws

*Value*

Overlap distance between `cov.x` and `cov.y`

*References*


---

**Partition2HypotMatrix**  
*Create binary hypothesis*

*Description*

Takes a vector 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

```r
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

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

## Default S3 method:

```r
PCAsimilarity(cov.x, cov.y, ret.dim = NULL, ...)
```

## S3 method for class 'list'

```r
PCAsimilarity(cov.x, cov.y = NULL, ..., repeat.vector = NULL, parallel = FALSE)
```

## S3 method for class 'mcmc_sample'

```r
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 backend must be registered, like `doParallel` or `doMC`.

Value

Ratio of projected variance to total variance
**Author(s)**

Edgar Zanella Alvarenga

**References**


**See Also**

KrzProjection, KrzCor, RandomSkewers, MantelCor

**Examples**

```r
   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**

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

**Arguments**

- **means**: list or array of species means being compared. array must have means in the rows.
- **cov.matrix**: ancestral covariance matrix for all populations
- **taxons**: names of taxons being compared. Must be in the same order of the means.
- **show.plots**: boolean. If TRUE, plot of eigenvalues of 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 put 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


Examples

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

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

PhyloCompare

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
PhyloMantel

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

```r
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


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)$Var)
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 within-group covariance matrix for each node.

Usage

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

Arguments

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

Value

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

Examples

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

PlotRarefaction

Plot Rarefaction analysis

Description

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

Usage

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

Plot results from TreeDriftTest

Description

Plot which labels reject drift hypothesis.

Usage

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

Arguments

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

Author(s)

Diogo Melo, Guilherme Garcia

See Also

BootstrapRep

Examples

```r
## Not run:
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)

## End(Not run)
```
PrintMatrix

Arguments

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

Author(s)

Diogo Melo

See Also

DriftTest TreeDriftTest

Examples

library(ape)
data(bird.orders)

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

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

PrintMatrix

Print Matrix to file

Description

Print a matrix or a list of matrices to file

Usage

PrintMatrix(x, ...)

## Default S3 method:
PrintMatrix(x, output.file = "./matrix.csv", ...)

## S3 method for class 'list'
PrintMatrix(x, output.file = "./matrix.csv", ...)
Arguments

- `x`  Matrix or list of matrices
- `...` Aditional parameters
- `output.file` Output file

Value

Prints coma separated matrices, with labels

Author(s)

Diogo Melo

Examples

```r
## Not run:
m.list <- RandomMatrix(10, 4)
PrintMatrix(m.list)
## End(Not run)
```

---

### 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
ProjectMatrix

References


See Also

EigenTensorDecomposition, RevertMatrix

Examples

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

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

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

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

dentus.mean.vcv <- aaply(dentus.post.vcv, 3, MeanMatrix)

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()

## End(Not run)
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 = TRUE, shape = 2)
RandomSkewers

Arguments

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

Value

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

Author(s)

Diogo Melo Edgar Zanella

Examples

- #single 10x10 correlation matrix
  RandomMatrix(10)
- #single 5x5 covariance matrix, variances between 3 and 4
  RandomMatrix(5, 1, 3, 4)
- #two 3x3 covariance matrices, with shared variances
  RandomMatrix(3, 2, variance= c(3, 4, 5))
- #large 10x10 matrix list, with wide range of variances
  RandomMatrix(10, 100, 1, 300)

RandomSkewers

Compare matrices via RandomSkewers

Description

Calculates covariance matrix correlation via random skewers
Usage

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

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

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

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

Arguments

cov.x
  Single covariance matrix or list of covariance matrices. If single matrix is supp-
  lied, it is compared to cov.y. If list is supplied and no cov.y is supplied, all mar-
  trices 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 backend 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

method compared to the common principal components model. Genetics and Molecular Biology,
30, 461-469.

See Also

KrzCor,MantelCor
Examples

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

reps <- unlist(lapply(list(c1, c2, c3), MonteCarloRep, sample.size = 10,
                      RandomSkewers, num.vectors = 100,
                      iterations = 10))
RandomSkewers(list(c1, c2, c3), repeat.vector = reps)
c4 <- RandomMatrix(10)
RandomSkewers(list(c1, c2, c3), c4)

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

---

**Rarefaction**

**Rarefaction analysis via resampling**

**Description**

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

**Usage**

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

**Arguments**

- `ind.data`: Matrix of residuals or individual measurements
- `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 backend must be registered, like doParallel or doMC.

Details

Samples of various sizes, with replacement, are taken from the full population, a statistic calculated and compared to the full population statistic.
A specialized plotting function displays the results in publication quality.
Bootstrapping may be misleading with very small sample sizes. Use with caution if original sample sizes are small.

Value

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

Author(s)

Diogo Melo, Guilherme Garcia

See Also

BootstrapRep

Examples

```r
## Not run:
ind.data <- iris[1:50,1:4]
results.RS <- Rarefaction(ind.data, PCAsimilarity, num.reps = 5)
results.Mantel <- Rarefaction(ind.data, MatrixCor, correlation = TRUE, num.reps = 5)
results.KrzCov <- Rarefaction(ind.data, KrzCor, num.reps = 5)
results.PCA <- Rarefaction(ind.data, PCAsimilarity, num.reps = 5)

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

#Easy access
library(reshape2)
melt(results.RS)
## 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 individual measurements
StatFunc Function for calculating the statistic
ComparisonFunc comparison function
... Additional 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 plotting 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

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

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

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

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

---

**RelativeEigenanalysis Relative Eigenanalysis**

**Description**

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

**Usage**

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

**Arguments**

- `cov.x`: covariance matrix
- `cov.y`: covariance matrix
- `symmetric`: compute symmetric eigenanalysis?

**Value**

list with two objects: eigenvalues and eigenvectors

**Author(s)**

Guilherme Garcia, Diogo Melo
References


Examples

data(dentus)
dentus.vcv <- dlply(dentus, .(species), function(df) var(df[, -5]))
dentus.eigrel <- RelativeEigenanalysis(dentus.vcv [[1]], dentus.vcv[[5]])

RemoveSize

Description

Removes first principal component effect in a covariance matrix.

Usage

RemoveSize(cov.matrix)

Arguments

cov.matrix Covariance matrix

Details

Function sets the first eigen value to zero.

Value

Altered covariance matrix with no variation on former first principal component

Author(s)

Diogo Melo, Guilherme Garcia

Examples

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

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


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


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 MCMC samples
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

Examples

```r
library(magrittr)
# small MCMCsample to reduce run time, actual 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))
## Not run:
rs_proj = RSProjection(cov.matrices, p = 0.8)
PlotRSprojection(rs_proj, cov.matrices, ncol = 5)
## End(Not run)
```

Description

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

Usage

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

Arguments

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

Value

Matrix of comparisons, matrix of probabilities.

Author(s)

Diogo Melo

See Also

- `MantelCor`,
- `KrzCor`,
- `RandomSkewers`
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.
... aditional parameters passed to other methods
iterations Number of random vectors used in comparison
parallel if TRUE computations are done in parallel. Some foreach backend 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
SRD

Note

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

Author(s)

Diogo Melo, Guilherme Garcia

References


See Also

RandomSkewers

Examples

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

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

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

#Plot
plot(srd.output)

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

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

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
...: aditional arguments passed to MantelModTest
landmark.dim: Used if permutations should be performed mantaining landmark structure in 
geometric morphomotric 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.

Value

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

Author(s)

Diogo Melo, Guilherme Garcia

References

"The Evolution of Modularity in the Mammalian Skull I: Morphological Integration Patterns and 

See Also

MantelModTest
**TreeDriftTest**

**Drift test along phylogeny**

**Description**

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

**Usage**

```r
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

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

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

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

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

*Topic **Autonomy**
  MeanMatrixStatistics, 37
*Topic **ConditionalEvolvability**
  MeanMatrixStatistics, 37
*Topic **Constraints**
  MeanMatrixStatistics, 37
*Topic **Evolvability**
  MeanMatrixStatistics, 37
*Topic **Flexibility**
  MeanMatrixStatistics, 37
*Topic **Krzanowski**
  KrzCor, 24
  KrzProjection, 26
  PCAsimilarity, 48
*Topic **PCA**
  PCAsimilarity, 48
*Topic **Pc1Percent**
  MeanMatrixStatistics, 37
*Topic **RandomSkewers**
  SRD, 70
*Topic **Responsability**
  MeanMatrixStatistics, 37
*Topic **SRD**
  SRD, 70
*Topic **bootstrap**
  PlotRarefaction, 53
  Rarefaction, 61
  RarefactionStat, 63
*Topic **bootstrap**
  BootstrapR2, 5
  BootstrapRep, 6
*Topic **correlation**
  CalcEigenVar, 9
  CalcR2, 12
  CalcR2CvCorrected, 13
*Topic **covariance**
  CalcICV, 11
*Topic **datasets**
  dentus, 19
dentus.tree, 19
*Topic **eigenvalues**
  CalcEigenVar, 9
*Topic **extension**
  ExtendMatrix, 23
*Topic **integration**
  BootstrapR2, 5
  CalcEigenVar, 9
  CalcICV, 11
  CalcR2, 12
  CalcR2CvCorrected, 13
*Topic **manteltest**
  MantelModTest, 32
*Topic **mantel**
  TestModularity, 72
*Topic **matrixDistance**
  MatrixDistance, 35
  RiemannDist, 67
*Topic **matrixcomparison**
  DeltaZCorr, 17
  KrzCor, 24
  KrzProjection, 26
  MantelCor, 30
  MantelModTest, 32
  MatrixDistance, 35
  PCAsimilarity, 48
  RandomSkewers, 59
  RiemannDist, 67
*Topic **matrixcorrelation**
  DeltaZCorr, 17
  KrzCor, 24
  KrzProjection, 26
  MantelCor, 30
  MantelModTest, 32
  PCAsimilarity, 48
INDEX

MonteCarloStat, 3, 6, 42
MultiMahalanobis, 43
MultivDriftTest, 45

Norm (Normalize), 46
Normalize, 46

OverlapDist, 35, 47
Partition2HypotMatrix, 47
Pc1Percent (MeanMatrixStatistics), 37
PCAsimilarity, 48
PCScoreCorrelation, 49
PhyloCompare, 50
PhyloMantel, 51
PhyloW, 53
pint, 10
plot.SRD (SRD), 70
PlotRarefaction, 53
PlotRSprojection (RSProjection), 68
PlotTreeDriftTest, 54
PrintMatrix, 55
ProjectMatrix, 21, 22, 56

RandCorr, 58
RandomMatrix, 58
RandomSkewers, 16, 25, 27, 31, 49, 59, 69, 71
Rarefaction, 61
RarefactionStat, 63
RelativeEigenanalysis, 64
RemoveSize, 65
Respondability (MeanMatrixStatistics), 37
RevertMatrix, 21, 22, 57, 66
RiemannDist, 35, 36, 67
RSProjection, 68

SingleComparisonMap, 69
SRD, 70

TestModularity, 31, 33, 72
TreeDriftTest, 73