Package ‘jordan’

April 7, 2021

Type Package
Title A Suite of Routines for Working with Jordan Algebras
Version 1.0-1
Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>
Description A Jordan algebra is an algebraic object originally
designed to study observables in quantum mechanics. Jordan
algebras are commutative but non-associative; they satisfy the
Jordan identity. The package follows the ideas and notation of
Algebras''.
License GPL (>= 2)
Suggests knitr,rmarkdown
Depends onion (>= 1.4-0)
VignetteBuilder knitr
Imports emulator,methods,mathjaxr
URL https://github.com/RobinHankin/jordan
BugReports https://github.com/RobinHankin/jordan/issues
RdMacros mathjaxr

R topics documented:

  jordan-package ..................................................... 2
  Arith .............................................................. 3
  c ................................................................. 5
  coerce ............................................................ 6
  Compare-methods ................................................... 7
  extract ............................................................ 8
  id ................................................................. 9
  jordan ............................................................ 10
  jordan-class ..................................................... 10
  misc ............................................................. 11
  random ........................................................... 12
  r_to_n ........................................................... 12
  show ............................................................. 14
  valid ............................................................ 15
  zero ............................................................. 16
Description

A Jordan algebra is an algebraic object originally designed to study observables in quantum mechanics. Jordan algebras are commutative but non-associative; they satisfy the Jordan identity. The package follows the ideas and notation of K. McCrimmon (2004, ISBN:0-387-95447-3) "A Taste of Jordan Algebras".

Details

A Jordan algebra is a non-associative algebra over the reals with a multiplication that satisfies the following identities:

\[ xy = yx \]
\[ (xy)(xx) = x(y(xx)) \]

(the second identity is known as the Jordan identity). In literature one usually indicates multiplication by juxtaposition but one sometimes sees \( x \circ y \). Package idiom is to use an asterisk, as in \( x*y \).

There are five types of Jordan algebras:

1. Real symmetric matrices, class real_symmetric_matrix, abbreviated in the package to rsm
2. Complex Hermitian matrices, class complex_herm_matrix, abbreviated to chm
3. Quaternionic Hermitian matrices, class quaternion_herm_matrix, abbreviated to qhm
4. Albert algebras, the space of \( 3 \times 3 \) octonionic matrices, class albert
5. Spin factors, class spin

(of course, the first two are special cases of the next). The jordan package provides functionality to manipulate jordan objects using natural R idiom.

Objects of all these classes are stored in dataframe (technically, a matrix) form with columns being elements of the jordan algebra.

The first four classes are matrix-based in the sense that the algebraic objects are symmetric or Hermitian matrices (the S4 class is "jordan_matrix"). The fifth class, spin factors, is not matrix based.

One can extract the symmetric or Hermitian matrix from objects of class jordan_matrix using as.list(), which will return a list of symmetric or Hermitian matrices. A function name preceded by a "1" (for example as.1matrix() or vec_to_qhm1()) means that it deals with a single (symmetric or Hermitian) matrix.

Algebraically, the matrix form of jordan_matrix objects is redundant (for example, a real_symmetric_matrix of size \( n \times n \) has only \( n(n+1)/2 \) independent entries, corresponding to the upper triangular elements).

Author(s)

NA

Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>
References


Examples

rrsm()  # Random Real Symmetric matrices
rchm()  # Random Complex Hermitian matrices
rqhm()  # Random Quaternionic Hermitian matrices
ralbert()  # Random Albert algebra
rspin()  # Random spin factor

x <- rqhm(n=1)
y <- rqhm(n=1)
z <- rqhm(n=1)

x/1.2 + 0.3*x*y  # Arithmetic works as expected ...
x*(y*z) -(x*y)*z  # ... but '*' is not associative

## Verify the Jordan identity for type 3 algebras:

LHS <- (x*y)*(x*x)
RHS <- x*(y*(x*x))
diff <- LHS-RHS  # zero to numerical precision
diff[,drop=TRUE]  # result in matrix form

Arith Methods for Function Arith in package Jordan

Description

Methods for Arithmetic functions for jordans: +, -, *, /, ^

Usage

jordan_negative(z)
jordan_plus_jordan(e1,e2)
jordan_plus_numeric(e1,e2)
jordan_prod_numeric(e1,e2)
jordan_power_jordan(e1,e2)
albert_arith_albert(e1,e2)
albert_arith_numeric(e1,e2)
albert_inverse(e1)
albert_power_albert(...) 
albert_power_numeric(e1,e2)
albert_power_single_n(e1,n)
albert_prod_albert(e1,e2)
chm_arith_cm(e1,e2)
chm_arith_numeric(e1,e2)
Arith

chm_inverse(e1)
chm_power_numeric(e1,e2)
chm_prod_chm(e1,e2)
numeric_arith_albert(e1,e2)
numeric_arith_chm(e1,e2)
numeric_arith_qhm(e1,e2)
numeric_arith_rsm(e1,e2)
qhm_arith_numeric(e1,e2)
qhm_arith_qhm(e1,e2)
qhm_inverse(x)
qhm_power_numeric(e1,e2)
qhm_prod_qhm(e1,e2)
rsm_arith_numeric(e1,e2)
rsm_arith_rsm(e1,e2)
rsm_inverse(e1)
rsm_power_numeric(e1,e2)
rsm_prod_rsm(e1,e2)
spin_plus_numeric(e1,e2)
spin_plus_spin(e1,e2)
spin_power_numeric(e1,e2)
spin_power_single_n(e1,n)
spin_power_spin(...)
spin_prod_numeric(e1,e2)
spin_prod_spin(e1,e2)
spin_inverse(...)
spin_negative(e1)
vec_albertprod_vec(x,y)
vec_chmprod_vec(x,y)
vec_qhmprod_vec(x,y)
vec_rsmprod_vec(x,y)

Arguments

z,e1,e2  Jordan objects or numeric vectors
n  Integer for powers
...  Further arguments (ignored)
x,y  Numeric vectors, Jordan objects in independent form

Details

The package implements the Arith group of S4 generics so that idiom like A + B*C works as expected with jordans.

Functions like jordan_inverse() and jordan_plus_jordan() are low-level helper functions. The only really interesting operation is multiplication; functions like jordan_prod_jordan().

Names are implemented and the rules are inherited (via onion::harmonize_oo() and onion::harmonize_on()) from rbind().

Value

generally return jordans
Author(s)

Robin K. S. Hankin

Examples

\[
a <- rspin()
da
\]

Description

Combines its arguments to form a single jordan object.

Usage

```
## S4 method for signature 'jordan'
c(x,...)
```

Arguments

\[x,...\] Jordan objects

Details

Returns a concatenated jordan of the same type as its arguments. Argument checking is not performed.

Value

An XXX

Note

Names are inherited from the behaviour of `cbind()`, not `c()`.

Author(s)

Robin K. S. Hankin

Examples

\[
c(rqhm(),rqhm()*10)
\]
### coerce

**Description**

Various coercions needed in the package

**Usage**

```r
as.jordan(x, class)
vec_to_rsm1(x)
vec_to_chm1(x)
vec_to_qhm1(x)
vec_to_albert1(x)
rsm1_to_vec(M)
chm1_to_vec(M)
qhm1_to_vec(M)
albert1_to_vec(H)
as.real_symmetric_matrix(x, d, single = FALSE)
as.complex_herm_matrix(x, d, single = FALSE)
as.quaternion_herm_matrix(x, d, single = FALSE)
as.albert(x, single = FALSE)
numeric_to_real_symmetric_matrix(x, d)
numeric_to_complex_herm_matrix(x, d)
numeric_to_quaternion_herm_matrix(x, d)
numeric_to_albert(e1)
as.list(x, ...)
matrix1_to_jordan(x)
```

**Arguments**

- `x`, `e1` Numeric vector of independent entries
- `M`, `H` A matrix
- `d` Dimensionality of algebra
- `single` Boolean, indicating whether a single value is to be returned
- `class` Class of object
- `...` Further arguments, currently ignored

**Details**

The numeral “1” in a function name means a single element, usually a matrix.

Functions `vec_to_rsm1()` et seq convert a numeric vector to a (symmetric, complex, quaternion, octonion) matrix, that is, elements of a matrix-based Jordan algebra.

Functions `rsm1_to_vec()` convert a (symmetric, complex, quaternion, octonion) matrix to a numeric vector of independent components. The upper triangular components are used; no checking for symmetry is performed (the lower triangular components, and non-real components of the diagonal, are discarded).

Function `as.1matrix()` is used to convert a jordan object to a list of matrices. Length one jordan objects are converted to a matrix.
Functions `as.real_symmetric_matrix()`, `as.complex_herm_matrix()`, `as.quaternion_herm_matrix()` and `as.albert()` take a numeric matrix and return a (matrix-based) Jordan object.
Functions `numeric_to_real_symmetric_matrix()` have not been coded up yet.
Function `matrix1_to_jordan()` takes a matrix and returns a length-1 (matrix based) Jordan vector. It uses the class of the entries (real, complex, quaternion, octonion) to decide which type of Jordan to return.

**Value**
Return a coerced value.

**Author(s)**
Robin K. S. Hankin

**Examples**

```R
vec_to_chm(1:16) # Hermitian matrix
as.lmatrix(rchm())
as.complex_herm_matrix(matrix(runif(75), ncol=3))
matrix1_to_jordan(cprod(matrix(rnorm(35), 7, 5)))
matrix1_to_jordan(matrix(c(1, 1+1i, 1-1i, 3), 2, 2))
matrix1_to_jordan(Oil + matrix(1, 3, 3))
```

---

**Compare-methods**

Methods for comparison (equal to, greater than, etc) of jordans. Only equality makes sense.

**Usage**

```R
jordan_compare_jordan(e1, e2)
```

**Arguments**

e1, e2 Jordan objects

**Value**
Return a boolean

**Examples**

```R
# rspin() > 0 # meaningless and returns an error
```
Extract and replace methods for Jordan objects

Description

Extraction and replace methods for Jordan objects should work as expected.
Replace methods can take a Jordan or a numeric, but the numeric must be zero.

Value

Generally return a Jordan object of the same class as the first argument.

Methods

- signature(x = "albert", i = "index", j = "missing", drop = "logical"): ...
- signature(x = "complex_herm_matrix", i = "index", j = "missing", drop = "logical"): ...
- signature(x = "jordan", i = "index", j = "ANY", drop = "ANY"): ...
- signature(x = "jordan", i = "index", j = "missing", drop = "ANY"): ...
- signature(x = "quaternion_herm_matrix", i = "index", j = "missing", drop = "logical"): ...
- signature(x = "real_symmetric_matrix", i = "index", j = "missing", drop = "logical"): ...
- signature(x = "spin", i = "index", j = "missing", drop = "ANY"): ...
- signature(x = "spin", i = "missing", j = "index", drop = "ANY"): ...
- <- signature(x = "albert", i = "index", j = "missing", value = "albert"): ...
- <- signature(x = "complex_herm_matrix", i = "index", j = "ANY", value = "ANY"): ...
- <- signature(x = "complex_herm_matrix", i = "index", j = "missing", value = "complex_herm_matrix"): ...
- <- signature(x = "jordan_matrix", i = "index", j = "missing", value = "numeric"): ...
- <- signature(x = "quaternion_herm_matrix", i = "index", j = "missing", value = "quaternion_herm_matrix"): ...
- <- signature(x = "real_symmetric_matrix", i = "index", j = "missing", value = "real_symmetric_matrix"): ...
- <- signature(x = "spin", i = "index", j = "index", value = "ANY"): ...
- <- signature(x = "spin", i = "index", j = "missing", value = "numeric"): ...
- <- signature(x = "spin", i = "index", j = "missing", value = "spin"): ...

Author(s)

Robin K. S. Hankin
Examples

```r
showClass("index") # taken from the Matrix package
a <- rspin(7)
a[2:4] <- 0
a[5:7] <- a[1]*10
a
```

---

### id

#### Multiplicative identities

**Description**

Multiplying a jordan object by the identity leaves it unchanged.

**Usage**

```r
as.identity(x)
rsm_id(n,d)
chm_id(n,d)
qhm_id(n,d)
albert_id(n)
spin_id(n=3,d=5)
```

**Arguments**

- `n` : Length of vector to be created
- `d` : Dimensionality
- `x` : In function `as.identity()`, a jordan object. Return value will be a jordan object of the same dimensionality but entries equal to the identity

**Details**

The identity object in the matrix-based classes `jordan_matrix` is simply the identity matrix. Class spin has identity $(1, 0)$.

**Value**

Returns a jordan object.

**Author(s)**

Robin K. S. Hankin

**Examples**

```r
2+4
```
*jordan-class*

**Description**

The functions documented here are the creation methods for the five types of jordan algebra.

**Usage**

```r
real_symmetric_matrix(M)
complex_herm_matrix(M)
albert(M)
quaternion_herm_matrix(M)
spin(a,V)
```

**Arguments**

- **M**: A matrix with columns representing independent entries in a matrix-based Jordan algebra.
- **a, V**: Scalar and vector components of a spin factor.

**Details**

 Details here

**Value**

Return jordans or Boolean as appropriate.

**Author(s)**

Robin K. S. Hankin

**Examples**

```r
4+5
```
References


Examples

showClass("jordan")

---

**Miscellaneous Jordan functionality**

Description

Miscellaneous Jordan functionality that should be documented somewhere

Usage

```
harmonize_spin_numeric(e1,e2)
harmonize_spin_spin(e1,e2)
```

Arguments

```
e1,e2                   Objects to harmonize
```

Details

Miscellaneous low-level helper functions.

The harmonize functions `harmonize_spin_numeric()` and `harmonize_spin_spin()` work for spin objects for the matrix-based classes `emulator::harmonize_foo()` is used.

Value

These are mostly low-level helper functions; they not particularly user-friendly. They generally return either numeric or Jordan objects.

Author(s)

Robin K. S. Hankin
**random**  
*Random Jordan objects*

**Description**

Random jordan objects with specified properties

**Usage**

- `ralbert(n=3)`
- `rrsm(n=3, d=5)`
- `rchm(n=3, d=5)`
- `rqhm(n=3, d=5)`
- `rspin(n=3, d=5)`

**Arguments**

- `n`  
  Length of random object returned
- `d`  
  Dimensionality of random object returned

**Details**

These functions give a quick “get you going” random Jordan object to play with.

**Value**

Return a jordan object

**Author(s)**

Robin K. S. Hankin

**Examples**

- `rrsm()`
- `ralbert()`
- `rspin()`

---

**r_to_n**  
*Sizes of Matrix-based Jordan algebras*

**Description**

Given the number of rows in a (matrix-based) Jordan object, return the size of the underlying associative matrix algebra
Usage

```
r_to_n_rsm(r)
r_to_n_chm(r)
r_to_n_qhm(r)
r_to_n_albert(r=27)
n_to_r_rsm(n)
n_to_r_chm(n)
n_to_r_qhm(n)
n_to_r_albert(n=3)
```

Arguments

- `n`: Integer, underlying associative algebra being matrices of size $n \times n$
- `r`: Integer, number of rows of independent representation of a matrix-based jordan object

Details

These functions are here for consistency, and the `albert` ones for completeness.

For the record, they are:

- Real symmetric matrices, $r_{sm}$, $r = n(n + 1)/2$, $n = (\sqrt{1 + 4r} - 1)/2$
- Complex Hermitian matrices, $r_{chm}$, $r = n^2$, $n = \sqrt{r}$
- Quaternion Hermitian matrices, $r_{qhm}$, $r = n(2n - 1)$, $n = \sqrt{1 + 8r}/4$
- Albert algebras, $r = 27$, $n = 3$

Value

Return non-negative integers

Note

I have not been entirely consistent in my use of these functions.

Author(s)

Robin K. S. Hankin

Examples

```
r_to_n_qhm(nrow(rqhm()))
```
show methods, to display objects at the prompt

Usage

albert_show(x)
spin_show(x)
jordan_matrix_show(x)
description(x, plural=FALSE)

Arguments

x | Jordan object
plural | Boolean, indicating whether plural form is to be given

Details

The matrix-based algebras use a show method that modifies the row and column names of the underlying matrix slightly.
Spin factors are displayed with the scalar component offset from the vector component.
Print methods for special algebras are sensitive to the value of option head_and_tail, a two-element integer vector indicating the number of start lines and end lines to print.
Function description() gives a natural-language description of its argument, used in the print method.

Value

Returns the argument

Author(s)

Robin K. S. Hankin

Examples

rspin()
rqhm()
rchm()
Description

Validity methods, to check that objects are well-formed

Usage

valid_rsm(object)
valid_chm(object)
valid_qhm(object)
valid_albert(object)
is_ok_rsm(r)
is_ok_chm(r)
is_ok_qhm(r)
is_ok_albert(r)
is_ok_rsm(r)

Arguments

object Putative jordan object
r Integer, number of rows in putative jordan object

Details

Validity methods. The validity_foo() functions test for an object to be the right type, and the
is_ok_foo() functions test the number of rows being appropriate for a jordan object of some type;
these functions return an error if not appropriate, or, for jordan_matrix objects, the size of the
matrix worked with.

Value

Return a Boolean

Author(s)

Robin K. S. Hankin

Examples

is_ok_qhm(45)  # 5x5 Hermitian quaternionic matrices
#is_ok_qhm(46)  # FALSE
zero  

The zero Jordan object

Description
Package idiom for the zero Jordan object, and testing

Usage
is.zero(e1, e2 = 0)

Arguments
  e1  Jordan object to test for zeroness
  e2  Dummy numeric object to make the Arith method work

Details
One often wants to test a jordan object for being zero, and natural idiom would be rchm() == 0. The helper function is is.zero().

Value
Returns a Boolean

Author(s)
Robin K. S. Hankin

Examples
rrsm() * 0 == 0
Index

* classes
  extract, 8
  jordan-class, 10

* math
  Arith, 3
  Compare-methods, 7

* methods
  Arith, 3
  Compare-methods, 7

* package
  jordan-package, 2
  [,albert,index,missing,logical-method (extract), 8
  [,complex_herm_matrix,index,missing,logical-method (extract), 8
  [,jordan,index,ANY,ANY-method (extract), 8
  [,jordan,index,missing,ANY-method (extract), 8
  [,quaternion_herm_matrix,index,missing,logical-method (extract), 8
  [,real_symmetric_matrix,index,missing,logical-method (extract), 8
  [,spin,index,missing,ANY-method (extract), 8
  [,spin,missing,ANY-method (extract), 8
  [,spin,missing,missing,numeric-method (extract), 8
  [<-,albert,index,missing,albert-method (extract), 8
  [<-,complex_herm_matrix,index,ANY,ANY-method (extract), 8
  [<-,complex_herm_matrix,index,missing,complex_herm_matrix-method (extract), 8
  [<-,jordan_matrix,index,missing,numerical-methods (extract), 8
  [<-,quaternion_herm_matrix,index,missing,quaternion_herm_matrix-method (extract), 8
  [<-,real_symmetric_matrix,index,missing,real_symmetric_matrix-method (extract), 8
  [<-,spin,ANY,missing,ANY-method (extract), 8
  [<-,spin,index,index,ANY-method (extract), 8
  [<-,spin,index,missing,numerical-method (extract), 8
  [<-,spin,index,missing,spin-method (extract), 8
  [<-,spin,missing,ANY,numeric-method (extract), 8
  [<-,spin,missing,missing,numeric-method (extract), 8
  [<-,spin,missing,missing,spin-method (extract), 8
  albert (jordan), 10
  albert-class (jordan-class), 10
  albert1_to_vec (coerce), 6
  albert_arith_albert (Arith), 3
  albert_arith_numeric (Arith), 3
  albert_id (id), 9
  albert_inverse (Arith), 3
  albert_power_albert (Arith), 3
  albert_power_numeric (Arith), 3
  albert_power_single_n (Arith), 3
  albert_prod_albert (Arith), 3
  albert_show (show), 14
  Arith, 3
  Arith,ANY,jordan-method (Arith), 3
  Arith,jordan,ANY-method (Arith), 3
  Arith,jordan,jordan-method (Arith), 3
  Arith,jordan,missing-method (Arith), 3
  Arith,jordan,numerical-method (Arith), 3
  Arith-methods (Arith), 3
  as.1matrix (coerce), 6
  as.1matrix,albert-method (coerce), 6
  as.1matrix,complex_herm_matrix-method (coerce), 6
  as.1matrix,quaternion_herm_matrix-method (coerce), 6
  as.1matrix,real_symmetric_matrix-method (coerce), 6
  as.albert (coerce), 6
  as.complex_herm_matrix (coerce), 6
  as.id (id), 9