Package 'symbolicQspray'

July 28, 2024

Title Multivariate Polynomials with Symbolic Parameters in their Coefficients

Version 1.1.0

Description Introduces the 'symbolicQspray' objects. Such an object represents a multivariate polynomial whose coefficients are fractions of multivariate polynomials with rational coefficients. The package allows arithmetic on such polynomials. It is based on the 'qspray' and 'ratioOfQsprays' packages. Some functions for 'qspray' polynomials have their counterpart for 'symbolicQspray' polynomials. A 'symbolicQspray' polynomial should not be seen as a polynomial on the field of fractions of rational polynomials, but should rather be seen as a polynomial with rational coefficients depending on some parameters, symbolically represented, with a dependence given by fractions of rational polynomials.

License GPL-3

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

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

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

Imports gmp, methods, Rcpp, utils

Suggests testthat (>= 3.0.0)

LinkingTo BH, qspray, ratioOfQsprays, Rcpp, RcppCGAL

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.1

SystemRequirements C++17, gmp, mpfr

Collate 'JacobiPolynomial.R' 'RcppExports.R' 'creation.R' 'evaluation.R' 'internal.R' 'symbolicQspray.R' 'queries.R' 'show.R' 'symmetricPolynomials.R' 'transformation.R'

NeedsCompilation yes

Author Stéphane Laurent [aut, cre]

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Repository CRAN

Date/Publication 2024-07-28 16:50:02 UTC

Contents

as.symbolicQspray — *Coercion to a 'symbolicQspray' object*

Description

Coercion to a 'symbolicQspray' object

Usage

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

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

## S4 method for signature 'ratioOfQsprays'
as.symbolicQspray(x)

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

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

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

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

Arguments

`x`	a `symbolicQspray` object or an object for which `as.ratioOfQsprays` is applicable

Value

A `symbolicQspray` object.

Examples

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

`changeParameters` *Change of parameters in a 'symbolicQspray' polynomial*

Description

Replaces the parameters of a `symbolicQspray` polynomial (which are `qspray` objects) with some `qspray` polynomials. E.g. you have a polynomial with two parameters $P_{a,b}(x)$ and you want the polynomial $P_{a+1,b+1}(x)$ (see example).

Usage

```
changeParameters(Qspray, newParameters)
```

Arguments

`Qspray`	a `symbolicQspray` polynomial
`newParameters`	a list containing at least `n` `qspray` objects, or objects coercible to `qspray` objects, where `n` is the number of parameters in the symbolic polynomial given in the `Qspray` argument; if this list is named, then its names will be used in the show options of the result

Value

The `symbolicQspray` polynomial obtained by replacing the parameters of the symbolic polynomial given in the `Qspray` argument with the polynomials given in the `newParameters` argument.

See Also

If you want to change the variables of a symbolic qspray, use `changeVariables`. If you want to assign some values to its parameters, use `substituteParameters`.

Examples

```
library(symbolicQspray)
( JP <- JacobiPolynomial(2) ) # a univariate polynomial with two parameters
a1 <- qlone(1)
a2 <- qlone(2)
changeParameters(JP, list(a1, a2)) == JP # should be TRUE
changeParameters(JP, list(a1+1, a2+1))
```

changeVariables — *Change of variables in a 'symbolicQspray' polynomial*

Description

Replaces the variables of a `symbolicQspray` polynomial with some `symbolicQspray` polynomials. E.g. you have a polynomial $P_a(x, y)$ and you want the polynomial $P_a(x + a, y + a)$ (see example).

Usage

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

Arguments

`x`	a `symbolicQspray` polynomial
`listOfQsprays`	a list containing at least `n` `symbolicQspray` objects, or objects coercible to `symbolicQspray` objects, where `n` is the number of variables in the polynomial given in the `x` argument; if this list is named, their its names will be used in the show options of the result

Value

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

See Also

If you want to change the parameters of a symbolic qspray, use `changeParameters`. If you want to assign some values to its variables, see `substituteVariables`.

Examples

```
library(symbolicQspray)
f <- function(a, X, Y) {
  a^2 / (a + 1) * X^2*Y  +  (3*a - 2) / a * Y^2
}
a <- qlone(1)
X <- Qlone(1)
Y <- Qlone(2)
Qspray <- f(a, X, Y)
U <- X + a
V <- Y + a
changeVariables(Qspray, list(U, V)) == f(a, U, V) # should be TRUE
```

compactSymmetricQspray *Compact symmetric qspray*

Description

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

Usage

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

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

Arguments

`qspray`	a `symbolicQspray` object which should correspond to a symmetric polynomial
`check`	Boolean, whether to check the symmetry

Value

A character string.

See Also

`MSPcombination`

derivSymbolicQspray *Partial derivative*

Description

Partial derivative of a `symbolicQspray` polynomial.

Usage

```
derivSymbolicQspray(Qspray, i, derivative = 1)
```

Arguments

`Qspray`	object of class `symbolicQspray`
`i`	integer, the dimension to differentiate with respect to, e.g. 2 to differentiate w.r.t. y
`derivative`	positive integer, how many times to differentiate

Value

A `symbolicQspray` object.

`dSymbolicQspray` *Partial differentiation*

Description

Partial differentiation of a `symbolicQspray` polynomial.

Usage

```
dSymbolicQspray(Qspray, orders)
```

Arguments

`Qspray`	object of class `symbolicQspray`
`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 `symbolicQspray` object.

`evalSymbolicQspray` *Evaluation of a 'symbolicQspray' polynomial*

Description

Evaluates a `symbolicQspray` polynomial by substituting some values to the parameters (same as `substituteParameters`) or to the variables (same as `substituteVariables`) or both.

Usage

```
evalSymbolicQspray(Qspray, a = NULL, X = NULL)
```

Arguments

`Qspray`	a `symbolicQspray` object
`a`	vector of values to be substituted to the parameters; these values must be coercible to `bigq` numbers
`X`	vector of values to be substituted to the variables; these values must be coercible to `bigq` numbers

Value

If both `a` and `X` are `NULL`, this returns the input `symbolicQspray` object; otherwise, if `a` is not `NULL`, this returns a `qspray` object, and if `X` is not `NULL`, this returns a `ratioOfQsprays` object.

Examples

```
library(symbolicQspray)
a1 <- qlone(1); a2 <- qlone(2)
X1 <- Qlone(1); X2 <- Qlone(2); X3 <- Qlone(3)
( Qspray <- (a1 + 2)*X1^2*X2 + (a2/(a1^2+a2))*X1*X2*X3 )
a <- c(2, 3)
X <- c(4, 3, 2)
( qspray <- evalSymbolicQspray(Qspray, a = a) )
( rOQ <- evalSymbolicQspray(Qspray, X = X) )
evalSymbolicQspray(Qspray, a = a, X = X)
evalQspray(qspray, X)
evalRatioOfQsprays(rOQ, a)
```

`getCoefficient` *Get a coefficient in a 'symbolicQspray' polynomial*

Description

Get the coefficient of the term with the given monomial.

Usage

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

Arguments

`qspray`	a `symbolicQspray` object
`exponents`	a vector of exponents, thereby defining a monomial; trailing zeros are ignored

Value

The coefficient, `ratioOfQsprays` object.

Examples

```
a1 <- qlone(1); a2 <- qlone(2)
X <- Qlone(1); Y <- Qlone(2)
p <- 2*(a1/a2)*X^2 + (a1/(a1+a2))*Y + a2^2/a1
getCoefficient(p, 2)       # coefficient of X^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, 3)       # coefficient of X^3
```

getConstantTerm — *Get the constant term of a 'symbolicQspray' polynomial*

Description

Get the constant term of a `symbolicQspray` polynomial.

Usage

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

Arguments

`qspray`	a `symbolicQspray` object

Value

A `ratioOfQsprays` object.

hasPolynomialCoefficientsOnly — *Whether the coefficients of a 'symbolicQspray' polynomially depend on its parameters*

Description

Checks whether the dependence of the coefficients of a `symbolicQspray` polynomial on their parameters is polynomial.

Usage

```
hasPolynomialCoefficientsOnly(Qspray)
```

Arguments

`Qspray`	a `symbolicQspray` object

Value

A Boolean value. The coefficients of a `symbolicQspray` polynomial always are fractions of polynomials. This function checks whether they are polynomials.

Examples

```
JP <- JacobiPolynomial(4)
hasPolynomialCoefficientsOnly(JP)
```

involvedVariables *Variables involved in a 'symbolicQspray' polynomial*

Description

Variables involved in a symbolicQspray object.

Usage

```
## S4 method for signature 'symbolicQspray'
involvedVariables(x)
```

Arguments

x	a symbolicQspray object

Value

A vector of integers. Each integer represents the index of a variable involved in x.

See Also

numberOfVariables.

Examples

```
a1 <- qlone(1); a2 <- qlone(2)
X <- Qlone(1); Z <- Qlone(3)
Qspray <- (a1/a2)*X^2 + (a1/(a1+a2))*X*Z + a2^2/a1
involvedVariables(Qspray) # should be c(1L, 3L)
```

isConstant *Whether a 'symbolicQspray' polynomial is constant*

Description

Checks whether a symbolicQspray object defines a constant polynomial.

Usage

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

Arguments

x	a symbolicQspray object

Value

A Boolean value.

`isQone` *Whether a 'symbolicQspray' polynomial is the unit polynomial*

Description

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

Usage

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

Arguments

`qspray`	a `symbolicQspray` object

Value

A Boolean value.

`isQzero` *Whether a 'symbolicQspray' polynomial is null*

Description

Checks whether a `symbolicQspray` object defines the zero polynomial.

Usage

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

Arguments

`qspray`	a `symbolicQspray` object

Value

A Boolean value.

isUnivariate — *Whether a 'symbolicQspray' polynomial is univariate*

Description

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

Usage

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

Arguments

`x`	a `symbolicQspray` object

Value

A Boolean value.

Note

It is considered that a constant `symbolicQspray` is univariate.

JacobiPolynomial — *Jacobi polynomial*

Description

Computes the n-th Jacobi polynomial as a `symbolicQspray`.

Usage

```
JacobiPolynomial(n)
```

Arguments

`n`	index (corresponding to the degree), a positive integer

Details

The Jacobi polynomials are univariate polynomials whose coefficients depend on two parameters.

Value

A `symbolicQspray` object representing the n-th Jacobi polynomial.

Examples

```
JP1 <- JacobiPolynomial(1)
showSymbolicQsprayOption(JP1, "showRatioOfQsprays") <-
  showRatioOfQspraysXYZ(c("alpha", "beta"))
JP1
```

`numberOfParameters` *Number of parameters*

Description

Number of parameters of a `symbolicQspray` polynomial, i.e. the number of variables occurring in its coefficients.

Usage

```
numberOfParameters(Qspray)
```

Arguments

`Qspray`	a `symbolicQspray` object

Value

An integer, the number of parameters involved in (the coefficients of) `Qspray`.

Examples

```
JP <- JacobiPolynomial(4) # Jacobi polynomials have two parameters
numberOfParameters(JP)
```

`numberOfTerms` *Number of terms in a 'symbolicQspray' polynomial*

Description

Number of terms in the polynomial defined by a `symbolicQspray` object.

Usage

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

Arguments

`qspray`	a `symbolicQspray` object

Value

An integer.

---

numberOfVariables *Number of variables of a 'symbolicQspray' polynomial*

Description

Number of variables involved in a `symbolicQspray` object.

Usage

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

Arguments

`x`	a `symbolicQspray` object

Value

An integer.

Note

The number of variables in the `symbolicQspray` object `Qlone(d)` is `d`, not `1`.

See Also

`involvedVariables`.

---

permuteVariables *Permute variables*

Description

Permute the variables of a `symbolicQspray` polynomial.

Usage

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

Arguments

`x`	a `symbolicQspray` object
`permutation`	a permutation

Value

A `symbolicQspray` object.

Examples

```
f <- function(a1, a2, X, Y, Z) {
  (a1^2 + 5*a2) / (a1 + 1) * X^2*Y  +  (3*a1 - a2) / a2 * Y^3
}
a1 <- qlone(1)
a2 <- qlone(2)
X <- Qlone(1)
Y <- Qlone(2)
Z <- Qlone(3)
Qspray <- f(a1, a2, X, Y, Z)
perm <- c(3, 1, 2)
permuteVariables(Qspray, perm) == f(a1, a2, Z, X, Y) # should be TRUE
```

`Qlone` *Polynomial variable*

Description

Creates a polynomial variable for a `symbolicQspray`.

Usage

```
Qlone(n)
```

Arguments

`n`	positive integer, the index of the variable

Value

A `symbolicQspray` object.

Examples

```
X <- Qlone(1)
Y <- Qlone(2)
(X + Y)^2
Qlone(0) == 1
```

`Qone` *The unit 'symbolicQspray' polynomial*

Description

Returns the `symbolicQspray` polynomial identically equal to 1.

Usage

```
Qone()
```

Value

A `symbolicQspray` object.

`Qzero` *The null 'symbolicQspray' polynomial*

Description

Returns the `symbolicQspray` polynomial identically equal to 0.

Usage

```
Qzero()
```

Value

A `symbolicQspray` object.

`rSymbolicQspray` *Random 'symbolicQspray'*

Description

Generates a random `symbolicQspray` object.

Usage

```
rSymbolicQspray()
```

Value

A `symbolicQspray` object.

showSymbolicQspray *Print a 'symbolicQspray' object*

Description

Prints a `symbolicQspray` object given a function to print a `ratioOfQsprays` object.

Usage

```
showSymbolicQspray(
  showRatioOfQsprays,
  showMonomial,
  lbrace = "{ ",
  rbrace = " }",
  addition = "  +  ",
  multiplication = " * "
)
```

Arguments

`showRatioOfQsprays`	a function which prints a `ratioOfQsprays` object
`showMonomial`	a function which prints a monomial, such as `showMonomialXYZ()` (and not `showMonomialXYZ`!)
`lbrace, rbrace`	used to enclose the coefficients
`addition`	used to separate the terms
`multiplication`	used to separate the coefficient and the monomial within a term

Value

A function which prints a `symbolicQspray` object.

Note

The function returned by this function is appropriate for usage in `showSymbolicQsprayOption<-` as the option `"showSymbolicQspray"` but in general we would rather use `showSymbolicQsprayX1X2X3` or `showSymbolicQsprayXYZ`, or rather set the options `"a"`, `"X"` and `"quotientBar"`.

See Also

`showSymbolicQsprayX1X2X3`, `showSymbolicQsprayXYZ`.

Examples

```
set.seed(421)
( Qspray <- rSymbolicQspray() )
showRatioOfQsprays <-
  showRatioOfQspraysXYZ(c("a", "b", "c"), quotientBar = " / ")
showMonomial <- showMonomialX1X2X3("X")
f <- showSymbolicQspray(showRatioOfQsprays, showMonomial, "{{{", "}}}")
f(Qspray)
# setting a show option:
showSymbolicQsprayOption(Qspray, "showSymbolicQspray") <- f
Qspray
# the show options are preserved by certain operations, e.g.:
2*Qspray
```

---

showSymbolicQsprayABCXYZ

Print a 'symbolicQspray' object

Description

Prints a `symbolicQspray` object.

Usage

```
showSymbolicQsprayABCXYZ(
  params,
  vars = c("X", "Y", "Z"),
  quotientBar = " %//% ",
  ...
)
```

Arguments

`params`	vector of strings, usually some letters, to denote the parameters of the polynomial
`vars`	a vector of strings, usually some letters, to denote the variables of the polynomial
`quotientBar`	a string for the quotient bar between the numerator and the denominator of a `ratioOfQsprays` object, including surrounding spaces, e.g. `" / "`
`...`	arguments other than `showRatioOfQsprays` and `showMonomial` passed to `showSymbolicQspray`

Value

A function which prints `symbolicQspray` objects.

Note

This function is built by applying `showSymbolicQspray` to `showRatioOfQspraysXYZ(params)` and `showMonomialXYZ(vars)`.

Examples

```
set.seed(421)
( Qspray <- rSymbolicQspray() )
showSymbolicQsprayABCXYZ(c("a", "b", "c"), c("U", "V"))(Qspray)
```

`showSymbolicQsprayOption<-` *Set a show option to a 'symbolicQspray' object*

Description

Set show option to a `symbolicQspray` object

Usage

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

Arguments

`x`	a `symbolicQspray` object
`which`	which option to set; this can be `"a"`, `"X"`, `"quotientBar"`, `"showMonomial"`, `"showRatioOfQsprays"` or `"showSymbolicQspray"`
`value`	the value for the option

Value

This returns the updated `symbolicQspray`.

Examples

```
set.seed(421)
Qspray <- rSymbolicQspray()
showSymbolicQsprayOption(Qspray, "a") <- "x"
showSymbolicQsprayOption(Qspray, "X") <- "A"
showSymbolicQsprayOption(Qspray, "quotientBar") <- " / "
Qspray
showSymbolicQsprayOption(Qspray, "showRatioOfQsprays") <-
  showRatioOfQspraysXYZ()
Qspray
```

showSymbolicQsprayX1X2X3 *Print a 'symbolicQspray' object*

Description

Prints a `symbolicQspray` object.

Usage

```
showSymbolicQsprayX1X2X3(a = "a", X = "X", quotientBar = " %//% ", ...)
```

Arguments

`a`	a string, usually a letter, to denote the non-indexed variables of the `ratioOfQsprays` coefficients
`X`	a string, usually a letter, to denote the non-indexed variables
`quotientBar`	a string for the quotient bar between the numerator and the denominator of a `ratioOfQsprays` object, including surrounding spaces, e.g. `"/"`
`...`	arguments other than `showRatioOfQsprays` and `showMonomial` passed to `showSymbolicQspray`

Value

A function which prints `symbolicQspray` objects.

Note

This function is built by applying `showSymbolicQspray` to `showRatioOfQspraysX1X2X3(a)` and `showMonomialX1X2X3(X)`.

Examples

```
set.seed(421)
Qspray <- rSymbolicQspray()
showSymbolicQsprayX1X2X3(quotientBar = " / ")(Qspray)
```

showSymbolicQsprayXYZ *Print a 'symbolicQspray' object*

Description

Prints a symbolicQspray object.

Usage

```
showSymbolicQsprayXYZ(
  a = "a",
  letters = c("X", "Y", "Z"),
  quotientBar = " %//% ",
  ...
)
```

Arguments

a	a string, usually a letter, to denote the non-indexed variables of the ratioOfQsprays coefficients
letters	a vector of strings, usually some letters, to denote the variables of the polynomial
quotientBar	a string for the quotient bar between the numerator and the denominator of a ratioOfQsprays object, including surrounding spaces, e.g. " / "
...	arguments other than showRatioOfQsprays and showMonomial passed to showSymbolicQspray

Value

A function which prints symbolicQspray objects.

Note

This function is built by applying showSymbolicQspray to showRatioOfQspraysX1X2X3(a) and showMonomialXYZ(letters).

Examples

```
set.seed(421)
Qspray <- rSymbolicQspray()
showSymbolicQsprayX1X2X3(quotientBar = " / ")(Qspray)
```

substituteParameters *Assign values to the parameters of a 'symbolicQspray'*

Description

Substitutes some values to the parameters of a `symbolicQspray` polynomial.

Usage

```
substituteParameters(Qspray, values)
```

Arguments

`Qspray`	a `symbolicQspray` object
`values`	vector of values to be substituted to the parameters; these values must be coercible to `bigq` numbers

Value

A `qspray` object.

See Also

Use `changeParameters` to apply a transformation of the parameters. Use `substituteVariables` to substitute some values to the variables.

Examples

```
library(symbolicQspray)
f <- function(a1, a2, X, Y) {
  (a1 + 2)*X^2*Y + (a2/(a1^2+a2))*X*Y
}
Qspray <- f(qlone(1), qlone(2), Qlone(1), Qlone(2))
a <- c(2, "2/3")
( qspray <- substituteParameters(Qspray, values = a) )
a <- gmp::as.bigq(a)
qspray == f(a[1], a[2], qlone(1), qlone(2)) ## should be TRUE
```

substituteVariables *Assign values to the variables of a 'symbolicQspray'*

Description

Substitutes some values to the variables of a `symbolicQspray` polynomial.

Usage

```
substituteVariables(Qspray, values)
```

Arguments

`Qspray`	a `symbolicQspray` object
`values`	vector of values to be substituted to the variables; these values must be coercible to `bigq` numbers

Value

A `ratioOfQsprays` object.

See Also

Use `changeVariables` to apply a transformation of the variables. Use `substituteParameters` to substitute some values to the parameters.

Examples

```
library(symbolicQspray)
f <- function(a1, a2, X, Y) {
  (a1 + 2)*X^2*Y + (a2/(a1^2+a2))*X*Y
}
a1 <- qlone(1); a2 <- qlone(2)
Qspray <- f(a1, a2, Qlone(1), Qlone(2))
values <- c(3, "2/3")
( rOQ <- substituteVariables(Qspray, values) )
values <- gmp::as.bigq(values)
rOQ == f(a1, a2, values[1], values[2]) ## should be TRUE
```

swapVariables *Swap variables*

Description

Swap two variables of a `symbolicQspray`.

Usage

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

Arguments

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

Value

A `symbolicQspray` object.

Examples

```
library(symbolicQspray)
f <- function(a1, a2, X, Y, Z) {
  (a1^2 + 5*a2) / (a1 + 1) * X^2*Y  +  (3*a1 - a2) / a2 * Y^3
}
a1 <- qlone(1)
a2 <- qlone(2)
X <- Qlone(1)
Y <- Qlone(2)
Z <- Qlone(3)
Qspray <- f(a1, a2, X, Y, Z)
swapVariables(Qspray, 2, 3) == f(a1, a2, X, Z, Y) # should be TRUE
```

symbolicQspray-unary *Unary operators for 'symbolicQspray objects*

Description

Unary operators for `symbolicQspray` objects.

Usage

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

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

Arguments

`e1`	object of class `symbolicQspray`
`e2`	nothing

Value

A `symbolicQspray` object.

`symbolicQspray_from_list` *(internal) Make a 'symbolicQspray' 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

```
symbolicQspray_from_list(x)
```

Arguments

`x`	list returned by the Rcpp function `returnSymbolicQspray`

Value

A `symbolicQspray` object.

Index