Advanced array operations in the **gRbase** package

Søren Højsgaard

March 11, 2013

Contents

1 Tables

This note describes various functions in the **gRbase** package for operations on tables / arrays in R. Notice that there is a distinction between a `table` and an `array` in R. For the purpose of what is described here the concepts can be used interchangably. The important point is that we are working on vectors which have a `dim` and a `dimnames` attribute. (Arrays do not need a `dimnames` attribute, but `dimnames` are essential in what follows here).

Consider the `lizard` data in **gRbase**:

```
R> data(lizard)
R> lizard


, , species = anoli

     height
diam  >4.75 <=4.75
  <=4    32     86
  >4     11     35
```

```
, , species = dist

     height
diam  >4.75 <=4.75
  <=4    61     73
  >4     41     70
```

Data is of class `table` and has `dim` and `dimnames` attributes

```
R> class(lizard)

[1] "table"

R> is.array(lizard)

[1] TRUE

R> dim(lizard)

[1] 2 2 2

R> dimnames(lizard)

$diam
[1] "<=4" ">4"

$height
[1] ">4.75"  "<=4.75"

$species
[1] "anoli" "dist"
```

Notice from the output above that the first variable (`diam`) varies fastest.

2   Notation

A formal description of an array is as follows: Let $\Delta = \{\delta_1, \ldots, \delta_R\}$ be a set of factors where δ_r has a finite set I_r of levels. Let $|I_r|$ denote the number of levels of δ_r and let $i_r \in I_r$ denote a value of δ_r. A configuration of the variables in Δ is $i = i_\Delta = (i_1, \ldots, i_R) \in I_1 \times \ldots \times I_R = I$. The total number of configurations is $|\Delta| = \prod_r |I_r|$. An array T is a function which maps I into some domain.

3 Algebraic operations on tables

Let U and V be non–empty subsets of Δ with configurations I_U and I_V and let T_U^1 and T_V^2 be corresponding arrays.

- The *product* and *quotient* of T_U^1 and T_V^2 are arrays defined on $U \cup V$ given by

$$T_{U \cup V}(i_{U \cup V}) := T_U^1(i_U) \times T_V^2(i_V) \text{ and } T_{U \cup V}(i_{U \cup V}) := T_U^1(i_U)/T_V^2(i_V)$$

  respectively, with the convention that $0/0 = 0$.

- If $W \subset U$ is non–empty[1] then *marginalization* of T_U^1 onto W is defined as

$$T_W^1(i_W) := \sum_{i_{U \setminus W}} T_U^1(i_{U \setminus W}, i_W)$$

- If $W \subset U$ is non–empty then a configuration i_W^* defines a *slice* of T_U^1 as

$$T_{U \setminus V}^1(i_{U \setminus V}) := T_U^1(i_{U \setminus V}, i_V^*)$$

In a less abstract setting let $U = \{A, B, C\}$, $V = \{C, D, B\}$ and $W = \{C, B\}$ where (a, b, c) denotes a specific configuration of $\{A, B, C\}$ and so on. Then the product and quotient become

$$T_{ABCD}(a, b, c, d) = T_{ABC}^1(a, b, c)T_{CDB}^2(c, d, b)$$

The marginal becomes

$$T_{CB}^1 = \sum_a T^1(a, b, c) \text{ and}$$

Finally the slice defined by $C = c^*$ and $B = b^*$ becomes

$$T_A^1(a) = T_{ABC}^1(a, b^*, c^*)$$

To illustrate we find two marginal tables

```
R> T1.U <- tableMargin(lizard, c("species","height"))


        height
species >4.75 <=4.75
  anoli    43    121
  dist    102    143


R> T1.V <- tableMargin(lizard, c("diam","species"))
```

---

[1]Marginalization onto an empty set is not implemented.

3

```
        species
diam   anoli dist
  <=4    118  134
  >4      46  111
```

Multiplication of these is done with

```
R> T1.UV<-tableOp(T1.U, T1.V, op = "*")
```

```
, , height = >4.75

       species
diam   anoli   dist
  <=4   5074  13668
  >4    1978  11322

, , height = <=4.75

       species
diam   anoli   dist
  <=4  14278  19162
  >4    5566  15873
```

A slice of a table is obtained with `tableSlice`:

```
R> tableSlice(lizard, "species", "anoli")
```

```
       height
diam   >4.75 <=4.75
  <=4     32     86
  >4      11     35
```

A reorganization of the table can be made with `tablePerm`:

```
R> tablePerm(T1.UV, c("species","height","diam"))
```

```
, , diam = <=4

         height
species  >4.75 <=4.75
  anoli   5074  14278
  dist   13668  19162

, , diam = >4

         height
species  >4.75 <=4.75
  anoli   1978   5566
  dist   11322  15873
```

4 Defining tables / arrays

As mentioned above, a table can be represented as an array. In general, arrays do not need dimnames in R, but for the functions described here, the dimnames are essential.

The examples here relate to the chest clinique example of Lauritzen and Spiegelhalter. The following two specifications are equivalent:

```
R> yn <- c('y','n')
R> T.U <- array(c(5,95,1,99), dim=c(2,2), dimnames=list("tub"=yn, "asia"=yn))
R> T.U <- parray(c("tub","asia"), levels=list(yn, yn), values=c(5,95,1,99))
```

Using `parray()`, arrays can be normalized in two ways: Normalization can be over the first variable for *each* configuration of all other variables or over all configurations. We illustrate this by defining the probability of tuberculosis given a recent visit to Asia and by defining the marginal probability of a recent visit to Asia:

```
R> T.U <- parray(c("tub","asia"), levels=list(yn, yn),
+                values=c(5,95,1,99), normalize="first")


     asia
tub     y     n
  y  0.05  0.01
  n  0.95  0.99


R> T.V <- parray("asia", list(yn), values=c(1,99),
+                normalize="all")

asia
   y     n
0.01  0.99
```

5 Calculations with probability tables

The joint distributions is

```
R> T.all <- tableOp(T.U, T.V, op="*")


      tub
asia       y       n
   y  0.0005  0.0095
   n  0.0099  0.9801
```

The marginal distribution of `"tub"` is

```
R> T.W <- tableMargin(T.all, "tub")
```

```
tub
     y      n
0.0104 0.9896
```

The conditional distribution of **"asia"** given **"tub"** is

```
R> tableOp(T.all, T.W, op="/")
```

```
   asia
tub            y          n
  y 0.048076923 0.9519231
  n 0.009599838 0.9904002
```