Package 'onion'

July 2, 2014

Version 1.2-4

Date 31-10-2007

Title octonions and quaternions

Author Robin K. S. Hankin

Description A collection of routines to manipulate and visualize quaternions and octonions.

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

License GPL

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2011-12-27 10:06:16

NeedsCompilation yes

R topics documented:

`onion-package` *Manipulation of quaternions and octonions*

Description

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.

Details

Package:	onion
Type:	Package
Version:	1.0
Date:	2007-05-01
License:	GPL

The package is intended to provide transparent access to quaternions and octonions.

Package currently S3 but will use S4 methods shortly

Author(s)

Robin K. S. Hankin

Maintainer: hankin.robin@gmail.com

References

R News article

Examples

```
as.quaternion(1:10)  # quaternionic vector with zero imaginary components

1:10 + Hj  # Simple nontrivial quaternion; note appropriate behaviour of '+'

1:10 + Oil   # simple octonionic vector ('Oil' is one of the octonionic bases).

a <- rquat(5)
b <- rquat(5)  #Quaternionic vectors with random integer components

a*b - b*a     # Nonzero!  (quaternions are not commutative)

Re(a)  #  Re() extracts the real component

i(a) <- 1000 ; a  # individual components may be manipulated intuitively

as.octonion(a)  # 'upgrades' to octonion

x <- roct(5)  # random octonionic vector with integer components
y <- roct(5)
z <- roct(5)

(x*y)*z - z*(y*z)  # Nonzero!  (octonions are not associative)

Norm(x)
Mod(x)  # Modulus and Norm work as expected

# Now some plotting:

a <- as.octonion(c(7,8,3,3,7,1,3,3),single=TRUE)
b <- as.octonion(c(8,4,2,8,3,7,3,7),single=TRUE)
plot(exp(seq(from=a,to=b,len=50)))
# Note operation of seq(), exp(), and plot()
```

---

as.matrix.onion *Coerces an onionic vector into a matrix*

Description

Coerces an onionic vector into a matrix.

Usage

```
## S3 method for class 'onion'
as.matrix(x, ...)
```

Arguments

x	Onionic vector
...	Further arguments (currently ignored)

Details

The matrix returned has 4 rows for a quaternion, 8 for an octonion

Author(s)

Robin K. S. Hankin

Examples

```
det(as.matrix(roct(8)))
```

---

biggest *Returns the biggest type of a set of onions*

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".

Author(s)

Robin K. S. Hankin

Examples

```
biggest(O1,rquat(100),1:4)
```

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.

Source

http://www-graphics.stanford.edu

Examples

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

c.onion *Combine onionic vectors into a single vector*

Description

Combines its arguments to form an onionic vector.

Usage

```
## S3 method for class 'onion'
c(...)
```

Arguments

...	onionic vectors

Details

Returns an onionic vector of type `biggest()` of the arguments.

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

Author(s)

Robin K. S. Hankin

Examples

```
a <- roct(3)
b <- seq(from=Oil,to=Oj,len=6)
c(a,b)

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

c(Hi,Oil,1:2)
```

`condense.onion` *Condense an onionic vector into a short form*

Description

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

Usage

```
## S3 method for class 'onion'
condense(x)
```

Arguments

`x`	An onionic vector

Value

Returns a string vector of the same length as `x` whose elements are length 4 or 8 strings for quaternions or octonions respectively. 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))
```

Conj.onion *Onionic conjugation*

Description

Returns the conjugate of an onionic vector

Usage

```
## S3 method for class 'onion'
Conj(z)
```

Arguments

z	An onionic vector

Details

The method is common to all onions: `Im(x) <- -Im(x)`

Author(s)

Robin K. S. Hankin

Examples

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

cumsum.onion *Cumulative sums for onions*

Description

Cumulative sum of an onionic vector

Usage

```
## S3 method for class 'onion'
cumsum(x)
```

Arguments

x	Onionic vector

Author(s)

Robin K. S. Hankin

Examples

```
a <- roct(7)
cumsum(a)
```

`dotprod` *Euclidean even product*

Description

Euclidean even product; dot product of two onions

Usage

```
dotprod(x, y)
```

Arguments

`x,y`	Onionic vectors

Details

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).

Author(s)

Robin K. S. Hankin

See Also

`prods`

Examples

```
a <- roct(5)
b <- roct(1)

a %.% b
a %.% (a+3)
```

exp *Elementary transcendental functions*

Description

Elementary transcendental functions: exponential and trig

Usage

```
## S3 method for class 'onion'
exp(x)
## S3 method for class 'onion'
log(x,base=exp(1))
## S3 method for class 'onion'
sin(x)
## S3 method for class 'onion'
cos(x)
## S3 method for class 'onion'
tan(x)
## S3 method for class 'onion'
asin(x)
## S3 method for class 'onion'
acos(x)
## S3 method for class 'onion'
atan(x)
## S3 method for class 'onion'
sinh(x)
## S3 method for class 'onion'
cosh(x)
## S3 method for class 'onion'
tanh(x)
## S3 method for class 'onion'
asinh(x)
## S3 method for class 'onion'
acosh(x)
## S3 method for class 'onion'
atanh(x)
## S3 method for class 'onion'
sqrt(x)
```

Arguments

x	An onionic vector
base	In `log()`, the base of the logarithm

Details

Trig and exponential functions, and a square root. **Warning:** these functions do not obey all the identities that one might expect; quaternions are not commutative, and octonions are not associative. The examples section illustrates this.

Author(s)

Robin K. S. Hankin

Examples

```
x <- roct(3)/10
sin(x)^2 + cos(x)^2  #should be close to O1

a <- rquat(5)
b <- roct(5)

log(a*b) -log(a) -log(b)  #zero for real or complex a & b, but not quaternions
log(b*a) -log(a) -log(b)  #different (and still nonzero)
```

`Extract.onion` *Extract or Replace Parts of an onion*

Description

Extract or replace subsets of onions.

Arguments

`x`	An onionic vector
`index`	elements to extract or replace
`value`	replacement value

Details

These methods work as expected: an octonionic vector is a eight-row matrix and element selection/replacement operate, on a per-column basis, on objects coerced to class "matrix".

Value

Always returns an onion of the same class as `x`.

Examples

```
a <- octonion(Re=1,j=1:10)
a[1:3]  <- Oil
a[3:5]

a[2:4] <- a[1]
a[2:3] <- c(10,11)
```

`length.onion` *Length of an octonionic vector*

Description

Get or set the length of octonionic vectors.

Usage

```
## S3 method for class 'onion'
length(x)
## S3 replacement method for class 'onion'
length(x) <-  value
```

Arguments

`x`	An onionic vector
`value`	An integer

Details

Operates on the columns of the matrix as expected.

In `length(x) <- value`, if `value>length(x)`, the onion is padded with `NA`s.

Author(s)

Robin K. S. Hankin

Examples

```
a <- roct(5)
length(a)
length(a) <- 3
a
length(a) <- 10
a
```

names.onion *Names of an onionic vector*

Description

Functions to get or set the names of an onionic vector.

Usage

```
## S3 method for class 'onion'
names(x)
## S3 replacement method for class 'onion'
names(x) <- value
```

Arguments

x	Onionic vector
value	a character vector of the same length as 'x', or NULL

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]
```

Norm *Norm and modulus of an onionic vector*

Description

Norm of an onionic vector: the modulus and its square

Usage

```
## S3 method for class 'onion'
Norm(z)
## S3 method for class 'onion'
Mod(z)
## S3 replacement method for class 'onion'
Mod(z) <- value
## S3 method for class 'onion'
sign(x)
```

Arguments

x,z	Onionic vector
value	A real vector

Note

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

The *Mod* of onionic vector O is the square root of its norm.

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

Author(s)

Robin K. S. Hankin

Examples

```
a <- roct(5)
Norm(a)
Mod(a)
```

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.

Examples

```
Oall
seq(from=O1,to=Oil,len=6)

stopifnot(Hj*Hk ==  Hi)
stopifnot(Okl*Oil == -Oj )  # See tests/aaa.R for the full set
```

`onion` *Basic onion functions*

Description

Construct, coerce to, test for, and print onions

Usage

```
octonion(length.out = NULL, names = NULL, Re = 0, i = 0, j = 0,
    k = 0, l = 0, il = 0, jl = 0, kl = 0)
as.octonion(x, single = FALSE, names=NULL)
is.octonion(x)
quaternion(length.out = NULL, names = NULL, Re = 0, i = 0, j = 0, k = 0)
as.quaternion(x, single = FALSE, names=NULL)
```

```
is.quaternion(x)
is.onion(x)
as.onion(x,type,names=NULL,single=FALSE)
type(x)
```

Arguments

length.out	In functions quaternion() and octonion(), the length of the onionic vector returned.
names	In functions quaternion() and octonion(), the names of the octonionic vector returned.
Re	The real part of the onionic vector returned.
i	Component i of the onionic vector returned.
j	Component j of the onionic vector returned.
k	In function octonion(), component k of the octonionic vector returned.
l	In function octonion(), component l of the octonionic vector returned.
il	In function octonion(), component il of the octonionic vector returned.
jl	In function octonion(), component jl of the octonionic vector returned.
kl	In function octonion(), component kl of the octonionic vector returned.
x	Onion to be tested or printed
single	In functions as.octonion() and as.quaternion(), a Boolean variable with default FALSE meaning to interpret x as a vector of reals to be coerced into an onionic vector with zero imaginary part; and TRUE meaning to interpret x as a length 4 (or length 8) vector and return the corresponding single onion.
type	In function as.onion() 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.

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 terminology for such objects (I have seen n-ion, anion and others. But "onion" is pronouncable and a bona fide English word).

Creating further onions is intended to be straightforward; the following steps show how to add the sedenions but any number of onions may be added the same way.

- Add functions `sedenion_prod_single()` and `sedenion_prod()` to `src/onion.c`.
- Update the following functions:
  - `type()`
  - `harmonize()`
  - `as.onion()`
  - `AprodA()`

  This should be reasonably straightforward.
- create the following functions:
  - `rsed()` (by analogy with `roct()` and `rquat()`)
  - `SprodS()` (by analogy with `OprodO()` and `HprodH()`)
  - `R_SprodS()` (by analogy with `R_OprodO()` and `R_HprodH()`)
  - `sedenion()` and `is.sedenion()` (by analogy with `octonion()` and `is.octonion()`)
  - `as.sedenion()` by analogy with `as.octonion()`.
  - a whole bunch of functions (with appropriate names) to get and set individual onion components, by analogy with `i.quaternion()` and `i<-.quaternion()`. Sedenions have at least two different naming conventions.

Note that function `Ops.onion()` need not be changed, as it copes with generic onions.

Author(s)

Robin K. S. Hankin

Examples

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

as.quaternion(diag(4))
```

`Ops.onion` *Arithmetic Ops Group Methods for Octonions*

Description

Allows arithmetic operators to be used for octonion calculations, such as addition, multiplication, division, integer powers, etc.

Usage

```
## S3 method for class 'onion'
Ops(e1, e2)
OprodO(oct1, oct2)
HprodH(quat1, quat2)
R_OprodO(oct1, oct2)
R_HprodH(quat1, quat2)
AprodA(A, B, ur=getOption("use.R"))
AsumA(A, B)
Apower(A, B)
AequalsA(A, B)
AprodS(A, scalar)
Ainv(A)
Aneg(A)
Amassage(A, B)
harmonize(A, B)
```

Arguments

e1,e2, A, B	Objects of class "onion"
oct1,oct2	Octonionic vectors
quat1,quat2	Quaternionic vectors
scalar	Scalar vector
ur	In function AprodA(), Boolean with default FALSE meaning to use the c implementation; and TRUE meaning to use the interpreted R function. See details section

Details

The function Ops.onion() passes unary and binary arithmetic operators ("+", "-", "*", and "/") to the appropriate specialist function.

The most interesting operator is "*", which is passed to AprodA(). This function is sensitive to the value of option use.R. If this is TRUE, then arguments are passed, via Amassage(), to either R_HprodH() (for quaternions), or R_OprodO() (for octonions). If option use.R is anything other than TRUE (including being unset, which is the default), the massaged arguments are passed to HprodH() or OprodO(). This is what the user usually wants: it is much faster than using the R_ functions.

The relative performance of, say, OprodO() vs R_OprodO(), will be system dependent but on my little Linux system (Fedora; 256MB) OprodO() runs more than three hundred times faster than R_OprodO(). Your mileage may vary; see examples section for using options() to set argument ur.

Value

An object of the appropriate (ie biggest) class as went in, as per harmonize().

The only non obvious ones are Amassage(), which is used by the other functions to massage the two arguments into being the same length, thus emulating recycling.

The other one is `harmonize()` that coerces scalars into quaternions and quaternions into octonions if necessary, returning a list of two octonions or two quaternions of the same length, for passing to functions like `AprodA()`.

None of these functions are really intended for the end user: use the ops as shown in the examples section.

Note

The "`A`" at the beginning of a function name means **A**ny onion. Thus `Ainv()` takes quaternionic or octonionic arguments, but `OprodO()` takes only octonions.

Examples

```
x <- octonion(Re=1  , il=1:3, k=3:1)
y <- octonion(Re=1:3,  i=1  ,il=3:1)
z <- octonion(Re=3:1,  j=1  ,jl=1:3)
x*9
x+y
x*y
x/y
```

`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

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

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.onion` *Plot onions*

Description

Plotting method for onionic vectors

Usage

```
## S3 method for class 'onion'
plot(x, ...)
```

Arguments

`x`	Onionic vector
`...`	Further arguments passed to `plot.default()`

Details

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

Author(s)

Robin K. S. Hankin

Examples

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

print.onion *Print method for onions*

Description

Prints an onion

Usage

```
## S3 method for class 'quaternion'
print(x, h=getOption("horiz"), ...)
## S3 method for class 'octonion'
print(x, h=getOption("horiz"), ...)
```

Arguments

x	Onionic vector
h	Boolean, with default FALSE meaning to print horizontally and TRUE meaning to print by columns.
...	Further arguments (currently ignored)

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.

Author(s)

Robin K. S. Hankin

Examples

```
roct(4)
```

prods *Various products of two onionic vectors*

Description

Returns the various inner and outer products of two onionic vectors.

Usage

```
x %<*>% y
x %>*<% y
x %<.>% y
x %>.<% y
## S3 method for class 'onion'
g.even(x,y)
## S3 method for class 'onion'
g.odd(x,y)
## S3 method for class 'onion'
e.even(x,y)
## S3 method for class 'onion'
e.odd(x,y)
```

Arguments

x	Onionic vector
y	Onionic vector

Details

This page documents an attempt at a consistent notation for onionic products. The product used by `Ops.octonion()` and `Ops.quaternion()` (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.

There is no binary operator for the ordinary Euclidean product (it is not defined because there is no natural, consistent notation available; and it seems to be rarely needed in practice). Use `Conj(x)*y`.

Author(s)

Robin K. S. Hankin

Examples

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

Re *Octonion components*

Description

Get or set each component of an onionic vector

Usage

```
## S3 method for class 'octonion'
Re(z)
## S3 method for class 'octonion'
Im(z)
## S3 method for class 'octonion'
i(x)
## S3 method for class 'octonion'
j(x)
## S3 method for class 'octonion'
k(x)
## S3 method for class 'octonion'
l(x)
## S3 method for class 'octonion'
il(x)
## S3 method for class 'octonion'
jl(x)
## S3 method for class 'octonion'
kl(x)
## S3 replacement method for class 'octonion'
Re(x) <- value
## S3 replacement method for class 'octonion'
Im(x) <- value
## S3 replacement method for class 'octonion'
i(x) <- value
## S3 replacement method for class 'octonion'
j(x) <- value
## S3 replacement method for class 'octonion'
k(x) <- value
## S3 replacement method for class 'octonion'
l(x) <- value
## S3 replacement method for class 'octonion'
il(x) <- value
## S3 replacement method for class 'octonion'
jl(x) <- value
```

```
## S3 replacement method for class 'octonion'
kl(x) <- value
## S3 method for class 'quaternion'
Re(z)
## S3 method for class 'quaternion'
Im(z)
## S3 method for class 'quaternion'
i(x)
## S3 method for class 'quaternion'
j(x)
## S3 method for class 'quaternion'
k(x)
## S3 replacement method for class 'quaternion'
Re(x) <- value
## S3 replacement method for class 'quaternion'
Im(x) <- value
## S3 replacement method for class 'quaternion'
i(x) <- value
## S3 replacement method for class 'quaternion'
j(x) <- value
## S3 replacement method for class 'quaternion'
k(x) <- value
## S3 method for class 'onion'
get.comp(x,i)
## S3 replacement method for class 'onion'
set.comp(x,i) <- value
```

Arguments

`x,z`	An onionic vector
`value`	A real vector (or, in the case of `Im<-()` and `set.comp<-()`, an appropriately sized matrix)
`i`	In functions `get.comp()` and `set.comp<-()`, an integer between 1 and 2^n where n depends on the type of onion

Value

All return an onion of the appropriate class.

Note

In the case of `Im<-` methods, if `value` has the special value `0`, then all the imaginary parts will be set to zero, as though one had typed `Im(a) <- Im(a)*0`. Note that setting value to `rep(0,length(x))` will *not* work; neither will `Im(x) <- 3` (say).

These functions are all specific to their algebra; there is no onionic generalization. This is because the code is more structured. It also makes it easier to change the names of the bases.

Functions `get.comp()` and the various methods for `set.comp<-()` are not really intended for the end-user. It is better to use idioms such as `il(x)` and `i(x) <- 4`

Author(s)

Robin K. S. Hankin

See Also

octonion

Examples

```
x <- octonion(Re=1,il=1:3,j=3:1)
Re(x)
kl(x) <- 1000
```

`rep.onion` *Replicate elements of onionic vectors*

Description

Replicate elements of onionic vectors

Usage

```
## S3 method for class '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

Returns a random onionic vector of arbitrary length

Usage

```
roct(n, x=1:8, replace=TRUE, rand="sample", ...)
rquat(n, x=1:4, replace=TRUE, rand="sample", ...)
```

Arguments

n	Length of onionic vector returned
x	Argument x as passed to sample(); Only matters if rand takes its default value of "sample".
replace	Argument replace as passed to sample(); Only matters if rand takes its default value of "sample".
rand	String, with name being that of the distribution intended. Currently implemented values are "sample" (default), "norm", "unif", "binom" and "pois". Add an "r" to get the name of the function used; thus "unif" means to call runif().
...	Further arguments passed to the random number generator (such as mean and sd, which would be passed to rnorm())

Details

Function rquat() returns a quaternionic vector and function roct() returns an octonionic vector.

Note

Arguments x and replace are there (and have the default values they do) in order to make roct(n) return a "get you going" random onion that prints relatively compactly.

Author(s)

Robin K. S. Hankin

Examples

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

rotate *Rotates 3D vectors using quaternions*

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

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)
```

seq.onion *seq method for onions*

Description

Rough equivalent of seq() for onions.

Usage

```
## S3 method for class 'onion'
seq(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=O1,to=Oil,length.out=6)
seq(from=H1,to=(Hi+Hj)/2,len=10,slerp=TRUE)
```

---

str.onion *Compactly display an onion*

Description

A very basic implementation of str for onions

Usage

```
## S3 method for class 'onion'
str(object, vec.len=4, ...)
```

Arguments

object	Onionic vector
vec.len	Number of elements to display
...	Futher arguments (currently ignored)

Details

Displays the first vec.len elements of an onionic vector using condense.

Value

Returns NULL.

Author(s)

Robin K. S. Hankin

See Also

condense

Examples

```
str(roct(100))
```

`sum.onion` *Sum of onionic vectors*

Description

Returns the onionic sum of all values present in its arguments.

Usage

```
## S3 method for class 'onion'
sum(..., na.rm = FALSE)
```

Arguments

`...`	Onionic vectors
`na.rm`	Boolean with default `FALSE` meaning to sum with `NA` values in place and `TRUE` meaning to ignore them.

Author(s)

Robin K. S. Hankin

Examples

```
sum(roct(5))
sum(Oil,Oj,O1)
sum(roct(5),100*Ok)
```

t.onion *Onion transpose*

Description

Takes the transpose of an onionic vector

Usage

```
## S3 method for class 'onion'
t(x)
```

Arguments

x	Onionic vector

Details

Returns the transpose of the eight-row or four-row matrix

Author(s)

Robin K. S. Hankin

Examples

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

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 `Re(x1 * (Conj(x2) * x3) - x3 * (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)
```

Index