Package ‘symmoments’

February 5, 2020

Type Package

Title Symbolic Central and Noncentral Moments of the Multivariate Normal Distribution

Version 1.2.1

Date 2020-01-05

Author Kem Phillips

Depends mvtnorm, cubature, combinat, multipol

Suggests ape, mpoly

Maintainer Kem Phillips <kemphillips@comcast.net>

Description Symbolic central and non-central moments of the multivariate normal distribution. Computes a standard representation, \LaTeX{} code, and values at specified mean and covariance matrices.

License GPL

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-05 16:30:02 UTC

R topics documented:

  symmoments-package ........................................... 2
  callmultmoments ............................................. 5
  convert.mpoly ................................................ 6
  convert.multipol ............................................. 7
  evaluate ....................................................... 8
  evaluate_expected.polynomial ............................... 9
  evaluate_noncentral ........................................ 11
  integrate.polynomial ......................................... 12
  make.all.moments ............................................ 13
  multmoments ................................................ 15
  print.moment ................................................ 16
  simulate.moment ............................................. 17
symmoments-package

Symbolically compute and numerically evaluate multivariate central moments

Description

Symbolically computes and numerically evaluates multivariate normal moments \( E[X_1^{k_1} \ldots X_n^{k_n}] \), where \((X_1, \ldots, X_n) \sim N(\mu, S)\), in terms of \(\mu\) and \(S\) elements.

Produces LaTeX code for the moment.

Computes numerical moments at specified means and covariance matrices.

Also converts between moment L-matrices, phylo objects, and matching objects.

Details

Package: symmoments
Type: Package
Version: 1.2
Date: 2018-05-30
License: GPL 2
LazyLoad: yes

A representation of a central moment of the multivariate normal distribution, given by a positive integer vector \(c(k_1,k_2,\ldots,k_n)\), is obtained from the function callmultmoments. This function initializes variables and calls the function multmoments which determines a representation of a multivariate moment using a recursive algorithm. The representation is given class 'moment'.

The print method prints the representation of a multivariate moment.

The toLatex method uses the output of callmultmoments to determine the LaTeX code for the moment sorted lexicographically.

The generic evaluate method uses the output of callmultmoments to determine the value of the moment for a specified covariance matrix.

The simulate method is used to approximate a (possibly non-central) moment using Monte Carlo integration.

The following functions compute non-central moments and do related computations:

The toLatex_noncentral function computes the LaTeX representations of a non-central moment.
The `evaluate_noncentral` function computes the value of a non-central moment.

The `evaluate_expected.polynomial` function evaluates the expected value of a multivariate polynomial defined by a list, multipol object, or mpoly object.

The `convert.multipol` function converts between multipol objects and multivariate polynomials defined by lists.

The `convert.mpoly` function converts between mpoly objects and multivariate polynomials defined by lists.

The `tounsorted` function converts a sorted moment (e.g. m123) to an unsorted moment (e.g. m312).

The `make.all.moments` function computes all moments up to a specified size and places them in the symmoments environment.

The `integrate.polynomial` function integrates a multivariate polynomial against the normal distribution using ordinary integration.

The functions `toMoment`, `toNewick`, and `toMatching` convert among moment L-matrices, Newick trees, and ape matching objects.

**Note**

The mvtnorm package must be loaded for the simulate method. The cubature package must be loaded for the `integrate.polynomial` function. The combinat package must be loaded for the `toMoment` function.

**Author(s)**

Maintainer: Kem Phillips <kemphillips@comcast.net>

**References**


**Examples**

```r
# Compute the moment for the 4-dimensional moment c(1,2,3,4):
callmultmoments(c(1,2,3,4))

# Print the representation of the 4-dimensional moment c(1,2,3,4):
print(callmultmoments(c(1,2,3,4)))

# Compute the LaTeX representation of the central moment c(1,2,3,4):
toLatex(callmultmoments(c(1,2,3,4)))

# Write the LaTeX representation to a file using the standard R function (not run):
# writeLines(callmultmoments(c(1,2,3,4)),con="yourfilename", sep = "\n")

# evaluate the moment c(1,2,3,4) at the following variance-covariance matrix
# 4 2 1 1
# 2 3 1 1
```
# 1 1 2 1
evaluate(callmultmoments(c(1,2,3,4),c(4,2,1,1,3,1,1,2,1,2)))

# Using 100000 samples, estimate the central moment for c(2,4) at the covariance matrix (not run)
# 2 1
# 1 4

# and mean (0,0)
library(mvtnorm)
simulate(callmultmoments(c(2,4)),10000,NULL,c(0,0),c(2,1,1,4))

# Compute Latex representation of a non-central moment
# as.matrix(toLatex_noncentral(c(1,3)))

# Create all 2-dimensional moment objects with exponents up to 3
# First create the symmoments environment if it does not exist
# symmoments <- new.env()
# make.all.moments(c(3,3))

# Evaluate a non-central moment at a specified mean and covariance matrix
# Note that this invocation requires moments of order up to c(1,3)
# to exist in environment symmoments.
# evaluate_noncentral(c(1,3),c(1,2),c(1,0,1))

# Create an mpoly object
library(mpoly)
t0 <- mpoly(list(c(coef=3,x1=2),c(coef=2,x1=1,x2=3),
                 c(coef=-4,z=2),c(coef=1,x1=1,x2=2,z=1)))

# Convert an mpolyobject to a moment object
t1 <<- convert.mpoly(t0)

# Convert a moment object to a multipol object
t2 <<- convert.multipol(t1)

# Convert from multipol back to mpoly through moment
mpoly(convert.mpoly(convert.multipol(t2)))

# Evaluate the expected value of a multivariate polynomial
# Required moments must exist in environment symmoments.
# evaluate_expected.polynomial(t0,c(1,2,3),c(1,0,0,1,0,1))

# Create a Newick representation of a tree
exam.Newick <- "(((a,b),c),d);"

# Convert to phylo format
library(ape)
exam.phylo <- read.tree(text=exam.Newick)

# Convert to matching format
callmultmoments

```r
evaluated <- as.matching(phylo)
# Convert to L-matrix format
evaluated.L.matrix <- toMoment(evaluated.matching)
```

---

**callmultmoments**

*Compute multivariate moment symbolically*

**Description**

Computes a multivariate normal moment by initializing variables, calling multmoments, and constructing output.

**Usage**

```r
callmultmoments(moment)
```

**Arguments**

- `moment`
  - vector `c(k1, ..., kn)` specifying the moment `X1**k1 * ... * Xn**kn`

**Details**

Each row of the representation gives the exponents for a single product of covariance terms. For example, `(1,2,0)` represents `S11**1 S12**2 S22**0`, where the `Sij` are the covariances. The full moment is the sum of these terms multiplied by their respective coefficients. If the sum of the exponents is odd, the moment is 0.

**Value**

A object of class 'moment', which is a list with three components:

- `moment`
  - the input moment vector
- `representation`
  - a matrix containing the representation in terms of upper-triangular matrices
- `coefficients`
  - the coefficients corresponding to the rows of the representation

If the sum of the exponents is odd, returns -1 and prints "Sum of powers is odd. Moment is 0."

If any exponent is negative, returns -2 and prints "All components of the moment must be non-negative."

If any exponent is not an integer, returns -3 and prints "All components of the moment must be integers."

**Author(s)**

Kem Phillips <kemphillips@comcast.net>
References

See Also
multmoments and the methods toLatex, evaluate, and simulate in symmoments

Examples
# Compute the moment for the 4-dimensional moment c(1,2,3,4):
m.1234 <- callmultmoments(c(1,2,3,4))

convert.mpoly Convert between mpoly and list representations of multivariate polynomials

Description
Converts an mpoly object to a simple list representing a multivariate polynomial or a simple list to an mpoly object

Usage
convert.mpoly(poly)

Arguments
poly an mpoly object or a list giving powers and coefficients defining the polynomial

Details
The list representation consists of 2 components: 'powers' is a matrix with each row representing the powers of X in one term of the multivariate polynomial. 'coeff' is a vector with each element being the coefficient of the corresponding term in powers

Value
if poly is of class 'mpoly', it is a list with two components shown below. If poly is such a list, the value is the corresponding mpoly object

Author(s)
Kem Phillips <kemphillips@comcast.net>
convert.multipol

References


See Also

cert.convert.multipol, evaluate.expected.polynomial, integrate.polynomial

Examples

# create an mpoly object here (requires library mpoly) (not run)

# convert from mpoly to list representation
# t0 <- mpoly::mpoly(list(c(coef=3,x1=2),c(coef=2,x1=1,x2=3),
# c(coef=-4,z=2),c(coef=1,x1=1,x2=2,z=1)))
# t1 <- convert.multipol(t0)
# convert from list representation back to an mpoly object
# t2 <- convert.multipoly(t1)

convert.multipol Convert between multipol and list representations of multivariate polynomials

Description

Converts an multipol object to a simple list representing a multivariate polynomial or a simple list to an multipol object

Usage

convert.multipol(poly)

Arguments

poly a multipol object or a list giving powers and coefficients defining the polynomial

Details

The list representation consists of 2 components: 'powers' is a matrix with each row representing the powers of X in one term of the multivariate polynomial. 'coeff' is a vector with each element being the coefficient of the corresponding term in powers

Value

if poly is of class 'multipol', it is a list with two components described below. If poly is such a list, the value is the corresponding multipol object
evaluate

Evaluate a multivariate moment

description

Generic method for class moment to compute the numerical value of a moment at a specified covariance matrix from the output of callmultmoments

usage

## S3 method for class 'moment'
evaluate(object, sigma)

arguments

object an object of class 'moment'
sigma an upper-triangular matrix of covariance terms expressed as a vector at which the moment is to be evaluated
**Details**

object is normally the output of a call to callmultmoment. This is a list with first component the moment itself, the second component the set of upper-triangular matrices representing the moment, and the third component containing their corresponding coefficients. This is an object of class 'moment'.

**Value**

numeric value of the moment at the specified covariance matrix

**Author(s)**

Kem Phillips <kemphillips@comcast.net>

**References**


**See Also**

callmultmoments and the simulate and toLatex methods from the symmoments package

**Examples**

```r
evaluate(callmultmoments(c(1,2,3,4)),c(4,2,1,1,3,1,1,2,1,2))
# evaluates the moment at c(1,2,3,4) at the following covariance matrix
#  4 2 1 1
#  2 3 1 1
#  1 1 2 1
#  1 1 1 2
```

---

**evaluate_expected.polynomial**

*Evaluate the expected value of a multivariate polynomial*

**Description**

Evaluate the expected value of a multivariate polynomial assuming a specified non-central multivariate distribution.

**Usage**

```r
evaluate_expected.polynomial(poly, mu, sigma, envir='symmoments')
```
**evaluate_expected.polynomial**

**Arguments**

- **poly**: either an object of class 'mpoly' or 'multipol', or a list with components for coefficients and powers.
- **mu**: a vector of real numbers representing the mean of the multivariate distribution
- **sigma**: an vector giving an upper-triangular matrix representing the covariance matrix of the multivariate distribution
- **envir**: a character variable specifying the environment containing the central moments needed for the calculation

**Details**

This function looks in the environment specified in the envir argument for the central moments needed in the calculation. The default is the symmoments environment. The computation stops with an error message if a required moment is not found in envir.

**Value**

expected value of the multivariate polynomial at the specified multivariate normal mean and covariance matrix

**Author(s)**

Kem Phillips <kemphillips@comcast.net>

**References**


**See Also**

See the evaluate_noncentral and make.all.moments functions.

**Examples**

```r
# define a mpoly object for a multivariate polynomial and determine # its expected value at specified mean and covariance matrix:
# note that all moments up to c(2,3,2) must exist in the symmoments # environment. Use make.all.moments(c(2,3,2)) if necessary.
# use library(mpoly) for first statement below.

# t0 <- mpoly(list(c(coef=3,x1=2),c(coef=2,x1=1,x2=3),c(coef=-4,z=2),c(coef=1,x1=1,x2=2,z=1)))
# evaluate_expected.polynomial(t0,c(1,2,3),c(1,0,0,1,0,1))
```
evaluate_noncentral

---

**Evaluate a noncentral multivariate moment**

**Description**

Computes the numerical value of a non-central moment at a specified mean and specified covariance matrix.

**Usage**

```r
evaluate_noncentral(moment, mu, sigma, envir = 'symmoments')
```

**Arguments**

- `moment`: a vector of non-negative integers representing the non-central moment to be evaluated.
- `mu`: a vector of real numbers representing the mean of the multivariate normal distribution.
- `sigma`: an upper-triangular matrix of covariance terms for the multivariate normal distribution expressed as a vector at which the moment is to be evaluated.
- `envir`: a character variable specifying the environment containing the central moments needed for the calculation.

**Details**

This function looks in the environment specified in the `envir` argument for the central moments needed in the calculation. The default is the `symmoments` environment. All even moments less than or equal to the moment argument are required. The computation stops with an error message if a required moment is not found in `envir`.

**Value**

Numeric value of the moment at the specified mean and covariance matrix.

**Author(s)**

Kem Phillips <kemphillips@comcast.net>

**References**


**See Also**

See the `evaluate.moment` and `make.all.moments` functions.
Examples

# evaluate_noncentral(c(3,1,2),c(3,4,1),c(4,2,1,3,1,2))
# evaluates the expected value of X1**3 X2 X3**2 at mean c(1,2,3)
# and at the following covariance matrix
# 4 2 1
# 2 3 1
# 1 1 2

# requires all moments up to c(3,1,2) to exist in the symmoments environment.
# use make.all.moments(c(3,1,2)) if necessary.

# use moments in the global environment:
# evaluate_noncentral(c(3,1,2),c(3,4,1),c(4,2,1,3,1,2),'GlobalEnv')

integrate.polynomial

**Numerically integrate a multivariate polynomial**

Description

Integrates a multivariate polynomial against a specified non-central multivariate distribution using ordinary integration by invoking the adaptIntegrate function from the cubature package.

Usage

```r
integrate.polynomial(poly,mu,sigma,lower=NULL,upper=NULL)
```

Arguments

- **poly**: either an object of class ‘mpoly’ or ‘multipol’, or a list with two components for coefficients and powers.
- **mu**: a vector giving the mean of the multivariate distribution
- **sigma**: a square matrix giving the covariance matrix of the multivariate distribution
- **lower**: vectors of the lower limits of integration, one element for each dimension of the moment
- **upper**: vectors of the upper limits of integration, one element for each dimension of the moment

Details

Defaults for lower and upper are +/- 6 times the standard deviations (square roots of diagonal elements of the covariance matrix). If the polynomial is defined by a list, it has two components, coeff and powers. powers is a matrix. Each row represents the powers for a term in the polynomial. coeff is a vector. Each element is the coefficient of the corresponding power. Example corresponding to example below: list(coeff=c(3,2,-4,1),powers=matrix(c(2,0,0,1,3,0,0,0,2,1,2,1),ncol=3,byrow=TRUE))
Value

the expected value of the polynomial integrated against the multivariate normal distribution

Author(s)

Kem Phillips <kemphillips@comcast.net>

References


See Also

evaluate.expected.polynomial, multmoments, evaluate, and simulate in symmoments

Examples

# define a mpoly object for a multivariate polynomial, and
# determine its expected value at specified mean and covariance matrix:
# t0 <- mpoly(list(c(coef=3,x1=2),c(coef=2,x1=1,x2=3),c(coef=-4,z=2),c(coef=1,x1=1,x2=2,z=1)))

# integrate.polynomial(t0,c(1,2,3),matrix(c(1,0,0,0,1,0,0,0,1),nrow=3,byrow=TRUE))
make.all.moments

Details

Unsorted moments, those with exponents are not in numeric order, are created in the symmoments environment using the tounsorted function to transform from the sorted moment. If symmoments does not exist, the user is prompted to create it using symmoments <- new.env().

If the sorted moment does not exist, it is created.

Moments of lower dimension are not created; for example, if c(2,4) is input, m20 is created, but m2 is not.

Moments are named mij..l, e.g., m136. If any exponent is greater than 9, lower case letters and then upper case letters are used. For example, m3bA is the name of the moment c(3,11,36).

The largest exponent allowed by this scheme is 9+26+26=61,

If an object with a name of this form exists but is not an object of class "moment", it is replaced (overwritten) by the moment object.

Value

all objects of class 'moment' up to the value given in moment are created in environment symmoments

Author(s)

Kem Phillips <kemphillips@comcast.net>

References


See Also

callmultmoments, tounsorted

Examples

# Compute all moments up to c(3,3)
# First create the symmoments environment if it does not exist
# symmoments <- new.env()
# make.all.moments(c(3,3))
multmoments

Recursive function to compute a multivariate moment

Description
Called by callmultmoments to compute representation of a multivariate normal moment using recursive algorithm

Usage
multmoments(moment, current.matrix, current.cell, moment.rep, row_col)

Arguments
- moment: vector (k1,...,kn) specifying the moment X1**k1 *...* Xn**kn
- current.matrix: upper-triangular integer matrix under consideration in recursion
- current.cell: cell in current matrix under consideration in recursion
- moment.rep: current set of representations; mult.moments adds each satisfying matrix to moment.rep
- row_col: matrix giving rows and columns for square matrix for each cell

Details
Each row of the representation gives the exponents for a single product of covariance terms. For example, (1,2,0) represents S11**1 S12**2 S22**0, where the Sij are the covariances.
This function would normally only be called by callmultmoments.

Value
moment representation, moment.rep, augmented with additional representations

Author(s)
Kem Phillips <kemphillips@comcast.net>

References

See Also
callmultmoments (symmoments)
print(moment)

Print the representation of a multivariate moment

Description

Prints an object of class 'moment'

Usage

## S3 method for class 'moment'
print(x,...)

Arguments

x an object of class 'moment', usually the output of callmultmoments

... Included only for consistency with generic function

Details

Prints the moment as E[X1**k1 X2**k2 ...]: followed by the lines of the representation with the corresponding coefficient attached

Author(s)

Kem Phillips <kemphillips@comcast.net>

References


See Also

callmultmoments (symmoments)

Examples

print(callmultmoments(c(1,2,3)))
**simulate.moment**

*Method to compute a multivariate moment using Monte Carlo integration*

**Description**

Computes a multivariate normal moment by Monte Carlo integration

**Usage**

```r
## S3 method for class 'moment'
simulate(object, nsim, seed, Mean, Sigma, ...)
```

**Arguments**

- `object` object of class 'moment' representing $E[X_1^{k_1},...,X_n^{k_n}]$
- `nsim` the number of samples to generate in computing the integral
- `seed` integer for random number generator (set.seed)
- `Mean` the mean of $(X_1,...,X_n)$
- `Sigma` covariance of $(X_1^{k_1},...,X_n^{k_n})$, dimension $nXn$, expressed as a vector by row
- `...` Included only for consistency with generic function

**Value**

Approximate value of the moment

**Note**

Non-central moments can be approximated by specifying `Mean`. For central moments, set `Mean` to a vector of 0s.

The mvtnorm package must be loaded for the function rmvnorm.

**Author(s)**

Kem Phillips <kemphillips@comcast.net>

**References**


**See Also**

callmultmoments and the methods toLatex and evaluate from symmoments
Examples

```r
# Using 10000 samples, estimate the central moment for the moment c(2,4) at the covariance matrix
# 2 1
# 1 4

# and mean (0,0)
library(mvtnorm)
simulate(callmultmoments(c(2,4)),10000,NULL,c(0,0),c(2,1,1,4))
```

### toLatex.moment

**LaTeX a multivariate moment**

### Description

Computes a LaTeX representation sorted lexicographically of an object of class 'moment'

### Usage

```r
## S3 method for class 'moment'
toLatex(object,...)
```

### Arguments

- `object`: an object of class 'moment', usually the output of `callmultmoments`
- `...`: Included only for consistency with generic function

### Details

The first element of the result is the moment expressed as an expected value (E[...] =). The remaining lines are the LaTeX representation broken at appropriate intervals for printing. (Individual terms for high dimensions will still overrun a printed line.) Double backslashes are inserted where LaTeX requires a backslash. These can be reset to single backslashes by writing the output to a file using the R function `writeLines` from the base package.

### Value

Character vector giving the LaTeX code for the symbolic moment

### Author(s)

Kem Phillips <kemphillips@comcast.net>

### References

See Also

callmultmoments and the evaluate method (symmoments)

Examples

toLatex(callmultmoments(c(1,2,3)))

Compute a Latex expression for a noncentral moment

Description

Compute a Latex expression for a noncentral moment

Usage

toLatex_noncentral(moment,envir="symmoments")

Arguments

moment vector c(k1,...,kn) specifying the moment X1**k1***Xn**kn
envir character variable specifying the environment that contains the required central moments

Details

All required moment objects must exist in the specified environment, with default 'symmoments'. However, if the sorted version of an unsorted moment exists, the tounsorted function is used to obtain it.

Value

A text value giving the Latex representation of moment where X is multivariate normal

Author(s)

Kem Phillips <kemphillips@comcast.net>

References


See Also

make.all.moments, tounsorted, callmultmoments and the method toLatex
Examples

# Compute the Latex representation of the 2-dimensional moment c(1,3) (not run).
# This requires that all moments up to c(1,3) exist in the symmoments environment.
# toLatex_noncentral(c(1,3))

**toMatching**

*Convert representation of a phylogenetic tree as a moment L-matrix to matching form*

**Description**

Function converts a tree in moment format to matching format.

The input can be an L-matrix object, a square L matrix, or an L matrix in reduced upper-triangular (vector) form.

The `toMatching` function sets its list output to class `L-matching`, which has 5 components, including the tree in matching format.

**Usage**

`toMatching(L, type = NULL, tip.label = NULL)`

**Arguments**

- `L` An L-matrix object, a square L matrix, or an L matrix in reduced upper-triangular (vector) form.
- `type` If object is not of class "L-matrix" and is a square L matrix, then type should be "square". If it is an L matrix in upper triangular form, type should be "ut".
- `tip.label` Character vector containing labels for tips. If null, labels default to "a"-"z" it at most 26; otherwise, 3-letter labels of the form "aaa", "aab"....

**Details**

An L-matrix object is a list with 5 components: "L" is the L-matrix in square form. "L.ut" is the L-matrix in upper-triangular form. "Newick" is the Newick representation of the tree. "tip.label" is the character vector of tip labels. "tip.label.n" is the number of tips.

**Value**

a matching representation of the phylogenetic tree corresponding to the input

**Author(s)**

Kem Phillips <kemphillips@comcast.net>
toMoment

References


See Also

functions toMoment and toNewick

Examples

# create a Newick object
exam.Newick <- "(((a,b),c),d);
# convert to a moment L-matrix
exam.moment <- toMoment(exam.Newick)
# convert to matching format
exam.matching <- toMatching(exam.moment)

toMoment

Converts a tree from Newick or matching to moment format

Description

Converts a tree from Newick or matching to moment format

Usage

toMoment(inputobject, tip.label = NULL)

Arguments

inputobject a tree in Newick format or a matching object defined in the ape package
tip.label rearranged labels for tips; these must be the original labels

Details

The L-matrix class consists of $5$ components: "L" is the L-matrix in square form. "L.ut" is the L-matrix in upper-triangular form. "Newick" is the Newick representation of the tree. "tip.label" is the character vector of tip labels. "tip.label.n" is the number of tips.

Value

a moment L-matrix corresponding to the input phylogentic tree object

Author(s)

Kem Phillips <kemphillips@comcast.net>
**toNewick**

**Function**: Convert representation of phylogenetic tree as a moment L-matrix to Newick form.

**Description**

The function converts a tree in moment format to Newick format. The input can be an L-matrix object, a square $L$ matrix, or an $L$ matrix in reduced upper-triangular (vector) form. The toNewick function sets its list output to class L-Newick, which has 5 components, including the tree in Newick format.

**Usage**

```r
toNewick(L, type = NULL, tip.label = NULL)
```

**Arguments**

- **L**: L can be an L-matrix object, a square $L$ matrix, or an $L$ matrix in reduced upper-triangular (vector) form.
- **type**: if L is not a L-matrix object, either 'square' or 'ut' as listed above
- **tip.label**: Character vector containing labels for tips. If null, labels default to "a"-"z" it at most 26; otherwise, 3-letter labels of the form "aaa", "aab",...

**Examples**

```r
# create a Newick object
exam.Newick <- "(((a,b),c),d);"
# convert to a moment L-matrix
exam.moment <- toMoment(exam.Newick)
# convert to matching object
exam.matching <- toMatching(exam.moment)
# convert back to moment object
backto.moment <- toMoment(exam.matching)
```

**References**


**See Also**

functions toNewick and toMatching
**Details**

An L-matrix object is a list with 5 components: "L" is the L-matrix in square form. "L.ut" is the L-matrix in upper-triangular form. "Newick" is the Newick representation of the tree. "tip.label" is the character vector of tip labels. "tip.label.n" is the number of tips.

**Value**

a Newick representation of the phylogenetic tree corresponding to the input

**Author(s)**

Kem Phillips <kemphillips@comcast.net>

**References**


**See Also**

functions toMoment and toMatching

**Examples**

```r
# create a Newick object
exam.Newick <- "(((a,b),c),d);"
# convert to a moment L-matrix
exam.moment <- toMoment(exam.Newick)
# convert back to Newick format
backto.Newick <- toNewick(exam.moment)
```

---

**tounsorted**

*Compute an unsorted central moment object from a sorted object*

**Description**

-produces an unsorted central moment object from a sorted object of class "moment".

Unsorted moments are those with exponents not in numeric order, e.g., m312.

**Usage**

tounsorted(moment,sorted.moment)
Arguments

- moment: unsorted moment to obtain moment is in vector form, e.g., c(3,1,2)
- sorted.moment: sorted moment to use in obtaining unsorted moment

Details

The unsorted moment is obtained by resorting the rows and columns of the sorted moment successively.

Value

A object of class 'moment', which is a list with three components:

- moment: the input moment vector
- representation: a matrix containing the representation in terms of upper-triangular matrices
- coefficients: the coefficients corresponding to the rows of the representation

Author(s)

Kem Phillips <kemphillips@comcast.net>

References


See Also

multmoments

Examples

```r
# obtain moment m312 from m123
tounsorted(c(3,1,2),callmultmoments(c(1,2,3)))
```
Index

callmultmoments (callmultmoments), 5
callmultmoments, 5
callmultmoments (callmultmoments), 5
callmultmoments, 5
convert.mpoly (convert.mpoly), 6
convert.mpoly, 6
convert.mpoly (convert.mpoly), 6
convert.mpoly, 6
convert.multipol (convert.multipol), 7
convert.multipol, 7
evaluate (evaluate), 8
evaluate (evaluate), 8
evaluate_expected.polynomial (evaluate_expected.polynomial), 9
evaluate_noncentral (evaluate_noncentral), 11
evaluate_noncentral, 11
evaluate_noncentral (evaluate_noncentral), 11
integrate.polynomial (integrate.polynomial), 12
integrate.polynomial, 12
integrate.polynomial (integrate.polynomial), 12
integrate.polynomial, 12
make.all.moments (make.all.moments), 13
make.all.moments, 13
multmoments (multmoments), 15
multmoments, 15
simulate.moment (simulate.moment), 17
simulate.moment, 17
symmoments (symmoments-package), 2
symmoments-package (symmoments-package), 2
symmoments-package (symmoments-package), 2
symmoments-package, 2
toLatex_noncentral (toLatex_noncentral), 19
toLatex_noncentral, 19
toLatex_noncentral, 19
toMatching, 20
toMoment, 21
toNewick, 22
tounsorted (tounsorted), 23
tounsorted, 23
tounsorted (tounsorted), 23
toNewick, 22
tounsorted, 23
toNewick (toNewick), 22

callmultmoments, 5
callmultmoments, 5
callmultmoments (callmultmoments), 5
callmultmoments, 5
callmultmoments (callmultmoments), 5
callmultmoments, 5
callmultmoments (callmultmoments), 5
callmultmoments, 5
convert.mpoly, 6
convert.mpoly, 6
convert.mpoly (convert.mpoly), 6
convert.mpoly, 6
convert.mpoly (convert.mpoly), 6
convert.mpoly, 6
convert.multipol, 7
convert.multipol, 7
convert.multipol (convert.multipol), 7
convert.multipol, 7
evaluate, 8
evaluate, 8
evaluate_expected.polynomial, 9
evaluate_noncentral, 11
evaluate_noncentral, 11
integrate.polynomial, 12
integrate.polynomial, 12
make.all.moments, 13
make.all.moments, 13
multmoments, 15
multmoments, 15
print.moment, 16
print.moment (print.moment), 16
print.moment (print.moment), 16
print.moment, 16
symmoments-package, 2
symmoments-package, 2
symmoments-package, 2
symmoments-package, 2