

# Package ‘onion’

October 14, 2022

**Version** 1.5-0

**Title** Octonions and Quaternions

**LazyData** TRUE

**Description** Quaternions and Octonions are four- and eight- dimensional extensions of the complex numbers. They are normed division algebras over the real numbers and find applications in spatial rotations (quaternions), and string theory and relativity (octonions). The quaternions are noncommutative and the octonions nonassociative. See the package vignette for more details.

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

**License** GPL-2

**Depends** methods, R (>= 3.5.0)

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**Imports** emulator

**URL** <https://github.com/RobinHankin/onion>

**BugReports** <https://github.com/RobinHankin/onion/issues>

**NeedsCompilation** yes

**Author** Robin K. S. Hankin [aut, cre] (<<https://orcid.org/0000-0001-5982-0415>>)

**Repository** CRAN

**Date/Publication** 2021-02-11 07:00:02 UTC

## R topics documented:

onion-package . . . . .	2
Arith . . . . .	4
biggest . . . . .	6
bind . . . . .	6
bunny . . . . .	7
c . . . . .	8

Compare-methods . . . . .	9
Complex . . . . .	9
condense . . . . .	11
cumsum . . . . .	12
Extract . . . . .	13
length . . . . .	14
Logic . . . . .	15
Math . . . . .	16
names . . . . .	17
O1 . . . . .	18
onion . . . . .	19
onion-class . . . . .	21
onionmat . . . . .	22
orthogonal . . . . .	25
p3d . . . . .	26
plot . . . . .	27
prods . . . . .	27
rep . . . . .	29
roct . . . . .	30
rotate . . . . .	31
seq . . . . .	32
show . . . . .	32
sum . . . . .	33
threeform . . . . .	34
zapsmall . . . . .	35

<b>Index</b>	<b>37</b>
--------------	-----------

---

onion-package	<i>Octonions and Quaternions</i>
---------------	----------------------------------

---

## Description

Quaternions and Octonions are four- and eight- dimensional extensions of the complex numbers. They are normed division algebras over the real numbers and find applications in spatial rotations (quaternions), and string theory and relativity (octonions). The quaternions are noncommutative and the octonions nonassociative. See the package vignette for more details.

## Details

Package:	onion
Version:	1.5-0
Title:	Octonions and Quaternions
LazyData:	TRUE
Authors@R:	person(given=c("Robin", "K. S."), family="Hankin", role = c("aut","cre"), email="hankin.robin@gmail.com")
Description:	Quaternions and Octonions are four- and eight- dimensional extensions of the complex numbers. They are
Maintainer:	Robin K. S. Hankin <hankin.robin@gmail.com>

License: GPL-2  
 Depends: methods, R (>= 3.5.0)  
 Suggests: testthat, knitr, rmarkdown  
 VignetteBuilder: knitr  
 Imports: emulator  
 URL: <https://github.com/RobinHankin/onion>  
 BugReports: <https://github.com/RobinHankin/onion/issues>  
 Author: Robin K. S. Hankin [aut, cre] (<<https://orcid.org/0000-0001-5982-0415>>)

## Index of help topics:

Arith	Methods for Function Arith in package Onion
Compare-methods	Methods for compare S4 group
01	Unit onions
Re	Complex functionality for onions
biggest	Returns the biggest type of a set of onions
bind	Binding of onionmats
bunny	The Stanford Bunny
c	Concatenation
condense	Condense an onionic vector into a short form
cumsum	Cumulative sums and products of onions
i	Extract or Replace Parts of onions or glubs
length	Length of an octonionic vector
log	Various logarithmic and circular functions for onions
logic.onion	Logical operations on onions
names.onion	Names of an onionic vector
onion	Basic onion functions
onion-class	Class "onion"
onion-package	Octonions and Quaternions
onionmat	Onionic matrices
orthogonal	Orthogonal matrix equivalents
p3d	Three dimensional plotting
plot	Plot onions
prods	Various products of two onions
rep	Replicate elements of onionic vectors
roct	Random onionic vector
rotate	Rotates 3D vectors using quaternions
seq	seq method for onions
show	Print method for onions
sum	Various summary statistics for onions
threeform	Various non-field diagnostics
zapsmall	Concatenation

There are precisely four normed division algebras over the reals: the reals themselves, the complex numbers, the quaternions, and the octonions. The R system is well equipped to deal with the first two: the **onion** package provides some functionality for the third and fourth.

**Author(s)**

NA

Maintainer: Robin K. S. Hankin &lt;hankin.robin@gmail.com&gt;

**References**

R. K. S. Hankin 2006. "Normed division algebras in R: introducing the onion package". *R News*, Volume 6, number 2

**Examples**

```
rquat(10) # random quaternions

Ok + (0i + 0j1)/(0j-0i1) # basic octonions

x <- roct(10)
y <- roct(10)
z <- roct(10)

x*(y*z) - (x*y)*z # nonassociative!
```

---

Arith

---

*Methods for Function Arith in package Onion*


---

**Description**

Methods for Arithmetic functions for onions: +, -, \*, /, ^

**Usage**

```
onion_negative(z)
onion_inverse(z)
onion_arith_onion(e1, e2)
onion_arith_numeric(e1, e2)
numeric_arith_onion(e1, e2)
harmonize_oo(a, b)
harmonize_on(a, b)
onion_plus_onion(a, b)
onion_plus_numeric(a, b)
onion_prod_onion(e1, e2)
octonion_prod_octonion(o1, o2)
quaternion_prod_quaternion(q1, q2)
onion_prod_numeric(a, b)
onion_power_singleinteger(o, n)
onion_power_numeric(o, p)
```

**Arguments**

`z, e1, e2, a, b, o, o1, o2, n, q1, q2, p`  
 onions or numeric vectors

**Details**

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

Functions like `onion_inverse()` and `onion_plus_onion()` are low-level helper functions. The only really interesting operation is multiplication; functions `octionion_prod_octionion()` and `quaternion_prod_quaternion()` dispatch to `C`.

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

**Value**

generally return an onion

**Note**

Previous versions of the package included the option to use native `R` rather than the faster compiled `C` code used here. But this was very slow and is now discontinued.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- rquat()
b <- rquat()
a
Re(a)
j(a) <- 0.2
a*b
b*a # quaternions are noncommutative

x <- as.octionion(matrix(rnorm(40),nrow=8))
y <- roct()
z <- roct()

x*(y*z) - (x*y)*z # octionions are nonassociative [use associator()]
```

---

biggest	<i>Returns the biggest type of a set of onions</i>
---------	--

---

### Description

Returns the biggest type of a set of onions; useful for “promoting” a set of onions to the most general type.

### Usage

```
biggest(...)
```

### Arguments

...                    Onionic vectors

### Details

If any argument passed to `biggest()` is an octonion, then return the string “octonion”. Failing that, if any argument is a quaternion, return the string “quaternion”, and failing that, return “scalar”.

### Value

Character string representing the type

### Author(s)

Robin K. S. Hankin

### Examples

```
biggest(01,rquat(100),1:4)
```

---

bind	<i>Binding of onionmats</i>
------	-----------------------------

---

### Description

Methods for `rbind()` and `cbind()` of onionmats. These are implemented by specifying methods for `rbind2()` and `cbind2()`.

### Usage

```
bind_onion(x,bind,...)
bind_onion_onion(x,y,bind,...)
bind_onion_onionmat(x,y,bind,...)
bind_onionmat_onion(x,y,bind,...)
```

**Arguments**

<code>x, y</code>	Onions or onionmats
<code>bind</code>	Either <code>rbind</code> or <code>cbind</code> as appropriate
<code>...</code>	Further arguments

**Value**

Return onionmats

**Author(s)**

Robin K. S. Hankin

**Examples**

```
rbind(rquat(3), rquat(3))
```

```
cbind(diag(5), roct(1))
```

```
cbind(matrix(0:1, 4, 2), matrix(roct(12), 4, 3))
```

---

bunny

*The Stanford Bunny*

---

**Description**

A set of 3D points in the shape of a rabbit (the Stanford Bunny)

**Usage**

```
data(bunny)
```

**Format**

A three column matrix with 35947 rows. Each row is the Cartesian coordinates of a point on the surface of the bunny.

**Value**

as for format

**Source**

<https://graphics.stanford.edu/data/3Dscanrep/>

**Examples**

```
data(bunny)
p3d(rotate(bunny,Hk))
```

---

c

*Concatenation*


---

**Description**

Combines its arguments to form a single onion.

**Usage**

```
c_onionpair(x,y)
## S4 method for signature 'onion'
c(x,...)
```

**Arguments**

`x,y,...`          onions

**Details**

Returns an onion of the same type as its arguments. Names are inherited from the behaviour of `cbind()`, not `c()`.

**Value**

An onion

**Note**

The method is not perfect; it will not, for example, coerce its arguments to the `biggest()` type, so `c(rquat(),roct())` will fail. You will have to coerce the arguments by hand.

Dispatch is based on the class of the first argument, so `c(1,rquat())` will return a list (not an onion), and `c(rquat(),1)` will fail.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- roct(3)
b <- seq_onion(from=0i1,to=0j, len=6)
c(a,b)

c(rquat(3),H1,H0,Him)
```



---

 Compare-methods

 Methods for compare S4 group
 

---

### Description

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

### Value

Return a boolean

### Examples

```
# roct() > 0 # meaningless and returns an error

x <- as.octonion(matrix(sample(0:1,800,TRUE,p=c(9,1)),nrow=8))
y <- as.octonion(matrix(sample(0:1,800,TRUE,p=c(9,1)),nrow=8))
x==y

matrix(as.quaternion(100+1:12),3,4) == 102
```

---

 Complex

 Complex functionality for onions
 

---

### Description

Functionality in the Complex group.

The *norm* Norm( $O$ ) of onion  $O$  is the product of  $O$  with its conjugate:  $|O| = OO^*$  but a more efficient numerical method is used (see dotprod()).

The *Mod* Mod( $O$ ) of onion  $O$  is the square root of its norm.

The *sign* of onion  $O$  is the onion with the same direction as  $O$  but with unit Norm: sign( $O$ )= $O$ /Mod( $O$ ).

Function Im() sets the real component of its argument to zero, and Conj() flips the sign of its argument's non-real components.

**Usage**

```
## S4 method for signature 'onion'  
Re(z)  
## S4 method for signature 'onion'  
Im(z)  
Re(z) <- value  
Im(x) <- value  
## S4 method for signature 'onion'  
Conj(z)  
## S4 method for signature 'onion'  
Mod(z)  
onion_abs(x)  
onion_conjugate(z)  
## S4 method for signature 'onion'  
sign(x)
```

**Arguments**

<code>x, z</code>	Object of class <code>onion</code> or <code>glub</code>
<code>value</code>	replacement value

**Value**

All functions documented here return a numeric vector or matrix of the same dimensions as their argument, apart from functions `Im()` and `Conj()`, which return an object of the same class as its argument.

**Note**

If `x` is a numeric vector and `y` an `onion`, one might expect typing `x[1] <- y` to result in `x` being a `onion`. This is impossible, according to John Chambers.

Extract and set methods for components such as `i, j, k` are documented at `Extract.Rd`

**Author(s)**

Robin K. S. Hankin

**See Also**

[Extract](#)

**Examples**

```
a <- rquat()  
Re(a)  
Re(a) <- j(a)  
  
Im(a)
```

```
b <- roamat()

A <- roamat()
Im(A) <- Im(A)*10
```

---

condense	<i>Condense an onionic vector into a short form</i>
----------	---

---

### Description

Condense an onion into a string vector showing whether the elements are positive, zero or negative.

### Usage

```
condense(x, as.vector=FALSE)
```

### Arguments

x	An onionic vector
as.vector	Boolean, indicating whether to return a vector or matrix

### Value

If `as.vector` is TRUE, return a string vector of the same length as `x` whose elements are length 4 or 8 strings for quaternions or octonions respectively. If FALSE, return a matrix with these columns.

The characters are “+” for a positive, “-” for a negative, and “0” for a zero, element.

### Author(s)

Robin K. S. Hankin

### Examples

```
condense(roct(3))
condense(roct(3), as.vector=TRUE)
```

---

`cumsum`*Cumulative sums and products of onions*

---

**Description**

Cumulative sums and products of onions

**Usage**

```
onion_cumsum(x)
onion_cumprod(x)
```

**Arguments**

`x`                    `onion`

**Value**

An onion

**Note**

The octonions are nonassociative but `cumprod()` operates left-associatively, as in  $((a[1]*a[2])*a[3])*a[4]$  etc.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
cumsum(as.quaternion(matrix(runif(20),4,5)))
cumsum(roct(5))

cumprod(rquat(7))
```

---

 Extract

---

*Extract or Replace Parts of onions or glubs*


---

**Description**

Methods for "[" and "[<-", i.e., extraction or subsetting of onions.

**Usage**

```
## S4 method for signature 'onion'
i(z)
## S4 method for signature 'onion'
j(z)
## S4 method for signature 'onion'
k(z)
## S4 method for signature 'octionion'
l(z)
## S4 method for signature 'octionion'
il(z)
## S4 method for signature 'octionion'
jl(z)
## S4 method for signature 'octionion'
kl(z)
## S4 method for signature 'onionmat'
i(z)
## S4 method for signature 'onionmat'
j(z)
## S4 method for signature 'onionmat'
k(z)
## S4 method for signature 'onionmat'
il(z)
## S4 method for signature 'onionmat'
jl(z)
## S4 method for signature 'onionmat'
kl(z)
i(x) <- value
j(x) <- value
k(x) <- value
l(x) <- value
il(x) <- value
jl(x) <- value
kl(x) <- value
```

**Arguments**

x, z	Object of class onion
value	replacement value

**Value**

Extraction and methods return an onion or onionmat. Replacement methods return an object of the same class as x.

**Note**

If x is a numeric vector and y a onion, one might expect typing `x[1] <- y` to result in x being a onion. This is impossible, according to John Chambers.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- roct(9)
il(a)
Re(a) <- 1:9

j(a) <- l(a)
a
```

---

length

*Length of an octonionic vector*

---

**Description**

Get or set the length of onions

**Usage**

```
## S4 method for signature 'onion'
length(x)
```

**Arguments**

x                    An onion

**Details**

Operates on the columns of the matrix as expected.

**Value**

integer

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- roct(5)
length(a)
```

---

Logic

*Logical operations on onions*

---

**Description**

Logical operations on onions are not supported

**Usage**

```
onion_logic(e1,e2)
```

**Arguments**

e1, e2            onions

**Value**

none

**Note**

Carrying out logical operations in this group will report an error. Negation, “!”, is not part of this group.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
# roct() & roct()    # reports an error
```

**Description**

Various elementary functions for onions

**Usage**

```
onion_log(x,base=exp(1))
onion_exp(x)
onion_sign(x)
onion_sqrt(x)
onion_cosh(x)
onion_sinh(x)
onion_acos(x)
onion_acosh(x)
onion_asin(x)
onion_asinh(x)
onion_atan(x)
onion_atanh(x)
onion_cos(x)
onion_sin(x)
onion_tan(x)
onion_tanh(x)
onion_cos(x)
onion_sin(x)
onion_tan(x)
onion_tanh(x)
```

**Arguments**

x	Object of class onion
base	In function <code>log()</code> , the base of the logarithm

**Details**

Standard math stuff. I am not convinced that the trig functions (`sin()` etc) have any value.

**Author(s)**

Robin K. S. Hankin



**Examples**

```

x <- roct()
exp(x+x) - exp(x)*exp(x) # zero to numerical precision

jj <- exp(log(x)/2)      # use sqrt() here
jj*jj-x                  # also small

y <- roct()
exp(x+y) - exp(x)*exp(y) # some rules do not operate for onions

max(Mod(c(sin(asin(x))-x,asin(sin(x))-x))) # zero to numerical precision

```

---

names	<i>Names of an onionic vector</i>
-------	-----------------------------------

---

**Description**

Functions to get or set the names of an onion

**Usage**

```

## S4 method for signature 'onion'
names(x)
## S4 method for signature 'onionmat'
rownames(x)
## S4 method for signature 'onionmat'
colnames(x)
## S4 method for signature 'onionmat'
dimnames(x)
## S4 method for signature 'onionmat'
dim(x)

```

**Arguments**

x                   onion

**Details**

Names attributes refers to colnames of the internal matrix, which are retrieved or set using colnames() or colnames<-().

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- roct(5)
names(a) <- letters[1:5]

b <- romat()
dimnames(b) <- list(month = month.abb[1:5], location=names(islands)[1:6])
```

---

O1

*Unit onions*

---

**Description**

Each of the eight unit quaternions and octonions

**Usage**

H1  
Hi  
Hj  
Hk  
H0  
Him  
Hall  
O1  
Oi  
Oj  
Ok  
Ol  
Oil  
Ojl  
O0  
Oim  
Oall

**Format**

Each one is an onionic vector of length one.

## Details

Try `Hi (=quaternion(i=1))` to get the pattern for the first four. The next ones are the zero quaternion, the pure imaginary quaternion with all components 1, and the quaternion with all components 1. The ones beginning with “O” follow a similar pattern.

These are just variables that may be overwritten and thus resemble T and F whose value may be changed.

## Value

A length-one onion, either a quaternion or an octonion

## Examples

```
Oall
seq_onion(from=01,to=0i1,len=6)

stopifnot(Hj*Hk == Hi)
stopifnot(Ok1*0i1 == -0j ) # See tests/test_aaa.R for the full set
```

---

onion	<i>Basic onion functions</i>
-------	------------------------------

---

## Description

Construct, coerce to, test for, and print onions

## Usage

```
octonion(length.out = NULL, Re = 0, i = 0, j = 0,
          k = 0, l = 0, il = 0, jl = 0, kl = 0)
as.octonion(x, single = FALSE)
is.octonion(x)
quaternion(length.out = NULL, Re = 0, i = 0, j = 0, k = 0)
as.quaternion(x, single = FALSE)
is.quaternion(x)
is.onion(x)
as.onion(x, type, single=FALSE)
quaternion_to_octonion(from)
octonion_to_quaternion(from)
## S4 method for signature 'onion'
as.matrix(x)
## S4 method for signature 'onion'
as.numeric(x)
```

**Arguments**

<code>length.out</code>	In functions <code>quaternion()</code> and <code>octonion()</code> , the length of the onionic vector returned
<code>Re</code>	The real part of the onionic vector returned
<code>i, j, k</code>	In functions <code>quaternion()</code> and <code>octonion()</code> , component $i, j, k$ respectively of the returned onion
<code>l, il, jl, kl</code>	In function <code>octonion()</code> , component $l, il, jl, kl$ respectively of the returned octonion
<code>x, from</code>	Onion to be tested or printed
<code>single</code>	In functions <code>as.octonion()</code> and <code>as.quaternion()</code> , Boolean with default <code>FALSE</code> meaning to interpret <code>x</code> as a vector of reals to be coerced into an onion with zero imaginary part; and <code>TRUE</code> meaning to interpret <code>x</code> as a length 4 (or length 8) vector and return the corresponding single onion.
<code>type</code>	In function <code>as.onion()</code> a string either “quaternion” or “octonion” denoting the algebra to be forced into

**Details**

Functions `quaternion()` and `octonion()` use standard recycling where possible; `rbind()` is used.

Functions `as.quaternion()` and `as.octonion()` coerce to quaternions and octonions respectively. If given a complex vector, the real and imaginary components are interpreted as `Re` and `i` respectively.

The output of `type()` is accepted as the `type` argument of function `as.onion()`; thus `as.onion(out, type=type(x))` works as expected.

**Value**

Generally return onions

**Note**

An *onion* is any algebra (over the reals) created by an iterated Cayley-Dickson process. Examples include quaternions, octonions, and sedenions. There does not appear to be a standard generic term for such objects (I have seen n-ion, anion and others. But “onion” is pronouncable and a bona fide English word).

Creating further onions—such as the sedenions—is intended to be straightforward.

There is a nice example of the onion package in use in the `permutations` package, under `cayley.Rd`. This also shows the quaternion group  $Q_8$ , but from a different perspective.

**Author(s)**

Robin K. S. Hankin

**Examples**

```

x <- octonion(Re=1,il=1:3)
x
kl(x) <- 100
x

as.quaternion(diag(4))

# Cayley table for the quaternion group Q8:
a <- c(H1,-H1,Hi,-Hi,Hj,-Hj,Hk,-Hk)
names(a) <- c("+1","-1","+i","-i","+j","-j","+k","-k")

f <- Vectorize(function(x,y){names(a)[a==a[x]*a[y]]})
X <- noquote(outer(1:8,1:8, f))
rownames(X) <- names(a)
colnames(X) <- names(a)
X

```

---

onion-class

*Class "onion"*


---

**Description**

The formal S4 class for onion and onionmat objects

**Objects from the Class**

Class *onion* is a virtual S4 class extending classes *quaternion* and *octonion*. In package documentation, “*onion*” means an R object that behaves as a vector of quaternions or octonions, stored as a four- or eight- row numeric matrix.

Class *onionmat* is the S4 class for matrices whose elements are quaternions or octonions. An onionmat is stored as a two-element list, the first being an onion and the second an integer matrix which holds structural matrix attributes such as dimensions and dimnames. Most standard arithmetic R idiom for matrices should work for onionmats.

Class *index* is taken from the excellent **Matrix** package and is a `setClassUnion()` of classes `numeric`, `logical`, and `character`, which mean that it is an arity-one matrix index.

**Author(s)**

Robin K. S. Hankin

## Examples

```

as.octonion(1:8,single=TRUE)
as.quaternion(matrix(runif(20),nrow=4))

H <- matrix(rquat(21),3,7)
dimnames(H) <- list(foo=letters[1:3],bar=state.abb[1:7])

i(H) <- 0.1

I <- matrix(rquat(14),7,2)
dimnames(I) <- list(foo=state.abb[1:7],baz=LETTERS[1:2])
H %**% I

```

---

onionmat

*Onionic matrices*


---

## Description

Simple functionality for quaternionic and octonionic matrices, intended for use in the jordan package. Use idiom like `matrix(Him, 4, 5)` or `matrix(roct(6), 2, 3)` to create an onionmat object, a matrix of onions.

The package is intended to match base R's matrix functionality in the sense that standard R idiom just goes through for onionic matrices. Determinants are not well-defined for quaternionic or octonionic matrices, and matrix inverses are not implemented.

## Usage

```

newonionmat(d, M)
onionmat(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
as.onionmat(x)
is.onionmat(x)
onionmat_negative(e1)
onionmat_inverse(e1)
onionmat_prod_onionmat(e1,e2)
onionmat_power_onionmat(...)
onionmat_prod_single(x,y)
onionmat_power_single(e1,e2)
onionmat_plus_onionmat(e1,e2)
matrix_arith_onion(e1,e2)
onion_arith_matrix(e1,e2)
matrix_plus_onion(e1,e2)
matrix_prod_onion(e1,e2)
drop(x)
## S4 method for signature 'onionmat,onionmat'
cprod(x,y)

```

```
## S4 method for signature 'onionmat,missing'  
cprod(x,y)  
## S4 method for signature 'onionmat,ANY'  
cprod(x,y)  
## S4 method for signature 'ANY,ANY'  
cprod(x,y)  
## S4 method for signature 'onion,missing'  
cprod(x,y)  
## S4 method for signature 'onion,onion'  
cprod(x,y)  
## S4 method for signature 'onion,onionmat'  
cprod(x,y)  
## S4 method for signature 'onionmat,onion'  
cprod(x,y)  
## S4 method for signature 'onionmat,onionmat'  
tcprod(x,y)  
## S4 method for signature 'onionmat,missing'  
tcprod(x,y)  
## S4 method for signature 'onionmat,ANY'  
tcprod(x,y)  
## S4 method for signature 'ANY,ANY'  
tcprod(x,y)  
## S4 method for signature 'onion,missing'  
cprod(x,y)  
## S4 method for signature 'onion,onion'  
cprod(x,y)  
## S4 method for signature 'onion,onionmat'  
cprod(x,y)  
## S4 method for signature 'onionmat,onion'  
cprod(x,y)  
## S4 method for signature 'onionmat'  
t(x)  
## S4 method for signature 'onion'  
t(x)  
## S4 method for signature 'onionmat'  
ht(x)  
## S4 method for signature 'onion'  
ht(x)  
nrow(x)  
ncol(x)  
herm_onion_mat(real_diagonal, onions)  
onionmat_complex(z)  
onionmat_conjugate(z)  
onionmat_imag(z)  
onionmat_re(z)  
onionmat_mod(z)  
onionmat_matrixprod_onionmat(x,y)  
onion_matrixprod_onionmat(x,y)
```

```
onionmat_matrixprod_numeric(x,y)
onionmat_matrixprod_onion(x,y)
```

### Arguments

d,M	data and matrix index
x,y,z,e1,e2	Objects of class onionmat
data,nrow,ncol,byrow,dimnames	In function onionmat(), as for matrix()
...	Further arguments (currently ignored)
real_diagonal, onions	In function herm_onion_mat(), on- and off- diagonal elements of an Hermitian matrix

### Details

An object of class onionmat is a two-element list, the first of which is an onion, and the second an integer matrix used for tracking attributes such as dimensions and dimnames. This device makes the extraction and replacement methods easy.

The S4 method for matrix() simply dispatches to onionmat(), which is a drop-in replacement for matrix().

Function newonionmat() is lower-level: it also creates onionmat objects, but takes two arguments: an onion and a matrix; the matrix argument can be used to specify additional attributes via attr(), but this ability is not currently used in the package.

Functions such as onionmat\_plus\_onionmat() are low-level helper functions, not really designed for the end-user.

Vignette onionmat shows some use-cases.

### Author(s)

Robin K. S. Hankin

### Examples

```
matrix(rquat(28),4,7)
```

```
M <- onionmat(rquat(10),2,5)
cprod(M)
```

```
Re(M)
Re(M) <- 0.3
```

```
romat() %*% rquat(6)
```



---

 orthogonal

*Orthogonal matrix equivalents*


---

**Description**

Convert a quaternion to and from an equivalent orthogonal matrix

**Usage**

```
matrix2quaternion(M)
as.orthogonal(Q)
```

**Arguments**

M	A three-by-three orthogonal matrix
Q	A vector of quaternions

**Value**

Function `matrix2quaternion()` returns a quaternion.

Function `as.orthogonal()` returns either a  $3 \times 3$  matrix or a  $3 \times 3 \times n$  array of orthogonal matrices

**Note**

Function `matrix2quaternion()` is low-level; use `as.quaternion()` to convert arrays.

**Author(s)**

Robin K. S. Hankin

**See Also**

[rotate](#)

**Examples**

```
as.orthogonal(rquat(1))

o <- function(w){diag(3)-2*outer(w,w)/sum(w^2)} # Householder
matrix2quaternion(o(1:3)) # Booorrriiinnggg
matrix2quaternion(o(1:3) %*% o(3:1))

Q <- rquat(7)
Q <- Q/abs(Q)
as.quaternion(as.orthogonal(Q)) # +/- Q
```

```
A <- replicate(7,o(rnorm(3)) %% o(rnorm(3)))
max(abs(as.orthogonal(as.quaternion(A))-A))
```

---

p3d

*Three dimensional plotting*


---

### Description

Three dimensional plotting of points. Produces a nice-looking 3D scatterplot with greying out of further points givin a visual depth cue

### Usage

```
p3d(x, y, z, xlim = NULL, ylim = NULL, zlim = NULL, d0 = 0.2, h = 1, ...)
```

### Arguments

<code>x,y,z</code>	vector of $x, y, z$ coordinates to be plotted. If <code>x</code> is a matrix, interpret the rows as 3D Cartesian coordinates
<code>xlim,ylim,zlim</code>	Limits of plot in the $x, y, z$ directions, with default NULL meaning to use <code>range()</code>
<code>d0</code>	E-folding distance for graying out (depths are standardized to be between 0 and 1)
<code>h</code>	The hue for the points, with default value of 1 corresponding to red. If NULL, produce black points greying to white
<code>...</code>	Further arguments passed to <code>persp()</code> and <code>points()</code>

### Value

Value returned is that given by function `trans3d()`.

### Author(s)

Robin K. S. Hankin

### See Also

[bunny](#)

### Examples

```
data(bunny)
p3d(bunny, theta=3, phi=104, box=FALSE)
```

---

plot

*Plot onions*


---

**Description**

Plotting method for onionic vectors

**Usage**

```
## S4 method for signature 'onion'
plot(x,y, ...)
```

**Arguments**

<code>x,y</code>	Onions
<code>...</code>	Further arguments passed to <code>plot.default()</code>

**Details**

The function is `plot(Re(x), Mod(Im(x)), ...)`, and thus behaves similarly to `plot()` when called with a complex vector.

**Value**

Called for its side-effect of plotting a diagram

**Author(s)**

Robin K. S. Hankin

**Examples**

```
plot(roct(30))
```

---

prods

*Various products of two onions*


---

**Description**

Returns various inner and outer products of two onionic vectors.

**Usage**

```

x %<*>% y
x %>*%<% y
x %<.>% y
x %>.<% y
x %.% y
onion_g_even(x,y)
onion_g_odd (x,y)
onion_e_even(x,y)
onion_e_odd (x,y)
dotprod(x,y)

```

**Arguments**

x,y                    onions

**Details**

This page documents an attempt at a consistent notation for onionic products. The default product for onions (viz “\*”) is sometimes known as the “Grassman product”. There is another product known as the Euclidean product defined by  $E(p, q) = p'q$  where  $x'$  is the conjugate of  $x$ .

Each of these products separates into an “even” and an “odd” part, here denoted by functions `g_even()` and `g_odd()` for the Grassman product, and `e_even()` and `e_odd()` for the Euclidean product. These are defined as follows:

- $g\_even(x, y) = (xy + yx) / 2$
- $g\_odd(x, y) = (xy - yx) / 2$
- $e\_even(x, y) = (x'y + y'x) / 2$
- $e\_odd(x, y) = (x'y - y'x) / 2$

These functions have an equivalent binary operator.

The Grassman operators have a “\*”; they are “%<\*>%” for the even Grassman product and “%>\*%<%” for the odd product.

The Euclidean operators have a “.”; they are “%<.>%” for the even Euclidean product and “%>.<%” for the odd product.

Function `dotprod()` returns the Euclidean even product of two onionic vectors. That is, if  $x$  and  $y$  are eight-element vectors of the components of two onions, return `sum(x*y)`.

Note that the returned value is a numeric vector (compare `%<.>%`, `e.even()`, which return onionic vectors with zero imaginary part).

There is no binary operator for the ordinary Euclidean product (it seems to be rarely needed in practice). For `Conj(x)*x`, `Norm(x)` is much more efficient and accurate.

Function `prod()` is documented at `Summary.Rd`.

**Note**

Frankly if you find yourself using these operators you might be better off using the **clifford** package, which has an extensive and consistent suite of product operators.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
Oj %<.>% Oall
```

---

rep

*Replicate elements of onionic vectors*

---

**Description**

Replicate elements of onionic vectors

**Usage**

```
## S4 method for signature 'onion'  
rep(x, ...)
```

**Arguments**

x	Onionic vector
...	Further arguments passed to seq.default()

**Author(s)**

Robin K. S. Hankin

**Examples**

```
a <- roct(3)  
rep(a,2) + a[1]  
rep(a,each=2)  
rep(a,length.out=5)
```

---

`roct`*Random onionic vector*

---

**Description**

Random quaternion or octonion vectors and matrices

**Usage**

```
rquat(n=5)
roct(n=5)
romat(type="quaternion", nrow=5, ncol=6, ...)
```

**Arguments**

<code>n</code>	Length of random vector returned
<code>nrow, ncol, ...</code>	Further arguments specifying properties of the returned matrix
<code>type</code>	string specifying type of elements

**Details**

Function `rquat()` returns a quaternionic vector, `roct()` returns an octonionic vector, and `romat()` a quaternionic matrix.

Functions `rquat()` and `roct()` give a quick “get you going” random onion to play with. Function `romat()` gives a simple onionmat, although arguably `matrix(roct(4), 2, 2)` is as convenient.

**Author(s)**

Robin K. S. Hankin

**References**

K. Shoemake 1992. “Uniform random rotations”. In D. Kirk, editor, *Graphics Gems III* pages 129-130. Academic, New York.

**Examples**

```
rquat(3)
roct(3)
plot(roct(30))

romat()
```

---

rotate	<i>Rotates 3D vectors using quaternions</i>
--------	---

---

**Description**

Rotates a three-column matrix whose rows are vectors in 3D space, using quaternions

**Usage**

```
rotate(x, H)
```

**Arguments**

x	A matrix of three columns whose rows are points in 3D space
H	A quaternion. Does not need to have unit modulus

**Value**

Returns a matrix of the same size as x

**Author(s)**

Robin K. S. Hankin

**See Also**

[orthogonal](#)

**Examples**

```
data(bunny)
par(mfrow=c(2,2))
par(mai=rep(0,4))
p3d(rotate(bunny,Hi),box=FALSE)
p3d(rotate(bunny,H1-Hi+Hj),box=FALSE)
p3d(rotate(bunny,Hk),box=FALSE)
p3d(rotate(bunny,Hall),box=FALSE)

o <- function(w){diag(3)-2*outer(w,w)/sum(w^2)} # Householder
O <- o(1:3) %*% o(3:1)

rotate(bunny,as.quaternion(O))
bunny %*% t(O) # should be the same; note transpose
```

---

seq	<i>seq method for onions</i>
-----	------------------------------

---

**Description**

Rough equivalent of seq() for onions.

**Usage**

```
seq_onion(from=1, to=1, by=((to-from)/(length.out-1)), length.out=NULL, slerp=FALSE, ...)
```

**Arguments**

from	Onion for start of sequence
to	Onion for end of sequence
by	Onion for interval
length.out	Length of vector returned
slerp	Boolean, with default FALSE meaning to use linear interpolation and TRUE meaning to use spherical linear interpolation (useful for animating 3D rotation)
...	Further arguments (currently ignored)

**Author(s)**

Robin K. S. Hankin

**Examples**

```
seq(from=0i, to=0i1, length.out=6)
seq(from=H1, to=(Hi+Hj)/2, len=10, slerp=TRUE)
```

---

show	<i>Print method for onions</i>
------	--------------------------------

---

**Description**

Show methods for onions

**Usage**

```
## S4 method for signature 'onion'
show(object)
onion_show(x, h=getOption("show_onions_horizontally"))
```



**Arguments**

x, object	Onions
h	Boolean, with default FALSE meaning to print horizontally and TRUE meaning to print by columns.

**Details**

If options("horiz") is TRUE, then print by rows rather than columns (provided that the default value of argument h is not overridden). The default behaviour is to print by columns; do this by setting horiz to anything other than TRUE, including leaving it unset.

**Note**

Print method for onionmat objects is also sensitive to this option.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
roct(4)
```

---

sum

*Various summary statistics for onions*

---

**Description**

Various summary statistics for onions

**Usage**

```
onion_allsum(x)
## S4 method for signature 'onion'
sum(x)
## S4 method for signature 'quaternion'
prod(x)
## S4 method for signature 'octonion'
sum(x)
## S4 method for signature 'onionmat'
sum(x)
## S4 method for signature 'octonion'
prod(x)
## S4 method for signature 'onion'
str(object, ...)
str_onion(object, vec.len = 4, ...)
onion_allsum(x)
onionmat_allsum(x)
quaternion_allprod(x)
```

**Arguments**

`x, object, ...`    Objects of class `onion`  
`vec.len`            number of elements to display

**Details**

For a `onion` object, return the sum or product accordingly

**Value**

Return an `onion`

**Note**

Function `str()` uses functionality from `condense()`.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
sum(roct())
str(roct())
```

---

threeform

*Various non-field diagnostics*

---

**Description**

Diagnostics of non-field behaviour: `threeform`, `associator`, `commutator`

**Usage**

```
threeform(x1, x2, x3)
associator(x1, x2, x3)
commutator(x1, x2)
```

**Arguments**

`x1, x2, x3`            onionic vectors

**Details**

The `threeform` is defined as  $\text{Re}(x1 * (\text{Conj}(x2) * x3) - x3 * (\text{Conj}(x2) * x1))/2$ ;  
the `associator` is  $(x1 * x2) * x3 - x1 * (x2 * x3)$ ;  
the `commutator` is  $x1 * x2 - x2 * x1$ .

**Value**

Returns an octonionic vector.

**Author(s)**

Robin K. S. Hankin

**Examples**

```
x <- roct(7) ; y <- roct(7) ; z <- roct(7)
associator(x,y,z)
```

---

zapsmall

*Concatenation*


---

**Description**

Zapping small components to zero

**Usage**

```
## S4 method for signature 'onion'
zapsmall(x,digits=getOption("digits"))
## S4 method for signature 'onionmat'
zapsmall(x,digits=getOption("digits"))
```

**Arguments**

x                    An onion or onionmat  
digits                integer indicating the precision to be used as in `base::zapsmall()`

**Details**

Uses `base::zapsmall()` to zap small elements to zero.

**Value**

An onion

**Author(s)**

Robin K. S. Hankin

**Examples**

```
zapsmall(as.octonion(0.01^(1:8),single=TRUE))
```

```
a <- roct(7)  
x <- a^1/a  
x  
zapsmall(x)
```

# Index

- \* **array**
  - c, 8
  - condense, 11
  - cumsum, 12
  - length, 14
  - names, 17
  - plot, 27
  - prods, 27
  - rep, 29
  - seq, 32
  - show, 32
  - threeform, 34
- \* **classes**
  - onion-class, 21
- \* **datasets**
  - bunny, 7
  - 01, 18
- \* **hplot**
  - p3d, 26
- \* **math**
  - Arith, 4
  - biggest, 6
  - Compare-methods, 9
  - Complex, 9
  - Extract, 13
  - Logic, 15
  - Math, 16
  - sum, 33
- \* **methods**
  - Arith, 4
  - Compare-methods, 9
- \* **misc**
  - onion, 19
  - orthogonal, 25
  - rotate, 31
- \* **package**
  - onion-package, 2
- + ,onion,missing-method (onionmat), 22
- + ,onionmat,missing-method (onionmat), 22
- ,onion,missing-method (onionmat), 22
- ,onionmat,missing-method (onionmat), 22
- [ (Extract), 13
- [ ,onion,ANY,ANY-method (Extract), 13
- [ ,onion,index,ANY,ANY-method (Extract), 13
- [ ,onion,index,ANY-method (Extract), 13
- [ ,onion,index,missing,ANY-method (Extract), 13
- [ ,onion,index,missing-method (Extract), 13
- [ ,onion-method (Extract), 13
- [ ,onionmat,ANY,ANY,ANY-method (Extract), 13
- [ ,onionmat,ANY,ANY-method (Extract), 13
- [ ,onionmat,index,index,ANY-method (Extract), 13
- [ ,onionmat,index,index-method (Extract), 13
- [ ,onionmat,index,missing,ANY-method (Extract), 13
- [ ,onionmat,index,missing,missing-method (Extract), 13
- [ ,onionmat,index,missing-method (Extract), 13
- [ ,onionmat,matrix,missing,ANY-method (Extract), 13
- [ ,onionmat,matrix,missing-method (Extract), 13
- [ ,onionmat,missing,index,ANY-method (Extract), 13
- [ ,onionmat,missing,index-method (Extract), 13
- [ ,onionmat,missing,missing,ANY-method (Extract), 13
- [ ,onionmat,missing,missing-method (Extract), 13
- [ .onion (Extract), 13
- [ .onionmat (onionmat), 22

- [<- (Extract), 13
- [<-, onion, ANY, ANY-method (Extract), 13
- [<-, onion, index, ANY, ANY-method (Extract), 13
- [<-, onion, index, missing, ANY-method (Extract), 13
- [<-, onion, index, missing, numeric-method (Extract), 13
- [<-, onion, index, missing, onion-method (Extract), 13
- [<-, onion, missing, missing, numeric-method (Extract), 13
- [<-, onion, missing, missing, onion-method (Extract), 13
- [<-, onion-method (Extract), 13
- [<-, onionmat, ANY, missing, numeric-method (Extract), 13
- [<-, onionmat, ANY, missing, onion-method (Extract), 13
- [<-, onionmat, index, index, numeric-method (Extract), 13
- [<-, onionmat, index, index, onion-method (Extract), 13
- [<-, onionmat, index, missing, numeric-method (Extract), 13
- [<-, onionmat, index, missing, onion-method (Extract), 13
- [<-, onionmat, missing, index, numeric-method (Extract), 13
- [<-, onionmat, missing, index, onion-method (Extract), 13
- [<-.onion (Extract), 13
- [<-.onionmat (onionmat), 22
- %% (onionmat), 22
- %%, numeric, onion-method (onionmat), 22
- %%, numeric, onionmat-method (onionmat), 22
- %%, onion, onionmat-method (onionmat), 22
- %%, onionmat, numeric-method (onionmat), 22
- %%, onionmat, onion-method (onionmat), 22
- %%, onionmat, onionmat-method (onionmat), 22
- %.% (prods), 27
- %<\*>% (prods), 27
- %<.>% (prods), 27
- %>\*<% (prods), 27
- %>.<% (prods), 27
- abs, onion-method (Math), 16
- acos (Math), 16
- acos, onion-method (Math), 16
- acosh (Math), 16
- acosh, onion-method (Math), 16
- Arith, 4
- Arith, ANY, onion-method (Arith), 4
- Arith, onion, ANY-method (Arith), 4
- Arith, onion, missing-method (Arith), 4
- Arith, onion, numeric-method (Arith), 4
- Arith, onion, onion-method (Arith), 4
- Arith-methods (Arith), 4
- as.matrix (onion), 19
- as.matrix, onion-method (onion), 19
- as.numeric, onion-method (onion), 19
- as.octonion (onion), 19
- as.octonionmat (onionmat), 22
- as.onion (onion), 19
- as.onionmat (onionmat), 22
- as.orthogonal (orthogonal), 25
- as.quaternion (onion), 19
- as.quaternionmat (onionmat), 22
- asin (Math), 16
- asin, onion-method (Math), 16
- asinh (Math), 16
- asinh, onion-method (Math), 16
- associator (threeform), 34
- atan (Math), 16
- atan, onion-method (Math), 16
- atanh (Math), 16
- atanh, onion-method (Math), 16
- biggest, 6
- bind, 6
- bind\_onion (bind), 6
- bind\_onion\_matrix (bind), 6
- bind\_onion\_onion (bind), 6
- bind\_onion\_onionmat (bind), 6
- bind\_onionmat\_onion (bind), 6
- bind\_onionmat\_onionmat (bind), 6
- bunny, 7, 26
- c, 8
- c, onion-method (c), 8
- c.onion (c), 8
- c\_onionpair (c), 8
- cbind (bind), 6
- cbind2, matrix, onion-method (bind), 6
- cbind2, matrix, onionmat-method (bind), 6

- cbind2,numeric,onion-method (bind), 6
- cbind2,numeric,onionmat-method (bind), 6
- cbind2,onion,matrix-method (bind), 6
- cbind2,onion,numeric-method (bind), 6
- cbind2,onion,onion-method (bind), 6
- cbind2,onion,onionmat-method (bind), 6
- cbind2,onionmat,matrix-method (bind), 6
- cbind2,onionmat,numeric-method (bind), 6
- cbind2,onionmat,onion-method (bind), 6
- cbind2,onionmat,onionmat-method (bind), 6
- colnames (names), 17
- colnames,onion-method (names), 17
- colnames,onionmat-method (names), 17
- colnames<- (onionmat), 22
- colnames<- ,onionmat-method (names), 17
- commutator (threeform), 34
- Compare,ANY,onionmat-method (Compare-methods), 9
- Compare,numeric,onion-method (Compare-methods), 9
- Compare,onion,numeric-method (Compare-methods), 9
- Compare,onion,onion-method (Compare-methods), 9
- Compare,onionmat,ANY-method (Compare-methods), 9
- Compare,onionmat,onionmat-method (Compare-methods), 9
- Compare-methods, 9
- Complex, 9
- concatenate.onion (c), 8
- condense, 11
- Conj (Complex), 9
- Conj,onion-method (Complex), 9
- Conj,onionmat-method (Complex), 9
- cos (Math), 16
- cos,onion-method (Math), 16
- cosh (Math), 16
- cosh,onion-method (Math), 16
- cprod (onionmat), 22
- cprod,ANY,ANY-method (onionmat), 22
- cprod,ANY,missing-method (onionmat), 22
- cprod,ANY,onionmat-method (onionmat), 22
- cprod,onion,missing-method (onionmat), 22
- cprod,onion,onion-method (onionmat), 22
- cprod,onion,onionmat-method (onionmat), 22
- cprod,onionmat,ANY-method (onionmat), 22
- cprod,onionmat,missing-method (onionmat), 22
- cprod,onionmat,onion-method (onionmat), 22
- cprod,onionmat,onionmat-method (onionmat), 22
- cumsum, 12
- diag (onionmat), 22
- diag,onion-method (onionmat), 22
- diag,onionmat-method (onionmat), 22
- diag.onion (onionmat), 22
- diag.onionmat (onionmat), 22
- diag<- ,onionmat-method (onionmat), 22
- diag<-.onionmat (onionmat), 22
- dim (names), 17
- dim,onionmat-method (names), 17
- dim<- (names), 17
- dim<- ,onionmat-method (names), 17
- dimnames,onionmat-method (names), 17
- dimnames<- ,onionmat-method (names), 17
- dotprod (prods), 27
- drop (onionmat), 22
- drop,onionmat-method (onionmat), 22
- e\_even.onion (prods), 27
- e\_odd.onion (prods), 27
- exp (Math), 16
- exp,onion-method (Math), 16
- Extract, 10, 13
- g\_even.onion (prods), 27
- g\_odd.onion (prods), 27
- getd (onionmat), 22
- getM (onionmat), 22
- H0 (O1), 18
- H1 (O1), 18
- Hall (O1), 18
- harmonize\_on (Arith), 4
- harmonize\_oo (Arith), 4
- herm\_onion\_mat (onionmat), 22
- Hi (O1), 18
- Him (O1), 18
- Hj (O1), 18
- Hk (O1), 18
- ht (onionmat), 22

- ht, onion-method (onionmat), 22
- ht, onionmat-method (onionmat), 22
- i (Extract), 13
- i, onion-method (Extract), 13
- i, onionmat-method (Extract), 13
- i.octonion (Extract), 13
- i.quaternion (Extract), 13
- i<- (Extract), 13
- i<-, onion-method (Extract), 13
- i<-, onionmat-method (Extract), 13
- i<- .octonion (Extract), 13
- i<- .quaternion (Extract), 13
- il (Extract), 13
- il, octonion-method (Extract), 13
- il, onionmat-method (Extract), 13
- il.octonion (Extract), 13
- il<- (Extract), 13
- il<-, octonion-method (Extract), 13
- il<-, onionmat-method (Extract), 13
- il<- .octonion (Extract), 13
- Im (Complex), 9
- Im, onion-method (Complex), 9
- Im, onionmat-method (Complex), 9
- Im<- (Complex), 9
- Im<-, onion-method (Complex), 9
- Im<-, onionmat-method (Complex), 9
- Im<- .quaternion (Extract), 13
- index-class (onion-class), 21
- is.octonion (onion), 19
- is.onion (onion), 19
- is.onionmat (onionmat), 22
- is.quaternion (onion), 19
- is\_orthogonal (orthogonal), 25
- j (Extract), 13
- j, onion-method (Extract), 13
- j, onionmat-method (Extract), 13
- j.octonion (Extract), 13
- j.quaternion (Extract), 13
- j<- (Extract), 13
- j<-, onion-method (Extract), 13
- j<-, onionmat-method (Extract), 13
- j<- .octonion (Extract), 13
- j<- .quaternion (Extract), 13
- jl (Extract), 13
- jl, octonion-method (Extract), 13
- jl, onionmat-method (Extract), 13
- jl.octonion (Extract), 13
- jl<- (Extract), 13
- jl<-, octonion-method (Extract), 13
- jl<-, onionmat-method (Extract), 13
- jl<- .octonion (Extract), 13
- jl<- .quaternion (Extract), 13
- kl (Extract), 13
- kl, octonion-method (Extract), 13
- kl, onionmat-method (Extract), 13
- kl.octonion (Extract), 13
- kl<- (Extract), 13
- kl<-, octonion-method (Extract), 13
- kl<-, onionmat-method (Extract), 13
- kl<- .octonion (Extract), 13
- kl<- .quaternion (Extract), 13
- l (Extract), 13
- l, octonion-method (Extract), 13
- l, onion-method (Extract), 13
- l, onionmat-method (Extract), 13
- l.octonion (Extract), 13
- l<- (Extract), 13
- l<-, octonion-method (Extract), 13
- l<-, onionmat-method (Extract), 13
- l<- .octonion (Extract), 13
- length, 14
- length, onion-method (length), 14
- length.onion (length), 14
- length<- (length), 14
- length<-, onion-method (length), 14
- length<- .onion (length), 14
- log (Math), 16
- log, onion-method (Math), 16
- Logic, 15
- Logic, ANY, onion-method (Logic), 15
- Logic, onion, ANY-method (Logic), 15
- Logic, onion, missing-method (Logic), 15
- Logic, onion-method (Logic), 15
- logic.onion (Logic), 15
- Math, 16



- matrix,onion-method (onionmat), 22
- matrix2quaternion (orthogonal), 25
- matrix\_arith\_onion (onionmat), 22
- matrix\_arith\_onionmat (onionmat), 22
- matrix\_plus\_onion (onionmat), 22
- matrix\_plus\_onionmat (onionmat), 22
- matrix\_prod\_onion (onionmat), 22
- max (sum), 33
- min (sum), 33
- Mod (Complex), 9
- Mod,onion-method (Complex), 9
- Mod,onionmat-method (Complex), 9
- names, 17
- names,onion-method (names), 17
- names,onionmat-method (names), 17
- names.onion (names), 17
- names.onion<- (names), 17
- names<- (names), 17
- names<-,onion-method (names), 17
- names<-,onionmat-method (names), 17
- names<-.onion (names), 17
- ncol (onionmat), 22
- ncol,ANY-method (onionmat), 22
- ncol,onionmat-method (names), 17
- ncol-methods (onionmat), 22
- ncol.onionmat (onionmat), 22
- newonionmat (onionmat), 22
- Norm (Complex), 9
- Norm,onion-method (Complex), 9
- Norm,onionmat-method (Complex), 9
- Norm.onion (Complex), 9
- nrow (onionmat), 22
- nrow,ANY-method (onionmat), 22
- nrow,onionmat-method (names), 17
- nrow-methods (onionmat), 22
- nrow.onionmat (onionmat), 22
- numeric\_arith\_onion (Arith), 4
- numeric\_arith\_onionmat (onionmat), 22
- numeric\_matrixprod\_onionmat (onionmat), 22
- 00 (01), 18
- 01, 18
- 0all (01), 18
- Octonion (onion), 9
- octonion (onion), 19
- octonion-class (onion-class), 21
- octonion\_prod\_octonion (Arith), 4
- octonion\_to\_quaternion (onion), 19
- Oi (01), 18
- Oil (01), 18
- Oim (01), 18
- Oj (01), 18
- Ojl (01), 18
- Ok (01), 18
- Ok1 (01), 18
- Ol (01), 18
- om\_cprod (onionmat), 22
- om\_ht (onionmat), 22
- om\_prod (onionmat), 22
- om\_tcprod (onionmat), 22
- onion, 19
- onion-class, 21
- onion-package, 2
- onion\_abs (Complex), 9
- onion\_acos (Math), 16
- onion\_acosh (Math), 16
- onion\_allsum (sum), 33
- onion\_arith\_matrix (onionmat), 22
- onion\_arith\_numeric (Arith), 4
- onion\_arith\_onion (Arith), 4
- onion\_arith\_onionmat (onionmat), 22
- onion\_arith\_single (onionmat), 22
- onion\_asin (Math), 16
- onion\_asinh (Math), 16
- onion\_atan (Math), 16
- onion\_atanh (Math), 16
- onion\_compare (Compare-methods), 9
- onion\_complex (Complex), 9
- onion\_conjugate (Complex), 9
- onion\_cos (Math), 16
- onion\_cosh (Math), 16
- onion\_cumprod (cumsum), 12
- onion\_cumsum (cumsum), 12
- onion\_e\_even (prods), 27
- onion\_e\_odd (prods), 27
- onion\_exp (Math), 16
- onion\_g\_even (prods), 27
- onion\_g\_odd (prods), 27
- onion\_imag (Complex), 9
- onion\_inverse (Arith), 4
- onion\_log (Math), 16
- onion\_logic (Logic), 15
- onion\_matrixprod\_onionmat (onionmat), 22
- onion\_mod (Complex), 9
- onion\_negative (Arith), 4

- onion\_plus\_numeric (Arith), 4
- onion\_plus\_onion (Arith), 4
- onion\_power\_matrix (onionmat), 22
- onion\_power\_numeric (Arith), 4
- onion\_power\_singleinteger (Arith), 4
- onion\_prod\_numeric (Arith), 4
- onion\_prod\_onion (Arith), 4
- onion\_re (Complex), 9
- onion\_show (show), 32
- onion\_sign (Math), 16
- onion\_sin (Math), 16
- onion\_sinh (Math), 16
- onion\_sqrt (Math), 16
- onion\_tan (Math), 16
- onion\_tanh (Math), 16
- onionmat, 22
- onionmat-class (onion-class), 21
- onionmat\_allsum (sum), 33
- onionmat\_arith\_matrix (onionmat), 22
- onionmat\_arith\_onion (onionmat), 22
- onionmat\_arith\_onionmat (onionmat), 22
- onionmat\_arith\_single (onionmat), 22
- onionmat\_compare\_onionmat  
(Compare-methods), 9
- onionmat\_compare\_single  
(Compare-methods), 9
- onionmat\_complex (onionmat), 22
- onionmat\_conjugate (onionmat), 22
- onionmat\_equal\_onionmat  
(Compare-methods), 9
- onionmat\_equal\_single  
(Compare-methods), 9
- onionmat\_imag (onionmat), 22
- onionmat\_inv (onionmat), 22
- onionmat\_inverse (onionmat), 22
- onionmat\_matrixprod\_numeric (onionmat),  
22
- onionmat\_matrixprod\_onion (onionmat), 22
- onionmat\_matrixprod\_onionmat  
(onionmat), 22
- onionmat\_mod (onionmat), 22
- onionmat\_neg (onionmat), 22
- onionmat\_negative (onionmat), 22
- onionmat\_plus\_matrix (onionmat), 22
- onionmat\_plus\_onionmat (onionmat), 22
- onionmat\_plus\_single (onionmat), 22
- onionmat\_power\_matrix (onionmat), 22
- onionmat\_power\_onionmat (onionmat), 22
- onionmat\_power\_single (onionmat), 22
- onionmat\_prod\_matrix (onionmat), 22
- onionmat\_prod\_onionmat (onionmat), 22
- onionmat\_prod\_single (onionmat), 22
- onionmat\_re (onionmat), 22
- onionmat\_show (show), 32
- onionmat\_unary (onionmat), 22
- onionmatprod (onionmat), 22
- Ops.onionmat (onionmat), 22
- orthogonal, 25, 31
- p3d, 26
- plot, 27
- plot, onion-method (plot), 27
- plot.onion (plot), 27
- print (show), 32
- print, onion-method (show), 32
- print.octonion (show), 32
- print.onion (show), 32
- print.onionmat (show), 32
- print.quaternion (show), 32
- prod (sum), 33
- prod, octonion-method (sum), 33
- prod, quaternion-method (sum), 33
- prods, 27
- Quaternion (onion), 19
- quaternion (onion), 19
- quaternion-class (onion-class), 21
- quaternion\_allprod (sum), 33
- quaternion\_prod\_quaternion (Arith), 4
- quaternion\_to\_octonion (onion), 19
- range (sum), 33
- rbind (bind), 6
- rbind2, matrix, onion-method (bind), 6
- rbind2, matrix, onionmat-method (bind), 6
- rbind2, numeric, onion-method (bind), 6
- rbind2, numeric, onionmat-method (bind), 6
- rbind2, onion, matrix-method (bind), 6
- rbind2, onion, numeric-method (bind), 6
- rbind2, onion, onion-method (bind), 6
- rbind2, onion, onionmat-method (bind), 6
- rbind2, onionmat, matrix-method (bind), 6
- rbind2, onionmat, numeric-method (bind), 6
- rbind2, onionmat, onion-method (bind), 6
- rbind2, onionmat, onionmat-method (bind),  
6
- Re (Complex), 9

- Re, onion-method (Complex), 9
- Re, onionmat-method (Complex), 9
- Re<- (Complex), 9
- Re<- , onion-method (Complex), 9
- Re<- , onionmat-method (Complex), 9
- Re<- . quaternion (Extract), 13
- rep, 29
- rep, onion-method (rep), 29
- rep.onion (rep), 29
- roct, 30
- romat (roct), 30
- ronionmat (roct), 30
- rotate, 25, 31
- rownames (names), 17
- rownames, ANY-method (onionmat), 22
- rownames, onionmat-method (names), 17
- rownames-methods (onionmat), 22
- rownames.onionmat (onionmat), 22
- rownames<- (onionmat), 22
- rownames<- , ANY-method (onionmat), 22
- rownames<- , onionmat-method (names), 17
- rownames<-methods (onionmat), 22
- rownames<- .onionmat (onionmat), 22
- rquat (roct), 30
- seq, 32
- seq, onion-method (seq), 32
- seq.onion (seq), 32
- seq\_onion (seq), 32
- show, 32
- show, onion-method (show), 32
- sign, onion-method (Complex), 9
- sin (Math), 16
- sin, onion-method (Math), 16
- single\_arith\_onionmat (onionmat), 22
- single\_compare\_onionmat (Compare-methods), 9
- single\_power\_onionmat (onionmat), 22
- single\_prod\_onionmat (onionmat), 22
- sinh (Math), 16
- sinh, onion-method (Math), 16
- SLERP (seq), 32
- slerp (seq), 32
- sqrt (Math), 16
- str, onion-method (sum), 33
- str\_onion (sum), 33
- sum, 33
- sum, octonion-method (sum), 33
- sum, onion-method (sum), 33
- sum, onionmat-method (sum), 33
- sum, quaternion-method (sum), 33
- Summary, onion-method (sum), 33
- t, onion-method (onionmat), 22
- t, onionmat-method (onionmat), 22
- t.onion (onionmat), 22
- t.onionmat (onionmat), 22
- tan (Math), 16
- tan, onion-method (Math), 16
- tanh (Math), 16
- tanh, onion-method (Math), 16
- tcprod (onionmat), 22
- tcprod, ANY, ANY-method (onionmat), 22
- tcprod, ANY, missing-method (onionmat), 22
- tcprod, ANY, onionmat-method (onionmat), 22
- tcprod, onion, missing-method (onionmat), 22
- tcprod, onion, onion-method (onionmat), 22
- tcprod, onion, onionmat-method (onionmat), 22
- tcprod, onionmat, ANY-method (onionmat), 22
- tcprod, onionmat, missing-method (onionmat), 22
- tcprod, onionmat, onion-method (onionmat), 22
- tcprod, onionmat, onionmat-method (onionmat), 22
- threeform, 34
- type (onion), 19
- zap (zapsmall), 35
- zapsmall, 35
- zapsmall, onion-method (zapsmall), 35
- zapsmall, onionmat-method (zapsmall), 35