Package 'disordR'

August 5, 2021

Type Package

Title Non-Ordered Vectors

Version 0.0-7

Depends methods,digest

Suggests mvp,knitr,rmarkdown,testthat

VignetteBuilder knitr

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

Description Functionality for manipulating values of associative maps. Ordinary R vectors are unsuitable for working with values of associative maps because elements of an R vector may be accessed by reference to their location in the vector, but associative maps are stored in arbitrary order. However, when associating keys with values one needs both parts to be in 1-1 correspondence, so one cannot dispense with the order entirely. The 'disordR' package includes a single S4 class, disord. This class allows one to perform only those operations appropriate for manipulating values of associative maps and prevents any other operation (such as accessing an element at a particular location). A useful heuristic is that one is only allowed to access or modify a disord object using a python list comprehension. The idea is to prevent ill-defined operations on values (or keys) of associative maps, whose order is undefined or at best implementation-specific, while allowing and facilitating sensible operations. The package is needed for development versions of 'mvp', 'hyper2', 'spray', 'clifford', and 'freealg'.

License GPL (>= 2)

URL https://github.com/RobinHankin/disordR

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

NeedsCompilation no

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

Repository CRAN

Date/Publication 2021-08-05 07:50:11 UTC

R topics documented:

Arith 2
c 3
disord 3
disord-class 4
drop 5
elements 6
extract 7
misc 8
rdis 9

Index **11**

`Arith` *Arithmetic and logical operations*

Description

Arithmetic operations including low-level helper functions

Usage

```
disord_inverse(a)
disord_mod_disord(a,b)
disord_mod_numeric(a,b)
disord_negative(a)
disord_plus_disord(a,b)
disord_plus_numeric(a,b)
disord_power_disord(a,b)
disord_power_numeric(a,b)
numeric_power_disord(a,b)
disord_prod_disord(a,b)
disord_prod_numeric(a,b)
disord_logical_negate(x)
```

Arguments

`a,b,x`	at least one is a disord object

Details

Basic low-level arithmetic operations, intended to be called from S4 dispatch.

These functions return a `disord` object or a regular vector as appropriate. Consistency is required. The hash is set to be that of the disord object if appropriate.

Value

Return a disord object or logical

Author(s)

Robin K. S. Hankin

Examples

```
a <- disord(sample(20))
a + 2*a
a[a%%2==1] <- a[a%%2==1] + 5
```

---

c *Concatenation*

Description

Concatenation simply does not make sense for `disord` objects.

Value

Returns an error.

Note

I could not figure out how to stop idiom like "`c(1,rdis())`" from returning a result. Just don't use it, OK?

Author(s)

Robin K. S. Hankin

---

disord *Functionality for* `disord` *objects*

Description

Allows arithmetic operators to be used for disord objects; the canonical application is coefficients of multivariate polynomials (as in the **mvp** package). The issue is that the storage order of disord objects is implementation-specific but the order (whatever it is) must be consistent between the list of keys and values in an associative array.

Usage

```
is.disord(x)
hash(x)
hashcal(x)
disord(v,h)
consistent(x,y)
x %~% y
```

Arguments

x,y	Objects of class disord
v	Vector of coefficients
h	Hash code

Details

The package provides a single S4 class, disord. A detailed vignette is provided that motivates the package.

Function hash() is the extractor function:

```
`hash` <-function(x){x@hash}
```

Compare hashcal() which is used to actually calculate the hash code for an object. Currently

```
`hashcal` <-function(x){digest::sha1(x)}
```

Value

Boolean, hash code, or object of class disord as appropriate.

Author(s)

Robin K. S. Hankin

Examples

```
a <- rdis()
b <- rdis()

a + 2*a + 2^a  # fine
# a + b # error

a[a<0.5] <- 0       # round small entries down
```

---

disord-class *Class* "disord"

Description

The disord class provides basic arithmetic and extract/replace methods for disord objects.

Objects from the Class

Objects can be created by calls of the form new("disord",...), although functions disord() and (eventually) as.disord() are more user-friendly.

Slots

`.Data`: Object of class `vector` that specifes the elements

`hash`: Object of class `character` that specifies the hash code

Author(s)

Robin K. S. Hankin

Examples

```
showClass("disord")
```

---

drop *Drop redundant information*

Description

Coerce `disord` objects to vector when this makes sense

Usage

```
drop(x)
allsame(x)
```

Arguments

`x`	`disord` object

Details

If one has a disord object all of whose elements are identical, one very frequently wants to drop the disord attribute and coerce to a vector. This can be done without ambiguity. Function `drop()` takes a disord object, and if all elements are identical returns the elements in the form of a vector. Some extraction methods take a `drop` argument, which does the same thing if `TRUE`.

Value

Function `drop()` returns either a vector or object of class `disord` as appropriate; `allsame()` returns a Boolean.

Author(s)

Robin K. S. Hankin

Examples

```
drop(disord(c(3,3,3,3,3,3,2,3,3,3,3,3)))
drop(disord(c(3,3,3,3,3,3,3,3,3,3,3,3)))

## In extraction, argument drop discards disorderliness when possible:
a <- rdis()
a[] <- 6             # a becomes a vector
b <- rdis()
b[drop=FALSE] <- 6   # b stays a disord
```

`elements` *Extract elements of a disord object*

Description

A disord object contains elements in an arbitrary order. Function `elements()` shows these elements in the form of an ordinary vector.

Usage

```
elements(x)
```

Arguments

`x`	Object of class `disord`

Value

Returns a vector

Note

This function imposes an arbitrary order on the elements. The value returned is thus implementation-specific.

Author(s)

Robin K. S. Hankin

Examples

```
elements(rdis())
```

`extract` *Extraction and replacement methods for class* `"disord"`

Description

The `disord` class provides basic arithmetic and extract/replace methods for disord objects.

Class *index* is taken from the excellent **Matrix** package and is a `setClassUnion()` of classes `numeric`, `logical`, and `character`.

Methods

[`signature(x = "disord", i = "ANY", j = "ANY")`: ...

[`signature(x = "disord", i = "index", j = "index")`: ...

[`signature(x = "disord", i = "index", j = "missing")`: ...

[`signature(x = "disord", i = "missing", j = "index")`: ...

[`signature(x = "disord", i = "missing", j = "missing")`: ...

[`signature(x = "disord", i = "matrix", j = "missing")`: ...

[<- `signature(x = "disord", i = "index", j = "index")`: ...

[<- `signature(x = "disord", i = "index", j = "missing")`: ...

[<- `signature(x = "disord", i = "missing", j = "index")`: ...

[<- `signature(x = "disord", i = "matrix", j = "missing")`: ...

[<- `signature(x = "disord", i = "missing", j = "missing")`: ...

Arith `signature(e1 = "ANY", e2 = "disord")`: ...

Arith `signature(e1 = "disord", e2 = "ANY")`: ...

Arith `signature(e1 = "disord", e2 = "disord")`: ...

Arith `signature(e1 = "disord", e2 = "missing")`: ...

The extraction method takes a `drop` argument which if `TRUE`, returns the `drop()` of its value.

Author(s)

Robin K. S. Hankin

See Also

`drop`

misc *Miscellaneous functions*

Description

This page documents various functions that work for disords, and I will add to these from time to time as I add new functions that make sense for disord objects. Functions like `sin()` and `abs()` work as expected: they take and return `disord` objects with the same hash as `x` (which means that idiom like `x + sin(x)` is accepted). However, there are a few functions that are a little more involved:

- `rev()` reverses its argument and returns a `disord` object with a reversed hash, which ensures that `rev(rev(x))==x`.
- `sort()` returns a vector of sorted elements.
- length() returns the length of the data component of the object.
- `sapply(X,f)` returns a disord object which is the result of applying `f()` to each element of `X`.
- `match(x,table)` should behave as expected but note that if `table` is a `disord`, the result is not defined (because it is not known where the elements of `x` occur in `table`). Nevertheless `x %in% table` is defined and returns a `disord` object.

Arguments

`x`	Object of class `disord`

Value

Returns a disord

Author(s)

Robin K. S. Hankin

Examples

```
a <- disord(c(a=1,b=2,c=7))
names(a)
length(a)
sqrt(a)
is.na(a) <- c(FALSE,FALSE,TRUE)

# powers() and vars() in the mvp package return lists; see the vignette
# for more discussion.

l <- disord(list(3,6:9,1:10))
sapply(l,length)
```

rdis *Random disord objects*

Description

Returns a random disord object

Usage

```
rdis(n=9)
```

Arguments

n	Number of elements

Details

A simple `disord` object, intended as a quick "get you going" example

Value

A `disord` object.

Author(s)

Robin K. S. Hankin

Examples

```
rdis()

a <- rdis()
b <- rdis()


## Following are OK:
a
a + 2*a - a^2
a>8
a[a<8]
a[a<6] <-  a[a<6] + 100


## Following forbidden:

# a+b
# a[1:4]
# a[b<4]
# a[a<4] <- 1:4
```

```
# a[a<4] <- a[b<4] + 100
```

Index

!,disord-method *(misc)*, 8
∗ **classes**
    disord-class, 4
∗ **symbolmath**
    disord, 3
[*(extract)*, 7
[,ANY,disord,ANY-method *(extract)*, 7
[,disord,ANY,ANY-method *(extract)*, 7
[,disord,disord,missing,ANY-method *(extract)*, 7
[,disord,disord,missing-method *(extract)*, 7
[,disord,index,ANY,ANY-method *(extract)*, 7
[,disord,index,ANY-method *(extract)*, 7
[,disord,index,index-method *(extract)*, 7
[,disord,index,missing,ANY-method *(extract)*, 7
[,disord,index,missing-method *(extract)*, 7
[,disord,missing,index-method *(extract)*, 7
[,disord,missing,missing-method *(extract)*, 7
[,disord-method *(extract)*, 7
[.disord *(extract)*, 7
[<- *(extract)*, 7
[<-,disord,ANY,ANY-method *(extract)*, 7
[<-,disord,disord,missing,ANY-method *(extract)*, 7
[<-,disord,disord,missing,disord-method *(extract)*, 7
[<-,disord,disord,missing-method *(extract)*, 7
[<-,disord,index,ANY,ANY-method *(extract)*, 7
[<-,disord,index,index-method *(extract)*, 7
[<-,disord,index,missing,ANY-method *(extract)*, 7
[<-,disord,index,missing,disord-method *(extract)*, 7
[<-,disord,index,missing,numeric-method *(extract)*, 7
[<-,disord,index,missing-method *(extract)*, 7
[<-,disord,missing,index-method *(extract)*, 7
[<-,disord,missing,missing,ANY-method *(extract)*, 7
[<-,disord,missing,missing,disord-method *(extract)*, 7
[<-,disord,missing,missing,numeric-method *(extract)*, 7
[<-,disord,missing,missing-method *(extract)*, 7
[<-,disord-method *(extract)*, 7
[<-.disord *(extract)*, 7
%~% *(disord)*, 3
%in% *(misc)*, 8
%in%,ANY,disord-method *(misc)*, 8
%in%,disord,ANY-method *(misc)*, 8
%in%,disord,disord-method *(misc)*, 8
%in%,disord-method *(misc)*, 8

accessors *(disord)*, 3
allsame *(drop)*, 5
an_y_logic_disord *(Arith)*, 2
any_compare_disord *(Arith)*, 2
any_logic_disord *(Arith)*, 2
Arith, 2
Arith,ANY,disord-method *(extract)*, 7
Arith,disord,ANY-method *(extract)*, 7
Arith,disord,disord-method *(extract)*, 7
Arith,disord,missing-method *(extract)*, 7
as_disord *(disord)*, 3

c, 3
c,disord-method *(c)*, 3

c.disord (c), 3
consistent (disord), 3

disord, 3
disord-class, 4
disord<- (disord), 3
disord_arith_disord (Arith), 2
disord_arith_numeric (Arith), 2
disord_arith_unary (Arith), 2
disord_compare_any (Arith), 2
disord_compare_disord (Arith), 2
disord_inverse (Arith), 2
disord_logic (Arith), 2
disord_logic_any (Arith), 2
disord_logic_disord (Arith), 2
disord_logic_missing (Arith), 2
disord_logic_unary (Arith), 2
disord_logical_negate (Arith), 2
disord_mod_disord (Arith), 2
disord_mod_numeric (Arith), 2
disord_negative (Arith), 2
disord_plus_disord (Arith), 2
disord_plus_numeric (Arith), 2
disord_positive (Arith), 2
disord_power_disord (Arith), 2
disord_power_numeric (Arith), 2
disord_prod_disord (Arith), 2
disord_prod_numeric (Arith), 2
disord_show (Arith), 2
disord_unary (Arith), 2
drop, 5, 7
drop,disord-method (drop), 5

elements, 6
extract, 7

hash (disord), 3
hashcal (disord), 3

index-class (extract), 7
is.disord (disord), 3
is.na (misc), 8
is.na,disord-method (misc), 8
is.na.disord (misc), 8
is.na<- (misc), 8
is.na<-,disord-method (misc), 8
is.na<-.disord (misc), 8

lapply (misc), 8
lapply,disord-method (misc), 8
lapply.disord (misc), 8
length (misc), 8
length,disord-method (misc), 8
length.disord (misc), 8
length<- (misc), 8
length<-,disord-method (misc), 8
length<-.disord (misc), 8
Logic (Arith), 2

match (misc), 8
match,ANY,disord-method (misc), 8
match,disord,ANY-method (misc), 8
match,disord,disord-method (misc), 8
match,disord-method (misc), 8
misc, 8

numeric_arith_disord (Arith), 2
numeric_mod_disord (Arith), 2
numeric_power_disord (Arith), 2

rdis, 9
rdisord (rdis), 9
rdisordR (rdis), 9
rev (misc), 8
rev,disord-method (misc), 8
rev.disord (misc), 8

sapply (misc), 8
sapply,disord-method (misc), 8
sapply.disord (misc), 8
sort (misc), 8
sort,disord-method (misc), 8
sort.disord (misc), 8