Table operations in the gRbase package

Søren Højsgaard

February 3, 2009

Contents

1 Tables

This note describes various functions in the `gRbase` package for opetations on tables / arrays. Consider the `HairEyeColor` data:

```
> data(HairEyeColor)
> hec <- HairEyeColor
> hec

, , Sex = Male

       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8

, , Sex = Female

       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

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

```
> class(hec)

[1] "table"

> dim(hec)

[1] 4 4 2

> dimnames(hec)

$Hair
[1] "Black" "Brown" "Red"   "Blond"

$Eye
[1] "Brown" "Blue"  "Hazel" "Green"

$Sex
[1] "Male"   "Female"

> str(hec)

 table [1:4, 1:4, 1:2] 32 53 10 3 11 50 10 30 10 25 ...
 - attr(*, "dimnames")=List of 3
  ..$ Hair: chr [1:4] "Black" "Brown" "Red" "Blond"
  ..$ Eye : chr [1:4] "Brown" "Blue" "Hazel" "Green"
  ..$ Sex : chr [1:2] "Male" "Female"
```

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

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. What is important is that we are working on a vector which has a `dim` and `dimnames` attribute. (Arrays do not need a `dimnames` attribute, but they are essential in what follows here).

A formal description of a table is as follows: Let $\Delta = \{\delta_1, \dots, \delta_R\}$ be a set of discrete variables 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 $i = i_\Delta = (i_1, \dots, i_R) \in I_1 \times \dots \times I_R = I_\Delta$. The total number of configurations is $|\Delta| = \prod_r |I_r|$.

2 Algebraic operations on tables

To define algebraic operations on tables, let U be a non–empty subsets of Δ with configurations I_U and let i_U denote a specific configuration. A table T_U defined on I_U is a function which maps i_U into some domain for all $i_U \in I_U$. Let U and V be non–empty subsets of Δ with configurations I_U and I_V and let T^1_U and T^2_V be corresponding potentials.

The *product* and *quotient* of T^1_U and T^2_V are potentials defined on $U \cup V$ given by

$$T_{U\cup V} := T^1_U \times T^2_V \text{ and } T_{U\cup V} := T^1_U / T^2_V$$

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

If $V \subset U$ is non–empty[1] then *marginalization* of T^1_U onto V is defined as

$$T^1_V := \sum_{U \setminus V} T^1_U$$

If $V \subset U$ is non–empty then a configuration i^*_V defines a *slice* of T^1_U as

$$T^1_{U \setminus V}(i_{U \setminus V}) := T^1_U(i_{U \setminus V}, i^*_V)$$

To illustrate we find two marginal tables

```
> T1.U <- tableMargin(hec, c("Hair", "Eye"))

       Eye
Hair    Brown Blue Hazel Green
  Black    68   20    15     5
  Brown   119   84    54    29
  Red      26   17    14    14
  Blond     7   94    10    16

> T1.V <- tableMargin(hec, c("Hair", "Sex"))

       Sex
Hair    Male Female
  Black   56     52
  Brown  143    143
  Red     34     37
  Blond   46     81
```

Multiplication of these is done with

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

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

, , Eye = Brown

       Sex
Hair     Male Female
  Black  3808   3536
  Brown 17017  17017
  Red     884    962
  Blond   322    567

, , Eye = Blue

       Sex
Hair     Male Female
  Black  1120   1040
  Brown 12012  12012
  Red     578    629
  Blond  4324   7614

, , Eye = Hazel

       Sex
Hair    Male Female
  Black  840    780
  Brown 7722   7722
  Red    476    518
  Blond  460    810

, , Eye = Green

       Sex
Hair    Male Female
  Black  280    260
  Brown 4147   4147
  Red    476    518
  Blond  736   1296
```

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

```
> tablePerm(T1.UV, c("Hair", "Eye", "Sex"))

, , Sex = Male

       Eye
Hair    Brown  Blue Hazel Green
  Black  3808  1120   840   280
  Brown 17017 12012  7722  4147
  Red     884   578   476   476
  Blond   322  4324   460   736

, , Sex = Female

       Eye
Hair    Brown  Blue Hazel Green
  Black  3536  1040   780   260
  Brown 17017 12012  7722  4147
  Red     962   629   518   518
  Blond   567  7614   810  1296
```

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

```
> tableSlice(hec, "Sex", "Female")

       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8
```

3 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:

```
> yn <- c("y", "n")
> T.U <- array(c(5, 95, 1, 99), dim = c(2, 2), dimnames = list(tub = yn, asia = yn))
> T.U <- ptable(c("tub", "asia"), nLevels = list(yn, yn), values = c(5, 95, 1,
+     99))
```

Using `ptable()`, 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:

```
> T.U <- ptable(c("tub", "asia"), nLevels = 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

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

asia
   y    n
0.01 0.99
```

The joint distributions is

```
> 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

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

tub
     y      n
0.0104 0.9896
```

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

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

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