Package ‘jack’

September 16, 2019

Type Package

Title Jack, Zonal, and Schur Polynomials

Version 1.1.1

Date 2019-09-16

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.

License GPL-3

Encoding UTF-8

LazyData true

Imports partitions, DescTools, gmp, mvp, multicool

RoxygenNote 6.1.1

Suggests testthat

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

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

NeedsCompilation no

Repository CRAN

Date/Publication 2019-09-16 09:50:02 UTC

R topics documented:

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)
```

`ESFpoly` *Elementary symmetric function*

Description

Returns an elementary symmetric function as a polynomial.

Usage

```
ESFpoly(m, lambda)
```

Arguments

`m`	integer, the number of variables
`lambda`	an integer partition, given as a vector of decreasing integers

Value

A polynomial (`mvp` object; see mvp-package).

Examples

```
ESFpoly(3, c(3,1))
```

`Jack` *Evaluation of Jack polynomials*

Description

Evaluates a Jack polynomial.

Usage

```
Jack(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`	positive 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.math.upenn.edu/~peal/polynomials/jack.htm`

See Also

`JackPol`

Examples

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

`JackPol` *Jack polynomial*

Description

Returns the Jack polynomial.

Usage

```
JackPol(n, lambda, alpha, algorithm = "DK", basis = "canonical")
```

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, always a positive number for `algorithm = "DK"`, a positive number or a positive `bigq` rational number for `algorithm = "naive"`
`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

Value

A polynomial (`mvp` object; see mvp-package) or a character string if `basis = "MSF"`.

Examples

```
JackPol(3, lambda = c(3,1), alpha = gmp::as.bigq(2,3),
                  algorithm = "naive")
JackPol(3, lambda = c(3,1), alpha = 2/3, algorithm = "DK")
JackPol(3, lambda = c(3,1), alpha= gmp::as.bigq(2,3),
        algorithm = "naive", basis = "MSF")
```

`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)
```

`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)
```

MSFpoly *Monomial symmetric function*

Description

Returns a monomial symmetric function as a polynomial.

Usage

```
MSFpoly(m, lambda)
```

Arguments

m	integer, the number of variables
lambda	an integer partition, given as a vector of decreasing integers

Value

A polynomial (mvp object; see mvp-package).

Examples

```
MSFpoly(3, c(3,1))
```

Schur *Evaluation of Schur polynomials*

Description

Evaluates a Schur polynomial.

Usage

```
Schur(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

`SchurPol`

Examples

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

SchurPol *Schur polynomial*

Description

Returns the Schur polynomial.

Usage

```
SchurPol(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 when using algorithm = "naive"; ignored if algorithm = "DK"

Value

A polynomial (mvp object; see mvp-package) or a character string if basis = "MSF".

Examples

```
SchurPol(3, lambda = c(3,1), algorithm = "naive")
SchurPol(3, lambda = c(3,1), algorithm = "DK")
SchurPol(3, lambda = c(3,1), algorithm = "naive", basis = "MSF")
```

Zonal *Evaluation of zonal polynomials*

Description

Evaluates a zonal polynomial.

Usage

```
Zonal(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/zonal.pdf`

See Also

`ZonalPol`

Examples

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

`ZonalPol` *Zonal polynomial*

Description

Returns the zonal polynomial.

Usage

```
ZonalPol(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 when using `algorithm = "naive"`; ignored if `algorithm = "DK"`

Value

A polynomial (`mvp` object; see mvp-package) or a character string if `basis = "MSF"`.

Examples

```
ZonalPol(3, lambda = c(3,1), algorithm = "naive")
ZonalPol(3, lambda = c(3,1), algorithm = "DK")
ZonalPol(3, lambda = c(3,1), algorithm = "naive", basis = "MSF")
```

`ZonalQ` *Evaluation of quaternionic zonal polynomials*

Description

Evaluates a quaternionic (or symplectic) zonal polynomial.

Usage

```
ZonalQ(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

ZonalQPol

Examples

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

ZonalQPol *Quaternionic zonal polynomial*

Description

Returns the quaternionic (or symplectic) zonal polynomial.

Usage

```
ZonalQPol(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 when using `algorithm = "naive"`; ignored if `algorithm = "DK"`

Value

A polynomial (`mvp` object; see mvp-package) or a character string if `basis = "MSF"`.

Examples

```
ZonalQPol(3, lambda = c(3,1), algorithm = "naive")
ZonalQPol(3, lambda = c(3,1), algorithm = "DK")
ZonalQPol(3, lambda = c(3,1), algorithm = "naive", basis = "MSF")
```

Index