Package 'kStatistics'

October 13, 2022

Type Package

Title Unbiased Estimators for Cumulant Products and Faa Di Bruno's Formula

Version 2.1.1

Date 2022-06-8

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 (joint) products of cumulants of a random sample 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 and 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 2022-06-08 11:20:02 UTC

R topics documented:

`kStatistics-package` *Unbiased Estimators for Cumulant Products and Faa Di Bruno's Formula*

Description

`kStatistics` is a package producing estimates of (joint) cumulants and (joint) cumulant products of a given dataset, 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 tools are given useful in the construction of these estimations such as integer partitions, set partitions, multiset subdivisions or multi-index partitions, pairing and merging of multisets. In the package, there are also functions to recover univariate and multivariate cumulants from a sequence of univariate and multivariate moments (and vice-versa), using Faa di Bruno's formula. The function producing Faa di Bruno's formula returns coefficients of exponential power series compositions such as `f[g(z)]` with `f` and `g` both univariate, or `f[g(z1,...,zm)]` with `f` univariate and `g` multivariate, or `f[g1(z1,...,zm),...,gn(z1,...,zm)]` with `f` and `g` both multivariate. Let us recall that Faa di Bruno's formula might also be employed to recover iterated (partial) derivatives 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 and 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

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

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

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

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

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

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 random sample data1 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 random sample data1
# 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 random sample data2  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 random sample data2
# 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 the subdivisions of a multiset with only one element of multiplicity 3
mkmSet(3)

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

# Return the same output of the previous example but in a compact expression.
mkmSet(c(2,1), TRUE)

# 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)
# OR (same output)
mkT(c(1,0,1),2, FALSE)

# Return the same output of the previous example but in a compact expression.
mkT(c(1,0,1),2, TRUE)

# Return all the partitions of the integer 4, that is
# [1,1,1,1],[1,1,2],[1,3],[2,2],[4]
intPart(4)
# OR (same output)
intPart(4, FALSE)

# Return the same output of the previous example but in a compact expression.
```

```
intPart(4, TRUE)

# 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 x in f(x) with x=g(z)
# f[2] is the coefficient of x^2 in f(x) with  x=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 x in f(x) with x=g(z1,z2)
# f[2]   is the coefficient of x^2 in f(x) with x=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 x1 in f(x1,x2) with x1=g1(z) and x2=g2(z)
# f[0,1] is the coefficient of x2 in f(x1,x2) with x1=g1(z) and x2=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 z1z2
# in f[g1(z1,z2),g2(z1,z2)] 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))

# The 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))

# The 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))

# The partition polynomial F[5]
pPart(5)

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

# The complete ordinary Bell Polynomial for 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)
oBellPol(5,)
```

```
# OR (same output)
oBellPol(5,0)

# The partial ordinary Bell polynomial for n=5 and m=3, that is
# 30(y1)(y2^2) + 60(y1^2)(y3)
oBellPol(5,3)

# The complete exponential Bell Polynomial for 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)
eBellPol(5)
# OR (same output)
eBellPol(5,0)

# The partial exponential Bell Polynomial for n=5 and m=3, that is
# 15(y1)(y2^2) + 10(y1^2)(y3)
eBellPol(5,3)

# The Stirling number of second kind S(5,3) = 25
e_eBellPol(5,3)
# OR (same output)
e_eBellPol(5,3,c(1,1,1,1,1))

# The Bell number B5 = 52
e_eBellPol(5)
# OR (same output)
e_eBellPol(5,0)
# OR (same output)
e_eBellPol(5,0,c(1,1,1,1,1) )

# 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 generalized complete Bell Polynomial for n=1, m=2 and g1=g
# 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
# 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" )
```

`countP` *Multiplicity of a multi-index partition.*

Description

The function computes the multiplicity of a multi-index partition. Note that a multi-index partition corresponds to a subdivision of a multiset having the input multi-index as multiplicities.

Usage

```
countP( v=c(1) )
```

Arguments

v	vector or list

Details

Consider the partitions of an integer, let's say 3, that are `[1,1,1]`,`[1,2]`,`[3]` such that `1+1+1=1+2=3`. Consider the partitions of a set with cardinality 3, let's say `[a1, a2, a3]`

(I) `[[a1], [a2], [a3]]`, `[[a1],[a2,a3]]`, `[[a2],[a1,a3]]`, `[[a3],[a1,a2]]`, `[[a1,a2,a3]]`.

The multiplicity of a partition of 3 is the number of partitions of `[a1, a2, a3]` in blocks having that partition as cardinalities. In the example, we have

a) 1 partition in 3 blocks of cardinalities 1, that is `[[a1], [a2], [a3]]` corresponding to `[1,1,1]`
b) 1 partition in 1 block of cardinalities 3, that is `[[a1,a2,a3]]` corresponding to `[3]`
c) 3 partitions in 2 blocks of cardinalities 1 and 2 respectively, that is `[[a1],[a2,a3]]`, `[[a2],[a1,a3]]`, `[[a3],[a1,a2]]`.

So running `countP(c(1,2))` we get 3, running `countP(c(1,1,1))` or `countP(c(3))` we get 1.

The same device can be used to find the multiplicity of a subdivision. The subdivisions of a multiset are obtained as follows: assume all distinct the elements of the multiset, determine all the corresponding set partitions and then replace each element in each block with the original one. For example, consider the multiset `[a, a, a]` having the integer 3 as multiplicity. Assuming all distinct the elements of the multiset, the partitions of `[a1, a2, a3]` are given in `(I)`. Now replace `a1<-a, a2<-a, a3<-a` and get

```
[[a], [a], [a]], [[a], [a,a]], [[a], [a,a]], [[a], [a,a]], [[a,a,a]]
```

. Note that the partitions

```
[[a1,a2], [a3]], [[a1,a3], [a2]], [[a2,a3], [a1]]
```

give rise to the same subdivision `[[a,a], [a]]`. Then the multiplicity of `[[a,a], [a]]` is 3. Therefore the subdivisions of `[a, a, a,]` are

```
[[a], [a], [a]], [[a], [a,a]], [[a,a,a]].
```

The multiplicity of the subdivision `[[a], [a,a]]` is 3, as for the partition `[1,2]` of the integer 3. The multiplicity of the subdivisions `[[a], [a], [a]]` and `[[a,a,a]]` is 1, as for the partitions `[1,1,1]` and `[3]` of the integer 3 respectively. Thus running `countP(c(1,2))` we get 3, that is

a) the multiplicity of the subdivision `[[a,a], [a]]`
b) the number of partitions of the set `[a1, a2, a3]` in two blocks, one of cardinality 1 and the other of cardinality 2
c) the number of partitions of the set `[a1, a2, a3]` corresponding to the partition `[1,2]` of the integer 3.

Now consider the partition of a multi-index, let's say (4,2). One of the partitions of (4,2) is the matrix (see the output of the `mkmSet` function)

```
               0  1  1  2
(II)           1  0  0  1
```

as `1+1+2=4` and `1+1=2`. The partition `(II)` corresponds to the subdivision `[[b], [a], [a], [a,a,b]]` of the multiset `[a,a,a,a,b,b]` having the multi-index (4,2) as multiplicities. Indeed each column `(i,j)` gives the instances `i` and `j` of `a` and `b` in a block of the subdivision. Running `countP (list(c(0,1), c(1,0), c(1,0), c(2,1)))` we get 12. This multiplicity gives the number of partitions of the set `[a1,a2,a3, a4,b1,b2]` corresponding to the subdivision `[[b], [a], [a], [a,a,b]]` after the replacement `a1<-a, a2<-a, a3<-a, a4<-a, b1<-b, b2<-b`.

Note that the input of the `countP` function does not necessarily be in lexicographic order. To find all the partitions of a multi-index see the `mkmSet` function.

Value

integer the multiplicity of the given item

Note

Called by the mkmSet and nPS functions in the kStatistics 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 https://arxiv.org/pdf/math/0607623.pdf)

See Also

mkmSet, mCoeff, nStirling2, intPart, ff

Examples

```
# Return 3 which is the multiplicity of [1,2], partition of the integer 3, or
# of [[a],[a,a]], subdivision of the multiset [a,a,a]
countP(c(1,2))

# Return 15 which is the multiplicity of [4,2], partition of the integer 6, or
# of [[a,a,a,a],[a,a]], subdivision of the multiset [a,a,a,a,a,a]
countP(c(4,2))

# Return 18 which is the multiplicity of
#  0 0 1 1 2
#  1 2 0 0 0
# partition of the multi-index (4,3), or of [[b],[b,b],[a],[a],[a,a]], subdivision  of
# the multiset [a,a,a,a,b,b,b]
countP( list(c(2,0), c(1,0), c(1,0), c(0,1),c(0,2)) )
```

---

cum2mom *Cumulants in terms of moments*

Description

The function computes a simple or a multivariate cumulant in terms of simple or multivariate moments.

Usage

```
cum2mom(n = 1)
```

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 f[g()] where f and g are exponential formal power series too. Simple cumulants are expressed in terms of simple 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 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". In such a case the coefficients of g are the multivariate moments.

Value

string	the expression of the cumulant 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 kStatistics 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 https://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 https://arxiv.org/abs/1012.6008)

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

See Also

MFB

Examples

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

# Return the 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))
```

---

eBellPol *Exponential Bell polynomials*

Description

The function generates a complete or a partial exponential Bell polynomial.

Usage

```
eBellPol(n = 1, m = 0)
```

Arguments

n	integer, the degree of the polynomial
m	integer, the fixed degree of each monomial in the polynomial

Details

Faa di Bruno's formula gives the coefficients of the exponential formal power series composition `f[g()]` obtained from the composition of the exponential formal power series `f` with `g`. Complete exponential Bell polynomials in the variables `y[1],...,y[n]` are generated by setting `f[i]=1` and `g[i]=y[i]`, for each `i` from `1` to `n`. Partial exponential Bell polynomials are polynomials in the variables `y[1],...,y[n-m+1]` with fixed degree `m` for each of the involved monomials. Partial exponential Bell polynomials are recovered from Faa di Bruno's formula by setting `g[i]=y[i]` for each `i` from `1` to `n` and `f[i]=1` if `i=m`, `f[i]=0` otherwise.

Value

string	the expression of the exponential Bell polynomial

Warning

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

Note

This function calls the MFB function in the kStatistics package.

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 (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from https://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 https://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 https://arxiv.org/abs/1012.6008)

See Also

MFB

Examples

```
# Return the complete exponential Bell Polynomial for 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)
eBellPol(5)
# OR (same output)
eBellPol(5,0)

# Return the partial exponential Bell Polynomial for n=5 and m=3, that is
# 15(y1)(y2^2) + 10(y1^2)(y3)
eBellPol(5,3)
```

---

e_eBellPol *Evaluation of exponential Bell polynomials*

Description

The function evaluates a complete or a partial exponential Bell polynomial (output of the eBellPol function) when its variables are substituted with numerical values.

Usage

```
e_eBellPol(n=1,m=0,v=c(rep(1,n)))
```

Arguments

n	integer, the degree of the polynomial
m	integer, the fixed degree of each monomial in the polynomial
v	vector, the numerical values in place of the variables of the polynomial

Details

The eBellPol function generates a complete or a partial exponential Bell polynomial in the variables y[1], . . . , y[n-m+1]. The e_eBellPol function computes the value assumed by this polynomial when its variables are substituted with numerical values.

Value

numerical value
: the value assumed by the polynomial.

Warning

By default, the function returns the Stirling numbers of second kind.

Note

This function calls the eBellPol function in the kStatistics package.

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 (2008) An unifying framework for k-statistics, polykays and their generalizations. Bernoulli. 14(2), 440-468. (download from https://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 https://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 https://arxiv.org/abs/1012.6008)

See Also

eBellPol

Examples

```
# Return S(5,3) = 25 (where S=Stirling number of second kind)
e_eBellPol(5,3)
#
# OR (same output)
#
e_eBellPol(5,3,c(1,1,1,1,1))

# Return  B5=52 (where B5 is the 5-th Bell number)
e_eBellPol(5)
#
# OR (same output)
#
e_eBellPol(5,0)
#
# OR (same output)
#
e_eBellPol(5,0,c(1,1,1,1,1))

# Return  s(5,3) = 35 (where s=Stirling number of first kind)
e_eBellPol(5,3,c(1,-1,2,-6,24))
```

---

e_GCBellPol *Evaluation of Generalized Complete Bell Polynomials*

Description

The function evaluates a generalized complete Bell polynomial (output of the GCBellPol function) when its variables and/or its coefficients are substituted with numerical values.

Usage

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

Arguments

pv	vector of integers, the subscript of the polynomial
pn	integer, the number of variables
pyc	vector, the numerical values into the variables [optional], or the string with the direct assignment into the variables and/or the coefficients
pc	vector, the numerical values into the coefficients, [optional if pyc is a string]
b	boolean, if TRUE the function prints the list of all the assignments

Details

The function GCBellPol returns the coefficient of the multivariate exponential formal power series exp(y[1] g1(z1,...,zm) + ... + y[n] gn(z1,...,zm)), where y[1],...,y[n] are variables corresponding to the subscript pv. The function e_GCBellPol allows us to substitute the coefficients of the power series g1,...,gn and/or the variables y[1],...,y[n] with numerical values. These values are passed to the e_GCBellPol function through the third and the fourth input parameter. In the resulting expression, the y's and the g's are managed in lexicographic order. There is one further input boolean parameter: when equal to TRUE, the function prints the list of all the assignments. See the examples for more details on the employment of this boolean parameter when the coefficients and/or the variables of the polynomial are substituted with numerical values.

Value

`string` or `numerical`
: the evaluation of the polynomial

Warning

The value of the first parameter is the same as the mkmSet function.

Note

Called by the GCBellPol function in the kStatistics package.

Author(s)

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

References

E. Di Nardo (2016) On multivariable cumulant polynomial sequence with applications. Jour. Algebraic Statistics 7(1), 72-89. (download from https://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 https://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 with subscript 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 parameter equals to 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 parameter equals to 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 with subscript (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 parameter equals to 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 \code{\link{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 parameter equals to 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 with subscript (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 parameter equals to 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 e_GCBellPol 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]
```

```
# To assign correctly the 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 - for example - the
#    first 6 integer 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 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, output of the MFB function, when the coefficients of the exponential formal power series f and g1, . . . , gn in the composition f[g1(), . . . , gn()] are substituted with numerical values.

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, the number of the inner formal power series "g"
pf	vector, the numerical values in place of the coefficients of the outer formal power series "f" or the string with the direct assignments in place of the coefficients of both "f" and "g"
pg	vector, the numerical values in place of the coefficients of the inner formal power series "g" [Optional if pf is a string]
b	boolean

Details

The output of the MFB function is a coefficient of the exponential formal power series compositions in the cases

a) univariate f with univariate g
b) univariate f with multivariate g
c) multivariate f with multivariates {gi}

The e_MFB function evaluates this coefficient when the coefficients of f and {gi} are substituted with numerical values. These values are passed to the e_MFB function trough the third and the fourth input parameter. There is one further input boolean parameter: when equal to 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 expression of the coefficients of f and {gi} becomes more complex.

Value

numerical the evaluation of Faa di Bruno's formula

Warning

The value of the first parameter is the same as the mkmSet function.

Note

Called from the MFB function in the kStatistics 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. 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 https://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 z1z2 in
# f(g1(z1,z2),g2(z1,z2))) 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, 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]

# To assign correctly the 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 - for example - the first 5 integer
#    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 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

The function 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=1, k )
```

Arguments

`n`	integer
`k`	integer

Details

Run `ff(n, k)` to get `n(n-1)(n-2).....(n-k+1)`.

Value

`integer`	the descending factorial

Author(s)

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

See Also

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

Examples

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

`GCBellPol` *Generalized Complete Bell Polynomial*

Description

The function generates a generalized complete Bell polynomial, that is a coefficient of the composition `exp(y[1] g1(z1,...,zm) + ... + y[n] gn(z1,...,zm))`, where `y[1],...,y[n]` are variables. The input vector of integers identifies the subscript of the polynomial.

Usage

```
GCBellPol(nv = c(), m = 1, b = FALSE)
```

Arguments

`nv`	vector of integers, the subscript of the polynomial, corresponding to the powers of the product among `z1, z2, ..., zm`
`m`	integer, the number of `z`'s variables
`b`	boolean, `TRUE` if the inner formal power series `"g"` are all equal

Details

The multivariate Faa di Bruno's formula, output of the MFB function, gives a coefficient of the multivariate exponential power series obtained from the composition of the multivariate exponential power series f(x1,...,xn) with xi=gi(z1,...,zm) for each i from 1 to n. Now, set f(y[1],...,y[n];x1,...,xn)=exp(y[1] x1 + ... + y[n] xn). In such a case, the coefficients are the generalized complete Bell polynomials, see the referred papers. In particular, the GCBellPol function gives the expression of these polynomials when n=1 or when n>1 and g1=...=gn=g or when n>1 and g1, ..., gn are all different. See the e_GCBellPol function for evaluating this polynomial when its variables y[1], ..., y[n] or/and its coefficients are substituted with numerical values.

Value

string	the expression of the polynomial

Warning

The value of the first parameter is the same as the mkmSet function.

Note

This function calls the MFB function in the kStatistics package.

Author(s)

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

References

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

E. Di Nardo (2016) On multivariable cumulant polynomial sequence with applications. Jour. Algebraic Statistics 7(1), 72-89. (download from https://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 https://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,
# 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,
# (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 a general partition polynomial.

Usage

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

Arguments

n	integer

Details

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

Value

string	the expression of the polynomial

Warning

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

Note

This function calls the MFB function in the kStatistics package.

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 https://arxiv.org/abs/1012.6008)

See Also

MFB

Examples

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

# Return the general partition polynomial G[a1,a2,a3,a4,a5; y1,y2,y3,y4,y5], 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

The function generates all possible (unique) decomposition of a positive integer n in the sum of positive integers less or equal to n.

Usage

```
intPart(n=0 ,vOutput = FALSE)
```

Arguments

n	integer
vOutput	optional boolean parameter, if equal to TRUE the function produces a compact output that is easy to read.

Details

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

Value

list	all the partitions of n

Note

Called by the mkmSet function in the kStatistics package.

Author(s)

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

References

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

See Also

mkmSet, mCoeff, nStirling2, countP, ff

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)
# OR (same output)
intPart(4, FALSE)

# Return the same output as the previous example but in a compact expression
intPart(4, TRUE)
```

`list2m` *List To Multiset*

Description

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

Usage

```
list2m( v=c(0) )
```

Arguments

`v`	single vector or list of vectors

Details

Given a list in input, the `list2m` 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 vector `v`.

Value

`multiset`	the list of multisets

Note

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

Author(s)

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

References

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

See Also

`list2Set`, `m2Set`, `countP`

Examples

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

# Return the list of multisets [[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, the function deletes the instances of an element in the list, leaving the order inalterated.

Usage

```
list2Set(v=c(0))
```

Arguments

v	single vector or list of vectors

Value

set	the sequence of distinct elements

Note

Called by the list2m function in the kStatistics package.

Author(s)

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

See Also

list2m, m2Set

Examples

```
# Return the vector 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 *Detecting equal columns in multi-index partitions*

Description

The function returns the vectors (only counted once) of all the multi-index partitions output of the mkmSet function. These vectors correspond also to the blocks of the subdivisions of the multiset having the given multi-index as multeplicites.

Usage

```
m2Set( v=c(0) )
```

Arguments

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

Details

Consider the multi-index (2,1). The partitions are

```
0 1 1      0 2      1 1     2
1 0 0      1 0      0 1     1
```

with multiplicities 1, 1, 2, 1 respectively. The m2Set function deletes column repetitions, that is transforms the given list in [[0,1],[1,0],[2,0],[1,1],[2,1]] according to the order given in the input. In terms of subdivisions, suppose to consider the multiset [a,a,b] with multiplicities (2,1). The subdivisions are

```
[[[b],[a],[a]],1], [[[a,a],[b]],1], [[[a],[a,b]],2], [[a,a,b],1].
```

The m2Set function deletes block repetitions, that is transforms the given list in

```
[[b],[a],[a,a],[a,b],[a,a,b]]
```

according to the order given in the input. See also the examples.

Value

set	the sequence with distinct elements

Note

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

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
#  list(
#       list( list(  c(0,1), c(1,0), c(1,0) )  ,1),
#       list( list(  c(0,1), c(2,0)         )  ,1),
#       list( list(  c(1,0), c(1,1)         )  ,2),
#       list( list(  c(2,1)                 )  ,1),
#      )
# To print all the partitions of the multi-index (2,1) run mkmSet(c(2,1),TRUE)
#   [( 0 1 )( 1 0 )( 1 0 ),  1 ]
#   [( 0 1 )( 2 0 ),  1 ]
#   [( 1 0 )( 1 1 ),  2 ]
#   [( 2 1 ),  1 ]
#
# Then m2Set(M1) returns the following set:  [[0,1],[1,0],[2,0],[1,1],[2,1]]
#
m2Set( M1 )
```

---

mCoeff *Extraction of a number from a list*

Description

Given a list containing vectors paired with numbers, the function returns the number paired with the vector matching the one passed in input.

Usage

```
mCoeff( v=NULL,L=NULL )
```

Arguments

v	vector to be searched in the list
L	two-dimensional list: in the first there is a vector and in the second a number

Details

The input variable L of the mCoeff function is a list containing vectors and numbers. The input variable v of the mCoeff function is one of vectors contained in the list. The function searches the vector v in the list and returns the number which is paired with v in the list. This function is useful in the construction of k-statistics but also to manage monomials and their coefficients.

Value

float	the number paired with the input vector

Note

Called by the nPS, nKM and nPM functions in the kStatistics 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 https://arxiv.org/pdf/math/0607623.pdf)

See Also

mkmSet, countP, nStirling2, intPart, ff

Examples

```
# Run mkmSet(c(3)) to get the list L1 = [[1,1,1],1], [[1,2],3], [[3],1]
L1 <- mkmSet(c(3))

# Return the number 3, which is the number paired with [1,2] in L1
mCoeff( c(1,2), L1)
```

MFB *Multivariate Faa di Bruno's formula*

Description

The function returns the coefficient indexed by the integers i1,i2,... of an exponential formal power series composition through the univariate or multivariate Faa di Bruno's formula.

Usage

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

Arguments

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

Details

The MFB function computes a coefficient of an exponential formal power series composition:

a) univariate f with univariate g, that is f[g(z)],
b) univariate f with multivariate g, that is f[g(z1,z2,...,zm)],
c) multivariate f with multivariate g's, that is f[g1(z1,z2,...,zm),...,gn(z1,z2,...,gm)].

If i1 is the power of z1, i2 is the power of z2 and so on up to im power of zm, then (i1,i2,....im) is the subscript of the output coefficient corresponding to the product z1^i1 z2^i2 ....zm^im. Note that this coefficient gives also the (partial) derivative of order (i1,i2,...,im) of the composition of the multivariate functions f and g's in terms of the partial derivatives of f and g's respectively. See the e_MFB function, for evaluating this coefficient when the coefficients of f and to the coefficients of g's are substituted with numerical values.

Value

string	the expression of Faa di Bruno's formula

Warning

The value of the first parameter is the same as the mkmSet function

Note

Called by the e_MFB function in the kStatistics package. 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

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

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 https://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 https://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 https://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 x in f(x) with x=g(z)
# f[2] is the coefficient of x^2 in f(x) with  x=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 x in f(x) with x=g(z1,z2)
# f[2]   is the coefficient of x^2 in f(x) with x=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 Univariate/Multivariate g1, g2, ..., gn   #
#-----------------------------------------------------------------#

# 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 x1 in f(x1,x2) with x1=g1(z) and x2=g2(z)
# f[0,1] is the coefficient of x2 in f(x1,x2) with x1=g1(z) and x2=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 x1 in f(x1,x2) with x1=g1(z1,z2) and x2=g2(z1,z2)
# f[0,1]  is the coefficient of x2 in f(x1,x2) with x1=g1(z1,z2) and x2=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 into a vector*

Description

Secondary function useful for manipulating the result of the MFB function.

Usage

```
MFB2Set(sExpr="")
```

Arguments

sExpr	the output of the MFB function

Value

set	a set

Author(s)

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

See Also

MFB, Set2expr

Examples

```
# Run MFB(c(3),1) to 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) into a vector using
# MFB2Set(MFB(c(3),1)). The result is the following:
# "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 *Partitions of a multi-index*

Description

The function returns all the partitions of a multi-index, that is a vector of non-negative integers. Note that these partitions correspond to the subdivisions of a multiset having the input multi-index as multiplicities.

Usage

```
mkmSet(vPar = NULL, vOutput = FALSE)
```

Arguments

vPar	vector of non-negative integers
vOutput	optional boolean variable. If equal to TRUE, the function produces a compact output that is easy to read.

Details

The mkmSet function finds all the vectors, different from the zero vector, whose sum (in column) is equal to the vector (in column) of nonnegative integers given in input. When the input vector is just an integer, let's say n, the function returns the partitions of n. Each partition is paired with the number of set partitions having that partition as their class. For example, if n=3 the output is

```
[[1,1,1],1], [[1,2],3], [[3],1],
```

where 1+1+1=1+2=3. From this output, the subdivisions of a multiset with multiplicity 3 can be recovered. For example, the subdivisions of [a,a,a] are [[a], [a], [a]] corresponding to [1,1,1], [[a], [a,a]] corresponds to [1,2] and [[a,a,a]] corresponds to [3]. When the input vector is a multi-index, the function returns all the partitions of the multi-index. For example, if the input is (2,1) then the function returns

```
0 1 1     0 2     1 1     2
1 0 0     1 0     0 1     1
```

with multiplicities 1, 1, 2, 1 respectively, which corresponds to the output (the columns become rows) of the mkmSet function when the flag variable vOutput is set equal to TRUE

```
[( 0 1 )( 1 0 )( 1 0 ), 1 ]
 [( 0 1 )( 2 0 ), 1 ]
 [( 1 0 )( 1 1 ), 2 ]
 [( 2 1 ), 1 ]
```

From this output, the subdivisions of a multiset with multiplicity (2,1) can be easily recovered. For example the previous partitions correspond to the following subdivisions of the multiset [a,a,b]

```
[[b], [a], [a]]
 [[b], [a,a]]
 [[a], [a,b]]
 [[a,a,b]]
```

The mkmSet function is the core of the kStatistics package. The strategy to find all the partitions of a multi-index is described in the refereed papers. To find the multiplicities of the multi-index partitions see the countP function.

Value

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

Note

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

In the output list, the sum of all multiplicities is the Bell Number whereas the sum of all multiplicities of the partitions with the same lenght is the Stirling number of the second kind. For example, `mkmSet(4, TRUE)` returns

```
[(1)(1)(1)(1), 1]
 [(1)(1)(2), 6]
 [(2)(2), 3]
 [(1)(3), 4]
 [(4), 1]
```

Observe that `3 + 4 = 7 = S(4,2)`, where 4 is the input integer, 2 is the length of the partitions `[1,3]` and `[2,2]` and `S(i,j)` denotes the Stirling numbers of the second kind, see also the nStirling2 function. Similarly, we have `1 = S(4,4) = S(4,1)` and `6 = S(4,3)`. Note that `1 + 6 + 4 + 3 + 1 = 15 = Bell(4)` which is the number of partitions of the integer 4.

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 https://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 https://arxiv.org/abs/0806.0129)

See Also

mCoeff, countP, nStirling2, intPart, ff

Examples

```
# Return [ [[1,1,1],1], [[1,2],3], [[3],1]  ]
# 3 is the multiplicity of a multiset with 3 elements all equal
mkmSet(3)

# Return [ [[1,1],[1,0],2], [[1,0],[1,0],[0,1],1],[[2,0],[0,1],1], [[2,1],1] ]
# (2,1) is the multiplicity of a multiset with 2 equal elements and a third distinct element
mkmSet(c(2,1))
# OR (same output)
mkmSet(c(2,1), FALSE)
```

```
# Returns the same output of the previous example but in a compact form.
mkmSet(c(2,1), TRUE)
```

---

mkT *Scomposition of a multi-index*

Description

Given a multi-index, that is a vector of non-negative integers and a positive integer n, the function returns all the lists (v1,...,vn) of non-negative integer vectors, with the same lenght of the multi-index and such that v=v1+...+vn.

Usage

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

Arguments

v	vector of integers
n	integer, number of addends
vOutput	optional boolean variable. If equal to TRUE, the function produces a compact output that is easy to read.

Details

From the input vector v of non-negative integers, which represents the multi-index, the function produces 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 permuted.

Value

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

Warning

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

Note

Called by the MFB function in the kStatistics package. 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

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 https://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)
# OR (same output)
mkT(c(1,1),2,FALSE)

# 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)
# OR (same output)
mkT(c(1,0,1),2, FALSE)

# Return the same output of the previous example but in a compact form.
mkT(c(1,0,1),2, TRUE)

# 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 given in a compact form.
for (m in mkT(c(1,1,1),3)) {for (n in m) cat(n," - "); cat("\n")}
```

---

mom2cum *Moments in terms of cumulants*

Description

The function compute a simple or a multivariate moment in terms of simple or multivariate cumulants.

Usage

```
mom2cum(n = 1)
```

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 f[g()] where f and g are exponential formal power series too. Simple moments are expressed in terms of simple 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". In such a case the coefficients of g are the multivariate cumulants.

Value

string	the expression of the moment 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 kStatistics 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 https://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 https://arxiv.org/abs/1012.6008)

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

See Also

MFB

Examples

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

# Return the 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 *Join two lists*

Description

Given two lists with elements of the same type, the function returns a new list whose elements are the joining of the two original lists, except for the last elements, which are multiplied.

Usage

```
mpCart( M1 = NULL, M2 = NULL )
```

Arguments

M1	list of vectors
M2	list of vectors

Details

The input of the mpCart function are two lists. Each list might contain multiple lists of two vectors: the first vector contains multisets whose elements are of the same type (integers or vectors with the same lenght), the second vector is a number (for example a multiplicity if the multiset is a subdivision). The mpCart function generates a new list of two vectors: the first is obtained by joining the first vectors in the two input lists, the second is just the product of the numbers in the second vectors. See the examples.

Value

list	the list with the joined input lists

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 https://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 https://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) )
#  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))

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

A <- list( list( list( c(1,0),c(1,0) ), c(-1)),  list( list( c(2,0)), c(1) ))
# where
# - 1 is the multiplicative factor of list( c(1,0),c(1,0) )
#   1 is the multiplicative factor of list( c(2,0) )
B <- list( list( list( c(1,0)), c(1)) )
# where 1 is the multiplicative factor of list( c(1,0))

# Return [[[1,0],[1,0],[1,0]], -1], [[[2,0],[1,0]],1]
mpCart(A,B)
```

---

nKM *Multivariate K-Statistics*

Description

Given a multivariate data sample, the function returns an estimate of a joint (or multivariate) cumulant with a fixed order.

Usage

```
nKM( v = NULL, V = NULL)
```

Arguments

v	vector of integers
V	vector of a multivariate data sample

Details

For a sample of i.i.d. random vectors, multivariate k-statistics are unbiased estimators of the population joint cumulants with minimum variance and are expressed in terms of power sum symmetric polynomials in the random vectors of the sample. See the referred papers to read more about these estimators. Thus, for the input multivariate sample `data`, running `nKM( c(r, s, ...), data)` with fixed order `v=(r, s, ...)` returns an estimate of the joint cumulant `k[r, s, ...]` of the population distribution.

Value

`float`	the value of the multivariate k-statistics

Warning

The size of each data vector must be equal to the length of the vector passed trough the first input variable.

Note

Called by the master `nPolyk` function in the `kStatistics` 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 https://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 https://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 data sample, the function returns an estimate of a cumulant with a fixed order.

Usage

```
nKS( v = NULL, V = NULL)
```

Arguments

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

Details

For a sample of i.i.d. random variables, k-statistics are unbiased estimators with minimum variance of the population cumulants and are expressed in terms of power sum symmetric polynomials in the random variables of the sample. See the referred papers to read more about these estimators. Thus, for the input sample `data`, running `nKS(v,data)` or `nKS(c(v),data)` returns an estimate of the `v`-th cumulant of the population distribution.

Value

`float` the value of the k-statistics

Note

Called by the master `nPolyk` function in the `kStatistics` 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 https://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 https://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 *Permutations of a list or a vector*

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 the permutations of L

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.

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", "Bob","Jack"
nPerm( c("Alice", "Bob","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 data sample, the function returns an estimate of a product of joint cumulants with fixed orders.

Usage

```
nPM( v = NULL, V = NULL)
```

Arguments

v	list of integer vectors
V	vector of a multivariate data sample

Details

Multivariate polykays or multivariate generalized k-statistics are unbiased estimators of joint cumulant products with minimum variance. See the referred papers to read more about these estimators. Multivariate polykays are usually expressed in terms of power sum symmetric polynomials in the random vectors of the sample. Thus, for the input multivariate sample `data`, running `nPM( list( c(r1, s1, ...), c(r1, s2, ...),.. ), data)` returns an estimate of the product `k[r1, s1,....]*k[r2, s2, ...]*...` where `k[r1, s1,....]`, `k[r2, s2, ...]`, `...` are the joint cumulants of the population distribution.

Value

`float`	the estimate of the multivariate polykay

Warning

The size of each data vector must be equal to the length of the vector passed trough the first input variable. The vectors in the list must have the same length.

Note

Called by the master `nPolyk` function in the `kStatistics` 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 https://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 https://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 the
# cross-correlation of order (2,1) and the marginal mean of the population distribution
# respectively
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 compute simple k-statistics (nKS), multivariate k-statistics (nKM), simple polykays (nPS) or multivariate polykays (nPM).

Usage

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

Arguments

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

Details

The master function analizes the first two input variables and recalls one of the nKS, nKM, nPS or nPM functions in the kStatistics package.

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	the estimate of the (joint) cumulant or of the (joint) cumulant product

Note

The dimension of the vector with the sample data 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 https://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 https://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:[1] -1.44706" which indicates the type of subfunction (nKS) called by
# the master function nPolyk and gives the estimate of the third cumulant
nPolyk(c(3),data1, TRUE)

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

# Display "PS:[1] 177.4233" which indicates the type of subfunction (nPS) called by
# the master function nPolyk and gives the estimate of the product between the
# variance k[2] and the mean k[1]
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))

# Display "KM:[1] -23.7379" which indicates the type of subfunction (nKM) called by
# the master function nPolyk and gives the estimate of k[2,1]
nPolyk(c(2,1),data2,TRUE)

# Display "PM:[1] 48.43243" which indicates the type of subfunction (nPM) called by
# the master function nPolyk and gives the estimate of k[2,1]*k[1,0]
nPolyk( list(  c(2,1), c(1,0) ),data2,TRUE)
```

---

nPS *Simple Polykays*

Description

Given a data sample, the function returns an estimate of a product of cumulants with fixed orders.

Usage

```
nPS( v = NULL, V = NULL)
```

Arguments

v	vector of integers
V	vector of a data sample

Details

Simple polykays or generalized k-statistics are unbiased estimators of cumulant products with minimum variance. See the referred papers to read more about these estimators. Simple polykays are usually expressed in terms of power sum symmetric polynomials in the i.i.d. random variables of the sample. Thus, for the input sample `data`, running `nPS(c(i,j,...),data)` returns an estimate of the product `k[i]*k[j]*...` with `k[i]`, `k[j]`, ... the cumulants of the population distribution and `v=(i,j,...)` their fixed orders.

Value

`float`	the estimate of the polykay

Note

Called by the master `nPolyk` function in the `kStatistics` 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 https://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 https://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 an estimate of the product k[2]*k[1], where k[1] and k[2] are the mean and
# the variance of the population distribution respectively
nPS(c(2,1), data)
```

---

nStirling2 *Stirling number of second kind*

Description

The function computes the Stirling number of the second kind.

Usage

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

Arguments

n	integer
k	integer less or equal to n

Details

The Stirling number of the second kind `S(n,k)` is equal to the number of ways to split a set of cardinality `n` into `k` nonempty subsets. For example, if the set is `[a,b,c,d]`, then 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 equal to 7. Note that (1,3) and (2,2) are also the partitions of the integer 4 in 2 parts.

Value

integer	the Stirling number of the second kind

Note

Called by the nKS and nKM functions in the `kStatistics` package.

Author(s)

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

References

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

See Also

mkmSet, mCoeff, intPart, countP, ff

Examples

```
# Return the number of ways to split a set of 6 objects into 2 nonempty subsets
nStirling2(6,2)
```

---

oBellPol *Ordinary Bell polynomials*

Description

The function generates a complete or a partial ordinary Bell polynomial.

Usage

```
oBellPol(n = 1, m = 0)
```

Arguments

n	integer, the degree of the polynomial
m	integer, the fixed degree of each monomial in the polynomial

Details

Faa di Bruno's formula gives the coefficients of the exponential formal power series obtained from the composition f[g()] of the exponential formal power series f with g. The partial ordinary Bell polynomials B[n,m] can be expressed in the terms of the partial exponential Bell polynomials B(n,m)(y[1],...,y[n-m+1]) using the following formula:

B[n,m](y[1],...,y[n-m+1])=k!/n!B(n,m)(y[1],...,y[n-m+1]).

The complete ordinary Bell polynomials are given by B[n]=B[n,1]+B[n,2]+...B[n,n], where B[n,m] is the partial ordinary Bell polynomial of order (n,m) for m from 1 to n.

Value

string	the expression of the polynomial

Warning

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

Note

This function calls the MFB function in the kStatistics package.

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 https://arxiv.org/abs/1012.6008)

See Also

MFB

Examples

```
# Return the complete ordinary Bell Polynomial for 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)
oBellPol(5)
#
# OR (same output)
#
oBellPol(5,0)

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

---

pCart *Cartesian product*

Description

The function returns the cartesian product between vectors.

Usage

```
pCart( L )
```

Arguments

L	vectors in a list

Details

The pCart 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	the list with the cartesian product

Note

Called by the nPS function in the kStatistics package.

Author(s)

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

References

D. E. Knuth (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]]
# and assign the result to L3
L3<-pCart ( list(L1, L2) )

# Return the cartesian product between L3 and [7].
# The result is [[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

The function returns the value of the power sum symmetric polynomial, with fixed degrees and in one or more sets of variables, when the variables are substituted with the input lists of numerical values.

Usage

```
powS(vn = NULL, lvd = NULL)
```

Arguments

vn	vector of integers (the powers of the indeterminates)
lvd	list of numerical values in place of the variables

Details

Given the lists of numerical values `(x[1],x[2],...)`, `(y[1],y[2],...)`, `(z[1],z[2],...)`, `...` in the input parameter `lvd` and the integers `(n,m,j,...)` in the input parameter `vn`, the powS function returns the value of `(x[1]^n)*(y[1]^m)*(z[1]^j)*...+(x[2]^n)*(y[2]^m)*(z[2]^j)*+....`

Value

integer	the value of the polynomial

Note

Called by the nKS, nKM, nPS and nPM functions in the `kStatistics` 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) Symbolic computation of moments of sampling distributions. Comp. Stat. Data Analysis. 52(11), 4909-4922. (download from https://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 polynomials*

Description

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

Usage

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

Arguments

n	integer, the degree of the polynomial

Details

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

Value

string	the expression of the 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 kStatistics package.

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 https://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 = NULL)
```

Arguments

L	lists of the coefficients of the polynomials

Value

vector	the coefficients of the polynomial output of the product

Note

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

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
# Return 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)
# Return 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 *Conversion of a vector into a string*

Description

The function converts a set into a string.

Usage

```
Set2expr(v = NULL )
```

Arguments

v	Set

Value

string	the string

Note

Called by the MFB and MFB functions in the kStatistics package being useful for manipulating the result before its print.

Author(s)

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

Examples

```
# To print 6f[3]^2g[2]^5 run
Set2expr( list(c("1","2","f","3","2"),c("1","3","g","2","5")))

# Run MFB(c(3),1) to recover f[3]g[1]^3 + 3f[2]g[1]g[2] + f[1]g[3]
# Run S<-MFB2Set(MFB(c(3),1)) to convert the output of MFB(c(3),1) into a vector.
# The result is
# "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"
# To set f[2]=1, run S[[3]][4]<-"" and S[[3]][3]<-"".
# Then run Set2expr(S) to recover
# 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)
```

Index