Primitive array operations in the gRbase package

Søren Højsgaard

January 18, 2012

Contents


```
> library(gRbase)
```

1 Introduction

This note describes some operations on arrays in R. These operations have been implemented to facilitate implementation of graphical models and Bayesian networks in R.

1.1 Arrays in R

The documentation of R states the following about arrays:

> An array in R can have one, two or more dimensions. It is simply a vector which is stored with additional attributes giving the dimensions (attribute "dim") and optionally names for those dimensions (attribute "dimnames").
>
> A two-dimensional array is the same thing as a matrix.
>
> One-dimensional arrays often look like vectors, but may be handled differently by some functions.

Hence the defining characterstic of an array is that it is a vector with a dim attribute. For example

```
> ## 1-dimensional array
> ##
> x1 <- 1:8
> dim(x1) <- 8
> x1

[1] 1 2 3 4 5 6 7 8

> is.array(x1)

[1] TRUE

> is.matrix(x1)

[1] FALSE

> ## 2-dimensional array (matrix)
> ##
> x2 <- 1:8
> dim(x2) <- c(2,4)
> x2

     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8

> is.array(x2)

[1] TRUE

> is.matrix(x2)

[1] TRUE

> ## 3-dimensional array
> ##
> x3 <- array(1:8, dim=c(2,2,2))
> x3

, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4


, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8

> is.array(x3)

[1] TRUE

> is.matrix(x3)

[1] FALSE
```

1.2 Terminology

Consider a set $\Delta = \{\delta_1, \ldots, \delta_K\}$ of $|\Delta| = K$ factors where the factor δ_k has levels I_k. The cross product $I = I_1 \times \ldots \times I_K$ defines an array where $i = (i_1, \ldots, i_K) \in I$ is a cell. It is the convention here that the first factor varies fastest. To each cell $i \in I$ there is often a value $f(i)$.

As shown above, an array is implemented as a vector x of length $L = |I|$, that is $x \equiv (f(i), i \in I)$. In practice x is indexed by an entry e as $x[e]$ for $e = 1, \dots, L$.

The factor levels $(I_1, \dots, I_K)$ are denoted `flevels` in the code below. As an example we take the following:

```
> flevels <- c(2,3,2,4)
```

2 `cell2entry()` and `entry2cell()`

The map from a cell to the corresponding entry is provided by `cell2entry()`. The reverse operation, going from an entry to a cell (which is much less needed) is provided by `entry2cell()`.

```
> cell2entry(c(1,1,1,1), flevels)

[1] 1

> entry2cell(1, flevels)

[1] 1 1 1 1

> cell2entry(c(2,1,2,1), flevels)

[1] 8

> entry2cell(8, flevels)

[1] 2 1 2 1
```

3 `nextCell()` and `nextCellMarg()`

Given a cell, say $i = (1,1,2,1)$ we often want to find the next cell in the table following the convention that the first factor varies fastest, that is $(2,1,2,1)$. This is provided by `nextCell()`. Notice the result of finding the next cell to the final cell.

```
> nextCell(c(1,1,2,1), flevels)

[1] 2 1 2 1

> nextCell(c(2,3,2,4), flevels)

NULL
```

Given $A \subset \Delta$ and a cell $i_A \in I_A$ consider the cells $I(i_A) = \{j \in I | j_A = i_A\}$. For example, the cells satisfying that factor 2 is at level 1. Given such a cell, say $(2,1,1,2)$ we often want to find the next cell also satisfying this constraint following the convention that the first factor varies fastest, that is $(1,1,2,2)$. This is provided by `nextCellMarg()`.

```
> nextCellMarg(c(1,3,2,1),    margset=c(2,3), flevels)

[1] 2 3 2 1

> nextCellMarg(c(2,3,2,1),    margset=c(2,3), flevels)

[1] 1 3 2 2
```

4 margcell2entry()

Given $A \subset \Delta$ and a cell $i_A \in I_A$. This cell defines a slice of the original array, namely the cells $I(i_A) = \{j \in I | j_A = i_A\}$. We often want to find the entries in x for the cells $I(i_A)$. This is provided by `margcell2entry()`. To be specific, we may want the entries for the cells $(*, 1, 2, *)$ or $(2, 2, *, *)$:

```
> margcell2entry(margcell=c(1,2), margset=c(2,3), flevels)

[1]  7  8 19 20 31 32 43 44

> margcell2entry(margcell=c(2,2), margset=c(1,2), flevels)

[1]  4 10 16 22 28 34 40 46
```

5 Factorial grids

Using the operations above we can obtain the combinations of the factors as a matrix:

```
> ff <- factgrid(flevels)
> head(ff)

     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    2    1    1    1
[3,]    1    2    1    1
[4,]    2    2    1    1
[5,]    1    3    1    1
[6,]    2    3    1    1

> tail(ff)

      [,1] [,2] [,3] [,4]
[43,]    1    1    2    4
[44,]    2    1    2    4
[45,]    1    2    2    4
[46,]    2    2    2    4
[47,]    1    3    2    4
[48,]    2    3    2    4

> ff <- factgrid(flevels, margcell=c(1,2), margset=c(2,3))
> ff

     [,1] [,2] [,3] [,4]
[1,]    1    1    2    1
[2,]    2    1    2    1
[3,]    1    1    2    2
[4,]    2    1    2    2
[5,]    1    1    2    3
[6,]    2    1    2    3
[7,]    1    1    2    4
[8,]    2    1    2    4
```