The doBy package

by Søren Højsgaard

This article is not about rocket science; in fact it is not about science at all. It is a description of yet another package with utility functions.

I have used R in connection with teaching generalized linear models and related topics to Ph.d. students within areas like agronomy, biology, and veterinary science at the Danish Institute of Agricultural Sciences.

These students, many of whom are familiar with the SAS system, have almost all come to appreciate R very quickly. However, they have also from time to time complained that certain standard tasks are hard to do in R – and certainly harder than in SAS. The **doBy** package is an attempt to make some of these standard tasks easier.

Airquality data

The presentation of the package is based on the `airquality` dataset which contains air quality measurements in New York, May to September 1973. (Note that months are coded as $5, \ldots, 9$).

```
> head(airquality)
```

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

The `summaryBy` function

With the summary procedure of SAS (`PROC SUMMARY`) one can easily calculate things like "the mean and variance of x for each combination of two factors A and B". To calculate mean and variance of Ozone and Wind for each combination of Month, do:

```
> summaryBy(Ozone + Wind ~ Month,
+     data = airquality, FUN = c(mean,
+         var), prefix = c("m",
+         "v"), na.rm = TRUE)
```

```
  Month m.Ozone m.Wind v.Ozone v.Wind
1     5   23.62 11.623   493.9 12.471
2     6   29.44 10.267   331.5 14.207
3     7   59.12  8.942  1000.8  9.217
4     8   59.96  8.794  1574.6 10.407
5     9   31.45 10.180   582.8 11.980
```

The result above can clearly be obtained in other ways. For example by using the `aggregate` function, the `summarize` function in the `Hmisc` package or by

```
> a <- by(airquality, airquality$Month,
+     function(d) {
+         c(mean(d[, c("Ozone",
+             "Wind")], na.rm = T),
+             diag(var(d[, c("Ozone",
+                 "Wind")], na.rm = T)))
+     })
> do.call("rbind", a)
```

However, my students have found this somewhat cumbersome!

The `orderBy` function

Ordering (or sorting) a data frame is possible with the `orderBy` function. Suppose we want to order the dataframe by `Temp` and by `Month` (within `Temp`) and that the ordering should be decreasing. This can be achieved by:

```
> x <- orderBy(~Temp + Month,
+     data = airquality, decreasing = T)
```

The first lines of the result are:

```
    Ozone Solar.R Wind Temp Month Day
120    76     203  9.7   97     8  28
122    84     237  6.3   96     8  30
121   118     225  2.3   94     8  29
123    85     188  6.3   94     8  31
126    73     183  2.8   93     9   3
127    91     189  4.6   93     9   4
```

Again, this can clearly be achieved in other ways, but presumably not with so few commands as above.

The `splitBy` function

Suppose we want to split data into a list of dataframes, e.g. one dataframe for each month. This can be achieved by:

```
> x <- splitBy(~Month, data = airquality)
```

Information about the grouping is stored as a dataframe in an attribute called `groupid`:

```
> attr(x, "groupid")
```

```
  Month
1     1
2     2
3     3
4     4
5     5
```

The `sampleBy` function

Suppose we want a random sample of 50% of the observations from a dataframe. This can be achieved with:

```
> sampleBy(~1, frac = 0.5, data = airquality)
```

Suppose instead that we want a systematic sample of every fifth observation within each month. This is achieved with:

```
> sampleBy(~Month, frac = 0.2,
+     data = airquality, systematic = T)
```

The `subsetBy` function

Suppose we want to take out those rows within each month for which the the wind speed is larger than the mean wind speed (within the month). This is achieved by:

```
> subsetBy(~Month, subset = "Wind>mean(Wind)",
+     data = airquality)
```

Note that the statement `"Wind>mean(Wind)"` is evaluated within each month.

The `esticon` function

Consider a linear model which explains `Ozone` as a linear function of `Month` and `Wind`:

```
> airquality <- transform(airquality,
+     Month = factor(Month))
> m <- lm(Ozone ~ Month * Wind,
+     data = airquality)
> coefficients(m)

(Intercept)      Month6      Month7
     50.748     -41.793      68.296
     Month8      Month9        Wind
     82.211      23.439      -2.368
Month6:Wind Month7:Wind Month8:Wind
      4.051      -4.663      -6.154
Month9:Wind
     -1.874
```

When a parameter vector β of (systematic) effects have been estimated, interest is often in a particular estimable function, i.e. linear combination $\lambda^\top\beta$ and/or testing the hypothesis $H_0 : \lambda^\top\beta = \beta_0$ where λ is a specific vector defined by the user.

Suppose for example we want to calculate the expected difference in ozone between consequtive months at wind speed 10 mph (which is about the average wind speed over the whole period).

The `esticon` function provides a way of doing so. We can specify several λ vectors at the same time by row–binding of $\lambda^\top$:

```
> Lambda <- rbind(c(0, -1, 0,
+     0, 0, 0, -10, 0, 0, 0),
+     c(0, 1, -1, 0, 0, 0, 10,
+         -10, 0, 0), c(0, 0,
+         1, -1, 0, 0, 0, 10,
+         -10, 0), c(0, 0, 0,
+         1, -1, 0, 0, 0, 10,
+         -10))
> esticon(m, Lambda)

Confidence interval ( WALD ) level = 0.95
  beta0 Estimate Std.Error t.value
1     0   1.2871    10.238  0.1257
2     0 -22.9503    10.310 -2.2259
3     0   0.9954     7.094  0.1403
4     0  15.9651     6.560  2.4337
   DF Pr(>|t|) Lower.CI Upper.CI
1 106  0.90019  -19.010   21.585
2 106  0.02814  -43.392   -2.509
3 106  0.88867  -13.069   15.060
4 106  0.01662    2.959   28.971
```

In other cases, interest is in testing a hypothesis of a contrast $H_0 : \Lambda\beta = \beta_0$ where Λ is a matrix. For example a test of no interaction between `Month` and `Wind` can be made by testing jointly that the last four parameters in `m` are zero (observe that the test is a Wald test):

```
> Lambda <- rbind(c(0, 0, 0, 0,
+     0, 0, 1, 0, 0, 0), c(0,
+     0, 0, 0, 0, 0, 0, 1, 0,
+     0), c(0, 0, 0, 0, 0, 0,
+     0, 0, 1, 0), c(0, 0, 0,
+     0, 0, 0, 0, 0, 0, 1))
> esticon(m, Lambda, joint.test = T)

  X2.stat DF Pr(>|X^2|)
1   22.11  4  0.0001906
```

For a linear normal model, one would typically prefer to do a likelihood ratio test instead. However, for generalized estimating equations of glm–type (as dealt with in the packages **geepack** and **gee**) there is no likelihood. In this case `esticon` function provides an operational alternative.

Observe that another function for calculating contrasts as above is the `contrast` function in the **Design** package but it applies to a narrower range of models than `esticon` does.

Final remarks

The ease in using the data oriented functions lies in that 1) the formula language is used in the specification of both the variables and the grouping and 2) the functions take a `data` argument. My "biologically oriented" students (and ditto collegues) seem to appreciate that.

On their wishlist are facilities along the line of the `ESTIMATE` and `LSMEANS` statements available in many SAS procedures (including `GLM`, `MIXED` and `GENMOD`) for easy specification of various contrasts (`LSMEANS` is sometimes also denoted "population means"). While `LSMEANS` are often misused (and certainly misinterpreted) such facilities would be nice to have in R. I would like to encourage anyone who has implemented such facilities in a reasonable level of generality to come forward.

Søren Højsgaard
Statistics and Decision Analysis Unit
Department of Genetics and Biotechnology
Danish Institute of Agricultural Sciences
Tjele, Denmark
sorenh@agrsci.dk

Normed division algebras with R: Introducing the onion package

The eccentric cousin and the crazy old uncle

by Robin K. S. Hankin

Preface

An *algebra* is a vector space V over a field (here the real numbers) in which the vectors may be multiplied. In addition to the usual vector space axioms, one requires, for any $\lambda, \mu \in \mathbb{R}$ and $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$:

- $\mathbf{x}(\lambda\mathbf{y} + \mu\mathbf{z}) = \lambda\mathbf{xy} + \mu\mathbf{xz}$
- $(\lambda\mathbf{y} + \mu\mathbf{z})\mathbf{x} = \lambda\mathbf{yx} + \mu\mathbf{zx}$

where multiplication is denoted by juxtaposition; note the absence of associativity. A *division algebra* is a nontrivial algebra in which division is possible: for any $\mathbf{a} \in V$ and any nonzero $\mathbf{b} \in V$, there exists precisely one element $\mathbf{x}$ with $\mathbf{a} = \mathbf{bx}$ and precisely one element $\mathbf{y}$ with $\mathbf{a} = \mathbf{yb}$. A *normed division algebra* is a division algebra with a norm $||\cdot||$ satisfying $||\mathbf{xy}|| = ||\mathbf{x}||\ ||\mathbf{y}||$.

There are precisely four normed division algebras: the reals themselves ($\mathbb{R}$), the complex numbers ($\mathbb{C}$), the quaternions ($\mathbb{H}$) and the octonions ($\mathbb{O}$); the generic term is "onion", although the term includes other algebras such as the sedenions.

The R programming language (Development Core Team, 2004) is well-equipped to deal with the first two: here, I introduce the **onion** package, which provides some functionality for the quaternions and the octonions, and illustrate these interesting algebras using numerical examples.

Introduction

Historically, the complex numbers arose from a number of independent lines of inquiry and our current understanding of them (viz $z = x + iy$; the Argand plane) developed over the eighteenth and nineteenth centuries.

Hamilton was one of many mathematicians to attempt to extend the complex numbers to a third dimension and discover what would in our terminology be a three dimensional normed division algebra. We now know that no such thing exists: division algebras all have dimension 2^n for n some non-negative integer.

Hamilton came upon the multiplicative structure of the quaternions in a now-famous flash of inspiration on 16th October 1843: the quaternions are obtained by adding the elements i, j, and k to the real numbers and requiring that

$$i^2 = j^2 = k^2 = ijk = -1. \qquad (1)$$

A general quaternion is thus written $a + bi + cj + dk$ with a, b, c, d being real numbers; complex arithmetic is recovered if $c = d = 0$. Hamilton's relations above, together with distributivity and associativity, yield the full multiplication table and the quaternions are the unique four dimensional normed division algebra.

However, Hamilton's scheme was controversial as his multiplication was noncommutative: a shocking suggestion at the time. Today we recognize many more noncommutative operations (such as matrix multiplication), but even Hamilton had difficulty convincing his contemporaries that such operations were consistent, let alone worth studying.

The octonions

The fourth and final normed division algebra is that of the octonions. These were discovered around 1844 and are an eight-dimensional algebra over the reals. The full multiplication table is given by Baez (2001).

Package onion in use

A good place to start is function `rquat()`, which returns a quaternionic vector of a specified length, whose elements are random small integers: