Package: evitaicossa (via r-universe)

September 10, 2024

Type Package

Title Antiassociative Algebra

Version 0.0-2

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

Description Methods to deal with the free antiassociative algebra over the reals with an arbitrary number of indeterminates. Antiassociativity means that (xy)z = -x(yz). Antiassociative algebras are nilpotent with nilindex four (Remm, 2022, <doi:10.48550/arXiv.2202.10812>) and this drives the design and philosophy of the package. Methods are defined to create and manipulate arbitrary elements of the antiassociative algebra, and to extract and replace coefficients. A vignette is provided.

License GPL (>= 2)

Depends R (>= 3.5.0)

Suggests knitr, markdown, rmarkdown, testthat, mvtnorm, covr

VignetteBuilder knitr

Imports Rcpp (>= 1.0-7), disordR (>= 0.9-8-2), methods, Rdpack

LinkingTo Rcpp

URL https://github.com/RobinHankin/evitaicossa, https://robinhankin.github.io/evitaicossa/

BugReports https://github.com/RobinHankin/evitaicossa/issues

RdMacros Rdpack

Config/testthat/edition 3

Repository https://robinhankin.r-universe.dev

RemoteUrl https://github.com/robinhankin/evitaicossa

RemoteRef HEAD

RemoteSha 0e63063039f11a4e6561a5e6da725680bf84cf74

Contents

`evitaicossa-package` *Antiassociative Algebra*

Description

Methods to deal with the free antiassociative algebra over the reals with an arbitrary number of indeterminates. Antiassociativity means that (xy)z = -x(yz). Antiassociative algebras are nilpotent with nilindex four (Remm, 2022, <doi:10.48550/arXiv.2202.10812>) and this drives the design and philosophy of the package. Methods are defined to create and manipulate arbitrary elements of the antiassociative algebra, and to extract and replace coefficients. A vignette is provided.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Functionality to work with the free antiassociative algebra in R. The hex sticker features an image taken from Hoffnung (1959) in which musical concepts [pizzicato, crescendo, etc] are given whimsical visual form. The character on the hex sticker is captioned “A Discord”: Hoffnung’s interpretation of the musical concept of dissonance. In the book, the preceding image was a “*chord*”, evoking harmony. The discord, on the other hand, embodies–for me at least–antiassociativity: everything is wrong, wrong, wrong.

Author(s)

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

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

References

Hoffnung G (1959). *Hoffnung’s Acoustics*. Dobson.

See Also

aaa

Examples

```
x <- raaa()
x
y <- raaa()

x+y
x*y
```

---

aaa *Function to create objects of class* aaa

Description

Objects of class aaa

Usage

```
aaa(s1 = character(0), sc = numeric(0), d1 = character(0), d2 =
character(0), dc = numeric(0), t1 = character(0), t2 = character(0), t3
= character(0), tc = numeric(0))
lavter(cout)
as.aaa(s)
thing_to_aaa(L)
```

Arguments

s1, d1, d2, t1, t2, t3	single, double, triple symbols
sc, dc, tc	single, double, triple coefficients
L	A list with elements s1 etc
cout	list
s	Object that function as.aaa() will coerce to an aaa object

Details

Function lavter() is the formal creation method for aaa objects; it is the only place that new() is called. It takes a single argument cout, which is a list as returned by C function retval(). But it is a little awkward to use and the user should use other functions for creation, which are more user-friendly and have sensible defaults:

- Function aaa() takes named arguments s1 etc, with defaults corresponding to "not present"
- Function thing_to_aaa() takes a list with names s1 etc
- Function as.aaa() tries hard to coerce its argument to an aaa object

Value

Return objects of class aaa

Author(s)

Robin K. S. Hankin

Examples

```
aaa(s1 = "x", sc = 13)
aaa(d1 = "z", d2 = "w", dc = 14)
aaa(t1 = "x", t2 = "y", t3 = "z", tc = 15)

aaa(
    s1 = c("a","d"),
    sc = c( 4 , 2 ),
    d1 = c("a", "a", "a", "b"),
    d2 = c("a", "b", "d", "a"),
    dc = c( 3 ,  4 ,  4 ,  3 ),
    t1 = c("a", "a", "a", "b", "b"),
    t2 = c("c", "d", "d", "c", "c"),
    t3 = c("a", "c", "d", "a", "b"),
    tc = c(-4 , -1 , -4 , 11 , 20 )
)


aaa() # the zero object

aaa(s1=letters,sc=seq_along(letters))
aaa(d1=state.abb,d2=rev(state.abb),dc=seq_along(state.abb))

as.aaa(state.abb)


evita <- aaa(s1=letters[1:5],sc=1:5)
icossa <- aaa(d1=c("fish","chips"),d2=c("x","y"),dc=c(6,7))

evita
evita + icossa
evita * icossa
evita^2

f <- function(o){aaa(state.abb[o],seq_along(o))}
f(8:9) - (f(1:2) - f(6:8)^2)^2
```

---

aaa-class *Class* "aaa"

Description

Class `aaa` is for elements of the free antiassociative algebra

Objects from the Class

Objects can be created by calls of the form `new("aaa", ...)`.

Slots

`single_indeterminate_name1`: Object of class `"character"`

`single_indeterminate_coeff`: Object of class `"numeric"`

`double_indeterminate_name1`: Object of class `"character"`

`double_indeterminate_name2`: Object of class `"character"`

`double_indeterminate_coeff`: Object of class `"numeric"`

`triple_indeterminate_name1`: Object of class `"character"`

`triple_indeterminate_name2`: Object of class `"character"`

`triple_indeterminate_name3`: Object of class `"character"`

`triple_indeterminate_coeff`: Object of class `"numeric"`

Author(s)

Robin K. S. Hankin

Examples

```
showClass("aaa")
```

`allsymbols` *All symbols in an aaa object*

Description

Function `allsymbols()` returns a character vector whose entries include all symbols of its argument.

Usage

```
allsymbols(a)
```

Arguments

`a`	Object of class `aaa`

Value

Returns a character vector

Author(s)

Robin K. S. Hankin

Examples

```
a <- raaaa()
a
allsymbols(a)

a[cbind(allsymbols(a))] == single(a)
```

`Arith-methods` *Arithmetic methods for `aaa` objects*

Description

Arithmetic methods for objects of class `aaa`.

Methods

`signature(e1 = "aaa", e2 = "aaa")` Dispatches to `aaa_arith_aaa()`

`signature(e1 = "aaa", e2 = "numeric")` Dispatches to `aaa_arith_numeric()`

`signature(e1 = "numeric", e2 = "aaa")` Dispatches to `numeric_arith_aaa()`

The S4 methods call lower-level functions `aaa_plus_aaa()`, `aaa_prod_aaa()`, `aaa_prod_numeric()`, `aaa_negative()`, and `aaa_plus_numeric()`.

These functions call the Rcpp functions `aaa_identity()`, `c_aaa_add()`, and `c_aaa_prod()`.

`Compare-methods` *Comparison methods for antiassociative algebra*

Description

Comparison methods generally do not make sense for elements of an antiassociative algebra. The only exception is equality: `x == y` returns `TRUE` if `aaa` objects `x` and `y` are identical.

The test for equality follows the **frab** package: go through the keys of `x`, compare the corresponding values of `y`, and return `FALSE` when any difference is detected. This is faster than `is.zero(x-y)`.

Technically, `x==0` makes sense but I thought consistency was more important: in the package, numeric values cannot be compared with `aaa` objects.

Functions `aaa_compare_aaa()` etc. are used in S4 dispatch; `c_aaa_equal()` is a low-level helper function that uses Rcpp to call the appropriate C routine.

Methods

```
signature(e1 = "aaa", e2 = "aaa")
signature(e1 = "aaa", e2 = "ANY")
signature(e1 = "aaa", e2 = "numeric")
signature(e1 = "ANY", e2 = "aaa")
signature(e1 = "numeric", e2 = "aaa")
```

`Extract` *Extract or Replace Parts of* `aaa` *objects*

Description

Extraction methods for `aaa` objects. The names of the two-letter functions and arguments follow a pattern: the initial letter (s, d, t) stands for “single”, “double”, or “triple”; the second symbol is c for “coefficients”, or a number (1, 2, 3) denoting first, second, or third. Thus “`dc()`” gets the coefficients of the double-symbol components, and “`t2()`” gets the second symbol of the triple-symbol components.

Usage

```
## S4 method for signature 'aaa'
s1(a)
## S4 method for signature 'aaa'
sc(a)
## S4 method for signature 'aaa'
d1(a)
## S4 method for signature 'aaa'
d2(a)
## S4 method for signature 'aaa'
dc(a)
## S4 method for signature 'aaa'
t1(a)
## S4 method for signature 'aaa'
t2(a)
## S4 method for signature 'aaa'
t3(a)
## S4 method for signature 'aaa'
tc(a)
single(a)
double(a)
triple(a)
```

Arguments

`a`	Object of class `aaa`

Details

An aaa object is a list of 9 vectors, three numeric and six character, which are extracted by functions s1() etc.

Functions single(), double() and triple() extract the single, double, and triple components of their argument, and return the corresponding aaa object.

There is no function evitaicossa::coeffs() because the three types of elements are qualitatively different; use sc(), dc(), and tc() to get the coefficients in disord format.

Functions getthings(), extracter() and overwriter() are lower-level methods, not really intended for the end-user. Function getthings() takes an aaa object and returns a named list with elements being disord objects corresponding to components s1,sc,d1 etc. Function extracter() takes an aaa object and arguments s1, d1,d2,t1 etc. and returns the aaa object corresponding to the specified index elements. Function overwriter takes

Functions single(), double(), and triple() return the index-1, index-2, and index-3 components of their arguments respectively. Functions single<-() *et seq.* are the corresponding setting methods which overwrite the index-1 (resp. 2,3) components with the right hand side. The right hand side must be purely the correct component otherwise an error is returned; thus in double(a) <- x, for example, the single-symbol and triple-symbol components of x must be zero.

Square bracket extraction and replacement methods are more user-friendly. These operate in two distinct modes. If given named arguments (s1, d1,d2, *et seq.*) then these are interpreted as symbols and coefficients of the different orders. If given an unnamed argument, this is interpreted as a character vector of length one, two, or three specifying a particular term in the object. See examples.

Value

Return disord or aaa objects

Author(s)

Robin K. S. Hankin

Examples

```
x <- linear1(1:3) + (linear1(1:2) + linear2(1:3))^2
x
x[d1=c("a","a"),d2=c("a","b")]
x[s1="a", t1="b", t2="c", t3="c"]


x[s1="a", t1="b", t2="c", t3="c"] <- 88
x
x[c("c","c","b")] <- -777
x


a <- raaaa()
sc(a)
t2(a)
single(a)
single(a) + double(a) + triple(a) == a  # should be TRUE

aaa(d1=d1(a),d2=d2(a), dc=dc(a)) == double(a)

x <- raaaa()
single(x) <- 0
double(x) <- double(raaa())
```

linear *Linear functions*

Description

Linear functions returning single, double, or triple-symbol aaa objects.

Usage

```
linear1(x)
linear2(x)
linear3(x)
```

Arguments

x	A numeric vector

Details

These functions return an antiassociative algebra element with the specified coefficients. Given a numeric vector v with elements $v_1, v_2, \ldots, v_n$ then

linear1(v) returns $v_1\mathbf{a} + v_2\mathbf{b} + \cdots + v_n\mathbf{L_n}$, where $\mathbf{L_n}$ is the n^{th} letter of the alphabet. Similarly, linear2(v) returns $v_1\mathbf{aa} + \cdots + v_n\mathbf{L_nL_n}$, and linear3(v) returns $v_1(\mathbf{aa})\mathbf{a} + \cdots + v_n(\mathbf{L_nL_n})\mathbf{L_n}$. They are linear in the sense that

$$f(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha f(\mathbf{x}) + \beta f(\mathbf{y})$$

where $\alpha, \beta \in \mathbb{R}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Value

These functions return an object of class aaa.

Author(s)

Robin K. S. Hankin

Examples

```
linear1(sample(8))
linear2(sample(8))
linear3(sample(8))

a <- 3
b <- 7
x <- sample(9)
y <- sample(9)

linear1(a*x + b*y) == a*linear1(x) + b*linear1(y)
linear2(a*x + b*y) == a*linear2(x) + b*linear2(y)
linear3(a*x + b*y) == a*linear3(x) + b*linear3(y)
```

---

raaa *Random elements of the free antiassociative algebra*

Description

Random elements of the free antiassociative algebra, intended as quick “get you going” examples of aaa objects

Usage

```
raaa(n = 4, s = 3)
raaaa(n = 10, s = 30)
```

Arguments

n	Number of terms to generate
s	Number of symbols to use in the alphabet

Details

Function raaa() returns a random aaa object. Function raaaa() returns, by default, a more complicated aaa object.

Value

Returns an object of class aaa

Author(s)

Robin K. S. Hankin

Examples

```
raaa()
raaaa()
```

`show` *Print method for antiassociative algebra objects*

Description

Show methods for `aaa` objects

Usage

```
## S4 method for signature 'aaa'
show(object)
aaa_show(a)
```

Arguments

`a`, `object`	Object of class `aaa`

Details

A bunch of functionality to print `aaa` objects.

Function `putsign()` is a low-level helper function that puts the sign (that is, `+` or `-`) before each element of a numeric vector. Functions `single_string()`, `double_string()`, and `triple_string()` process the 1,2, and 3- symbols for printing.

Value

No return value, called for side-effects

Author(s)

Robin K. S. Hankin

Examples

```
aaa_show(raaa())
aaa_show(aaa())
```

`zero` *The additive zero in antiassociative algebras*

Description

Function `is.zero()` tests for its argument being the additive zero.

Package idiom to create the zero element of the antiassociative algebra is `aaa()`.

Usage

```
is.zero(x)
```

Arguments

`x`	Object of class `aaa`

Value

Returns a Boolean.

Note

In any antiassociative algebra, the only scalar is zero.

Author(s)

Robin K. S. Hankin

Examples

```
is.zero(raaa())
is.zero(raaa()*0)
is.zero(aaa())
```

Index