Package 'qspray'

April 23, 2024

Type Package

Title Multivariate Polynomials with Rational Coefficients

Version 3.0.0

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Symbolic calculation and evaluation of multivariate polynomials with rational coefficients. This package is strongly inspired by the 'spray' package. It provides a function to compute Gröbner bases (reference <doi:10.1007/978-3-319-16721-3>). The header file of the C++ code can be used by other packages. It provides the templated class 'Qspray' that can be used to represent and to deal with multivariate polynomials with another type of coefficients.

License GPL-3

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

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

Imports DescTools, gmp, methods, partitions, purrr, RationalMatrix, Rcpp (>= 1.0.9), Ryacas, utils

LinkingTo BH, Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.3.1

SystemRequirements C++17, gmp

Collate 'RcppExports.R' 'characteristicPolynomial.R' 'creation.R' 'evaluation.R' 'groebner.R' 'integrateOnSimplex.R' 'internal.R' 'qspray.R' 'qsprayDivision.R' 'queries.R' 'show.R' 'symmetricPolynomials.R' 'transformation.R' 'yacas.R'

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Stéphane Laurent [aut, cre], Robin Hankin [ctb, cph] (author of the 'spray' package, which strongly inspired this package)

Repository CRAN

Date/Publication 2024-04-23 08:20:02 UTC

R topics documented:

`as.function.qspray` *Multivariate polynomial as function*

Description

Coerces a `qspray` polynomial into a function.

Usage

```
## S3 method for class 'qspray'
as.function(x, N = FALSE, ...)
```

Arguments

`x`	object of class `qspray`
`N`	Boolean, whether the function must numerically approximate the result
`...`	ignored

Value

A function having the same variables as the polynomial. If `N=FALSE`, it returns a string. If `N=TRUE`, it returns a number if the result does not contain any variable, otherwise it returns a R expression.

Examples

```
library(qspray)
P <- (qlone(1) + "1/2"*qlone(2))^2 + 5
f <- as.function(P)
g <- as.function(P, N = TRUE)
f(2, "3/7")
g(2, "3/7")
f("x", "y")
g("x", "y")
# the evaluation is performed by (R)yacas and complex numbers are
# allowed; the imaginary unit is denoted by `I`
f("2 + 2*I", "Sqrt(2)")
g("2 + 2*I", "Sqrt(2)")
```

as.qspray *Coercion to a 'qspray' object*

Description

Coercion to a 'qspray' object

Usage

```
## S4 method for signature 'character'
as.qspray(x)

## S4 method for signature 'qspray'
as.qspray(x)

## S4 method for signature 'numeric'
as.qspray(x)

## S4 method for signature 'bigz'
as.qspray(x)

## S4 method for signature 'bigq'
as.qspray(x)
```

Arguments

x	a qspray object or an object yielding a quoted integer or a quoted fraction after an application of as.character, e.g. a bigq number

Value

A qspray object.

Examples

```
as.qspray(2)
as.qspray("1/3")
```

changeVariables *Change of variables in a 'qspray' polynomial*

Description

Replaces the variables of a qspray polynomial with some qspray polynomials. E.g. you have a polynomial $P(x, y)$ and you want the polynomial $P(x^2, x+y+1)$. This is an alias of composeQspray.

Usage

```
## S4 method for signature 'qspray,list'
changeVariables(x, listOfQsprays)
```

Arguments

x	a qspray polynomial
listOfQsprays	a list containing at least n qspray objects, or objects coercable to qspray objects, where n is the number of variables of the polynomial given in the x argument

Value

The qspray polynomial obtained by replacing the variables of the polynomial given in the x argument with the polynomials given in the listOfQsprays argument.

Examples

```
library(qspray)
f <- function(x, y) x*y/2 + 4*y
x <- qlone(1)
y <- qlone(2)
P <- f(x, y)
X <- x^2
Y <- x + y + 1
changeVariables(P, list(X, Y)) == f(X, Y) # should be TRUE
```

---

characteristicPolynomial

Characteristic polynomial

Description

Characteristic polynomial of a matrix.

Usage

```
characteristicPolynomial(A)
```

Arguments

A	a square matrix with numeric, character, or bigq entries

Value

A univariate qspray polynomial.

Examples

```
set.seed(666)
A <- matrix(rpois(9L, 10), nrow = 3, ncol = 3)
( P <- characteristicPolynomial(A) )
# check the roots are the eigen values:
f <- as.function(P, N = TRUE)
sapply(eigen(A)$values, f) # approx c(0, 0, 0)
```

`collinearQsprays` *Whether two 'qspray' polynomials are collinear*

Description

Checks whether the polynomials represented by two `qspray` objects are collinear, that is, whether they are equal up to a scalar factor.

Usage

```
collinearQsprays(qspray1, qspray2)
```

Arguments

`qspray1, qspray2`
: two `qspray` objects

Value

A Boolean value.

Examples

```
library(qspray)
qspray1 <- qsprayMaker(string = "1/2 x^(1, 1) + 4 x^(0, 2) + 5")
qspray2 <- "4/7" * qspray1
collinearQsprays(qspray1, qspray2)
```

`compactSymmetricQspray`
Compact symmetric qspray

Description

Prints a symmetric qspray polynomial as a linear combination of the monomial symmetric polynomials.

Usage

```
## S4 method for signature 'qspray,logical'
compactSymmetricQspray(qspray, check)

## S4 method for signature 'qspray,ANY'
compactSymmetricQspray(qspray)
```

Arguments

qspray	a qspray object, which should correspond to a symmetric polynomial
check	Boolean, whether to check the symmetry (default TRUE)

Value

A character string.

See Also

MSPcombination

Examples

```
library(qspray)
( qspray <- PSFpoly(4, c(3, 1)) - ESFpoly(4, c(2, 2)) + 4L )
compactSymmetricQspray(qspray, check = TRUE)
```

---

composeQspray *Compose 'qspray' polynomials*

Description

Substitutes the variables of a qspray polynomial with some qspray polynomials. E.g. you have a polynomial $P(x, y)$ and you want the polynomial $P(x^2, x + y + 1)$ (see example).

Usage

```
composeQspray(qspray, listOfQsprays)
```

Arguments

qspray	a qspray polynomial
listOfQsprays	a list containing at least n qspray polynomials where n is the number of variables of the polynomial given in the qspray argument

Value

The qspray polynomial obtained by composing the polynomial given in the qspray argument with the polynomials given in the listOfQsprays argument.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
P <- x*y/2 + 4*y
X <- x^2
Y <- x + y + 1
composeQspray(P, list(X, Y)) # this is P(x^2, x+y+1)
```

`derivQspray` *Partial derivative*

Description

Partial derivative of a `qspray` polynomial.

Usage

```
derivQspray(qspray, i, derivative = 1)
```

Arguments

`qspray`	object of class `qspray`
`i`	integer, the dimension to differentiate with respect to, e.g. `1` to differentiate with respect to x
`derivative`	integer, how many times to differentiate

Value

A `qspray` object.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
qspray <- 2*x  + 3*x*y
derivQspray(qspray, 2) # derivative w.r.t. y
```

dQspray *Partial differentiation*

Description

Partial differentiation of a qspray polynomial.

Usage

```
dQspray(qspray, orders)
```

Arguments

qspray	object of class qspray
orders	integer vector, the orders of the differentiation; e.g. c(2, 0, 1) means that you differentiate two times with respect to x, you do not differentiate with respect to y, and you differentiate one time with respect to z

Value

A qspray object.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
qspray <- x + 2*y  + 3*x*y
dQspray(qspray, c(1, 1))
derivQspray(derivQspray(qspray, 1), 2)
```

ESFpoly *Elementary symmetric polynomial*

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 positive integers

Value

A qspray object.

Examples

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

`evalQspray` *Evaluate a 'qspray' object*

Description

Evaluation of the multivariate polynomial represented by a qspray object.

Usage

```
evalQspray(qspray, values_re, values_im = NULL)
```

Arguments

qspray	a qspray object
values_re	vector of the real parts of the values; each element of `as.character(values_re)` must be a quoted integer or a quoted fraction
values_im	vector of the imaginary parts of the values; each element of `as.character(values_im)` must be a quoted integer or a quoted fraction

Value

A bigq number if `values_im=NULL`, a pair of bigq numbers otherwise: the real part and the imaginary part of the result.

Examples

```
x <- qlone(1); y <- qlone(2)
P <- 2*x + "1/2"*y
evalQspray(P, c("2", "5/2", "99999")) # "99999" will be ignored
```

getCoefficient *Get a coefficient in a 'qspray' polynomial*

Description

Get the coefficient corresponding to the given sequence of exponents.

Usage

```
## S4 method for signature 'qspray,numeric'
getCoefficient(qspray, exponents)
```

Arguments

qspray	a qspray object
exponents	a vector of exponents

Value

The coefficient as a bigq number.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
p <- 4*x^2 + 3*y - 5
getCoefficient(p, 2)
getCoefficient(p, c(2, 0)) # same as getCoefficient(p, 2)
getCoefficient(p, c(0, 1)) # coefficient of y because y=x^0*y^1
getCoefficient(p, 0) # the constant term
getCoefficient(p, integer(0)) # the constant term
getCoefficient(p, 3) # there's no cubic term
```

getConstantTerm *Get the constant term of a 'qspray' polynomial*

Description

Get the constant term of a qspray polynomial.

Usage

```
## S4 method for signature 'qspray'
getConstantTerm(qspray)
```

Arguments

qspray	a qspray object

Value

A bigq number.

---

groebner *Gröbner basis*

Description

Returns a Gröbner basis following Buchberger's algorithm using the lexicographical order.

Usage

```
groebner(G, minimal = TRUE, reduced = TRUE)
```

Arguments

G	a list of qspray polynomials, the generators of the ideal
minimal	Boolean, whether to return a minimal basis
reduced	Boolean, whether to return the reduced basis

Value

A Gröbner basis of the ideal generated by G, given as a list of qspray polynomials.

References

Cox, Little & O'Shea. *Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Fourth edition, Springer 2015.

Examples

```
library(qspray)
f <- qsprayMaker(string = "x^(3) - 2 x^(1,1)")
g <- qsprayMaker(string = "x^(2,1) - 2 x^(0,2) + x^(1)")
groebner(list(f, g), FALSE, FALSE)
# other example
x <- qlone(1); y <- qlone(2); z <- qlone(3)
f1 <- x^2 + y + z^2 - 1
f2 <- x^2 + y + z - 1
f3 <- x + y^2 + z - 1
gb <- groebner(list(f1, f2, f3))
lapply(gb, prettyQspray, vars = c("x", "y", "z"))
```

HallInnerProduct *Hall inner product*

Description

Hall inner product of two symmetric polynomials.

Usage

```
HallInnerProduct(qspray1, qspray2, alpha = 1)
```

Arguments

qspray1, qspray2	two symmetric qspray polynomials
alpha	parameter equal to 1 for the usual Hall inner product, otherwise this is the "Jack parameter", an integer or a bigq number

Value

A bigq number.

implicitization *Implicitization with Gröbner bases*

Description

Implicitization of a system of parametric equations (see examples).

Usage

```
implicitization(nvariables, parameters, equations, relations)
```

Arguments

nvariables	number of variables
parameters	character vector of the names of the parameters, or NULL if there's no parameter
equations	list of qspray polynomials representing the parametric equations
relations	list of qspray polynomials representing the relations between the variables and the parameters, or NULL if there is none

Value

A list of qspray polynomials.

Examples

```
library(qspray)
# ellipse example ####
# variables
cost <- qlone(1)
sint <- qlone(2)
# parameters
a <- qlone(3)
b <- qlone(4)
#
nvariables <- 2
parameters <- c("a", "b")
equations <- list(
  "x" = a * cost,
  "y" = b * sint
)
relations <- list(
  cost^2 + sint^2 - 1
)
#
eqs <- implicitization(nvariables, parameters, equations, relations)
```

---

integratePolynomialOnSimplex

Integral of a multivariate polynomial over a simplex

Description

Returns the exact value of the integral of a multivariate polynomial with rational coefficients over a simplex whose vertices have rational coordinates.

Usage

```
integratePolynomialOnSimplex(P, S)
```

Arguments

P	a qspray object
S	the simplex, a (n+1)xn matrix such that each entry of the matrix as.character(S) is a quoted integer or a quoted fraction

Value

A bigq number, the exact value of the integral.

Examples

```
library(qspray)
x <- qlone(1); y <- qlone(2)
P <- x/2 + x*y
S <- rbind(c("0", "0"), c("1", "0"), c("1", "1")) # a triangle
integratePolynomialOnSimplex(P, S)
```

`isConstant` *Whether a 'qspray' polynomial is constant*

Description

Checks whether a `qspray` object defines a constant polynomial.

Usage

```
## S4 method for signature 'qspray'
isConstant(x)
```

Arguments

`x`	a `qspray` object

Value

A Boolean value.

`isPolynomialOf` *Whether a 'qspray' is a polynomial of some given 'qsprays'*

Description

Checks whether a `qspray` polynomial can be written as a polynomial of some given `qspray` polynomials. If `TRUE`, this polynomial is returned.

Usage

```
isPolynomialOf(qspray, qsprays)
```

Arguments

`qspray`	a `qspray` object
`qsprays`	a list of `qspray` objects

Value

A Boolean value indicating whether the polynomial defined by `qspray` can be written as a polynomial of the polynomials defined by the `qspray` objects given in the `qsprays` list. If this is `TRUE`, this polynomial is returned as an attribute named `"polynomial"`.

Examples

```
library(qspray)
P <- function(X, Y) X^2*Y + 2*X + 3
x <- qlone(1); y <- qlone(2); z <- qlone(3)
q1 <- x + y
q2 <- x*z^2 + 4
qspray <- P(q1, q2)
( check <- isPolynomialOf(qspray, list(q1, q2)) )
POLYNOMIAL <- attr(check, "polynomial")
composeQspray(POLYNOMIAL, list(q1, q2)) == qspray # should be TRUE
```

---

isQone *Whether a 'qspray' polynomial is the unit polynomial*

Description

Checks whether a `qspray` object defines the unit polynomial.

Usage

```
## S4 method for signature 'qspray'
isQone(qspray)
```

Arguments

`qspray`	a `qspray` object

Value

A Boolean value.

isQzero *Whether a 'qspray' polynomial is null*

Description

Checks whether a qspray object defines the zero polynomial.

Usage

```
## S4 method for signature 'qspray'
isQzero(qspray)
```

Arguments

qspray	a qspray object

Value

A Boolean value.

isSymmetricQspray *Check symmetry of a polynomial*

Description

Check whether a qspray polynomial is symmetric.

Usage

```
isSymmetricQspray(qspray)
```

Arguments

qspray	a qspray polynomial

Value

A Boolean value indicating whether the polynomial defined by qspray is symmetric.

See Also

MSPcombination, compactSymmetricQspray

Examples

```
e1 <- ESFpoly(3, 1)
e2 <- ESFpoly(3, 2)
e3 <- ESFpoly(3, 3)
q <- e1 + 2*e2 + 3*e3 + 4*e1*e3
isSymmetricQspray(q)
```

`isUnivariate` *Whether a 'qspray' is univariate*

Description

Checks whether a `qspray` object defines a univariate polynomial.

Usage

```
## S4 method for signature 'qspray'
isUnivariate(x)
```

Arguments

`x`	a `qspray` object

Value

A Boolean value.

Note

It is considered that a constant `qspray` is univariate, and that the `qspray` object `y^2+1` where `y=qlone(2)` is not univariate, although only one variable is present (see the note in the documentation of `numberOfVariables`).

`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 positive integers

Value

A qspray object.

Examples

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

MSPcombination *Symmetric polynomial in terms of the monomial symmetric polynomials*

Description

Expression of a symmetric polynomial as a linear combination of the monomial symmetric polynomials.

Usage

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

Arguments

qspray	a qspray object defining a symmetric polynomial
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 * MSFpoly(n, lambda), where n is the number of variables in the symmetric polynomial.

Examples

```
qspray <- PSFpoly(4, c(3, 1)) + ESFpoly(4, c(2, 2)) + 4L
MSPcombination(qspray)
```

numberOfTerms *Number of terms in a 'qspray' polynomial*

Description

Number of terms in the polynomial defined by a qspray object.

Usage

```
## S4 method for signature 'qspray'
numberOfTerms(qspray)
```

Arguments

qspray	a qspray object

Value

An integer.

numberOfVariables *Number of variables in a 'qspray' polynomial*

Description

Number of variables involved in a qspray object.

Usage

```
## S4 method for signature 'qspray'
numberOfVariables(x)
```

Arguments

x	a qspray object

Value

An integer.

Note

The number of variables in the qspray object `y^2+1` where `y=qlone(2)` is 2, not 1, although only one variable is present. Strictly speaking, the function returns the maximal integer `d` such that the variable `qlone(d)` occurs in the polynomial.

orderedQspray *Ordered 'qspray'*

Description

Reorders the terms of a qspray object according to the lexicographic order of the powers. This function is rather used internally only but it is exported for internal usage in other packages.

Usage

```
orderedQspray(qspray)
```

Arguments

qspray	a qspray object

Value

A qspray object. It defines the same polynomial as the input qspray object but it is ordered.

Examples

```
qspray <- rQspray()
qspray == orderedQspray(qspray) # should be TRUE
```

permuteVariables *Permute variables*

Description

Permute the variables of a qspray polynomial.

Usage

```
## S4 method for signature 'qspray,numeric'
permuteVariables(x, permutation)
```

Arguments

x	a qspray object
permutation	a permutation

Value

A qspray object.

Examples

```
library(qspray)
f <- function(x, y, z) {
  x^2 + 5*y + z - 1
}
x <- qlone(1)
y <- qlone(2)
z <- qlone(3)
P <- f(x, y, z)
permutation <- c(3, 1, 2)
Q <- permuteVariables(P, permutation)
Q == f(z, x, y) # should be TRUE
```

---

prettyQspray *Pretty polynomial*

Description

Pretty form of a qspray polynomial.

Usage

```
prettyQspray(qspray, vars = NULL)
```

Arguments

qspray	a qspray object
vars	variable names; NULL for "x1", "x2", ...

Value

A character string.

Examples

```
library(qspray)
P <- (qlone(1) + "1/2"*qlone(2))^2 + 5
prettyP <- prettyQspray(P, vars = c("x", "y"))
prettyP
cat(Ryacas::yac_str(sprintf("PrettyForm(%s)", prettyP)))
Ryacas::yac_str(sprintf("TeXForm(%s)", prettyP))
```

PSFpoly *Power sum polynomial*

Description

Returns a power sum function as a polynomial.

Usage

```
PSFpoly(m, lambda)
```

Arguments

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

Value

A `qspray` object.

Examples

```
library(qspray)
PSFpoly(3, c(3, 1))
```

PSPexpression *Symmetric polynomial in terms of the power sum polynomials.*

Description

Expression of a symmetric `qspray` polynomial as a polynomial in the power sum polynomials.

Usage

```
PSPexpression(qspray)
```

Arguments

`qspray`	a symmetric `qspray` polynomial (an error is returned if it is not symmetric)

Value

A `qspray` polynomial, say P, such that $P(p_1, ..., p_n)$ equals the input symmetric polynomial, where p_i is the i-th power sum polynomial (`PSFpoly(n, i)`).

Note

This function has been exported because it is used in the **jack** package.

qdivision *Division of a qspray polynomial*

Description

Division of a qspray polynomial by a list of qspray polynomials. See the reference for the definition.

Usage

```
qdivision(qspray, divisors)
```

Arguments

qspray	the dividend, a qspray object
divisors	the divisors, a list of qspray objects

Value

The remainder of the division, a qspray object.

References

Michael Weiss, 2010. Computing Gröbner Bases in Python with Buchberger's Algorithm.

Examples

```
# a univariate example
library(qspray)
x <- qlone(1)
f <- x^4 - 4*x^3 + 4*x^2 - x # 0 and 1 are trivial roots
g <- x * (x - 1)
qdivision(f, list(g)) # should be zero
```

qlone *Polynomial variable*

Description

Creates a polynomial variable. Using this function is the main way to build qspray objects.

Usage

```
qlone(n)
```

Arguments

n	positive integer, the index of the variable

Value

A qspray object.

Examples

```
x <- qlone(1)
y <- qlone(2)
(x + y) * (x - y)
```

---

qone *The unit 'qspray' polynomial*

Description

Returns the qspray polynomial identically equal to 1.

Usage

```
qone()
```

Value

A qspray object.

---

qspray-unary *Unary operators for qspray objects*

Description

Unary operators for qspray objects.

Usage

```
## S4 method for signature 'qspray,missing'
e1 + e2

## S4 method for signature 'qspray,missing'
e1 - e2
```

Arguments

e1	object of class qspray
e2	nothing

Value

A qspray object.

qsprayDivision *Division of two polynomials*

Description

Division of two polynomials

Usage

```
qsprayDivision(qsprayA, qsprayB)
```

Arguments

qsprayA	a qspray object, the dividend
qsprayB	a qspray object, the divisor

Value

A list with two qspray objects, the quotient and the remainder.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
z <- qlone(3)
B <- x*y^2 + z*x^2 + 1
A <- B * (x^2*y^2*z^2 - 3) + x*y
divis <- qsprayDivision(A, B)
B * divis[["Q"]] + divis[["R"]] == A # should be TRUE
```

qsprayMaker *Make a 'qspray' object*

Description

Make a qspray object from a list of exponents and a vector of coefficients.

Usage

```
qsprayMaker(powers, coeffs, string = NULL)
```

Arguments

powers	list of positive integer vectors
coeffs	a vector such that each element of as.character(coeffs) is a quoted integer or a quoted fraction; it must have the same length as the powers list
string	if not NULL, this argument takes precedence over powers and coeffs; it must be a string representing a multivariate polynomial; see the example

Value

A qspray object.

Examples

```
powers <- list(c(1, 1), c(0, 2))
coeffs <- c("1/2", "4")
qsprayMaker(powers, coeffs)
qsprayMaker(string = "1/2 x^(1, 1) + 4 x^(0, 2)")
```

---

qspray_from_list *(internal) Make a 'qspray' object from a list*

Description

This function is for internal usage. It is exported because it is also used for internal usage in others packages.

Usage

```
qspray_from_list(qspray_as_list)
```

Arguments

qspray_as_list	list returned by the Rcpp function returnQspray

Value

A qspray object.

qzero *The null 'qspray' polynomial*

Description

Returns the qspray polynomial identically equal to 0.

Usage

```
qzero()
```

Value

A qspray object.

rQspray *Random 'qspray'*

Description

Generates a random qspray object.

Usage

```
rQspray()
```

Value

A qspray object with at most 4 terms and at most 3 variables.

showMonomialOld *Print a monomial*

Description

Prints a monomial like "x^(1, 0, 2)". This way of showing a monomial was used by default in previous versions of this package.

Usage

```
showMonomialOld(x = "x")
```

Arguments

x	a string, usually a letter such as "x" or "X", to denote the variable

Value

A function which takes as argument a sequence of exponents and which prints the corresponding monomial.

See Also

showMonomialX1X2X3, showMonomialXYZ, showQspray, showQsprayOption<-.

Examples

```
showMonomialOld("X")(c(1, 0, 2))
showMonomialOld("X")(NULL)
```

---

showMonomialX1X2X3 *Print a monomial*

Description

Prints a monomial in the style of "x1.x3^2".

Usage

```
showMonomialX1X2X3(x = "x", collapse = ".")
```

Arguments

x	a string, usually a letter such as "x" or "X", to denote the non-indexed variables
collapse	a string to denote the symbol representing the multiplication, e.g. "*" or ""

Value

A function which takes as argument a sequence of exponents and which prints the corresponding monomial.

Note

The function returned by this function can be used as the option "showMonomial" in the showQsprayOption<- function. But if you are happy with the default collapse argument, then you can equivalently set the "x" option instead, thereby typing less code. See the example.

See Also

showQsprayX1X2X3, showMonomialXYZ, showQsprayOption<-.

Examples

```
showMonomialX1X2X3("X")(c(1, 0, 2))
showMonomialX1X2X3("X", collapse = "*")(c(1, 0, 2))
showMonomialX1X2X3("X")(c(1, 0, 2)) ==
  showMonomialXYZ(c("X1", "X2", "X3"))(c(1, 0, 2))
showMonomialX1X2X3()(NULL)
# setting a show option:
set.seed(3141)
( qspray <- rQspray() )
showQsprayOption(qspray, "showMonomial") <- showMonomialX1X2X3("X")
qspray
# this is equivalent to:
showQsprayOption(qspray, "showQspray") <- showQsprayX1X2X3("X")
# and also equivalent to:
showQsprayOption(qspray, "x") <- "X"
```

---

showMonomialXYZ *Print a monomial*

Description

Prints a monomial like `"x.z^2"` if possible (see details).

Usage

```
showMonomialXYZ(letters = c("x", "y", "z"), collapse = ".")
```

Arguments

`letters`	a vector of strings, usually some letters such as `"x"` and `"y"`, to denote the variables
`collapse`	a string to denote the symbol representing the multiplication, e.g. `"*"` or `"."`

Details

If the function returned by this function is applied to a vector of exponents whose length is higher than the length of the `letters` vector, then `showMonomialX1X2X3(x=letters[1])` is applied (see the last example).

Value

A function which takes as argument a sequence of exponents and which prints the corresponding monomial.

Note

The function returned by this function can be used as the option `"showMonomial"` in the `showQsprayOption<-` function.

See Also

showQsprayXYZ, showMonomialX1X2X3, showQsprayOption<-.

Examples

```
showMonomialXYZ()(c(1, 0, 2))
showMonomialXYZ(collapse = "*")(c(1, 0, 2))
showMonomialXYZ()(NULL)
# what happens if there are more exponents than letters:
showMonomialXYZ(c("a", "b"), collapse = "*")(c(1, 2, 3))
# same as:
showMonomialX1X2X3("a", collapse = "*")(c(1, 2, 3))
# setting a show option:
set.seed(3141)
( qspray <- rQspray() )
showQsprayOption(qspray, "showMonomial") <- showMonomialXYZ(c("A", "B", "C"))
qspray
# this is equivalent to:
showQsprayOption(qspray, "showQspray") <- showQsprayXYZ(c("A", "B", "C"))
```

---

showQspray *Print a 'qspray' object*

Description

Prints a qspray object given a function which prints the monomials.

Usage

```
showQspray(showMonomial, compact = FALSE, multiplication = "*")
```

Arguments

showMonomial	a function which takes as argument a sequence of exponents and which returns a string representing the corresponding monomial
compact	a Boolean value; if TRUE, then the + sign and the - sign will not be surrounded by spaces
multiplication	used to separate the coefficient and the monomial within a term

Value

A function which prints a qspray object.

Note

The function returned by this function can be used as the option "showQspray" in the showQsprayOption<- function. But one generally prefers to use showQsprayX1X2X3 or showQsprayXYZ instead, which are both built with showQspray.

See Also

showQsprayX1X2X3, showQsprayXYZ, showQsprayOption<-.

Examples

```
set.seed(3141)
( qspray <- rQspray() )
f <- showQspray(showMonomialX1X2X3("X"), compact = TRUE)
f(qspray)
# this is equivalent to:
f <- showQsprayX1X2X3("X", compact = TRUE)
f(qspray)
# if you want to adopt this way to show a qspray, use
# the setter function \code{\link{showQsprayOption<-}}:
showQsprayOption(qspray, "showQspray") <-
  showQsprayX1X2X3("X", compact = TRUE)
qspray
# then this show option will be preserved by some operations on the qspray:
qspray^2
```

---

showQsprayOption<- *Set a show option to a 'qspray' object*

Description

Set a show option to a qspray object

Usage

```
showQsprayOption(x, which) <- value
```

Arguments

x	a qspray object
which	which option to set; this can be "x", "showMonomial", or "showQspray"
value	the value of the option

Value

This returns the updated qspray.

Note

The interest of setting some show options to a 'qspray' is that these options are preserved by some operations. See the examples and the README.

Examples

```
set.seed(3141)
( qspray <- rQspray() )
showQsprayOption(qspray, "x") <- "a"
qspray
# this is identical to:
showQsprayOption(qspray, "showMonomial") <- showMonomialX1X2X3("a")
# and also identical to:
showQsprayOption(qspray, "showQspray") <- showQsprayX1X2X3("a")
# old show method:
showQsprayOption(qspray, "showMonomial") <- showMonomialOld()
qspray
# show options are preserved by some operations:
qspray^2
3*qspray
derivQspray(qspray, 1)
swapVariables(qspray, 1, 2)
substituteQspray(qspray, c(NA, NA, "3/2"))
# for the binary arithmetic operations, the show options of the first
# operand are transferred to the result when possible:
( qspray2 <- rQspray() )
qspray + qspray2
```

---

showQsprayX1X2X3 *Print a 'qspray' object*

Description

Prints a qspray object given a string for the variable.

Usage

```
showQsprayX1X2X3(x = "x", collapse = ".", ...)
```

Arguments

x, collapse	see showMonomialX1X2X3
...	arguments passed to showQspray, such as compact=TRUE

Value

A function which prints a qspray object.

Note

The way qspray objects are displayed can be controlled with the help of the function showQsprayOption<-, and showQsprayX1X2X3() is a possible option to pass in showQsprayOption<-.

See Also

showMonomialX1X2X3, showQspray, showQsprayOption<-.

Examples

```
set.seed(3141)
( qspray <- rQspray() )
showQsprayX1X2X3("X")(qspray)
# setting a show option:
showQsprayOption(qspray, "showQspray") <- showQsprayX1X2X3("A")
qspray
# this is equivalent to:
showQsprayOption(qspray, "showMonomial") <- showMonomialX1X2X3("A")
# and also equivalent to:
showQsprayOption(qspray, "x") <- "A"
```

---

showQsprayXYZ *Print a polynomial*

Description

Prints a polynomial by printing monomials like "x^2.yz".

Usage

```
showQsprayXYZ(letters = c("x", "y", "z"), collapse = ".", ...)
```

Arguments

letters, collapse	see showMonomialXYZ
...	arguments passed to showQspray, such as compact=TRUE

Value

A function which prints a qspray object. It is constructed with showQspray and showMonomialXYZ.

Note

The function returned by this function can be used as the option "showQspray" in the showQsprayOption<- function.

See Also

showMonomialXYZ, showQspray, showQsprayOption<-.

Examples

```
set.seed(3141)
( qspray <- rQspray() )
showQsprayXYZ(c("X", "Y", "Z"))(qspray)
showQsprayXYZ(c("X", "Y", "Z"))(qlone(1) + qlone(2) + qlone(3) + qlone(4))
# setting a show option:
showQsprayOption(qspray, "showQspray") <- showQsprayXYZ(c("A", "B", "C"))
qspray
# this is equivalent to:
showQsprayOption(qspray, "showMonomial") <- showMonomialXYZ(c("A", "B", "C"))
```

`substituteQspray` *Substitutions in a 'qspray' polynomial*

Description

Substitute some variables in a `qspray` polynomial.

Usage

```
substituteQspray(qspray, values)
```

Arguments

`qspray`	a `qspray` object
`values`	the values to be substituted; this must be a vector whose length equals the number of variables of `qspray`, and whose each entry is either `NA` (for non-substitution) or a 'scalar' `x` such that `as.character(x)` is a quoted integer or a quoted fraction

Value

A `qspray` object.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
z <- qlone(3)
p <- x^2 + y^2 + x*y*z - 1
substituteQspray(p, c("2", NA, "3/2"))
```

swapVariables *Swap variables*

Description

Swap two variables in a qspray polynomial.

Usage

```
## S4 method for signature 'qspray,numeric,numeric'
swapVariables(x, i, j)
```

Arguments

x	a qspray object
i, j	indices of the variables to be swapped

Value

A qspray object.

Examples

```
library(qspray)
f <- function(x, y, z) {
  x^2 + 5*y + z - 1
}
x <- qlone(1)
y <- qlone(2)
z <- qlone(3)
P <- f(x, y, z)
Q <- swapVariables(P, 2, 3)
Q == f(x, z, y) # should be TRUE
```

Index