Package: keep (via r-universe)

September 2, 2024

Type Package

Title Arrays with Better Control over Dimension Dropping

Version 1.0

Date 2015-12-11

Author Paavo Jumppanen

Maintainer Paavo Jumppanen<paavo.jumppanen@csiro.au>

Description Provides arrays with flexible control over dimension dropping when subscripting.

Depends methods

Suggests Oarray

License GPL

NeedsCompilation no

Date/Publication 2015-12-16 00:36:24

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

RemoteUrl https://github.com/cran/keep

RemoteRef HEAD

RemoteSha 0c49cffce185df001ae33d14ccd1faa49e9f336f

Contents

keep 2

Index **5**

keep *Arrays with Better Control over Dimension Dropping*

Description

Carrying out an array a subset operation in traditional R will result in dimensions being dropped when those dimensions have a size of 1. We can circumvent this behaviour by setting `drop=FALSE` but this is an all or nothing approach and often we require a more refined control over which dimensions will be dropped. This extension provides the means to better control dropping behaviour.

Usage

```
karray(data = NA, dim = length(data), dimnames = NULL)
kOarray(data = NA, dim = length(data), dimnames = NULL, offset = rep(1, length(dim)),
        drop.negative = TRUE)
as.karray(x, ...)
as.kOarray(x, offset = rep(1, length(dim)), drop.negative = TRUE)
keep(index)
## S3 method for class 'keep'
as.array(x, ...)
```

Arguments

`data`, `dim`, `dimnames`, `drop`	As in the function `array`
`offset`	Vector of first index values for each extent (defaults to 1s); a length-one argument will be silently recycled to the appropriate length
`drop.negative`	Should negative subscripts indicate exclusion?
`x`	an array or Oarray created with `karray` or `kOarray` respectively to be subsetted
`...`	arguments to specify how to subset `x`
`index`	an index or index range in a subsetting operation

Details

`karray` delegates to `array` to create an array of given size and initialisation but adds the S3 class attribute `keep` to it. The `keep` S3 class designation is used to direct the dispatch of array subsetting to the overloaded `[` operator method, which in turn provides the added control over dimension dropping. `[` method merely acts as a man in the middle, delegating the actual subsetting to the handler for the `array` S3 class. It then re-attributes the result with the appropriate dimension attributes. `kOarray` behaves in the same manner except the base S3 class is now `Oarray` instead of array. Note that `Oarray` is a package extension that must be installed and loaded if you wish to make use of `kOarray`.

`as.karray` casts an `array` into a `karray` in a similar manner to `as.array`. Likewise `as.kOarray` casts an `Oarray` into a `kOarray`.

Traditional R will drop dimensions if the size of that dimension is 1. This can cause major headaches if you wish to write programs in R that use array subsetting with variable subsetting constraints. Consider for example,

```
M <- array(1:12, c(1,3,4))
for (i in 1:4) print(dim(M[,,i:4]))
```

which produces the output,

```
[1] 3 4
[1] 3 3
[1] 3 2
NULL
```

The first thing to notice is that the first dimension has been dropped even though we wouldn't necessarily expect it to as we provided an empty argument in the subsetting expression. As it has a size of 1 it has been dropped. The second point to notice is that for the case of `i = 4` another dimension was dropped as `4:4` results in a dimension of size 1. Now look at the same case but with karray instead.

```
M <- karray(1:12, c(1,3,4))
for (i in 1:4) print(dim(M[,,i:4]))
```

which produces the output,

```
[1] 1 3 4
[1] 1 3 3
[1] 1 3 2
[1] 1 3
```

In this case the first dimension is preserved because for S3 class `keep` an empty indexing argument implies we want to keep the dimension. However, we still have the issue that if an indexing argument evaluates to 1 then it will be dropped. However, we can stop this behaviour by making use of the `keep()` function which flags the index dimension to be kept as a side effect of the function call. In this case,

```
M <- karray(1:12, c(1,3,4))
for (i in 1:4) print(dim(M[,,keep(i:4)]))
```

produces the output,

```
[1] 1 3 4
[1] 1 3 3
[1] 1 3 2
[1] 1 3 1
```

Value

karray and as.karray returns an array with additional S3 class designation of keep

kOarray and as.kOarray returns an Oarray with additional S3 class designation of keep

as.array returns an array with the additional S3 class designation removed

keep returns the index passed in. The function is used for its side effect of marking a dimension for preservation.

Note

Whilst this package references the Oarray package internally it does not make this dependency explicit. It does so to avoid having to have the package installed in cases where you are not going to be using Oarray. If you wish to use the Oarray related support you will need to ensure that the package is installed and loaded yourself.

If you are going to be using keep arrays in S4 classes then you can make reference to the appropriate array type through the package defined class unions karray and kOarray. For example,

```
setClass("MyS4Class", representation(Array1="karray",Array2="karray", ...))
```

Author(s)

Paavo Jumppanen

See Also

array,Oarray

Examples

```
# Normal R array
M <- array(1:12, c(4,3,1))

# First dimension dropped because it has size 1
print(M[,2,])

# Normal R array but with keep class
M2 <- karray(1:12, c(4,3,1))

# First dimension preserved
print(M2[,2,])

# middle dimension dropped for i=3 because 3:3 has length 1
for (i in 1:3) print(M2[,i:3,])

# use keep() to preserve middle dimension
for (i in 1:3) print(M2[,keep(i:3),])

# indexing through arrays works as normal
ind <- as.matrix(expand.grid(1:4,1:3,1:1))
M2[ind]
```

Index

∗ **array**
keep, 2
[.keep *(*keep*)*, 2

array, *4*
as.array.keep *(*keep*)*, 2
as.karray *(*keep*)*, 2
as.kOarray *(*keep*)*, 2

karray *(*keep*)*, 2
keep, 2
kOarray *(*keep*)*, 2

Oarray, *4*