Package 'jack'

July 29, 2024

Type Package

Title Jack, Zonal, Schur, and Other Symmetric Polynomials

Version 6.1.0

Date 2024-07-29

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Schur polynomials appear in combinatorics and zonal polynomials appear in random matrix theory. They are particular cases of Jack polynomials. This package allows to compute these polynomials and other symmetric multivariate polynomials: flagged Schur polynomials, factorial Schur polynomials, t-Schur polynomials, Hall-Littlewood polynomials, Macdonald polynomials, and modified Macdonald polynomials. In addition, it can compute the Kostka-Jack numbers, the Kostka-Foulkes polynomials, the Kostka-Macdonald polynomials, and the Hall polynomials. Mainly based on Demmel & Koev's paper (2006) <doi:10.1090/S0025-5718-05-01780-1> and Macdonald's book (1995) <doi:10.1093/oso/9780198534891.003.0001>.

License GPL-3

URL https://github.com/stla/jackR

BugReports https://github.com/stla/jackR/issues

Depends qspray (>= 3.1.0), ratioOfQsprays (>= 1.1.0), symbolicQspray (>= 1.1.0)

Imports DescTools, gmp, methods, multicool, mvp, partitions, RationalMatrix, Rcpp, spray, syt (>= 0.5.0), utils

Suggests testthat

LinkingTo BH, qspray, ratioOfQsprays, Rcpp, RcppCGAL, symbolicQspray

Encoding UTF-8

RoxygenNote 7.3.1

SystemRequirements C++17, gmp, mpfr

NeedsCompilation yes

Author Stéphane Laurent [aut, cre]

Repository CRAN

Date/Publication 2024-07-29 17:50:02 UTC

Contents

ESF *Evaluation of elementary symmetric functions*

Description

Evaluates an elementary symmetric function.

Usage

```
ESF(x, lambda)
```

Arguments

x	a numeric vector or a bigq vector
lambda	an integer partition, given as a vector of decreasing integers

Value

A number if x is numeric, a bigq rational number if x is a bigq vector.

Examples

```
x <- c(1, 2, 5/2)
lambda <- c(3, 1)
ESF(x, lambda)
library(gmp)
x <- c(as.bigq(1), as.bigq(2), as.bigq(5,2))
ESF(x, lambda)
```

`factorialSchurPol` *Factorial Schur polynomial*

Description

Computes a factorial Schur polynomial.

Usage

```
factorialSchurPol(n, lambda, a)
```

Arguments

`n`	number of variables
`lambda`	integer partition
`a`	vector of `bigq` numbers, or vector of elements coercible to `bigq` numbers; this vector corresponds to the sequence denoted by a in the reference paper, section **6th Variation** (in this paper a is a doubly infinite sequence, but in the case of a non-skew partition, the non-positive indices of this sequence are not involved); the length of this vector must be large enough (an error will be thrown if it is too small) but it is not easy to know the minimal possible length

Value

A `qspray` polynomial.

References

I.G. Macdonald. *Schur functions: theme and variations*. Publ. IRMA Strasbourg, 1992.

Examples

```
# for a=c(0, 0, ...), the factorial Schur polynomial is the Schur polynomial
n <- 3
lambda <- c(2, 2, 2)
a <- c(0, 0, 0, 0)
factorialSchurPoly <- factorialSchurPol(n, lambda, a)
schurPoly <- SchurPol(n, lambda)
factorialSchurPoly == schurPoly # should be TRUE
```

flaggedSchurPol *Flagged Schur polynomial*

Description

Computes a flagged Schur polynomial (which is not symmetric in general). See Chains in the Bruhat order for the definition.

Usage

```
flaggedSchurPol(lambda, a, b)
```

Arguments

lambda	integer partition
a, b	lower bounds and upper bounds, weakly increasing vectors of integers; lambda, a and b must have the same length

Value

A qspray polynomial.

Examples

```
lambda <- c(3, 2, 2)
n <- 3
a <- c(1, 1, 1); b <- c(n, n, n)
flaggedPoly <- flaggedSchurPol(lambda, a, b)
poly <- SchurPol(n, lambda)
flaggedPoly == poly # should be TRUE
```

flaggedSkewSchurPol *Flagged skew Schur polynomial*

Description

Computes a flagged skew Schur polynomial (which is not symmetric in general). See Schur polynomials (flagged) for the definition.

Usage

```
flaggedSkewSchurPol(lambda, mu, a, b)
```

Arguments

lambda, mu	integer partitions defining the skew partition: lambda is the outer partition and mu is the inner partition (so mu must be a subpartition of lambda)
a, b	lower bounds and upper bounds, weakly increasing vectors of integers; lambda, a and b must have the same length

Value

A qspray polynomial.

Examples

```
lambda <- c(3, 2, 2); mu <- c(2, 1)
n <- 3
a <- c(1, 1, 1); b <- c(n, n, n)
flaggedPoly <- flaggedSkewSchurPol(lambda, mu, a, b)
poly <- SkewSchurPol(n, lambda, mu)
flaggedPoly == poly # should be TRUE
```

HallLittlewoodPol *Hall-Littlewood polynomial*

Description

Hall-Littlewood polynomial of a given partition.

Usage

```
HallLittlewoodPol(n, lambda, which = "P")
```

Arguments

n	number of variables
lambda	integer partition
which	which Hall-Littlewood polynomial, "P" or "Q"

Value

The Hall-Littlewood polynomial in n variables of the integer partition lambda. This is a symbolicQspray polynomial with a unique parameter usually denoted by t and its coefficients are polynomial in this parameter. When substituting t with 0 in the Hall-Littlewood P-polynomials, one obtains the Schur polynomials.

HallPolynomials *Hall polynomials*

Description

Hall polynomials $g^{\lambda}_{\mu,\nu}(t)$ for given integer partitions μ and ν.

Usage

```
HallPolynomials(mu, nu)
```

Arguments

mu, nu	integer partitions

Value

A list of lists. Each of these lists has two elements: an integer partition λ in the field `lambda`, and a univariate `qspray` polynomial in the field `polynomial`, the Hall polynomial $g^{\lambda}_{\mu,\nu}(t)$. Every coefficient of a Hall polynomial is an integer.

Note

This function is slow.

Examples

```
HallPolynomials(c(2, 1), c(1, 1))
```

Jack *Evaluation of Jack polynomial - C++ implementation*

Description

Evaluates a Jack polynomial.

Usage

```
Jack(x, lambda, alpha)
```

Arguments

x	values of the variables, a vector of `bigq` numbers, or a vector that can be coerced as such (e.g. `c("2", "5/3")`)
lambda	an integer partition, given as a vector of decreasing integers
alpha	rational number, given as a string such as `"2/3"` or as a `bigq` number

Value

A bigq number.

Examples

```
Jack(c("1", "3/2", "-2/3"), lambda = c(3, 1), alpha = "1/4")
```

`JackCombination` *Symmetric polynomial in terms of Jack polynomials*

Description

Expression of a symmetric polynomial as a linear combination of Jack polynomials.

Usage

```
JackCombination(qspray, alpha, which = "J", check = TRUE)
```

Arguments

qspray	a qspray object or a symbolicQspray object defining a symmetric polynomial
alpha	Jack parameter, must be coercible to a bigq number
which	which Jack polynomials, "J", "P", "Q" or "C"
check	Boolean, whether to check the symmetry of qspray

Value

A list defining the combination. Each element of this list is a list with two elements: coeff, which is a bigq number if qspray is a qspray polynomial or a ratioOfQsprays if qspray is a symbolicQspray polynomial, and the second element of the list is lambda, an integer partition; then this list corresponds to the term coeff * JackPol(n, lambda, alpha, which), where n is the number of variables in the symmetric polynomial qspray.

`JackPol` *Jack polynomial - C++ implementation*

Description

Returns a Jack polynomial.

Usage

```
JackPol(n, lambda, alpha, which = "J")
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers
alpha	rational number, given as a string such as "2/3" or as a bigq number
which	which Jack polynomial, "J", "P", "Q", or "C"

Value

A qspray multivariate polynomial.

Examples

```
JackPol(3, lambda = c(3, 1), alpha = "2/5")
```

JackPolR *Jack polynomial*

Description

Returns the Jack polynomial.

Usage

```
JackPolR(n, lambda, alpha, algorithm = "DK", basis = "canonical", which = "J")
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers
alpha	parameter of the Jack polynomial, a number, possibly (and preferably) a bigq rational number
algorithm	the algorithm used, either "DK" or "naive"
basis	the polynomial basis for algorithm = "naive", either "canonical" or "MSF" (monomial symmetric functions); for algorithm = "DK" the canonical basis is always used and this parameter is ignored
which	which Jack polynomial, "J", "P" or "Q"; this argument is taken into account **only** if alpha is a bigq number and algorithm = "DK"

Value

A mvp multivariate polynomial (see mvp-package), or a qspray multivariate polynomial if alpha is a bigq rational number and algorithm = "DK", or a character string if basis = "MSF".

Examples

```
JackPolR(3, lambda = c(3,1), alpha = gmp::as.bigq(2,3),
                  algorithm = "naive")
JackPolR(3, lambda = c(3,1), alpha = 2/3, algorithm = "DK")
JackPolR(3, lambda = c(3,1), alpha = gmp::as.bigq(2,3), algorithm = "DK")
JackPolR(3, lambda = c(3,1), alpha= gmp::as.bigq(2,3),
        algorithm = "naive", basis = "MSF")
# when the Jack polynomial is a `qspray` object, you can
# evaluate it with `qspray::evalQspray`:
jack <- JackPolR(3, lambda = c(3, 1), alpha = gmp::as.bigq(2))
evalQspray(jack, c("1", "1/2", "3"))
```

`JackR` *Evaluation of Jack polynomials*

Description

Evaluates a Jack polynomial.

Usage

```
JackR(x, lambda, alpha, algorithm = "DK")
```

Arguments

`x`	numeric or complex vector or `bigq` vector
`lambda`	an integer partition, given as a vector of decreasing integers
`alpha`	ordinary number or `bigq` rational number
`algorithm`	the algorithm used, either `"DK"` (Demmel-Koev) or `"naive"`

Value

A numeric or complex scalar or a `bigq` rational number.

References

- I.G. Macdonald. *Symmetric Functions and Hall Polynomials*. Oxford Mathematical Monographs. The Clarendon Press Oxford University Press, New York, second edition, 1995.
- J. Demmel & P. Koev. *Accurate and efficient evaluation of Schur and Jack functions*. Mathematics of computations, vol. 75, n. 253, 223-229, 2005.
- *Jack polynomials*. https://www.symmetricfunctions.com/jack.htm

See Also

`JackPolR`

Examples

```
lambda <- c(2,1,1)
JackR(c(1/2, 2/3, 1), lambda, alpha = 3)
# exact value:
JackR(c(gmp::as.bigq(1,2), gmp::as.bigq(2,3), gmp::as.bigq(1)), lambda,
     alpha = gmp::as.bigq(3))
```

JackSymPol *Jack polynomial with symbolic Jack parameter*

Description

Returns the Jack polynomial with a symbolic Jack parameter.

Usage

```
JackSymPol(n, lambda, which = "J")
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers
which	which Jack polynomial, "J", "P", "Q", or "C"

Value

A symbolicQspray object.

Examples

```
JackSymPol(3, lambda = c(3, 1))
```

KostaFoulkesPolynomial *Kostka-Foulkes polynomial*

Description

Kostka-Foulkes polynomial for two given partitions.

Usage

```
KostaFoulkesPolynomial(lambda, mu)
```

Arguments

`lambda`, `mu`	integer partitions; in order for the Kostka-Foulkes polynomial to be non-zero, a necessary condition is that `lambda` and `mu` have the same weight; more precisely, `mu` must be dominated by `lambda`

Value

The Kostka-Foulkes polynomial associated to `lambda` and `mu`. This is a univariate `qspray` polynomial whose value at `1` is the Kostka number associated to `lambda` and `mu`.

`KostkaJackNumbers` *Kostka-Jack numbers with a given Jack parameter*

Description

Kostka numbers with Jack parameter, or Kostka-Jack numbers, for partitions of a given weight and a given Jack parameter.

Usage

```
KostkaJackNumbers(n, alpha = "1")
```

Arguments

`n`	positive integer, the weight of the partitions
`alpha`	the Jack parameter, a `bigq` number or an object coercible to a `bigq` number

Details

The Kostka-Jack number $K_{\lambda,\mu}(\alpha)$ is the coefficient of the monomial symmetric polynomial m_μ in the expression of the P-Jack polynomial $P_\lambda(\alpha)$ as a linear combination of monomial symmetric polynomials. For $\alpha = 1$ it is the ordinary Kostka number.

Value

The matrix of the Kostka-Jack numbers $K_{\lambda,\mu}(\alpha)$ given as character strings representing integers or fractions. The row names of this matrix encode the partitions λ and the column names encode the partitions μ

See Also

`KostkaJackNumbersWithGivenLambda`, `symbolicKostkaJackNumbers`, `skewKostkaJackNumbers`.

Examples

```
KostkaJackNumbers(4)
```

KostkaJackNumbersWithGivenLambda
Kostka-Jack numbers with a given partition λ

Description

Kostka numbers with Jack parameter, or Kostka-Jack numbers $K_{\lambda,\mu}(\alpha)$ for a given Jack parameter α and a given integer partition λ.

Usage

```
KostkaJackNumbersWithGivenLambda(lambda, alpha, output = "vector")
```

Arguments

`lambda`	integer partition
`alpha`	the Jack parameter, a `bigq` number or anything coercible to a `bigq` number
`output`	the format of the output, either `"vector"` or `"list"`

Details

The Kostka-Jack number $K_{\lambda,\mu}(\alpha)$ is the coefficient of the monomial symmetric polynomial m_μ in the expression of the P-Jack polynomial $P_\lambda(\alpha)$ as a linear combination of monomial symmetric polynomials. For $\alpha = 1$ it is the ordinary Kostka number.

Value

If `output="vector"`, this function returns a named vector. This vector is made of the non-zero (i.e. positive) Kostka-Jack numbers $K_{\lambda,\mu}(\alpha)$ given as character strings and its names encode the partitions μ. If `ouput="list"`, this function returns a list of lists. Each of these lists has two elements. The first one is named `mu` and is an integer partition, and the second one is named `value` and is a `bigq` rational number, the Kostka-Jack number $K_{\lambda,\mu}(\alpha)$.

See Also

`KostkaJackNumbers`, `symbolicKostkaJackNumbersWithGivenLambda`.

Examples

```
KostkaJackNumbersWithGivenLambda(c(3, 2), alpha = "2")
```

LRmult *Littlewood-Richardson rule for multiplication*

Description

Expression of the product of two Schur polynomials as a linear combination of Schur polynomials.

Usage

```
LRmult(mu, nu, output = "dataframe")
```

Arguments

mu, nu	integer partitions, given as vectors of decreasing integers
output	the type of the output, `"dataframe"` or `"list"`

Value

This computes the expression of the product of the two Schur polynomials associated to `mu` and `nu` as a linear combination of Schur polynomials. If `output="dataframe"`, the output is a dataframe with two columns: the column `coeff` gives the coefficients of this linear combination, these are positive integers, and the column `lambda` gives the partitions defining the Schur polynomials of this linear combination as character strings, e.g. the partition `c(4, 3, 1)` is encoded by the character string `"[4, 3, 1]"`. If `output="list"`, the output is a list of lists with two elements. Each of these lists with two elements corresponds to a term of the linear combination: the first element, named `coeff`, is the coefficient, namely the Littlewood-Richardson coefficient $c_{\mu,\nu}^{\lambda}$, where λ is the integer partition given in the second element of the list, named `lambda`, which defines the Schur polynomial of the linear combination.

Examples

```
library(jack)
mu <- c(2, 1)
nu <- c(3, 2, 1)
LR <- LRmult(mu, nu, output = "list")
LRterms <- lapply(LR, function(lr) {
  lr[["coeff"]] * SchurPol(3, lr[["lambda"]])
})
smu_times_snu <- Reduce(`+`, LRterms)
smu_times_snu == SchurPol(3, mu) * SchurPol(3, nu) # should be TRUE
```

LRskew *Littlewood-Richardson rule for skew Schur polynomial*

Description

Expression of a skew Schur polynomial as a linear combination of Schur polynomials.

Usage

```
LRskew(lambda, mu, output = "dataframe")
```

Arguments

lambda, mu	integer partitions defining the skew partition: lambda is the outer partition and mu is the inner partition (so mu must be a subpartition of lambda)
output	the type of the output, "dataframe" or "list"

Value

This computes the expression of the skew Schur polynomial associated to the skew partition defined by lambda and mu as a linear combination of Schur polynomials. Every coefficient of this linear combination is a positive integer, a so-called Littlewood-Richardson coefficient. If output="dataframe", the output is a dataframe with two columns: the column coeff gives the coefficients of this linear combination, and the column nu gives the partitions defining the Schur polynomials of this linear combination as character strings, e.g. the partition c(4, 3, 1) is given by "[4, 3, 1]". If output="list", the output is a list of lists with two elements. Each of these lists with two elements corresponds to a term of the linear combination: the first element, named coeff, is the coefficient, namely the Littlewood-Richardson coefficient $c_{\mu,\nu}^{\lambda}$, where ν is the integer partition given in the second element of the list, named nu, which defines the Schur polynomial of the linear combination.

Examples

```
library(jack)
LRskew(lambda = c(4, 2, 1), mu = c(3, 1))
```

MacdonaldPol *Macdonald polynomial*

Description

Returns the Macdonald polynomial associated to the given integer partition.

Usage

```
MacdonaldPol(n, lambda, which = "P")
```

Arguments

n	number of variables, a positive integer
lambda	integer partition
which	which Macdonald polynomial, "P", "Q", or "J"

Value

A symbolicQspray multivariate polynomial, the Macdonald polynomial associated to the integer partition lambda. It has two parameters usually denoted by q and t. Substituting q with 0 yields the Hall-Littlewood polynomials.

---

modifiedMacdonaldPol *Modified Macdonald polynomial*

Description

Returns the modified Macdonald polynomial associated to a given integer partition.

Usage

```
modifiedMacdonaldPol(n, mu)
```

Arguments

n	number of variables, a positive integer
mu	integer partition

Value

A symbolicQspray multivariate polynomial, the modified Macdonald polynomial associated to the integer partition mu. It has two parameters and its coefficients are polynomials in these parameters.

MSF *Evaluation of monomial symmetric functions*

Description

Evaluates a monomial symmetric function.

Usage

```
MSF(x, lambda)
```

Arguments

x	a numeric vector or a bigq vector
lambda	an integer partition, given as a vector of decreasing integers

Value

A number if x is numeric, a bigq rational number if x is a bigq vector.

Examples

```
x <- c(1, 2, 5/2)
lambda <- c(3, 1)
MSF(x, lambda)
library(gmp)
x <- c(as.bigq(1), as.bigq(2), as.bigq(5,2))
MSF(x, lambda)
```

---

qtKostkaPolynomials *qt-Kostka polynomials*

Description

qt-Kostka polynomials, aka Kostka-Macdonald polynomials.

Usage

```
qtKostkaPolynomials(mu)
```

Arguments

mu	integer partition

Value

A list. The qt-Kostka polynomials are usually denoted by $K_{\lambda,\mu}(q,t)$ where q and t denote the two variables and λ and μ are two integer partitions. One obtains the Kostka-Foulkes polynomials by substituting q with 0. For a given partition μ, the function returns the polynomials $K_{\lambda,\mu}(q,t)$ as qspray objects for all partitions λ of the same weight as μ. The generated list is a list of lists with two elements: the integer partition λ and the polynomial.

qtSkewKostkaPolynomials *Skew qt-Kostka polynomials*

Description

Skew qt-Kostka polynomials associated to a given skew partition.

Usage

```
qtSkewKostkaPolynomials(lambda, mu)
```

Arguments

lambda, mu	integer partitions defining the skew partition: lambda is the outer partition and mu is the inner partition (so mu must be a subpartition of lambda)

Value

A list. The skew qt-Kostka polynomials are usually denoted by $K_{\lambda/\mu,\nu}(q,t)$ where q and t denote the two variables, λ and μ are the two integer partitions defining the skew partition, and ν is an integer partition. One obtains the skew Kostka-Foulkes polynomials by substituting q with 0. For given partitions λ and μ, the function returns the polynomials $K_{\lambda/\mu,\nu}(q,t)$ as qspray objects for all partitions ν of the same weight as the skew partition. The generated list is a list of lists with two elements: the integer partition ν and the polynomial.

Schur *Evaluation of Schur polynomial - C++ implementation*

Description

Evaluates a Schur polynomial. The Schur polynomials are the Jack P-polynomials with Jack parameter $\alpha = 1$.

Usage

```
Schur(x, lambda)
```

Arguments

x	values of the variables, a vector of bigq numbers, or a vector that can be coerced as such (e.g. c("2", "5/3"))
lambda	an integer partition, given as a vector of decreasing integers

Value

A `bigq` number.

Examples

```
Schur(c("1", "3/2", "-2/3"), lambda = c(3, 1))
```

`SchurCombination` *Symmetric polynomial in terms of the Schur polynomials*

Description

Expression of a symmetric polynomial as a linear combination of some Schur polynomials.

Usage

```
SchurCombination(qspray, check = TRUE)
```

Arguments

`qspray`	a `qspray` object defining a symmetric polynomial
`check`	Boolean, whether to check the symmetry of `qspray`

Value

A list defining the combination. Each element of this list is a list with two elements: `coeff`, a `bigq` number, and `lambda`, an integer partition; then this list corresponds to the term `coeff * SchurPol(n, lambda)`, where `n` is the number of variables in the symmetric polynomial.

See Also

`JackCombination`.

`SchurPol` *Schur polynomial - C++ implementation*

Description

Returns a Schur polynomial. The Schur polynomials are the Jack P-polynomials with Jack parameter $\alpha = 1$.

Usage

```
SchurPol(n, lambda)
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers

Value

A qspray multivariate polynomial.

Examples

```
( schur <- SchurPol(3, lambda = c(3, 1)) )
schur == JackPol(3, lambda = c(3, 1), alpha = "1", which = "P")
```

SchurPolR *Schur polynomial*

Description

Returns the Schur polynomial.

Usage

```
SchurPolR(n, lambda, algorithm = "DK", basis = "canonical", exact = TRUE)
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers
algorithm	the algorithm used, either "DK" or "naive"
basis	the polynomial basis for algorithm = "naive", either "canonical" or "MSF" (monomial symmetric functions); for algorithm = "DK" the canonical basis is always used and this parameter is ignored
exact	logical, whether to use exact arithmetic

Value

A mvp multivariate polynomial (see mvp-package), or a qspray multivariate polynomial if exact = TRUE and algorithm = "DK", or a character string if basis = "MSF".

Examples

```
SchurPolR(3, lambda = c(3,1), algorithm = "naive")
SchurPolR(3, lambda = c(3,1), algorithm = "DK")
SchurPolR(3, lambda = c(3,1), algorithm = "DK", exact = FALSE)
SchurPolR(3, lambda = c(3,1), algorithm = "naive", basis = "MSF")
```

SchurR *Evaluation of Schur polynomials*

Description

Evaluates a Schur polynomial.

Usage

```
SchurR(x, lambda, algorithm = "DK")
```

Arguments

x	numeric or complex vector or bigq vector
lambda	an integer partition, given as a vector of decreasing integers
algorithm	the algorithm used, either "DK" (Demmel-Koev) or "naive"

Value

A numeric or complex scalar or a bigq rational number.

References

J. Demmel & P. Koev. *Accurate and efficient evaluation of Schur and Jack functions*. Mathematics of computations, vol. 75, n. 253, 223-229, 2005.

See Also

SchurPolR

Examples

```
x <- c(2,3,4)
SchurR(x, c(2,1,1))
prod(x) * sum(x)
```

`SkewFactorialSchurPol` *Skew factorial Schur polynomial*

Description

Computes the skew factorial Schur polynomial associated to a given skew partition.

Usage

```
SkewFactorialSchurPol(n, lambda, mu, a, i0)
```

Arguments

`n`	number of variables
`lambda`, `mu`	integer partitions defining the skew partition: `lambda` is the outer partition and `mu` is the inner partition (so `mu` must be a subpartition of `lambda`)
`a`	vector of `bigq` numbers, or vector of elements coercible to `bigq` numbers; this vector corresponds to the sequence denoted by a in the reference paper, section **6th Variation** (in this paper a is a doubly infinite sequence, but only a finite number of indices are not involved); the length of this vector must be large enough (an error will be thrown if it is too small) but it is not easy to know the minimal possible length
`i0`	positive integer, the index of `a` that must be considered as the zero index of the sequence denoted by a in the reference paper

Value

A `qspray` polynomial.

References

I.G. Macdonald. *Schur functions: theme and variations*. Publ. IRMA Strasbourg, 1992.

Examples

```
# for a=c(0, 0, ...), the skew factorial Schur polynomial is the
# skew Schur polynomial; let's check
n <- 4
lambda <- c(3, 3, 2, 2); mu <- c(2, 2)
a <- rep(0, 9)
i0 <- 3
skewFactorialSchurPoly <- SkewFactorialSchurPol(n, lambda, mu, a, i0)
skewSchurPoly <- SkewSchurPol(n, lambda, mu)
skewFactorialSchurPoly == skewSchurPoly # should be TRUE
```

SkewHallLittlewoodPol *Skew Hall-Littlewood polynomial*

Description

Returns the skew Hall-Littlewood polynomial associated to the given skew partition.

Usage

```
SkewHallLittlewoodPol(n, lambda, mu, which = "P")
```

Arguments

n	number of variables, a positive integer
lambda, mu	integer partitions defining the skew partition: lambda is the outer partition and mu is the inner partition (so mu must be a subpartition of lambda)
which	which skew Hall-Littlewood polynomial, "P" or "Q"

Value

A symbolicQspray multivariate polynomial, the skew Hall-Littlewood polynomial associated to the skew partition defined by lambda and mu. It has a single parameter usually denoted by t and its coefficients are polynomial in this parameter. When substituting t with 0 in the skew Hall-Littlewood P-polynomials, one obtains the skew Schur polynomials.

Examples

```
n <- 3; lambda <- c(3, 2, 1); mu <- c(1, 1)
skewHLpoly <- SkewHallLittlewoodPol(n, lambda, mu)
skewSchurPoly <- SkewSchurPol(n, lambda, mu)
substituteParameters(skewHLpoly, 0) == skewSchurPoly # should be TRUE
```

SkewJackPol *Skew Jack polynomial*

Description

Computes a skew Jack polynomial with a given Jack parameter.

Usage

```
SkewJackPol(n, lambda, mu, alpha, which = "J")
```

Arguments

n	positive integer, the number of variables
lambda	outer integer partition of the skew partition
mu	inner integer partition of the skew partition; it must be a subpartition of lambda
alpha	the Jack parameter, any object coercible to a bigq number
which	which skew Jack polynomial, "J", "P", "Q" or "C"

Value

A qspray polynomial.

See Also

SkewJackSymPol.

Examples

```
SkewJackPol(3, c(3,1), c(2), "2")
```

SkewJackSymPol *Skew Jack polynomial with symbolic Jack parameter*

Description

Computes a skew Jack polynomial with a symbolic Jack parameter.

Usage

```
SkewJackSymPol(n, lambda, mu, which = "J")
```

Arguments

n	positive integer, the number of variables
lambda	outer integer partition of the skew partition
mu	inner integer partition of the skew partition; it must be a subpartition of lambda
which	which skew Jack polynomial, "J", "P", "Q" or "C"

Value

A symbolicQspray polynomial.

Examples

```
SkewJackSymPol(3, c(3,1), c(2))
```

SkewKostkaFoulkesPolynomial *Skew Kostka-Foulkes polynomial*

Description

Computes a skew Kostka-Foulkes polynomial.

Usage

```
SkewKostkaFoulkesPolynomial(lambda, mu, nu)
```

Arguments

lambda, mu	integer partitions defining the skew partition: lambda is the outer partition and mu is the inner partition (so mu must be a subpartition of lambda)
nu	integer partition; the condition sum(nu)==sum(lambda)-sum(mu) is necessary in order to get a non-zero polynomial

Value

The skew Kostka-Foulkes polynomial associated to the skew partitiion defined by lambda and mu and to the partition nu. This is a univariate qspray polynomial whose value at 1 is the skew Kostka number associated to the skew partition defined by lambda and mu and to the partition nu.

skewKostkaJackNumbers *Skew Kostka-Jack numbers with given Jack parameter*

Description

Skew Kostka-Jack numbers associated to a given skew partition and a given Jack parameter.

Usage

```
skewKostkaJackNumbers(lambda, mu, alpha = NULL, output = "vector")
```

Arguments

lambda, mu	integer partitions defining the skew partition: lambda is the outer partition and mu is the inner partition (so mu must be a subpartition of lambda)
alpha	the Jack parameter, a bigq number or an object coercible to a bigq number; setting alpha=NULL is equivalent to set alpha=1
output	the format of the output, either "vector" or "list"

Details

The skew Kostka-Jack number $K_{\lambda/\mu,\nu}(\alpha)$ is the coefficient of the monomial symmetric polynomial m_ν in the expression of the skew P-Jack polynomial $P_{\lambda/\mu}(\alpha)$ as a linear combination of monomial symmetric polynomials. For $\alpha = 1$ it is the ordinary skew Kostka number.

Value

If `output="vector"`, the function returns a named vector. This vector is made of the non-zero skew Kostka-Jack numbers $K_{\lambda/\mu,\nu}(\alpha)$ given as character strings and its names encode the partitions ν. If `ouput="list"`, the function returns a list. Each element of this list is a named list with two elements: an integer partition ν in the field named `"nu"`, and the corresponding skew Kostka-Jack number $K_{\lambda/\mu,\nu}(\alpha)$ in the field named `"value"`. Only the non-null skew Kostka-Jack numbers are provided by this list.

Note

The skew Kostka-Jack numbers $K_{\lambda/\mu,\nu}(\alpha)$ are well defined when the Jack parameter α is zero, however this function does not work with `alpha=0`. A possible way to get the skew Kostka-Jack numbers $K_{\lambda/\mu,\nu}(0)$ is to use the function `symbolicSkewKostkaJackNumbers` to get the skew Kostka-Jack numbers with a symbolic Jack parameter α, and then to substitute α with 0.

See Also

`symbolicSkewKostkaJackNumbers`.

Examples

```
skewKostkaJackNumbers(c(4,2,2), c(2,2))
```

`SkewMacdonaldPol` *Skew Macdonald polynomial*

Description

Returns the skew Macdonald polynomial associated to the given skew partition.

Usage

```
SkewMacdonaldPol(n, lambda, mu, which = "P")
```

Arguments

`n`	number of variables, a positive integer
`lambda`, `mu`	integer partitions defining the skew partition: `lambda` is the outer partition and `mu` is the inner partition (so `mu` must be a subpartition of `lambda`)
`which`	which skew Macdonald polynomial, `"P"`, `"Q"` or `"J"`

Value

A `symbolicQspray` multivariate polynomial, the skew Macdonald polynomial associated to the skew partition defined by `lambda` and `mu`. It has two parameters usually denoted by q and t. Substituting q with 0 yields the skew Hall-Littlewood polynomials.

`SkewSchurPol` *Skew Schur polynomial*

Description

Returns the skew Schur polynomial.

Usage

```
SkewSchurPol(n, lambda, mu)
```

Arguments

`n`	number of variables, a positive integer
`lambda`, `mu`	integer partitions defining the skew partition: `lambda` is the outer partition and `mu` is the inner partition (so `mu` must be a subpartition of `lambda`)

Details

The computation is performed with the help of the Littlewood-Richardson rule (see `LRskew`).

Value

A `qspray` multivariate polynomial, the skew Schur polynomial associated to the skew partition defined by `lambda` and `mu`.

Examples

```
SkewSchurPol(3, lambda = c(3, 2, 1), mu = c(1, 1))
```

symbolicJackCombination

Symmetric polynomial in terms of symbolic Jack polynomials

Description

Expression of a symmetric polynomial as a linear combination of Jack polynomials with a symbolic Jack parameter.

Usage

```
symbolicJackCombination(qspray, which = "J", check = TRUE)
```

Arguments

qspray	a qspray object or a symbolicQspray object defining a symmetric polynomial
which	which Jack polynomials, "J", "P", "Q" or "C"
check	Boolean, whether to check the symmetry

Value

A list defining the combination. Each element of this list is a list with two elements: coeff, a bigq number, and lambda, an integer partition; then this list corresponds to the term coeff * JackSymPol(n, lambda, which), where n is the number of variables in the symmetric polynomial.

symbolicKostkaJackNumbers

Kostka-Jack numbers with symbolic Jack parameter

Description

Kostka-Jack numbers with a symbolic Jack parameter for integer partitions of a given weight.

Usage

```
symbolicKostkaJackNumbers(n)
```

Arguments

n	positive integer, the weight of the partitions

Value

A named list of named lists of ratioOfQsprays objects. Denoting the Kostka-Jack numbers by $K_{\lambda,\mu}(\alpha)$, the names of the outer list correspond to the partitions λ, and the names of the inner lists correspond to the partitions μ.

See Also

KostkaJackNumbers, symbolicKostkaJackNumbersWithGivenLambda.

Examples

```
symbolicKostkaJackNumbers(3)
```

---

symbolicKostkaJackNumbersWithGivenLambda

Kostka-Jack numbers with symbolic Jack parameter for a given λ

Description

Kostka-Jack numbers $K_{\lambda,\mu}(\alpha)$ with a symbolic Jack parameter α for a given integer partition λ.

Usage

```
symbolicKostkaJackNumbersWithGivenLambda(lambda)
```

Arguments

lambda	integer partition

Value

A named list of `ratioOfQsprays` objects. The elements of this list are the Kostka-Jack numbers $K_{\lambda,\mu}(\alpha)$ and its names correspond to the partitions μ.

See Also

KostkaJackNumbersWithGivenLambda, symbolicKostkaJackNumbers.

Examples

```
symbolicKostkaJackNumbersWithGivenLambda(c(3, 1))
```

symbolicSkewKostkaJackNumbers *Skew Kostka-Jack numbers with symbolic Jack parameter*

Description

Skew Kostka-Jack numbers associated to a given skew partition with a symbolic Jack parameter.

Usage

```
symbolicSkewKostkaJackNumbers(lambda, mu)
```

Arguments

lambda, mu	integer partitions defining the skew partition: lambda is the outer partition and mu is the inner partition (so mu must be a subpartition of lambda)

Value

The function returns a list. Each element of this list is a named list with two elements: an integer partition ν in the field named "nu", and the corresponding skew Kostka number $K_{\lambda/\mu,\nu}(\alpha)$ in the field named "value", a ratioOfQsprays object.

Examples

```
symbolicSkewKostkaJackNumbers(c(4,2,2), c(2,2))
```

tSchurPol *t-Schur polynomial*

Description

Returns the t-Schur polynomial associated to the given partition.

Usage

```
tSchurPol(n, lambda)
```

Arguments

n	number of variables, a positive integer
lambda	integer partition

Value

A `symbolicQspray` multivariate polynomial, the t-Schur polynomial associated to `lambda`. It has a single parameter usually denoted by t and its coefficients are polynomials in this parameter. Substituting t with 0 yields the Schur polynomials.

Note

The name "t-Schur polynomial" is taken from Wheeler and Zinn-Justin's paper *Hall polynomials, inverse Kostka polynomials and puzzles*.

`tSkewSchurPol` *Skew t-Schur polynomial*

Description

Returns the skew t-Schur polynomial associated to the given skew partition.

Usage

```
tSkewSchurPol(n, lambda, mu)
```

Arguments

`n`	number of variables, a positive integer
`lambda`, `mu`	integer partitions defining the skew partition: `lambda` is the outer partition and `mu` is the inner partition (so `mu` must be a subpartition of `lambda`)

Value

A `symbolicQspray` multivariate polynomial, the skew t-Schur polynomial associated to the skew partition defined by `lambda` and `mu`. It has a single parameter usually denoted by t and its coefficients are polynomials in this parameter. Substituting t with 0 yields the skew Schur polynomials.

`Zonal` *Evaluation of zonal polynomial - C++ implementation*

Description

Evaluates a zonal polynomial. The zonal polynomials are the Jack C-polynomials with Jack parameter $\alpha = Z$.

Usage

```
Zonal(x, lambda)
```

Arguments

x	values of the variables, a vector of bigq numbers, or a vector that can be coerced as such (e.g. c("2", "5/3"))
lambda	an integer partition, given as a vector of decreasing integers

Value

A bigq number.

Examples

```
Zonal(c("1", "3/2", "-2/3"), lambda = c(3, 1))
```

ZonalPol *Zonal polynomial - C++ implementation*

Description

Returns a zonal polynomial. The zonal polynomials are the Jack C-polynomials with Jack parameter $\alpha = Z$.

Usage

```
ZonalPol(n, lambda)
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers

Value

A qspray multivariate polynomial.

Examples

```
( zonal <- ZonalPol(3, lambda = c(3, 1)) )
zonal == JackPol(3, lambda = c(3, 1), alpha = "2", which = "C")
```

ZonalPolR *Zonal polynomial*

Description

Returns the zonal polynomial.

Usage

```
ZonalPolR(n, lambda, algorithm = "DK", basis = "canonical", exact = TRUE)
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers
algorithm	the algorithm used, either "DK" or "naive"
basis	the polynomial basis for algorithm = "naive", either "canonical" or "MSF" (monomial symmetric functions); for algorithm = "DK" the canonical basis is always used and this parameter is ignored
exact	logical, whether to get rational coefficients

Value

A mvp multivariate polynomial (see mvp-package), or a qspray multivariate polynomial if exact = TRUE and algorithm = "DK", or a character string if basis = "MSF".

Examples

```
ZonalPolR(3, lambda = c(3,1), algorithm = "naive")
ZonalPolR(3, lambda = c(3,1), algorithm = "DK")
ZonalPolR(3, lambda = c(3,1), algorithm = "DK", exact = FALSE)
ZonalPolR(3, lambda = c(3,1), algorithm = "naive", basis = "MSF")
```

ZonalQ *Evaluation of zonal quaternionic polynomial - C++ implementation*

Description

Evaluates a zonal quaternionic polynomial. The quaternionic zonal polynomials are the Jack C-polynomials with Jack parameter $\alpha = 1/Z$.

Usage

```
ZonalQ(x, lambda)
```

Arguments

x	values of the variables, a vector of bigq numbers, or a vector that can be coerced as such (e.g. c("2", "5/3"))
lambda	an integer partition, given as a vector of decreasing integers

Value

A bigq number.

Examples

```
ZonalQ(c("1", "3/2", "-2/3"), lambda = c(3, 1))
```

ZonalQPol *Quaternionic zonal polynomial - C++ implementation*

Description

Returns a quaternionic zonal polynomial. The quaternionic zonal polynomials are the Jack C-polynomials with Jack parameter $\alpha = 1/Z$.

Usage

```
ZonalQPol(n, lambda)
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers

Value

A qspray multivariate polynomial.

Examples

```
( zonalQ <- ZonalQPol(3, lambda = c(3, 1)) )
zonalQ == JackPol(3, lambda = c(3, 1), alpha = "1/2", which = "C")
```

ZonalQPolR *Quaternionic zonal polynomial*

Description

Returns the quaternionic (or symplectic) zonal polynomial.

Usage

```
ZonalQPolR(n, lambda, algorithm = "DK", basis = "canonical", exact = TRUE)
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers
algorithm	the algorithm used, either "DK" or "naive"
basis	the polynomial basis for algorithm = "naive", either "canonical" or "MSF" (monomial symmetric functions); for algorithm = "DK" the canonical basis is always used and this parameter is ignored
exact	logical, whether to get rational coefficients

Value

A mvp multivariate polynomial (see mvp-package), or a qspray multivariate polynomial if exact = TRUE and algorithm = "DK", or a character string if basis = "MSF".

Examples

```
ZonalQPolR(3, lambda = c(3,1), algorithm = "naive")
ZonalQPolR(3, lambda = c(3,1), algorithm = "DK")
ZonalQPolR(3, lambda = c(3,1), algorithm = "DK", exact = FALSE)
ZonalQPolR(3, lambda = c(3,1), algorithm = "naive", basis = "MSF")
```

ZonalQR *Evaluation of quaternionic zonal polynomials*

Description

Evaluates a quaternionic (or symplectic) zonal polynomial.

Usage

```
ZonalQR(x, lambda, algorithm = "DK")
```

Arguments

x	numeric or complex vector or bigq vector
lambda	an integer partition, given as a vector of decreasing integers
algorithm	the algorithm used, either "DK" (Demmel-Koev) or "naive"

Value

A numeric or complex scalar or a bigq rational number.

References

F. Li, Y. Xue. *Zonal polynomials and hypergeometric functions of quaternion matrix argument*. Comm. Statist. Theory Methods, 38 (8), 1184-1206, 2009

See Also

ZonalQPolR

Examples

```
lambda <- c(2,2)
ZonalQR(c(3,1), lambda)
ZonalQR(c(gmp::as.bigq(3),gmp::as.bigq(1)), lambda)
##
x <- c(3,1)
ZonalQR(x, c(1,1)) + ZonalQR(x, 2) # sum(x)^2
ZonalQR(x, 3) + ZonalQR(x, c(2,1)) + ZonalQR(x, c(1,1,1)) # sum(x)^3
```

ZonalR *Evaluation of zonal polynomials*

Description

Evaluates a zonal polynomial.

Usage

```
ZonalR(x, lambda, algorithm = "DK")
```

Arguments

x	numeric or complex vector or bigq vector
lambda	an integer partition, given as a vector of decreasing integers
algorithm	the algorithm used, either "DK" (Demmel-Koev) or "naive"

Value

A numeric or complex scalar or a bigq rational number.

References

- Robb Muirhead. *Aspects of multivariate statistical theory*. Wiley series in probability and mathematical statistics. Probability and mathematical statistics. John Wiley & Sons, New York, 1982.
- Akimichi Takemura. *Zonal Polynomials*, volume 4 of Institute of Mathematical Statistics Lecture Notes – Monograph Series. Institute of Mathematical Statistics, Hayward, CA, 1984.
- Lin Jiu & Christoph Koutschan. *Calculation and Properties of Zonal Polynomials*. http://koutschan.de/data/zonal/

See Also

ZonalPolR

Examples

```
lambda <- c(2,2)
ZonalR(c(1,1), lambda)
ZonalR(c(gmp::as.bigq(1),gmp::as.bigq(1)), lambda)
##
x <- c(3,1)
ZonalR(x, c(1,1)) + ZonalR(x, 2) # sum(x)^2
ZonalR(x, 3) + ZonalR(x, c(2,1)) + ZonalR(x, c(1,1,1)) # sum(x)^3
```

Index