Package 'jack'

May 3, 2024

Type Package

Title Jack, Zonal, and Schur Polynomials

Version 6.0.0

Date 2024-05-03

Author Stéphane Laurent

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Symbolic calculation and evaluation of the Jack polynomials, zonal polynomials, and Schur polynomials. Mainly based on Demmel & Koev's paper (2006) <doi:10.1090/S0025-5718-05-01780-1>. Zonal polynomials and Schur polynomials are particular cases of Jack polynomials. Zonal polynomials appear in random matrix theory. Schur polynomials appear in the field of combinatorics. The package can also compute the skew Schur polynomials.

License GPL-3

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

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

SystemRequirements C++17, gmp, mpfr

Depends qspray (>= 3.0.0), symbolicQspray

Imports DescTools, gmp, multicool, mvp, partitions, Rcpp, spray, utils

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

Suggests testthat, syt

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-05-03 06:30:02 UTC

R topics documented:

ESF . . . 2
Jack . . . 3
JackPol . . . 3
JackPolR . . . 4
JackR . . . 5
JackSymPol . . . 6
KostkaNumbers . . . 6
LRmult . . . 7
LRskew . . . 8
MSF . . . 8
Schur . . . 9
SchurPol . . . 10
SchurPolR . . . 10
SchurR . . . 11
SkewSchurPol . . . 12
Zonal . . . 12
ZonalPol . . . 13
ZonalPolR . . . 13
ZonalQ . . . 14
ZonalQPol . . . 15
ZonalQPolR . . . 15
ZonalQR . . . 16
ZonalR . . . 17

Index **19**

`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)
```

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

Description

Evaluates the 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")
```

JackPol *Jack polynomial - C++ implementation*

Description

Returns the 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 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))
```

KostkaNumbers *Kostka numbers*

Description

The Kostka numbers for partitions of a given weight.

Usage

```
KostkaNumbers(n)
```

Arguments

n	positive integer, the weight of the partitions

Value

A matrix of integers.

Examples

```
KostkaNumbers(4)
```

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 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, 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 given by `"4, 3, 1"`. If `output="list"`, the output is a list with two fields: the field `coeff` is the vector made of the coefficients of the linear combination, and the field `lambda` is the list of partitions defining the Schur polynomials of the linear combination given as integer vectors.

Examples

```
library(jack)
mu <- c(2, 1)
nu <- c(3, 2, 1)
LR <- LRmult(mu, nu, output = "list")
LRcoeffs <- LR$coeff
LRparts <- LR$lambda
LRterms <- lapply(1:length(LRcoeffs), function(i) {
  LRcoeffs[i] * SchurPol(3, LRparts[[i]])
})
smu_times_snu <- Reduce(`+`, LRterms)
smu_times_snu == SchurPol(3, mu) * SchurPol(3, nu)
```

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. 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 with two fields: the field coeff is the vector made of the coefficients of the linear combination, and the field nu is the list of partitions defining the Schur polynomials of the linear combination given as integer vectors.

Examples

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

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)
```

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

Description

Evaluates the Schur polynomial.

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))
```

SchurPol *Schur polynomial - C++ implementation*

Description

Returns the Schur polynomial.

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)
```

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))
```

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

Description

Evaluates the zonal polynomial.

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 the zonal polynomial.

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 the zonal quaternionic polynomial.

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 the quaternionic zonal polynomial.

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

bigq, *2*, *4*, *5*, *9*, *11*, *16*, *17*

ESF, 2

Jack, 3
JackPol, 3
JackPolR, 4, *5*
JackR, 5
JackSymPol, 6

KostkaNumbers, 6

LRmult, 7
LRskew, 8, *12*

MSF, 8
mvp-package, *4*, *11*, *14*, *16*

Schur, 9
SchurPol, 10
SchurPolR, 10, *11*
SchurR, 11
SkewSchurPol, 12

Zonal, 12
ZonalPol, 13
ZonalPolR, 13, *17*
ZonalQ, 14
ZonalQPol, 15
ZonalQPolR, 15, *16*
ZonalQR, 16
ZonalR, 17