LSmeans, population means, estimates and contrasts with the doBy package

Søren Højsgaard

doBy version 4.5-5 as of feb 07, 2013

Contents

1 Introduction

This is a working document; please feel free to suggest improvements.

```
R> library(doBy)
```

2 Linear functions of parameters, contrasts

For a regression model with parameters $\beta = (\beta^1, \beta^2, \dots, \beta^P)$ we shall refer to a weighted sum of the form

$$\sum_j w_j \beta^j$$

as a contrast. Notice that it is common in the litterature to require that $\sum_j w_j = 0$ for the sum $\sum_j w_j \beta^j$ to be called a contrast but we do not follow this tradition here.

3 The `esticon` function

Consider a linear model which explains `Ozone` as a linear function of `Month` (factor) and `Wind` (numeric) and an interaction between these effects:

```
R> data(airquality)
R> airquality <- transform(airquality, Month=factor(Month))
R> m <- lm(Ozone~Month*Wind, data=airquality)
R> coef(summary(m))

              Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  50.748472  15.747605  3.222615 0.0016879982
Month6      -41.793446  31.147533 -1.341790 0.1825314437
Month7       68.295661  20.995006  3.252948 0.0015324979
Month8       82.210682  20.313529  4.047090 0.0000987944
Month9       23.439210  20.662571  1.134380 0.2591939469
Wind         -2.368111   1.316221 -1.799175 0.0748364258
Month6:Wind   4.050636   2.490344  1.626537 0.1068048191
Month7:Wind  -4.663240   2.025735 -2.301999 0.0232898057
Month8:Wind  -6.154287   1.922637 -3.200961 0.0018078805
Month9:Wind  -1.873651   1.820335 -1.029289 0.3056867339
```

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 investigator. 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. For example

```
R> 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))
```

Notice that the specification of `Lambda` depends on the choice of reference level of the factors; if the reference levels are changed, `Lambda` must be changed too. Estimates and tests are obtained with:

```
R> esticon(m, Lambda)

  beta0    Estimate Std.Error    t.value  DF  Pr(>|t|)       Lower     Upper
1     0   1.2870834 10.237872  0.1257179 106 0.9001934 -19.010494 21.584661
2     0 -22.9503412 10.310435 -2.2259335 106 0.0281352 -43.391780 -2.508902
3     0   0.9954442  7.093972  0.1403225 106 0.8886715 -13.069046 15.059934
4     0  15.9651107  6.560102  2.4336681 106 0.0166188   2.959071 28.971151
```

In other cases, interest is in testing a hypothesis of a contrast $H_0 : \Lambda\beta = \beta_0$ where Λ is a matrix as a joint test. 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):

```
R> 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))
```

```
R> esticon(m, Lambda, joint.test=TRUE)

   X2.stat DF   Pr(>|X^2|)
1 22.10963  4 0.0001905969
```

For a linear normal model a more attractice alternative (which is also easier to specify) is a likelihood ratio test:

```
R> m2 <- update(m, .~.-Month:Wind)
R> anova(m, m2)

Analysis of Variance Table

Model 1: Ozone ~ Month * Wind
Model 2: Ozone ~ Month + Wind
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    106 56649
2    110 68465 -4    -11816 5.5274 0.0004423 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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. In fact, the `anova()` method in **geepack** is based on calling `esticon()`.

4 LSmeans (population means, marginal means)

4.1 A simulated dataset

In the following sections we consider these data:

```
R> library(doBy)
R> dd <- expand.grid(A=factor(1:3),B=factor(1:3),C=factor(1:2))
R> dd$y <- rnorm(nrow(dd))
R> dd$x <- rnorm(nrow(dd))^2
R> dd$z <- rnorm(nrow(dd))
R> head(dd,10)

   A B C           y          x           z
1  1 1 1  1.29937657 0.10518273  0.97300486
2  2 1 1  1.42620454 0.18639122 -0.61194871
3  3 1 1 -0.47386798 0.30860641 -0.29959724
4  1 2 1  2.08016078 0.87098602  0.88355491
5  2 2 1  1.29928782 3.11513627 -1.12004173
6  3 2 1  0.29179084 0.05240387 -0.56642883
7  1 3 1  0.47804974 4.01086685 -0.55524764
8  2 3 1  1.04426436 0.03674464 -0.29209647
9  3 3 1 -2.81944660 0.10580910 -1.04036748
10 1 1 2  0.07415363 0.14378647  0.03997111
```

Consider the additive model

$$y_i = \beta_0 + \beta^1_{A(i)} + \beta^2_{B(i)} + \beta^3_{C(i)} + e_i \tag{1}$$

where $e_i \sim N(0, \sigma^2)$. We fit this model:

```
R> mm <- lm(y~A+B+C, data=dd)
R> coef(mm)

(Intercept)          A2          A3          B2          B3          C2
  0.7484006   0.3467459  -0.8105740   0.4656698  -0.7051035  -0.2917864
```

Notice that the parameters corresponding to the factor levels `A1`, `B1` and `C2` are set to zero to ensure identifiability of the remaining parameters.

4.2 What are these quantities

LSmeans, population means and marginal means are used synonymously in the literature. These quantities are a special kind of contrasts as defined in Section 2. LSmeans seems to be the most widely used term, so we shall adopt this terms here too.

The model (1) is a model for the conditional mean $\mathbb{E}(y|A, B, C)$. Sometimes one is interested in quantities like $\mathbb{E}(y|A)$. This quantity can not formally be found unless B and C are random variables such that we may find $\mathbb{E}(y|A)$ by integration. However, suppose that A is a treatment of main interest, B is a blocking factor and C represents days on which the experiment was carried out. Then it is tempting to average $\mathbb{E}(y|A, B, C)$ over B and C (average over block and day) and think of this average as $\mathbb{E}(y|A)$.

The population mean for $A = 1$ is

$$\beta^0 + \beta^1_{A1} + \frac{1}{3}(\beta^2_{B1} + \beta^2_{B2} + \beta^2_{B3}) + \frac{1}{2}(\beta^3_{C1} + \beta^3_{C2}) \tag{2}$$

Recall that the parameters corresponding to the factor levels `A1`, `B1` and `C2` are set to zero to ensure identifiability of the remaining parameters. Therefore we may also write the population mean for $A = 1$ as

$$\beta^0 + \frac{1}{3}(\beta^2_{B2} + \beta^2_{B3}) + \frac{1}{2}(\beta^3_{C2}) \tag{3}$$

This quantity can be estimated as:

```
R> w <- c(1, 0, 0, 1/3, 1/3, 1/2)
R> coef(mm)*w

(Intercept)          A2          A3          B2          B3          C2
  0.7484006   0.0000000   0.0000000   0.1552233  -0.2350345  -0.1458932

R> sum(coef(mm)*w)

[1] 0.5226962
```

We may find the population mean for all three levels of A as

```
R> W <- matrix(c(1, 0, 0, 1/3, 1/3, 1/2,
+                1, 1, 0, 1/3, 1/3, 1/2,
+                1, 0, 1, 1/3, 1/3, 1/2),nr=3, byrow=TRUE)
```

Notice that the matrix W is based on that the first level of A is set as the reference level. If the reference level is changed then so must W be.

Given that one has specified W, we can use the `esticon()` function in the `doBy` as illustrated below:

```
R> esticon(mm, W)

  beta0   Estimate Std.Error    t.value DF  Pr(>|t|)      Lower     Upper
1     0  0.5226962 0.5258321  0.9940362 12 0.3398273 -0.6229936 1.6683859
2     0  0.8694421 0.5258321  1.6534594 12 0.1241396 -0.2762477 2.0151318
3     0 -0.2878779 0.5258321 -0.5474711 12 0.5940915 -1.4335676 0.8578119
```

4.3 Using `popMatrix()` and `popMeans()`

Writing the matrix W is somewhat tedious and hence error prone. In addition, there is a potential risk of getting the wrong answer if the the reference level of a factor has been changed. The `popMatrix()` function provides an automated way of generating such matrices. The above `W` matrix is constructed by

```
R> pma <- popMatrix(mm,effect='A')
R> summary(pma)

     (Intercept) A2 A3        B2        B3  C2
[1,]           1  0  0 0.3333333 0.3333333 0.5
[2,]           1  1  0 0.3333333 0.3333333 0.5
[3,]           1  0  1 0.3333333 0.3333333 0.5
grid:
'data.frame':        3 obs. of  1 variable:
 $ A: chr  "1" "2" "3"
at:
 NULL
```

The `effect` argument requires to calculate the population means for each level of A aggregating across the levels of the other variables in the data.

The `popMeans()` function is simply a wrapper around first a call to `popMatrix()` followed by a call to (by default) `esticon()`:

```
R> pme <- popMeans(mm, effect='A')
R> pme

  beta0   Estimate Std.Error    t.value DF  Pr(>|t|)      Lower     Upper A
1     0  0.5226962 0.5258321  0.9940362 12 0.3398273 -0.6229936 1.6683859 1
2     0  0.8694421 0.5258321  1.6534594 12 0.1241396 -0.2762477 2.0151318 2
3     0 -0.2878779 0.5258321 -0.5474711 12 0.5940915 -1.4335676 0.8578119 3
```

More details about how the matrix was constructed is provided by the `summary()` function:

```
R> summary(pme)

  beta0   Estimate Std.Error    t.value DF  Pr(>|t|)      Lower     Upper A
1     0  0.5226962 0.5258321  0.9940362 12 0.3398273 -0.6229936 1.6683859 1
2     0  0.8694421 0.5258321  1.6534594 12 0.1241396 -0.2762477 2.0151318 2
3     0 -0.2878779 0.5258321 -0.5474711 12 0.5940915 -1.4335676 0.8578119 3
Call:
NULL
Contrast matrix:
Length  Class   Mode
     0   NULL   NULL
```

As an additional example we may do:

```
R> popMatrix(mm,effect=c('A','C'))

     (Intercept) A2 A3        B2        B3 C2
[1,]           1  0  0 0.3333333 0.3333333  0
[2,]           1  1  0 0.3333333 0.3333333  0
[3,]           1  0  1 0.3333333 0.3333333  0
[4,]           1  0  0 0.3333333 0.3333333  1
[5,]           1  1  0 0.3333333 0.3333333  1
[6,]           1  0  1 0.3333333 0.3333333  1
```

This gives the matrix for calculating the estimate for each combination of `A` and `C` when averaging over `B`.

Omitting `effect` as in

```
R> popMatrix(mm)

     (Intercept)        A2        A3        B2        B3  C2
[1,]           1 0.3333333 0.3333333 0.3333333 0.3333333 0.5

R> popMeans(mm)

  beta0  Estimate Std.Error t.value DF  Pr(>|t|)      Lower    Upper
1     0 0.3680868 0.3035893 1.21245 12 0.2486709 -0.2933775 1.029551
```

gives the “total average”.

4.4 Using the `at` argument

We may be interested in finding the population means at all levels of A but only at $C = 1$. This is obtained by using the `at` argument (in which case the average is only over the remaining factor B):

```
R> popMatrix(mm, effect='A', at=list(C='1'))

     (Intercept) A2 A3        B2        B3 C2
[1,]           1  0  0 0.3333333 0.3333333  0
[2,]           1  1  0 0.3333333 0.3333333  0
[3,]           1  0  1 0.3333333 0.3333333  0
```

Another way of creating the population means at all levels of (A, C) is therefore

```
R> popMatrix(mm, effect='A', at=list(C=c('1','2')))

     (Intercept) A2 A3        B2        B3 C2
[1,]           1  0  0 0.3333333 0.3333333  0
[2,]           1  1  0 0.3333333 0.3333333  0
[3,]           1  0  1 0.3333333 0.3333333  0
[4,]           1  0  0 0.3333333 0.3333333  1
[5,]           1  1  0 0.3333333 0.3333333  1
[6,]           1  0  1 0.3333333 0.3333333  1
```

We may have several variables in the `at` argument:

```
R> popMatrix(mm, effect='A', at=list(C=c('1','2'), B='1'))

     (Intercept) A2 A3 B2 B3 C2
[1,]           1  0  0  0  0  0
[2,]           1  1  0  0  0  0
[3,]           1  0  1  0  0  0
[4,]           1  0  0  0  0  1
[5,]           1  1  0  0  0  1
[6,]           1  0  1  0  0  1
```

4.5 Ambiguous specification when using the `effect` and `at` arguments

There is room for an ambiguous specification if a variable appears in both the `effect` and the `at` argument, such as

```
R> popMatrix(mm, effect=c('A','C'), at=list(C='1'))

     (Intercept) A2 A3        B2        B3 C2
[1,]           1  0  0 0.3333333 0.3333333  0
[2,]           1  1  0 0.3333333 0.3333333  0
[3,]           1  0  1 0.3333333 0.3333333  0
```

This ambiguity is due to the fact that the `effect` argument asks for the populations means at all levels of the variables but the `at` chooses only specific levels.

This ambiguity is resolved as follows: Any variable in the `at` argument is removed from the `effect` argument such as the statement above is equivalent to

```
R> popMatrix(mm, effect='A', at=list(C='1'))

     (Intercept) A2 A3        B2        B3 C2
[1,]           1  0  0 0.3333333 0.3333333  0
[2,]           1  1  0 0.3333333 0.3333333  0
[3,]           1  0  1 0.3333333 0.3333333  0
```

4.6 Using covariates

Next consider the model where a covariate is included:

```
R> mm2 <- lm(y~A+B+C+C:x, data=dd)
R> coef(mm2)

(Intercept)          A2          A3          B2          B3          C2 
 1.23205168  0.09451548 -2.06913676  0.92781435 -0.79662775 -1.45431633 
      C1:x        C2:x 
-0.10604271  1.14920235 
```

In this case we get

```
R> popMatrix(mm2,effect='A', at=list(C='1'))

     (Intercept) A2 A3        B2        B3 C2      C1:x C2:x
[1,]           1  0  0 0.3333333 0.3333333  0 0.9491782    0
[2,]           1  1  0 0.3333333 0.3333333  0 0.9491782    0
[3,]           1  0  1 0.3333333 0.3333333  0 0.9491782    0
```

Above, x has been replaced by its average and that is the general rule for models including covariates. However we may use the `at` argument to ask for calculation of the population mean at some user-specified value of x, say 12:

```
R> popMatrix(mm2,effect='A', at=list(C='1',x=12))

     (Intercept) A2 A3        B2        B3 C2 C1:x C2:x
[1,]           1  0  0 0.3333333 0.3333333  0   12    0
[2,]           1  1  0 0.3333333 0.3333333  0   12    0
[3,]           1  0  1 0.3333333 0.3333333  0   12    0
```

```
R> mm22 <- lm(y~A+B+C+C:x+I(x^2), data=dd)
R> coef(mm22)

(Intercept)          A2          A3          B2          B3          C2
 1.11980976 -0.01107036 -2.01233636  0.75417173 -0.75756426 -1.38200904
     I(x^2)        C1:x        C2:x
-0.20147251  0.68339213  1.75123473
```

```
R> popMatrix(mm22,effect='A', at=list(C='1'))

     (Intercept) A2 A3        B2        B3 C2    I(x^2)      C1:x C2:x
[1,]           1  0  0 0.3333333 0.3333333  0 0.9009393 0.9491782    0
[2,]           1  1  0 0.3333333 0.3333333  0 0.9009393 0.9491782    0
[3,]           1  0  1 0.3333333 0.3333333  0 0.9009393 0.9491782    0
```

```
R> dd <- transform(dd, x.sq=x^2)
R> mm23 <- lm(y~A+B+C+C:x+x.sq, data=dd)
R> coef(mm23)

(Intercept)          A2          A3          B2          B3          C2
 1.11980976 -0.01107036 -2.01233636  0.75417173 -0.75756426 -1.38200904
       x.sq        C1:x        C2:x
-0.20147251  0.68339213  1.75123473

R> popMatrix(mm23,effect='A', at=list(C='1'))

     (Intercept) A2 A3        B2        B3 C2     x.sq      C1:x C2:x
[1,]           1  0  0 0.3333333 0.3333333  0 2.609603 0.9491782    0
[2,]           1  1  0 0.3333333 0.3333333  0 2.609603 0.9491782    0
[3,]           1  0  1 0.3333333 0.3333333  0 2.609603 0.9491782    0
```

4.7 Using transformed covariates

Next consider the model where a transformation of a covariate is included:

```
R> mm3 <- lm(y~A+B+C+C:I(log(x)), data=dd)
R> coef(mm3)

 (Intercept)           A2           A3           B2           B3           C2 
   1.2718654   -0.5347050   -2.3510833    1.0499175   -1.0311703    0.9628959 
C1:I(log(x)) C2:I(log(x)) 
  -0.1586710    0.7617714 
```

In this case we can not use `popMatrix()` (and hence `popMeans()`) directly. Instead we first have to generate a new variable, say `log.x`, with `log.x`$= \log(x)$, in the data and then proceed as

```
R> dd <- transform(dd, log.x = log(x))
R> mm32 <- lm(y~A+B+C+C:log.x, data=dd)
R> popMatrix(mm32, effect='A', at=list(C='1'))

     (Intercept) A2 A3        B2        B3 C2 C1:log.x C2:log.x
[1,]           1  0  0 0.3333333 0.3333333  0 -1.31699        0
[2,]           1  1  0 0.3333333 0.3333333  0 -1.31699        0
[3,]           1  0  1 0.3333333 0.3333333  0 -1.31699        0
```

4.8 The `engine` argument of `popMeans()`

The `popMatrix()`is a function to generate a linear tranformation matrix of the model parameters with emphasis on constructing such matrices for population means. `popMeans()` invokes by default the `esticon()` function on this linear transformation matrix for calculating parameter estimates and confidecne intervals. A similar function to `esticon()` is the `glht` function of the `multcomp` package.

The `glht()` function can be chosen via the `engine` argument of `popMeans()`:

```
R>  library(multcomp)
R> g<-popMeans(mm,effect='A', at=list(C='1'),engine="glht")
R> g

	 General Linear Hypotheses

Linear Hypotheses:
       Estimate
1 == 0   0.6686
2 == 0   1.0153
3 == 0  -0.1420
```

This allows to apply the methods available on the `glht` object like

```
R> summary(g,test=univariate())

         Simultaneous Tests for General Linear Hypotheses

Fit: lm(formula = y ~ A + B + C, data = dd)

Linear Hypotheses:
       Estimate Std. Error t value Pr(>|t|)
1 == 0   0.6686     0.6072   1.101    0.292
2 == 0   1.0153     0.6072   1.672    0.120
3 == 0  -0.1420     0.6072  -0.234    0.819
(Univariate p values reported)

R> confint(g,calpha=univariate_calpha())

         Simultaneous Confidence Intervals

Fit: lm(formula = y ~ A + B + C, data = dd)

Quantile = 2.1788
95% confidence level


Linear Hypotheses:
       Estimate lwr     upr
1 == 0  0.6686  -0.6543  1.9915
2 == 0  1.0153  -0.3076  2.3383
3 == 0 -0.1420  -1.4649  1.1809
```

which yield the same results as the `esticon()` function.

By default the functions will adjust the tests and confidence intervals for multiplicity

```
R> summary(g)

         Simultaneous Tests for General Linear Hypotheses

Fit: lm(formula = y ~ A + B + C, data = dd)

Linear Hypotheses:
       Estimate Std. Error t value Pr(>|t|)
1 == 0   0.6686     0.6072   1.101    0.612
2 == 0   1.0153     0.6072   1.672    0.295
3 == 0  -0.1420     0.6072  -0.234    0.993
(Adjusted p values reported -- single-step method)

R> confint(g)

         Simultaneous Confidence Intervals

Fit: lm(formula = y ~ A + B + C, data = dd)

Quantile = 2.7326
95% family-wise confidence level


Linear Hypotheses:
       Estimate lwr     upr
1 == 0  0.6686  -0.9906  2.3278
2 == 0  1.0153  -0.6439  2.6745
3 == 0 -0.1420  -1.8012  1.5172
```