Package 'kStatistics'

July 24, 2020

Type Package

Title Unbiased Estimators for Cumulant Products

Version 2.0

Date 2020-07-18

Author Elvira Di Nardo <elvira.dinardo@unito.it>, Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

Maintainer Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

Description Tools for estimate (joint) cumulants and products of (joint) cumulants of a given population distribution using (multivariate) k-statistics and (multivariate) polykays, unbiased estimators with minimum variance. Tools for generating univariate and multivariate Faa di Bruno's formula and related polynomials, such as Bell polynomials, generalized complete Bell polynomials, partition polynomials, generalized partition polynomials. For more details see Di Nardo E., Guarino G., Senato D. (2009) <arXiv:0807.5008>, <arXiv:1012.6008>.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-24 09:50:02 UTC

R topics documented:

`kStatistics-package` *Unbiased Estimators for Cumulant Products*

Description

kStatistics is a package allowing to estimate (joint) cumulants and products of (joint) cumulants of a given distribution using (multivariate) k-statistics and (multivariate) polykays, which are symmetric unbiased estimators. The procedures rely on a symbolic method arising from the classical umbral calculus and described in the referred papers. In the package, a set of combinatorial devices are given such as integer partitions, set partitions and multiset subdivisions, pairing and merging of multisets, useful in the construction of these estimators. In the package, there are also functions to recover univariate and multivariate cumulants from univariate and multivariate moments (and vicev-ersa) using Faa di Bruno's formula. The function returning Faa di Bruno's formula allows to recover coefficients of composition of exponential power series such as f(g(z)) with f and g both univariate, such as f(g(z1,...,zn)) with f univariate and g multivariate, such as f(g1(z1,...,zn),...,gm(z1,...,zn)) with f and g both multivariate. Faa di Bruno's formula might also be employed to recover iterated derivative of all these compositions. Lastly, using Faa di Bruno's formula, some special families of polynomials are also generated, such as Bell polynomials, generalized complete Bell polynomials, partition polynomials, generalized partition polynomials. Applications of these polynomials are described in the referred papers.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

Elvira Di Nardo <elvira.dinardo@unito.it>, Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

Maintainer: Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Charalambides C.A. (2002) Enumerative Combinatoris, Chapman & Haii/CRC.

Constantine G. M., Savits T. H. (1996) A Multivariate Faa Di Bruno Formula With Applications. Trans. Amer. Math. Soc. 348(2), 503–520.

E. Di Nardo E. (2016) On multivariable cumulant polynomial sequence with applications. Jour. Algebraic Statistics 7(1), 72-89. (download from http://arxiv.org/abs/1606.01004)

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2009) A new method for fast computing unbiased estimators of cumulants. Statistics and Computing, 19, 155-165. (download from https://arxiv.org/abs/0807.5008)

E. Di Nardo, G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

E. Di Nardo, M. Marena, P. Semeraro (2020) On non-linear dependence of multivariate subordinated Levy processes. In press Stat. Prob. Letters (download from https://arxiv.org/abs/2004.03933)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

Examples

```
# Some of the most important functions:

# Data assignment
data1<-c(16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68,
16.08, 19.43,8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10,
15.02, 16.83, 16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11)

# Data assignment
data2<-list(c(5.31,11.16),c(3.26,3.26),c(2.35,2.35),c(8.32,14.34),c(13.48,49.45),
c(6.25,15.05),c(7.01,7.01),c(8.52,8.52),c(0.45,0.45),c(12.08,12.08),c(19.39,10.42))

# Return an estimate of the third cumulant of the population distribution with the indication
# of which function has been employed
```

```
# KS:[1] -1.44706
nPolyk(c(3), data1, TRUE)

# Return an estimate of the product of the mean and the variance of the population distribution
# with the indication of which function has been employed
# PS:[1] 177.4233
nPolyk( list(  c(2), c(1) ), data1, TRUE)

# Return an estimate of the joint cumulant c[2,1] of the population distribution  with the
# indication of which function has been employed
# KM:[1] -23.7379
nPolyk(c(2,1), data2, TRUE);

# Return an estimate of the product of joint cumulants c[2,1]*c[1,0] of the population
# distribution with the indication of which function has been employed
# PM:[1] 48.43243
nPolyk( list(  c(2,1), c(1,0) ), data2, TRUE)

# Return all subdivisions of a multiset with only one element of multiplicity 3
mkmSet(3)

# Return all subdivisions of a multiset with two elements,
# having multiplicity respectively 2 and 1
mkmSet(c(2,1))

# Faa di Bruno's formula (Univariate with Univariate Case)
# The coefficient of z^2 in f(g(z)), that is f[2]g[1]^2 + f[1]g[2], where
# f[1] is the coefficient of t in f(t) with t=g(z)
# f[2] is the coefficient of t^2 in f(t) with  t=g(z)
# g[1] is the coefficient of z in g(z)
# g[2] is the coefficient of z^2 in g(z)
#
MFB( c(2), 1 )

# Faa di Bruno's formula (Univariate with Multivariate Case)
# The coefficient of z1 z2 in f(g(z1,z2)), that is f[1]g[1,1] + f[2]g[1,0]g[0,1]
# where
# f[1]   is the coefficient of t in f(t) with t=g(z1,z2)
# f[2]   is the coefficient of t^2 in f(t) with t=g(z1,z2)
# g[1,0] is the coefficient of z1 in g(z1,z2)
# g[0,1] is the coefficient of z2 in g(z1,z2)
# g[1,1] is the coefficient of z1z2 in g(z1,z2)
#
MFB( c(1,1), 1 )

# Faa di Bruno's formula (Multivariate with Multivariate Case)
# The coefficient of z in f((g1(z),g2(z)), that is  f[1,0]g1[1] + f[0,1]g2[1] where
# f[1,0] is the coefficient of t1 in f(t1,t2) with t1=g1(z) and t2=g2(z)
# f[0,1] is the coefficient of t2 in f(t1,t2) with t1=g1(z) and t2=g2(z)
# g1[1]  is the coefficient of z of g1(z)
# g2[1]  is the coefficient of z of g2(z)
MFB( c(1), 2 )
```

```
# The numerical value of f[1]g[1,1] + f[2]g[1,0]g[0,1], that is the coefficient of x1x2
# in f(g1(x1,x2),g2(x1,x2))) output of MFB(c(1,1),1) when
# f[1] = 5 and f[2] = 10
# g[0,1]=3,  g[1,0]=6,  g[1,1]=9
e_MFB(c(1,1),1, c(5,10), c(3,6,9))

# Multivariate cumulant k[3,1] in terms of the multivariate moments m[i,j] for i=0,1,2,3 and j=0,1.
cum2mom(c(3,1))

# Multivariate moment m[3,1] in terms of the multivariate cumulants k[i,j] for i=0,1,2,3 and j=0,1.
mom2cum(c(3,1))

# Partition polynomial F[5]
pPart(5)

#  The general partition polynomial G[2], that is a2(y1^2) + a1(y2)
gpPart(2)

# The complete exponential Bell Polynomial B[5], that is (y1^5) + 10(y1^3)(y2) +
# 15(y1)(y2^2) + 10(y1^2)(y3) + 10(y2)(y3) + 5(y1)(y4) + (y5)
BellPol(5)

# The partial ordinary Bell polynomial Bo[5,3], 330(y1)(y2^2) + 60(y1^2)(y3)
BellPol(5,3,FALSE);

# The generalized complete Bell Polynomial for n=1, m=1 and g1=g,
# that is (y^2)g[1]^2 + (y)g[2]
#
GCBellPol( c(2),1 )

# The eneralized complete Bell Polynomial for n=1, m=2 and g1=g, corresponding to the
# cumulant polinomial indexed by (2,1),
# that is 2(y^2)g[1,0]g[1,1] + (y^3)g[0,1]g[1,0]^2 + (y)g[2,1] + (y^2)g[0,1]g[2,0]
#
GCBellPol( c(2,1),1 )


# The generalized complete Bell Polynomial for n=2, m=2 and g1=g2=g, corresponding to the
# cumulant polinomial in y1+y2 indexed by (1,1),
# that is (y1)g[1,1] + (y1^2)g[0,1]g[1,0] + (y2)g[1,1] + (y2^2)g[0,1]g[1,0] + 2(y1)(y2)g[0,1]g[1,0]
#
GCBellPol( c(1,1),2, TRUE )

# The polynomial 2(y^2)g[1,1]g[1,0] + (y^3)g[1,0]^2g[0,1] + (y)g[2,1] + (y^2)g[2,0]g[0,1], output of
# GCBellPol( c(2,1),1 ), when g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5,
# that is 16(y^2) + 4(y^3) + 5(y)
#
e_GCBellPol(c(2,1),1,,c(1:5))
#
# OR (same output)
#
e_GCBellPol(c(2,1),1,,c(1,2,3,4,5))
#
```

```
# OR (same output)
#
e_GCBellPol( c(2,1),1,"g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5" )

# The polynomial 2(y^2)g[1,1]g[1,0] + (y^3)g[1,0]^2g[0,1] + (y)g[2,1] + (y^2)g[2,0]g[0,1], output of
# GCBellPol( c(2,1),1 ) when g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5 and y=7, that is 2191
#
e_GCBellPol( c(2,1),1,c(7),c(1:5) )
#
# OR (same output)
#
e_GCBellPol( c(2,1),1,"y=7, g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5" )
```

`BellPol` *Bell polynomials*

Description

The routine generates complete and partial exponential Bell polynomials, as well as complete and partial ordinary Bell polynomials.

Usage

```
BellPol(n = 0, m = 0, b = TRUE)
```

Arguments

`n`	integer, the degree of the polynomial
`m`	integer, the fixed degree of each monomial in the polynomial
`b`	boolean, TRUE (default) for exponential Bell polynomials, FALSE for ordinary Bell polynomials

Details

Faa di Bruno's formula gives the coefficients of the exponential formal power series obtained from the composition of the exponential formal power series f() and g(). Complete exponential Bell polynomials in the variables y1,...,yn are generated by setting f[i]=1 and g[i]=yi, for each i from 1 to n. Partial exponential Bell polynomials are polynomials in the variables y1,...,yn with fixed degree m for each of the involved monomials. Partial exponential Bell polynomials are recovered from Faa di Bruno's formula by setting f[i]=1 if i=m, f[i]=0 otherwise and g[i]=yi for each i from 1 to n. Observe that complete exponential Bell polynomials are the summation of partial exponential Bell polynomials for m from 1 to n.
Partial ordinary Bell polynomials in the variables y1,...,yn are recovered from Faa di Bruno's formula setting f[i]=1 if i=m, f[i]=0 otherwise and g[i]=i! yi for each i from 1 to n. Complete ordinary Bell polynomials are recovered from Faa di Bruno's formula by setting f[i]=1 and g[i]=i!yi for each i from 1 to n. Observe that complete ordinary Bell polynomials are the summation of partial ordinary Bell polynomials for m from 1 to n.

Value

`string` the expression of the generated Bell polynomial

Warning

The value of the first parameter is the same as the MFB function in the univariate with univariate case composition.

Note

This function calls the MFB function in the package `kStatistics`.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Charalambides C.A. (2002) Enumerative Combinatoris, Chapman & Haii/CRC.

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

See Also

MFB

Examples

```
# Complete ordinary Bell Polynomial n=5, that is
# (y1^5) + 20(y1^3)(y2) + 30(y1)(y2^2) + 60(y1^2)(y3) + 120(y2)(y3) + 120(y1)(y4) + 120(y5)
BellPol(5,,FALSE)
# or
BellPol(5,0,FALSE)

# Complete exponential Bell Polynomial n=5, that is
# (y1^5) + 10(y1^3)(y2) + 15(y1)(y2^2) + 10(y1^2)(y3) + 10(y2)(y3) + 5(y1)(y4) + (y5)
BellPol(5)

# Partial ordinary Bell polynomial n=5 and m=3, that is
# 30(y1)(y2^2) + 60(y1^2)(y3)
```

```
BellPol(5,3,FALSE)

# Partial exponential Bell Polynomial n=5 and m=3, that is
# 15(y1)(y2^2) + 10(y1^2)(y3)
BellPol(5,3)
```

---

countP *Multiplicity of a subdivision*

Description

Compute the multiplicity of a subdivision of a multiset.

Usage

```
countP( v )
```

Arguments

v	vector or list

Details

The subdivisions of a multiset are obtained as follows: assume all distinct the elements of the multiset, determine all set partitions and then replace each element in each block with the original one. For example, the partitions of the set [a1, a2, a3] are [[a1], [a2], [a3]], [[a1,a2], [a3]], [[a1,a3], [a2]], [[a2,a3], [a1]], [[a1,a2,a3]]. Replace a1<-a, a2<-a, a3<-a. The label does not matter. The subdivisions are [[a], [a], [a]], [[a,a], [a]], [[a,a], [a]], [[a,a], [a]], [[a,a,a]]. A short notation for the output is [[1,1,1],1], [[1,2],3], [[3],1]. The function countP(c(1,2)) returns 3, that is the multiplicity of the subdivision [[a,a], [a]]. If in the multiset, there is more than one element, we pass a list. For example, for the multiset [a,a,a,a,b,b], the multiplicity of the subdivision [[a,a,b], [a], [a], [b]] is obtained by calling countP(list(c(2,1), c(1,0), c(1,0), c(0,1))), where c(i,j) denotes a block with instances i and j for a and b respectively.

Value

integer	the multiplicity of the given item

Note

Called by the function mkmSet, nPS in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

See Also

mkmSet, umSet, mCoeff, nStirling2, intPart, df

Examples

```
# Return  the multiplicity of [[a],[aa]]
countP(c(1,2))

# Return  the multiplicity of [[a,a,a,a],[a,a]]
countP(c(4,2))

# Return  the multiplicity of [[aa],[a],[a],[b],[bb]]
countP( list(c(2,0), c(1,0), c(1,0), c(0,1),c(0,2)) )
```

---

cum2mom *Cumulants in terms of moments*

Description

Compute simple and multivariate cumulants in terms of simple and multivariate moments.

Usage

```
cum2mom(n = 0)
```

Arguments

n	integer or vector of integers

Details

Faa di Bruno's formula (the MFB function) gives the coefficients of the exponential formal power series obtained from the composition f(g()) of exponential formal power series. Simple cumulants are expressed in terms of moments using the Faa di Bruno's formula obtained from the MFB function in the case "composition of univariate f with univariate g" with f[i]=(-1)^(i-1)*(i-1)!, g[i]=m[i] for each i from 1 to n and m[i] moments. Multivariate cumulants are expressed in terms of multivariate moments using the Faa di Bruno's formula obtained from the MFB function in the case "composition of univariate f with multivariate g" whose coefficients are the multivariate moments.

Value

string	the expression of cumulants in terms of moments

Warning

The value of the first parameter is the same as the MFB function in the univariate with univariate case composition and in the univariate with multivariate case composition.

Note

This function calls the MFB function in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo E., G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

See Also

MFB

Examples

```
# Simple cumulant k[5] in terms of the moments m[1],..., m[5].
cum2mom(5)

# Multivariate cumulant k[3,1] in terms of the multivariate moments m[i,j] for i=0,1,2,3 and j=0,1.
cum2mom(c(3,1))
```

e_GCBellPol *Evaluation of Generalized Complete Bell Polynomial*

Description

The function assigns numerical values to the coefficients of the generalized complete Bell polynomial, output of the function GCBellPol. The input vector of integers identifies the subscripts of the polynomial coefficients. When numerical values are assigned to the variables y1,...,yn too, the function returns an overall numerical value.

Usage

```
e_GCBellPol(pv = c(), pn = 0, pyc = c(), pc = c(), b = FALSE)
```

Arguments

pv	vector of integers, the subscript of the generalized complete Bell polynomial
pn	integer, the number of y's variables
pyc	vector, the values to be assigned to the y's variables [optional], or string, the direct assignment of the y's variables and/or the related coefficients
pc	vector, the values to be assigned to the coefficients of the polynomial, [optional if pyc is a string]
b	boolean, if TRUE the function prints the list of all the assignments

Details

The output of the function GCBellPol is a coefficient of exp(y1 g1(z1,...,zm) + ... + yn gn(z1,...,zm)), where y1,...,yn are variables. The function e_GCBellPol allows to assign numerical values to the coefficients of the power series g1,...,gn together with the variables y1,...,yn (optional). These values are passed to e_GCBellPol through the third and the fourth input parameter. The y's and the g's in the resulting expression are managed in lexicographic order. There is one more input boolean parameter: when it is TRUE, the function prints the list of all the assignments. See the examples for more details on the employment of this boolean parameter for assigning values to the coefficients and/or to the variables of the generalized complete Bell polynomial.

Value

`string` or `numerical`
evaluation of the generalized complete Bell Polynomial

Warning

The value of the first parameter is the same as the mkmSet function, i.e. the number of blocks considered.

Note

Called by the function GCBellPol in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo E. (2016) On multivariable cumulant polynomial sequence with applications. Jour. Algebraic Statistics 7(1), 72-89. (download from http://arxiv.org/abs/1606.01004)

E. Di Nardo, G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

E. Di Nardo, M. Marena, P. Semeraro (2020) On non-linear dependence of multivariate subordinated Levy processes. In press Stat. Prob. Letters (download from https://arxiv.org/abs/2004.03933)

See Also

mkmSet, MFB, GCBellPol

Examples

```
#-------------------------------------------------------------------------------#
# Evaluation of the generalized complete Bell polynomial indexed by 2
#-------------------------------------------------------------------------------#
#
# The polynomial (y^2)g[1]^2 + (y^1)g[2], output of GCBellPol( c(2),1 ), when
# g[1]=3 and g[2]=4, that is 9(y^2) + 4(y)
#
e_GCBellPol( c(2),1,,c(3,4) )
#
# OR (same output)
#
e_GCBellPol( c(2),1,"g[1]=3,g[2]=4" )

# Check the assignments setting the boolean variable equals TRUE, that is g[1]=3 and g[2]=4
e_GCBellPol( c(2),1,,c(3,4),TRUE )

# The numerical value of (y^2)g[1]^2 + (y^1)g[2], output of GCBellPol( c(2),1 ), when
# g[1]=3 and g[2]=4 and y=7, that is 469
#
e_GCBellPol( c(2),1,c(7),c(3,4) )
#
# OR (same output)
#
e_GCBellPol( c(2),1,"y=7, g[1]=3,g[2]=4" )
```

```
# Check the assignments setting the boolean variable equals TRUE, that is g[1]=3 and g[2]=4
# and y=7
e_GCBellPol( c(2),1,c(7),c(3,4),TRUE )

#-----------------------------------------------------------------------------#
# Evaluation of the generalized complete Bell polynomial indexed by (2,1)
#-----------------------------------------------------------------------------#
#
# The polynomial 2(y^2)g[1,1]g[1,0] + (y^3)g[1,0]^2g[0,1] + (y)g[2,1] + (y^2)g[2,0]
# g[0,1], output of GCBellPol( c(2,1),1 ), when g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4,
# g[2,1]=5, that is 16(y^2) + 4(y^3) + 5(y)
#
e_GCBellPol(c(2,1),1,,c(1:5))
#
# OR (same output)
#
e_GCBellPol(c(2,1),1,,c(1,2,3,4,5))
#
# OR (same output)
#
e_GCBellPol( c(2,1),1,"g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5" )

# Check the assignments setting the boolean variable equals TRUE, that is
# g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5
e_GCBellPol( c(2,1),1,,c(1:5), TRUE )

# The numerical value of 2(y^2)g[1,1]g[1,0] + (y^3)g[1,0]^2g[0,1] + (y)g[2,1] + (y^2)g[2,0]
# g[0,1], output of GCBellPol( c(2,1),1 ) when g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4,
# g[2,1]=5 and y=7, that is 2191
#
e_GCBellPol( c(2,1),1,c(7),c(1:5) )
#
# OR (same output)
#
e_GCBellPol( c(2,1),1,"y=7, g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5" )

# Check the assignments setting the boolean variable equals TRUE, that is
# g[0,1]=1, g[1,0]=2, g[1,1]=3, g[2,0]=4, g[2,1]=5, y=7
e_GCBellPol( c(2,1),1,c(7),c(1:5) )

#-----------------------------------------------------------------------------#
# Evaluation of the generalized complete Bell Polynomial indexed by (1,1)
#-----------------------------------------------------------------------------#

# The polynomial (y1)g1[1,1] + (y1^2)g1[1,0]g1[0,1] + (y2)g2[1,1] + (y2^2)g2[1,0]
# g2[0,1] + (y1)(y2)g1[1,0]g2[0,1] + (y1)(y2)g1[0,1]g2[1,0], output of GCBellPol(c(1,1),2)
# when g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4, g2[1,0]=5, g2[1,1]=6, that is
# 3(y1) + 2(y1^2) + 6(y2) + 20(y2^2) + 13(y1)(y2)
#
e_GCBellPol( c(1,1),2,,c(1:6))
#
# OR (same output)
```

```
#
e_GCBellPol(c(1,1),2,,c(1,2,3,4,5,6))
#
# OR (same output)
#
e_GCBellPol( c(1,1),2,"g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4, g2[1,0]=5, g2[1,1]=6" )


# Check the assignments setting the boolean variable equals TRUE, that is
# g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4, g2[1,0]=5, g2[1,1]=6
e_GCBellPol( c(1,1),2,,c(1:6), TRUE )

# The numerical value of (y1)g1[1,1] + (y1^2)g1[1,0]g1[0,1] + (y2)g2[1,1] + (y2^2)g2[1,0]
# g2[0,1] + (y1)(y2)g1[1,0]g2[0,1] + (y1)(y2)g1[0,1]g2[1,0], output of GCBellPol(c(1,1),2)
# when g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4, g2[1,0]=5, y1=7 and y2=8, that is 2175
e_GCBellPol( c(1,1),2,c(7,8),c(1:6))
#
# OR (same output)
#
cVal<-"y1=7, y2=8, g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4, g2[1,0]=5,g2[1,1]=6"
e_GCBellPol(c(1,1),2,cVal)

# To recover which coefficients and variables are involved in the generalized complete
# Bell polynomial, run the function without any assignment.
# The error message prints which coefficients and variables are involved, that is
# Error in e_GCBellPol(c(1, 1), 2) :
#   The third parameter must contain the 2 values of y: y1 y2.
#   The fourth parameter must contain the 6 values of g:
#       g1[0,1] g1[1,0] g1[1,1] g2[0,1] g2[1,0] g2[1,1]

# Syntax to correctly assign values to the coefficients and the variables.
# 1) run e_GCBellPol(c(1, 1), 2) and get the errors with the indication of the involved
#    coefficients and variables, that is
#      The third parameter must contain the 2 values of y: y1 y2
#      The fourth parameter must contain the 6 values of g:
#          g1[0,1] g1[1,0] g1[1,1] g2[0,1] g2[1,0] g2[1,1]
# 2) initialize g1[0,1] g1[1,0] g1[1,1] g2[0,1] g2[1,0] g2[1,1] with the first 6 natural
#    numbers and do the same for y1 and y2, that is
#          e_GCBellPol(c(1,1),2, c(1,2), c(1,2,3,4,5,6), TRUE)
# 3) trought the boolean value TRUE, recover the string y1=1, y2=1, g1[0,1]=1, g1[1,0]=2,
#    g1[1,1]=3, g2[0,1]=4, g2[1,0]=5, g2[1,1]=6
# 4) copy and past the string as input in place of "..." when run
#    e_GCBellPol(c(1,1),2,"...")
# 5) change the assignments if necessary
cVal<-"y1=10,y2=11,g1[0,1]=1.1,g1[1,0]=-2,g1[1,1]=3.2,g2[0,1]=-4,g2[1,0]=10,g2[1,1]=6"
e_GCBellPol(c(1,1), 2,cVal)
```

---

e_MFB *Evaluation of Faa di Bruno's formula*

Description

The function evaluates the Faa di Bruno's formula obtained as output of the function MFB, when values are assigned to the coefficients of the exponential formal power series f() and g1(),...,gn() in f[g1(),...,gn()].

Usage

```
e_MFB(pv = c(), pn = 0, pf = c(), pg = c(), b = FALSE)
```

Arguments

pv	vector of integers, the subscript of Faa di Bruno's formula
pn	integer, number of the inner formal power series "g" in the composition
pf	vector, the values to be assigned to the coefficients of the outer formal power series f() or the string with the direct assignments to the coefficients of both f() and g()
pg	vector, the values to be assigned to the coefficients of the inner formal power series "g" [Optional if pf is the string with the assignments to the coefficients of f() and g()]
b	boolean

Details

The output of the function MFB is a coefficient of the composition of the exponential formal power series f() and g() in the cases univariate f univariate g, univariate f multivariate g and multivariate f multivariates g1,...,gn. The function e_MFB evaluates this coefficient by assigning values to the coefficients of "f" and "g". These values are passed to e_MFB trough the third and the fourth input parameters. There is one more input boolean parameter: when it is TRUE, the function prints the list of all the assignments. See the examples for more details on how to use this boolean parameter when the form of "f" and "g" becomes more complex.

Value

numerical	evaluation of the output of the function MFB

Warning

The value of the first parameter is the same as the mkmSet function, i.e. the number of considered blocks.

Note

This function is called from the function MFB of the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. Vol. 14(2), 440-468. (download from http://www.elviradinardo.it/lavori1.html)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis Vol. 52(11), 4909-4922, (download from http://www.elviradinardo.it/lavori1.html)

E. Di Nardo, G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

See Also

mkmSet, MFB

Examples

```
# The numerical value of f[1]g[1,1] + f[2]g[1,0]g[0,1], that is the coefficient of x1x2 in
# f(g1(x1,x2),g2(x1,x2))) output of MFB(c(1,1),1) when
# f[1] = 5 and f[2] = 10
# g[0,1]=3,  g[1,0]=6,  g[1,1]=9
e_MFB(c(1,1),1, c(5,10), c(3,6,9))

# Same as the previous example, with a string of assignments as third input parameter
e_MFB(c(1,1),1, "f[1]=5, f[2]=10, g[0,1]=3, g[1,0]=6, g[1,1]=9")


# Use the boolean parameter to verify the assignments to the coefficients of "f" and "g",
# that is f[1]=5, f[2]=10, g[0,1]=3, g[1,0]=6, g[1,1]=9
e_MFB(c(1,1),1, c(5,10), c(3,6,9), TRUE)

# To recover which coefficients are involved in the Faa di Bruno's formula, run the function
# without any assignment. The error message recalls which coefficients are necessary, that is
# e_MFB(c(1,1),1)
# Error in e_MFB(c(1, 1), 1) :
#   The third parameter must contain the 2 values of f: f[1] f[2].
#   The fourth parameter must contain the 3 values of g: g[0,1] g[1,0] g[1,1]

# Syntax to correctly assign values to the coefficients of "f" and "g" when the functions
# become more complex.
# 1) run e_MFB(c(1,1),2) and get the errors with the indication of the involved coefficients
#    of "f" and "g", that is
#      The third parameter must contain the 5 values of f:
#                      f[0,1] f[0,2] f[1,0] f[1,1] f[2,0]
#      The fourth parameter must contain the 6 values of g:
#                      g1[0,1] g1[1,0] g1[1,1] g2[0,1] g2[1,0] g2[1,1]"
# 2) initialize f[0,1] f[0,2] f[1,0] f[1,1] f[2,0] with the first 5 natural numbers and do
#    the same for g1[0,1] g1[1,0] g1[1,1] g2[0,1] g2[1,0] g2[1,1], that is
#    e_MFB(c(1,1),2, c(1:5), c(1:6), TRUE)
# 3) trought the boolean value TRUE, recover the string f[0,1]=1, f[0,2]=2, f[1,0]=3, f[1,1]=4,
```

```
#    f[2,0]=5, g1[0,1]=1, g1[1,0]=2, g1[1,1]=3, g2[0,1]=4, g2[1,0]=5, g2[1,1]=6
# 4) copy and past the string as input in place of " ... " when run e_MFB(c(1,1),1," ... ")
# 5) change the assignments if necessary
cfVal<-"f[0,1]=2, f[0,2]=5, f[1,0]=13, f[1,1]=-4, f[2,0]=0"
cgVal<-"g1[0,1]=-2.1, g1[1,0]=2,g1[1,1]=3.1, g2[0,1]=5, g2[1,0]=0, g2[1,1]=6.1"
cVal<-paste0(cfVal,",",cgVal)
e_MFB(c(1,1),2,cVal)
```

`ff` *Falling factorial*

Description

Computes the descending (falling) factorial of a positive integer n with respect to a positive integer k less or equal to n.

Usage

```
ff( n, k )
```

Arguments

`n`	integer
`k`	integer

Details

ff(n, k) returns n(n-1)(n-2).....(n-k+1).

Value

`integer`	descending factorial

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

See Also

`mkmSet`, `umSet`, `countP`, `nStirling2`, `intPart`, `mCoeff`

Examples

```
# Return 6*5*4 = 120
ff(6,3)
```

GCBellPol *Generalized Complete Bell Polynomial*

Description

The function returns generalized complete Bell polynomials, that are coefficients of exp(y1 g1(z1,...,zm) + ... + yn gn(z1,...,zm)), where y1,...,yn are the variables. The input vector of integers identifies the subscripts of the polynomial coefficients.

Usage

```
GCBellPol(n = 0, m = 1, b = FALSE)
```

Arguments

n	vector of integers, the subscript of the polynomial, corresponding to the powers of z1, z2, ..., zm
m	integer, the number of z's variables
b	boolean, TRUE if g1=g2=...=gn=g

Details

The multivariate Faa di Bruno's formula output of the MFB function gives the coefficients of the multivariate exponential power series obtained from the composition of the multivariate exponential power series f(t1,...,tn) with ti=gi(z1,...,zm) for each i from 1 to n. Choose f(t1,...,tn)=exp(y1 t1 + ... + yn tn). The coefficients obtained through multivariate Faa di Bruno's formula are the generalized complete Bell polynomials. In particular, the function GCBellPol gives the expression of these polynomials when n=1 or n> 1 with g1=...=gn=g or n>1 with g1, ..., gn all different. Among the various applications, we point out the cumulant polynomial sequences. Cumulant polynomials allow us to compute moments and cumulants of multivariate Levy processes and subordinated multivariate Levy processes, multivariate compound Poisson processes with applications in photocounting, multivariable Sheffer polynomial sequences, see the referred papers. See the function e_GCBellPol for evaluating the generalized complete Bell polynomials by assigning numerical values to the variables and their coefficients.

Value

string	expression of the generalized complete Bell polynomial

Warning

The value of the first parameter is the same as the mkmSet function, i.e. the number of blocks considered.

Note

It calls the function MFB in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Constantine G. M., Savits T. H. (1996) A Multivariate Faa Di Bruno Formula With Applications. Trans. Amer. Math. Soc. 348(2), 503–520.

E. Di Nardo E. (2016) On multivariable cumulant polynomial sequence with applications. Jour. Algebraic Statistics 7(1), 72-89. (download from http://arxiv.org/abs/1606.01004)

E. Di Nardo, G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

E. Di Nardo, M. Marena, P. Semeraro (2020) On non-linear dependence of multivariate subordinated Levy processes. In press Stat. Prob. Letters (download from https://arxiv.org/abs/2004.03933)

See Also

mkmSet, MFB, e_GCBellPol

Examples

```
# Return the generalized complete Bell Polynomial for n=1, m=1 and g1=g,
# that is (y^2)g[1]^2 + (y)g[2]
#
GCBellPol( c(2),1 )

# Return the generalized complete Bell Polynomial for n=1, m=2 and g1=g,
# corresponding to the cumulant polinomial indexed by (2,1), that is
# 2(y^2)g[1,0]g[1,1] + (y^3)g[0,1]g[1,0]^2 + (y)g[2,1] + (y^2)g[0,1]g[2,0]
#
GCBellPol( c(2,1),1 )


# Return the generalized complete Bell Polynomial for n=2, m=2 and g1=g2=g,
# corresponding to the cumulant polinomial in y1+y2 indexed by (1,1), that is
# (y1)g[1,1] + (y1^2)g[0,1]g[1,0] + (y2)g[1,1] + (y2^2)g[0,1]g[1,0] + 2(y1)(y2)g[0,1]g[1,0]
#
GCBellPol( c(1,1),2, TRUE )

# Return the generalized complete Bell Polynomial for n=2, m=2 and g1 different from g2,
# that is (y1)g1[1,1] + (y1^2)g1[1,0]g1[0,1] + (y2)g2[1,1] + (y2^2)g2[1,0]g2[0,1] +
# (y1)(y2)g1[1,0]g2[0,1] + (y1)(y2)g1[0,1]g2[1,0]
#
GCBellPol( c(1,1),2 )
```

gpPart *General partition polynomial*

Description

The function returns general partition polynomials.

Usage

```
gpPart(n = 0)
```

Arguments

n	integer

Details

Faa di Bruno's formula gives the coefficients of the exponential formal power series obtained from the composition f(g()) of exponential formal power series. General partition polynomials in the variables y1,...,yn are recovered from Faa di Bruno's formula obtained from the MFB function in the case "composition of univariate f with univariate g" by setting f[i]=ai and g[i]=yi, for each i from 1 to n.

Value

string	expression of the general partition polynomial

Warning

The value of the first parameter is the same as the MFB function in the univariate with univariate case composition.

Note

This function calls the function MFB in the package kStatistics

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

C.A. Charalambides (2002) Enumerative Combinatoris, Chapman & Haii/CRC.

E. Di Nardo, G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

See Also

MFB

Examples

```
# Return the general partition polynomial G[2], that is a2(y1^2) + a1(y2)
gpPart(2)

# Return the general partition polynomial G[5], that is a5(y1^5) + 10a4(y1^3)(y2) + 15a3(y1)(y2^2)
# + 10a3(y1^2)(y3) + 10a2(y2)(y3) + 5a2(y1)(y4) + a1(y5)
gpPart(5)
```

---

intPart *Integer partitions*

Description

Generate all possible (unique) ways to represent a positive integer n as sum of positive integers.

Usage

```
intPart(n)
```

Arguments

n	integer

Details

A partition of an integer n is a sequence of weakly increasing integers such that their sum returns n. The function generates all the partitions of a given integer in sorted order.

Value

list	all partitions of n

Note

Called by the function mkmSet in the package kStatistics

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Nijenhuis A. and Wilf. H., Combinatorial Algorithms for Computers and Calculators. Academic Press, Orlando FL, II edition, 1978.

See Also

mkmSet, umSet, mCoeff, nStirling2, countP, df

Examples

```
# Return the partition of the integer 3, that is
# [1,1,1],[1,2],[3]
intPart(3)

# Return the partition of the integer 4, that is
# [1,1,1,1],[1,1,2],[1,3],[2,2],[4]
intPart(4)
```

---

list2m *List To Multiset*

Description

The function returns a multiset representation of a vector or a list, in increasing order.

Usage

```
list2m( v )
```

Arguments

v	single vector or list of vectors

Details

Given a list as input, the function returns a structure as [[e1,e2,...], m1], [[f1,f2,...], m2],... where m1, m2,... are the instances of c(e1,e2,...), c(f1,f2,...), ... in the input v.

Value

multiset	list of multiset

Note

Called by the function countP in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Knuth, Donald E. (1998). The Art of Computer Programming. (3rd ed.). Addison Wesley.

See Also

list2Set, m2Set, countP

Examples

```
# Return the multiset [[1],3], [[2],1] from the input vector (1,2,1,1)
list2m(c(1,2,1,1 ))

# Return the multiset [[1,2],2], [[2,3],1] from the input list (c(1,2),c(2,3),c(1,2))
list2m(list(c(1,2),c(2,3),c(1,2)))
```

---

list2Set *List To Set*

Description

Given a list, delete the instances of an element in the list, leaving the order inalterated.

Usage

```
list2Set(v)
```

Arguments

v	single vector or list of vectors

Value

set	sequence of distinct elements

Note

Called by the function list2m in the package kStatistics

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

See Also

list2m, m2Set

Examples

```
# Return c(1,2,3,5,6)
list2Set(c(1,2,3,1,2,5,6))

# Return the list (c(1,2),c(10,11),c(7,8))
list2Set( list(c(1,2),c(1,2),c(10,11),c(1,2),c(7,8)))
```

---

m2Set *Multiset To Set*

Description

Transform a multiset in a set.

Usage

```
m2Set( v )
```

Arguments

v	multiset structure of type [[e1,e2,...], m1], [[f1,f2,...], m2],... with m1, m2 multiplicities

Details

Consider the multiset [a,a,b]. The subdivisions are [[[a,b],[a]],2],[[[a],[a],[b]],1], [[a,a,b],1], [[a,a],[b],1]. m2Set(c(2,1)) deletes block repetitions, that is transforms the given list in [a,b],[a],[b],[a,a,b],[a,a], according to the order given in the input. See also the examples.

Value

set	sequence of distinct elements

Note

Called by the functions nKM, nPM in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

See Also

list2m, list2Set

Examples

```
M1 <- mkmSet(c(2,1))
#
# M1 is the multiset of subdivisions
# [
#   [[1,1],[1,0]], 2],
#   [[1,0],[1,0],[0,1]], 1),
#   [[2,1]], 1],
#   [[2,0],[0,1]], 1]
# ]
#

# Return the following set:  [[1,1],[1,0],[0,1],[2,1],[2,0]]
m2Set( M1 )
```

---

mCoeff *Multiplicity in a multiset*

Description

Return the instances of an element in a multiset.

Usage

```
mCoeff( v, L )
```

Arguments

v	vector to be searched in a list
L	list of two-dimensional vectors: in the first there is a vector and in the second its instances

Details

This function is useful in the construction of kStatistics.

Compute how many times an element is in a multiset, that is its multiplicity.

Value

float	multiplicity of the researched item

Note

Called by the functions nPS, nKM, nPM in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

See Also

mkmSet, umSet, countP, nStirling2, intPart, df

Examples

```
# Return [[1,1,1],1], [[1,2],3], [[3],1]
L1 <- mkmSet(c(3))

# Return 3 that is the istance of [1,2] in L1
mCoeff( c(1,2), L1)
```

MFB *Faa di Bruno's formula*

Description

The routine generates the coefficients of the composition of (exponential) formal power series provided by the univariate and multivariate Faa di Bruno's formula indexed by the integers i1,i2,....

Usage

```
MFB(v = c(), n = 0)
```

Arguments

v	vector of integers, the subscript of the coefficient
n	integer, the number of inner functions "g"

Details

The function computes the coefficients of the composition of (exponential) formal power series:
1) Univariate with Univariate Case: f(g(z),z,z...)
2) Univariate with Multivariate Case: f(g(z1,z2,...),z1,z1,..,z2,z2...,..)
3) Multivariate with Multivariate Case: f(g1(z1,z2,..),g2(z1,z2,..),..), z1,z1,...,z2,z2,..,..)
If i1 is the number of z1, i2 is the number of z2 and so on, then (i1,i2, ...) is the index of the output coefficient corresponding to the product z1^i1 z2^i2 ....
Note that this coefficient is also related to the (partial) derivative of order (i1,i2, ...) of the composition of (exponential) formal power series evaluated at z1=0, z2=0, and so on. See the function e_MFB, for evaluating these coefficients by assigning numerical values to the coefficients of f() and to the coefficients of g1(),g2(),...

Value

`string`	the expression of Faa di Bruno's formula

Warning

The value of the first parameter is the same as the mkmSet function, i.e. the number of assigned blocks.

Note

Called by the function `e_MFB` in the package `kStatistics`. The routine uses the `mkmSet` function in the same package.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo E., G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

See Also

`mkmSet`, `e_MFB`

Examples

```
#---------------------------------------#
# Univariate f with Univariate g        #
#---------------------------------------#

# The coefficient of z^2 in f(g(z)), that is f[2]g[1]^2 + f[1]g[2], where
# f[1] is the coefficient of t in f(t) with t=g(z)
# f[2] is the coefficient of t^2 in f(t) with  t=g(z)
# g[1] is the coefficient of z in g(z)
# g[2] is the coefficient of z^2 in g(z)
#
MFB( c(2), 1 )

# The coefficient of z^3 in f(g(z)), that is f[3]g[1]^3 + 3f[2]g[1]g[2] + f[1]g[3]
#
MFB( c(3), 1 )

#---------------------------------------#
#  Univariate f with Multivariate g     #
#---------------------------------------#

# The coefficient of z1 z2 in f(g(z1,z2)), that is f[1]g[1,1] + f[2]g[1,0]g[0,1]
# where
# f[1]   is the coefficient of t in f(t) with t=g(z1,z2)
# f[2]   is the coefficient of t^2 in f(t) with t=g(z1,z2)
# g[1,0] is the coefficient of z1 in g(z1,z2)
# g[0,1] is the coefficient of z2 in g(z1,z2)
# g[1,1] is the coefficient of z1 z2 in g(z1,z2)
#
MFB( c(1,1), 1 )

# The coefficient of z1^2 z2 in f(g(z1,z2))
#
MFB( c(2,1), 1 )

# The coefficient of z1 z2 z3 in f(g(z1,z2,z3))
#
MFB( c(1,1,1), 1 )

#-------------------------------------------------#
#  Multivariate f with Multivariate g1, g2, ...   #
#-------------------------------------------------#

# The coefficient of z in f(g1(z),g2(z)), that is  f[1,0]g1[1] + f[0,1]g2[1] where
# f[1,0] is the coefficient of t1 in f(t1,t2) with t1=g1(z) and t2=g2(z)
# f[0,1] is the coefficient of t2 in f(t1,t2) with t1=g1(z) and t2=g2(z)
# g1[1]  is the coefficient of z of g1(z)
# g2[1]  is the coefficient of z of g2(z)
MFB( c(1), 2 )

# The coefficient of z1 z2 in f(g1(z1,z2),g2(z1,z2)), that is
```

```
# f[1,0]g1[1,1] + f[2,0]g1[1,0]g1[0,1] + f[0,1]g2[1,1] + f[0,2]g2[1,0]g2[0,1] +
# f[1,1]g1[1,0]g2[0,1] + f[1,1]g1[0,1]g2[1,0] where
# f[1,0]  is the coefficient of t1 in f(t1,t2) with t1=g1(z1,z2) and t2=g2(z1,z2)
# f[0,1]  is the coefficient of t2 in f(t1,t2) with t1=g1(z1,z2) and t2=g2(z1,z2)
# g1[1,1] is the coefficient of z1z2 in g1(z1,z2)
# g1[1,0] is the coefficient of z1 in g1(z1,z2)
# g1[0,1] is the coefficient of z2 in g1(z1,z2)
# g2[1,1] is the coefficient of z1 z2 in g2(z1,z2)
# g2[1,0] is the coefficient of z1 in g2(z1,z2)
# g2[0,1] is the coefficient of z2 in g1(z1,z2)
MFB( c(1,1), 2 )

# The coefficient of z1 in f(g1(z1,z2),g2(z1,z2),g3(z1,z2))
MFB( c(1,0), 3 )

# The coefficient of z1 z2 in f(g1(z1,z2),g2(z1,z2),g3(z1,z2))
MFB( c(1,1), 3 )

# The coefficient of z1^2 z2 in f(g1(z1,z2),g2(z1,z2))
MFB( c(2,1), 2 )

# The coefficient of z1^2 z2 in f(g1(z1,z2),g2(z1,z2),g3(z1,z2))
MFB( c(2,1), 3 )

# The previous result expressed in a compact form
for (m in unlist(strsplit( MFB(c(2,1),3), " + ", fixed=TRUE)) ) cat( m,"\n" )

# The coefficient of z1 z2 z3 in f(g1(z1,z2,z3),g2(z1,z2,z3),g3(z1,z2,z3))
MFB( c(1,1,1), 3 )

# The previous result expressed in a compact form
for (m in unlist(strsplit( MFB(c(1,1,1),3), " + ", fixed=TRUE)) ) cat( m,"\n" )
```

`MFB2Set` *Convert the output of the MFB function to a vector*

Description

Secondary function useful for manipulating the result of the MFB function

Usage

```
MFB2Set(sExpr)
```

Arguments

`sExpr`	MFB output

Value

set	output as set

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

See Also

MFB Set2expr

Examples

```
# MFB(c(3),1) generate f[3]g[1]^3 + 3f[2]g[1]g[2] + f[1]g[3]
# Convert the output of the MFB(c(3),1) to a vector using
# MFB2Set(MFB(c(5),1)). It generates the following result:
# "1" "1" "f" "3" "1"
# "1" "1" "g" "1" "3"
# "2" "3" "f" "2" "1"
# "2" "1" "g" "1" "1"
# "2" "1" "g" "2" "1"
# "3" "1" "f" "1" "1"
# "3" "1" "g" "3" "1"

MFB2Set(MFB(c(3),1))
```

---

mkmSet *Subdivisions of a Multiset*

Description

Generate all the subdivisions of a multiset, reducing the overall computational complexity by using integer partitions.

Usage

```
mkmSet(vPar)
```

Arguments

vPar	Number of blocks to consider or vector of numbers of blocks to consider

Value

list	two-dimensional list: in the first there is the partition type, while in the second there is its multiplicity

Note

Called by the functions nKS, nKM, nPS, nPM in the package kStatistics.

Note

This is the most important function for calculating kStatistics and Polykays.

The subdivisions of a multiset are obtained as follows: assume all distinct the elements of the multiset, determine all set partitions and then replace each element in each block with the original one. For example, the partitions of the set [a1, a2, a3] are [[a1], [a2], [a3]], [[a1,a2], [a3]], [[a1,a3], [a2]], [[a2,a3], [a1]], [[a1,a2,a3]]. Replace a1<-a, a2<-a, a3<-a. The label does not matter. The subdivisions are [[a], [a], [a]], [[a,a], [a]], [[a,a], [a]], [[a,a], [a]], [[a,a,a]]. A short notation for the output is [[1,1,1],1], [[1,2],3], [[3],1], with the multiplicity of each block. Note that [1,1,1], [1,2], [3] are the partitions of the integer 3.

In the list, the sum of all multiplicities is the Bell Number. Moreover the sum of multiplicities for the subdivisions having the same block numbers is the Stirling number of the second kind. For example, *mkmSet(4)* returns [[1,1,1,1],1], [[1,1,2],6] ,[[1,3],4], [[2,2],3], [[4],1], where in the second position there is the multiplicity of the subdivision indicated in the first position. Observe that 3 + 4 = 7 = S(4,2), where 4 is the input integer, 2 is the length of the blocks [1,3] and [2,2] and S(i,j) denotes the Stirling numbers of the second kind, see function nStirling2. In the same way 1 = S(4,4) = S(4,1) and 6 = S(4,3). Note that 7 + 1 + 1 + 6 = 15 = Bell(4) which is the number of partitions of 4.

When in the multiset there are two or more different elements, the resulting list is different. For example, for the multiset [a,a,b], the calling is *mkmSet(c(2,1))*. In the output list, [[1,1],[1,0],2] denotes the subdivision [[a,b],[a]] with multiplicities 2.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

See Also

umSet, mCoeff, countP, nStirling2, intPart, df

Examples

```
# Return [ [[1,1,1],1], [[1,2],3], [[3],1]  ]
# The support of the multiset has all the same eleents
mkmSet(3)
```

```
# Return [ [[1,1],[1,0],2], [[1,0],[1,0],[0,1],1],[[2,0],[0,1],1], [[2,1],1] ]
# The support of the multiset has 2 elements with instances respectively 2 and 1.
mkmSet(c(2,1))
```

---

mkT *Scomposition of a vector of non-negative integers*

Description

Given a vector v of non-negative integers, the routine finds all the lists (v1, ..., vn) of non-negative integer vectors, with the same lenght of v and such that v=v1+...+vn.

Usage

```
mkT(v = c(), n = 0)
```

Arguments

v	vector of integers
n	integer, number of addends

Details

From an input vector v of non-negative integers, the routine generates all the lists of n vectors (v1, ..., vn) of non-negative integers, including the zero vector, having the same lenght of v and such that their sum gives v. Note that two lists are different if they contain the same vectors but permutated.

Value

list	the list of n vectors (v1, ..., vn)

Warning

The vector of the first parameter must be not empty and must contain all non-negative values. The second parameter must be a positive integer.

Note

Called by the function MFB in the package kStatistics. It uses the mkmSet function in the same package.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Di Nardo E., Guarino G., Senato D. (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

See Also

mkmSet, MFB

Examples

```
# Return the scompositions of the vector (1,1) in 2 vectors of 2 non-negative integers such that
# their sum is (1,1), that is
# ([1,1],[0,0]) - ([0,0],[1,1]) - ([1,0],[0,1]) - ([0,1],[1,0])
mkT(c(1,1),2)

# Return the scompositions of the vector (1,0,1) in 2 vectors of 3 non-negative integers such that
# their sum gives (1,0,1), that is
# ([1,0,1],[0,0,0]) - ([0,0,0],[1,0,1]) - ([1,0,0],[0,0,1]) - ([0,0,1],[1,0,0]).
# Note that the second value in each resulting vector is always zero.
mkT(c(1,0,1),2)

# Return the scompositions of the vector (1,1,1) in 3 vectors of 3 non-negative integers such that
# their sum gives (1,1,1). The result is in a compact form expression.
for (m in mkT(c(1,1,1),3)) {for (n in m) cat(n," - "); cat("\n"); }
```

---

mom2cum *Moments in terms of cumulants*

Description

Compute simple and multivariate moments in terms of simple and multivariate cumulants.

Usage

```
mom2cum(n = 0)
```

Arguments

n	integer or vector of integers

Details

Faa di Bruno's formula (the MFB function) gives the coefficients of the exponential formal power series obtained from the composition f(g()) of exponential formal power series. Simple moments are expressed in terms of cumulants using the Faa di Bruno's formula obtained from the MFB function in the case "composition of univariate f with univariate g" with f[i]=1, g[i]=k[i] for each i from 1 to n and k[i] cumulants. Multivariate moments are expressed in terms of multivariate cumulants using the Faa di Bruno's formula obtained from the MFB function in the case "composition of univariate f with multivariate g" whose coefficients are the multivariate cumulants.

Value

`string`	the expression of moments in terms of cumulants

Warning

The value of the first parameter is the same as the MFB function in the univariate with univariate case composition and in the univariate with multivariate case composition.

Note

This function calls the `MFB` function in the package `kStatistics`.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo E., G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

See Also

`MFB`

Examples

```
# Simple moment m[5] in terms of the cumulants k[1],...,k[5].
mom2cum(5)

# Multivariate moment m[3,1] in terms of the multivariate cumulants k[i,j] for
# i=0,1,2,3 and j=0,1.
mom2cum(c(3,1))
```

mpCart *Pairing two multisets*

Description

Given two multisets, the function generates a new multiset whose elements are the pairing of the corresponding ones in the input multisets.

Usage

```
mpCart( M1,M2 )
```

Arguments

M1	list of vectors
M2	list of vectors

Details

Two lists are given in input. Each list might contains more lists of two vectors: the first vector is a multiset, the second vector is a multiplicative factor (for example its multiplicity if the multiset is a subdivision). The function generates a new multiset whose elements are the pairing of the corresponding ones in the two input lists, with a multiplicative factor which is just the product of the corresponding multiplicative factors.

Value

list	list with the paired multiset

Note

Called by the function nPM in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2009) A new method for fast computing unbiased estimators of cumulants. Statistics and Computing, 19, 155-165. (download from https://arxiv.org/abs/0807.5008)

See Also

pCart

Examples

```
A <- list(  list( list(c(1),c(2) ),c(-1)), list(list(c(3)),c(1))  )
# where -1 is the multiplicative factor of list(c(1),c(2) ) and 1 is the
# multiplicative factor of list(c(3))
B<-list( list( list(c(5)),c(7)))
# where 7 is the multiplicative factor of list(c(5))

mpCart(A,B)
# generates [[[1],[2],[5]], -7] , [[[3],[5]], 7]

A <- list( list( list( c(1,0),c(1,0) ), c(-1)),  list( list( c(2,0)), c(1) ))
B <- list( list( list( c(1,0)), c(1)) )
mpCart(A,B)
# generates  [[[1,0],[1,0],[1,0]], -1], [[[2,0],[1,0]],1]
```

---

nKM *Multivariate K-Statistics*

Description

Given a sample data, compute an estimate of the joint (multivariate) cumulant of a given order for the population distribution.

Usage

```
nKM( v, V)
```

Arguments

v	vector of integers
V	vector of sample data

Details

For a sample of i.i.d. random vectors, multivariate kstatistics are unbiased estimators of the population joint cumulants and are expressed in terms of the power sum symmetric polynomials in the random vectors of the sample. Thus, for the given multivariate sample data, nKM(c(r, s, ...), data) computes an estimate of the joint cumulant k[r, s, ...] of the population distribution.

Value

`float` the value of the multivariate kstatistics of order v

Warning

The size of each value of the data vector must coincide with the length of the first parameter

Note

Called by the master function `nPolyk` in the package `kStatistics`

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2009) A new method for fast computing unbiased estimators of cumulants. Statistics and Computing, 19, 155-165. (download from https://arxiv.org/abs/0807.5008)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

See Also

`nKS`, `nPS`, `nPM`

Examples

```
# Data assignment
data1<-list(c(5.31,11.16),c(3.26,3.26),c(2.35,2.35),c(8.32,14.34),c(13.48,49.45),
c(6.25,15.05),c(7.01,7.01),c(8.52,8.52),c(0.45,0.45),c(12.08,12.08),c(19.39,10.42))

# Return an estimate of the joint cumulant k[2,1]
nKM(c(2,1),data1)

# Data assignment
data2<-list(c(5.31,11.16,4.23),c(3.26,3.26,4.10),c(2.35,2.35,2.27),
c(4.31,10.16,6.45),c(3.1,2.3,3.2),c(3.20, 2.31, 7.3))

# Return an estimate of the joint cumulant k[2,2,2]
```

```
nKM(c(2,2,2),data2)

# Data assignment
data3<-list(c(5.31,11.16,4.23,4.22),c(3.26,3.26,4.10,4.9),c(2.35,2.35,2.27,2.26),
c(4.31,10.16,6.45,6.44),c(3.1,2.3,3.2,3.1),c(3.20, 2.31, 7.3,7.2))

# Return an estimate of the joint cumulant k[2,1,1,1]
nKM(c(2,1,1,1),data3)
```

---

nKS *Simple K-Statistics*

Description

Given a sample data, compute an estimate of the cumulant of a given order for the population distribution.

Usage

```
nKS( v, V )
```

Arguments

v	integer or one-dimensional vector
V	vector of sample data

Details

For a sample of i.i.d. random variables, kstatistics are unbiased estimators of the population cumulants and are expressed in terms of the power sum symmetric polynomials in the random variables of the sample. Thus, for the given sample data, nKS(n,data) or nKS(c(n),data) computes an estimate of the n-th cumulant of the population distribution.

Value

float	the value of the kstatistics of order v

Note

Called by the master function nPolyk in the package kStatistics

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2009) A new method for fast computing unbiased estimators of cumulants. Statistics and Computing, 19, 155-165. (download from https://arxiv.org/abs/0807.5008)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

See Also

nPolyk, nKM, nPS, nPM

Examples

```
# Data assignment
data<-c(16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08,
19.43,8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83,
16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11)

# Return an estimate of the cumulant of order 7
nKS(7, data)

# Return an estimate of the cumulant of order 1, that is the mean (R command: mean(data))
nKS(1, data)

# Return an estimate of the cumulant of order 2, that is the variance (R command: var(data))
nKS(2, data)

# Return an estimate of the skewness (R command: skewnes(data) in the library "moments")
nKS(3, data)/sqrt(nKS(2, data))^3

# Return an estimate of the kurtosis (R command: kurtosis(data) in the library "moments")
nKS(4, data)/nKS(2, data)^2 + 3
```

---

nPerm *Complete enumeration of all possible permutations*

Description

The function returns all possible different permutations of objects in a list or in a vector.

Usage

```
nPerm(L = c())
```

Arguments

L	List/Vector

Details

In order to manage permutations of numbers or vectors, the standard permutation process is applied.

Value

list	all permutations of L

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Charalambides C. A. (2002) Enumerative Combinatoris, Chapman & Haii/CRC.

Examples

```
# permutations of 1,2,3
nPerm( c(1,2,3) );

# permutations of 1,2,1 (two elements are equal)
nPerm( c(1,2,1) );

# permutations of the words "Alice", "Box","Jack"
nPerm( c("Alice", "Box","Jack") );

# permutations of the vectors c(0,1), c(2,3), c(7,3)
nPerm( list(c(0,1), c(2,3), c(7,3)) );
```

---

nPM *Multivariate Polykays*

Description

Given a multivariate sample data, the function returns an estimate of a product of population joint cumulants using multivariate polykays.

Usage

```
nPM( v, V )
```

Arguments

v	vector of integers
V	vector of multivariate sample data

Details

Multivariate polykays are symmetric statistics which generalize polykays. They are unbiased estimators of products of joint cumulants of a distribution and are expressed in terms of the power sum symmetric polynomials in the random vectors of the sample. Thus, for the given multivariate sample data, nPM(list(c(r1, s1, ...), c(r1, s2, ...),..), data) computes an estimate of the product k[r1, s1,....]*k[r2, s2, ...]*... where k[r1, s1,....], k[r2, s2, ...], ... are joint cumulants of the population distribution.

Value

float	estimate of the multivariate polykay of order v

Warning

The size of each value of the data vector must coincide with the length of the first parameter.
The vectors contained in the list must have the same length.

Note

Called by the master function nPolyk in the package kStatistics

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2009) A new method for fast computing unbiased estimators of cumulants. Statistics and Computing, 19, 155-165. (download from https://arxiv.org/abs/0807.5008)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

See Also

nPolyk, nKS, nKM, nPS

Examples

```
# Data assignment
data1<-list(c(5.31,11.16),c(3.26,3.26),c(2.35,2.35),c(8.32,14.34),c(13.48,49.45),
c(6.25,15.05),c(7.01,7.01),c(8.52,8.52),c(0.45,0.45),c(12.08,12.08),c(19.39,10.42))

# Return an estimate of the product k[2,1]*k[1,0]  where k[2,1] and k[1,0] are respectively
# the cross-correlation of order (2,1) and the marginal mean of the population distribution
nPM( list( c(2,1), c(1,0) ), data1)

# Data assignment
data2<-list(c(5.31,11.16,4.23),c(3.26,3.26,4.10),c(2.35,2.35,2.27),
c(4.31,10.16,6.45),c(3.1,2.3,3.2),c(3.20, 2.31, 7.3))

# Return an estimate of the product k[2,0,1]*k[1,1,0] where k[2,0,1] and k[1,1,0]
# are joint cumulants of the population distribution
nPM( list( c(2,0,1), c(1,1,0) ), data2)
```

---

nPolyk *K-Statistics Master function*

Description

The master function executes one of the functions to generate simple kstatistics (nKS), multivariate kstatistics (nKM), simple polykays (nPS), multivariate polykays (nPM) of given orders.

Usage

```
nPolyk( L, data, bhelp=NULL )
```

Arguments

L	vector of orders
data	vector of (multivariate) sample data
bhelp	T=true or F=false

Details

The master function analizes the first and the second parameters and recall one of the functions nKS, nKM, nPS, nPM in the package kStatistics.

Given a sample data:

1) simple k-statistics are computed using nPolyk(c(r), data)) or nPolyk(list(c(r)), data))

2) multivariate k-statistics are computed using nPolyk(c(r, s), data)) or nPolyk(list(c(r, s)), data))

3) simple polykays are computed using nPolyk(list(c(r),c(s)...),data))

4) multivariate polykays are computed using nPolyk(list(c(r1, r2,...),c(s1, s2,...),...),data))

Value

`float` estimate of (joint) cumulant (product)

Note

The sample data vector dimension depends on the first parameter.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2009) A new method for fast computing unbiased estimators of cumulants. Statistics and Computing, 19, 155-165. (download from https://arxiv.org/abs/0807.5008)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

See Also

nKS, nKM, nPS, nPM

Examples

```
# Data assignment
data1<-c(16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08,
19.43,8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83,
16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11)

# display "KS" which indicates the type of subfunction (nKS) called by the master function nPolyk
# KS:[1] -1.44706
nPolyk(c(3),data1, TRUE)
```

```
# Produce
# [1] -1.44706 (without indication of the employed subfunction)
nPolyk(c(3),data1, FALSE)

# Return the estimate of k[2]*k[1] with the indication of the employed subfunction
# PS:[1] 177.4233
nPolyk( list(  c(2), c(1) ),data1,TRUE)

# Data assignment
data2<-list(c(5.31,11.16),c(3.26,3.26),c(2.35,2.35),c(8.32,14.34),c(13.48,49.45),
c(6.25,15.05),c(7.01,7.01),c(8.52,8.52),c(0.45,0.45),c(12.08,12.08),c(19.39,10.42))

# Return the estimate of k[2,1] with the indication of the employed subfunction
# KM:[1] -23.7379
nPolyk(c(2,1),data2,TRUE)

# Return the estimate of k[2,1]*k[1,0] with the indication of the employed subfunction
# PM:[1] 48.43243
nPolyk( list(  c(2,1), c(1,0) ),data2,TRUE)
```

---

nPS *Simple Polykays*

Description

Given a sample data, compute an estimate of a product of population cumulants using simple Polykays (generalized k-statistics).

Usage

```
nPS( v, V )
```

Arguments

v	vector of integers
V	vector of sample data

Details

Products of kstatistics are known as polykays. They are unbiased estimators of products of cumulants of a distribution and are expressed in terms of the power sum symmetric polynomials in the random variables of the sample. Thus, for the given sample data, nPS(c(i,j,...),data) computes an estimate of the product k[i]*k[j]*... where k[i], k[j], ... are cumulants of the population distribution.

Value

float	estimate of the polykay of order v

Note

Called by the master function nPolyk in the package kStatistics

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

E. Di Nardo, G. Guarino, D. Senato (2009) A new method for fast computing unbiased estimators of cumulants. Statistics and Computing, 19, 155-165. (download from https://arxiv.org/abs/0807.5008)

P. McCullagh, J. Kolassa (2009), Scholarpedia, 4(3):4699. http://www.scholarpedia.org/article/Cumulants

See Also

nPolyk, nKS, nKM, nPM

Examples

```
# Data assignment
data<-c(16.34, 10.76, 11.84, 13.55, 15.85, 18.20, 7.51, 10.22, 12.52, 14.68, 16.08,
19.43,8.12, 11.20, 12.95, 14.77, 16.83, 19.80, 8.55, 11.58, 12.10, 15.02, 16.83,
16.98, 19.92, 9.47, 11.68, 13.41, 15.35, 19.11)

# Return the estimate of the product k[2]*k[1], where k[1] and k[2] are respectively
# the mean and the variance of the population distribution
nPS(c(2,1), data)
```

---

nStirling2 *Stirling number of second kind*

Description

Compute the Stirling number of the second kind.

Usage

```
nStirling2( n, k )
```

Arguments

n	integer
k	integer [k <= n]

Details

The Stirling numbers of the second kind, written S(n,k), count the number of ways to partition a set of n labelled objects into k nonempty unlabelled subsets. For example if the set is [a,b,c,d], the partitions in 2 blocks are: [[a], [bcd]], [[b], [acd]], [[c], [abd]], [[d],[abc]] with cardinalities (1,3) and [ab, cd], [ac, bd], [ad, bc] with cardinalities (2,2). Then S(4,2) is 7. S(4,2) is also the number of set partitions of class the partition of 4 in two parts.

Value

integer	Stirling number of the second kind

Note

Called by the functions nKS, nKM in the package kStatistics

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Stanley, R. P. Enumerative combinatorics. Vol.1. II edition. Cambridge Studies in Advanced Mathematics, 49. Cambridge University Press, Cambridge, 2012.

See Also

mkmSet, umSet, mCoeff, intPart, countP, df

Examples

```
# Generate the number of ways to partition a set of 6 objects into 2 non-empty subsets
nStirling2(6,2)
```

pCart *Cartesian product*

Description

The function returns the cartesian product between vectors.

Usage

```
pCart( L )
```

Arguments

L	vectors in a list

Details

The function pairs any element of the first vector with any element of the second vector, iteratively if there are more than two vectors in input. Repetitions are allowed. See examples.

Value

list	list with the cartesian product

Note

Called by the function nPS in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

Knuth, Donald E. (1998). The Art of Computer Programming. (3rd ed.). Addison Wesley.

See Also

mpCart

Examples

```
A <- c(1,2)
B <- c(3,4,5)
# Return the cartesian product [[1,3],[1,4],[1,5],[2,3],[2,4],[2,5]]
pCart( list( A, B) )


L1<-list( c(1,1), c(2))
L2<-list( c(5,5), c(7) )
# Return the Cartesian product [[1,1],[5,5]], [[1,1],[7]], [[2],[5,5]], [[2],[7]]
# assign the value to L3
L3<-pCart ( list(L1, L2) )

# Return the Cartesian product between L3 and [7]
# The result will be [[1,1],[5,5],[7]], [[1,1],[7],[7]], [[2],[5,5],[7]], [[2],[7],[7]]
pCart ( list(L3, c(7)) )
```

---

powS *Power sums*

Description

Given one or more lists of numbers, computes the value of the power sum symmetric polynomial with fixed degrees in one or more sets of variables replaced by a given list of numbers.

Usage

```
powS(vn,lvd)
```

Arguments

vn	vector of powers (degrees)
lvd	list of numbers

Details

Given the lists (x[1],x[2],...), (y[1],y[2],...), (z[1],z[2],...), ... in L and the integers (n,m,j,...) in vn, the function powS(vn ,L) returns (x[1]^n)*(y[1]^m)*(z[1]^j)*...+(x[2]^n)*(y[2]^m)*(z[2]^j)*+...See also the examples.

Value

integer	value of the power sum symmetric polynomial

Note

Called by the functions nKS, nKM, nPS, nPM in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from http://arxiv.org/abs/0806.0129)

Examples

```
# Return 1^3 + 2^3 + 3^3 = 36
powS(c(3), list(c(1),c(2),c(3)))

# Return (1^3 * 4^2) + (2^3 * 5^2) + (3^3 * 6^2) = 1188
powS(c(3,2),list(c(1,4),c(2,5),c(3,6)))
```

---

pPart *Partition polynomial*

Description

The function generates the partition polynomial of degree n, whose coefficients are the number of partitions of n with k parts for k from 1 to n.

Usage

```
pPart(n = 0)
```

Arguments

n	integer

Details

Faa di Bruno's formula gives the coefficients of the exponential formal power series obtained from the composition f(g()) of exponential formal power series. The partition polynomial F[n] of degree n is obtained using the Faa di Bruno's formula obtained from the MFB function in the case "composition of univariate f with univariate g" with f[i]=1/i!, g[i]^k=(i!)^k*k!*y for each i and k from 1 to n. Note the symbolic substitution g[i]^k=(i!)^k*k!*y, where the power of g[i] appears in its definition. This function is an example of application of Faa di Bruno's formula by using the symbolic calculus with two indexes.

Value

string	expression of the partition polynomial of degree n

Warning

The value of the first parameter is the same as the MFB function in the univariate with univariate case composition.

Note

This function calls the MFB function in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo E., G. Guarino, D. Senato (2011) A new algorithm for computing the multivariate Faa di Bruno's formula. Appl. Math. Comp. 217, 6286–6295. (download from http://arxiv.org/abs/1012.6008)

See Also

MFB

Examples

```
# Return the partition polynomial F[5]
pPart(5)

# Return the partition polynomial F[11] and its evaluation when y=7
#
s<-pPart(11)          # run the command
s<-paste0("1",s)      # add the coefficient to the first term (fixed command)
s<-gsub(" y","1y",s)  # replace the variable y without coefficient (fixed command)
s<-gsub("y", "*7",s)  # assignment y = 7
eval(parse(text=s))   # evaluation of the expression (fixed command)
```

---

pPoly *Product of Polynomials*

Description

The function returns the product between polynomials without constant term.

Usage

```
pPoly( L )
```

Arguments

L	lists of coefficients of the polynomials

Value

vector	coefficients of the polynomial output of the product

Note

Called by the functions nKS, nKM in the package kStatistics.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

Examples

```
# c(1,-3) are the coefficients of (x-3x^2), c(2) is the coefficient of 2x
# generate c(0, 2,-6), coefficients of 2x^2-6x^3 =(x-3x^2)*(2x)
pPoly(list(c(1,-3), c(2)))

# c(0,3,-2) are the coefficients of 3x^2-2x^3, c(0,2,-1) are the coefficients of (2x^2-x^3)
# generate c(0,0,0,6,-7,2), coefficients of 6x^4-7x^5+2x^6=(3x^2-2x^3)*(2x^2-x^3)
pPoly(list(c(0,3,-2),c(0,2,-1)))
```

Set2expr *converts a vector into a string*

Description

Converts a set into a string. Useful for manipulating the result before being printed again.

Usage

```
Set2expr(v = c())
```

Arguments

v	Set

Value

string	output as string

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

See Also

MFB MFB2Set

Examples

```
# MFB(c(3),1) generate f[3]g[1]^3 + 3f[2]g[1]g[2] + f[1]g[3]
# Convert the output of the MFB(c(3),1) to a vector using
# S<-MFB2Set(MFB(c(3),1)). It generates the following result:
# "1" "1" "f" "3" "1"
# "1" "1" "g" "1" "3"
# "2" "3" "f" "2" "1"
# "2" "1" "g" "1" "1"
# "2" "1" "g" "2" "1"
# "3" "1" "f" "1" "1"
# "3" "1" "g" "3" "1"
# for example I want to set f[2]=1 then I have to execute
# S[[3]][4]<-""; S[[3]][3]<-"";
# After I run Set2expr(S). It generate
# f[3]g[1]^3 + 3g[1]g[2] + f[1]g[3]
#
S<-MFB2Set(MFB(c(3),1))
S[[3]][4]<-""
S[[3]][3]<-""
Set2expr(S)
```

---

umSet *Disjoint union of multisets*

Description

The function returns the disjoint union of two or more multisets, by adding the multiplicity of equal elements.

Usage

```
umSet( pM )
```

Arguments

pM	list of two or more multisets

Details

Given the two multisets [a,a,a,b] and [a,b,b,c], the function returns [a,a,a,a,b,b,b,c]. Given the subdivisions [[a,b],[a]] with multiplicity 3 and [[a,b],[a]] with multiplicity 5, the function returns [[a,b],[a]] with multiplicity 8. See also examples.

Value

`list`	the multiset union

Note

Called by the function `nPM` in the package `kStatistics`.

Author(s)

Elvira Di Nardo <elvira.dinardo@unito.it>,
Giuseppe Guarino <giuseppe.guarino@rete.basilicata.it>

References

E. Di Nardo, G. Guarino, D. Senato (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from http://arxiv.org/pdf/math/0607623.pdf)

See Also

`mkmSet`, `mCoeff`, `countP`, `nStirling2`, `intPart`, `df`

Examples

```
# Return the list( list( c(1,1), c(1,0) ), c(8) )
M1 <- list( list( c(1,1), c(1,0) ), c(3) )
M2 <- list( list( c(1,1), c(1,0) ), c(5) )
umSet(list(M1,M2))
```

Index