Package 'jack'

May 10, 2023

Type Package

Title Jack, Zonal, and Schur Polynomials

Version 5.1.0

Date 2023-05-08

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

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

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

SystemRequirements C++ 17, gmp

Imports DescTools, gmp, JuliaConnectoR, multicool, mvp, partitions, qspray, Rcpp, Ryacas, spray

LinkingTo BH, Rcpp

Suggests testthat

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-05-10 09:00:02 UTC

R topics documented:

`as.function.exactmvp` *Exact multivariate polynomial as function*

Description

Coerces an exact multivariate polynomial into a function.

Usage

```
## S3 method for class 'exactmvp'
as.function(x, ...)
```

Arguments

`x`	object of class `exactmvp`; the functions returned by `Jack_julia` can return such objects
`...`	ignored

Value

A function having the same variables as the polynomial.

Examples

```
# library(jack)
if(JuliaConnectoR::juliaSetupOk()){
  julia <- Jack_julia()
  ( pol <- julia$JackPol(m = 2, lambda = c(3, 1), alpha = "3/2") )
  f <- as.function(pol)
  f(2, "3/7")
  # the evaluation is performed by (R)yacas and complex numbers are
  # allowed; the imaginary unit is denoted by `I`
  f("2 + 2*I", "1/4")
  JuliaConnectoR::stopJulia()
}
```

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 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.symmetricfunctions.com/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))
```

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

Description

Evaluates the Jack polynomial.

Usage

```
JackCPP(x, lambda, alpha)
```

Arguments

x	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	positive rational number, given as a string such as "2/3" or as a bigq number

Value

A bigq number.

Examples

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

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, a positive number, possibly 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

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

```
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 = "DK")
JackPol(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 <- JackPol(3, lambda = c(3, 1), alpha = gmp::as.bigq(2))
qspray::evalQspray(jack, c("1", "1/2", "3"))
```

JackPolCPP *Jack polynomial - C++ implementation*

Description

Returns the Jack polynomial.

Usage

```
JackPolCPP(n, lambda, alpha)
```

Arguments

n	number of variables, a positive integer
lambda	an integer partition, given as a vector of decreasing integers
alpha	positive rational number, given as a string such as "2/3" or as a bigq number

Value

A qspray multivariate polynomial.

Examples

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

Jack_julia *Evaluation with Julia*

Description

Evaluate the Jack polynomials with Julia. This is highly faster.

Usage

```
Jack_julia()
```

Value

A list of functions having the same names as the R functions of this package (`Jack`, `JackPol`, `Schur`, etc). The `XXXPol` functions have an argument `poly`, whose possible value is `"mvp"` or `"qspray"` (default), and this is the class of the polynomial returned by these functions.

Note

See `JuliaConnectoR-package` for information about setting up Julia. If you want to directly use Julia, you can use my package.

See Also

`as.function.exactmvp`

Examples

```
library(jack)
if(JuliaConnectoR::juliaSetupOk()){
  julia <- Jack_julia()
  # numerical evaluation ####
  julia$Jack(x = c(2, 2/3), lambda = c(3, 1), alpha = 3/2)
  # to pass rational numbers, use strings:
  julia$Jack(x = c("2", "2/3"), lambda = c(3, 1), alpha = "3/2")
  # symbolic polynomials ####
  # for `JackPol`, you must pass a rational `alpha` as a string if
  # you want an exact polynomial:
  ( pol <- julia$JackPol(m = 2, lambda = c(3, 1), alpha = "3/2") )
  class(pol)
  JuliaConnectoR::stopJulia()
}
```

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)
```

`prettyForm` *Pretty exact expression*

Description

Pretty form of the exact expression of a polynomial.

Usage

```
prettyForm(poly, asCharacter = FALSE)
```

Arguments

`poly`	an `exactmvp` object, that is, a polynomial with an exact expression
`asCharacter`	Boolean, whether to return a character string; if `FALSE`, the pretty form is printed

Value

A character string if `asCharacter=TRUE`, otherwise it is also returned but invisibly, and it is printed in the console.

Examples

```
library(jack)
if(JuliaConnectoR::juliaSetupOk()){
  julia <- Jack_julia()
  ( pol <- julia$ZonalPol(m = 2, lambda = c(3, 1), poly = "mvp") )
  prettyForm(pol)
  JuliaConnectoR::stopJulia()
}
```

print.exactmvp *Print an* exactmvp *object*

Description

Print an exactmvp object.

Usage

```
## S3 method for class 'exactmvp'
print(x, ...)
```

Arguments

x	object of class exactmvp; the functions returned by Jack_julia can return such objects
...	arguments passed to print.mvp

Value

Nothing.

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)
```

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

Description

Evaluates the Schur polynomial.

Usage

```
SchurCPP(x, lambda)
```

Arguments

x	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

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

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 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

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

SchurPolCPP *Schur polynomial - C++ implementation*

Description

Returns the Schur polynomial.

Usage

```
SchurPolCPP(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

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

---

toLaTeX *Exact expression to LaTeX*

Description

LaTeX form of the exact expression of a polynomial.

Usage

```
toLaTeX(poly, asCharacter = FALSE)
```

Arguments

poly	an exactmvp object, that is, a polynomial with an exact expression
asCharacter	Boolean, whether to return a character string; if FALSE, the LaTeX code is printed

Value

A character string if asCharacter=TRUE, otherwise it is also returned but invisibly, and it is printed in the console.

Examples

```
library(jack)
if(JuliaConnectoR::juliaSetupOk()){
  julia <- Jack_julia()
  ( pol <- julia$ZonalQPol(m = 2, lambda = c(3, 2), poly = "mvp") )
  toLaTeX(pol)
  JuliaConnectoR::stopJulia()
}
```

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/

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
```

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

Description

Evaluates the zonal polynomial.

Usage

```
ZonalCPP(x, lambda)
```

Arguments

x	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

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

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

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

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

ZonalPolCPP *Zonal polynomial - C++ implementation*

Description

Returns the Zonal polynomial.

Usage

```
ZonalPolCPP(m, lambda)
```

Arguments

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

Value

A `qspray` multivariate polynomial.

Examples

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

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
```

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

Description

Evaluates the zonal quaternionic polynomial.

Usage

```
ZonalQCPP(x, lambda)
```

Arguments

x	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

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

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

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

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

ZonalQPolCPP *Quaternionic zonal polynomial - C++ implementation*

Description

Returns the quaternionic zonal polynomial.

Usage

```
ZonalQPolCPP(m, lambda)
```

Arguments

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

Value

A qspray multivariate polynomial.

Examples

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

Index