# Population means (also called marginal means or LSMEANS), contrasts and estimable functions in the `doBy` package

Søren Højsgaard and Ulrich Halekoh

March 19, 2013

## Contents

## 1 Introduction

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

## 2 A simulated dataset

Consider these data:

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

   A B C           y           x          z
1  1 1 1 -1.35475575 0.467707669 -0.5429324
2  2 1 1 -1.94635511 0.011929191  0.2481751
3  3 1 1  0.24676731 0.623862930 -0.4094545
4  1 2 1 -0.30311159 0.565359711  0.6733511
5  2 2 1  0.06015903 7.059121129  0.8169155
6  3 2 1  0.29786608 0.172023215  1.6289320
7  1 3 1  1.54779038 0.214894614  1.3154760
8  2 3 1 -0.66183527 5.090941228 -0.9315802
9  3 3 1  2.24878538 1.536907918 -1.1327850
10 1 1 2 -0.75479109 0.002802541  1.4390123
```

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:

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

(Intercept)          A2          A3          B2          B3          C2
 -0.7879238  -0.6825508   0.8873026   0.8121865   1.3919367   0.3645339
```

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

## 3 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.

The effect of changing the factor $A$ from `A2` to `A3` can be found as

```
> w <- c(0,-1,1,0,0,0)
> sum(coef(mm)*w)

[1] 1.569853
```

The `esticon()` function provides this estimate, the standard error etc. as follows:

```
> esticon(mm, w)

  beta0 Estimate Std.Error  t.value DF  Pr(>|t|)      Lower    Upper
1     0 1.569853 0.6957783 2.256255 12 0.0435038 0.05388279 3.085824
```

## 4 Population means

Population means (sometimes also called marginal means) are in some sciences much used for reporting marginal effects (to be described below). Population means are known as lsmeans in SAS jargon. Population means is a special kind of contrasts as defined in Section 3.

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 $\mathbb{E}(y|A)$.

### 4.1 A brute–force calculation

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:

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

(Intercept)          A2          A3          B2          B3          C2 
 -0.7879238   0.0000000   0.0000000   0.2707288   0.4639789   0.1822669 

> sum(coef(mm)*w)

[1] 0.1290509
```

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

```
> 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)
> W

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

> W %*% coef(mm)

           [,1]
[1,]  0.1290509
[2,] -0.5535000
[3,]  1.0163534
```

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.

### 4.2 Using `esticon()`

Given that one has specified $W$, the `esticon()` function in the `doBy` package be used for the calculations above and the function also provides standard errors, confidence limits etc:

```
> esticon(mm, W)

  beta0   Estimate Std.Error    t.value DF  Pr(>|t|)       Lower     Upper
1     0  0.1290509 0.4919895  0.2623041 12 0.7975357 -0.94290226 1.2010040
```

```
2     0 -0.5535000 0.4919895 -1.1250239 12 0.2825783 -1.62545308 0.5184531
3     0  1.0163534 0.4919895  2.0658030 12 0.0611403 -0.05559967 2.0883066
```

## 5 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

```
> pma <- popMatrix(mm,effect='A')
> 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 `popMeans()` function is simply a wrapper around first a call to `popMatrix()` followed by a call to (by default) `esticon()`:

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

  beta0   Estimate Std.Error    t.value DF  Pr(>|t|)       Lower     Upper A
1     0  0.1290509 0.4919895  0.2623041 12 0.7975357 -0.94290226 1.2010040 1
2     0 -0.5535000 0.4919895 -1.1250239 12 0.2825783 -1.62545308 0.5184531 2
3     0  1.0163534 0.4919895  2.0658030 12 0.0611403 -0.05559967 2.0883066 3
```

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

```
> summary(pme)
```

```
  beta0   Estimate Std.Error    t.value DF  Pr(>|t|)       Lower     Upper A
1     0  0.1290509 0.4919895  0.2623041 12 0.7975357 -0.94290226 1.2010040 1
2     0 -0.5535000 0.4919895 -1.1250239 12 0.2825783 -1.62545308 0.5184531 2
3     0  1.0163534 0.4919895  2.0658030 12 0.0611403 -0.05559967 2.0883066 3
Call:
NULL
Contrast matrix:
Length  Class   Mode
     0   NULL   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.

Likewise we may do:

```
> 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`. Consequently

```
> popMeans(mm)

  beta0  Estimate Std.Error   t.value DF  Pr(>|t|)     Lower     Upper
1     0 0.1973014 0.2840503 0.6946004 12 0.5005336 -0.421591 0.8161939
```

gives the "total average".

## 5.1 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:

```
> 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
```

Notice here that average is only taken over $B$. Another way of creating the population means at all levels of $(A, C)$ is therefore

```
> 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:

```
> 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
```

## 5.2 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

```
> 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

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

## 5.3 Using covariates

Next consider the model where a covariate is included:

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

(Intercept)          A2          A3          B2          B3          C2
-0.76250736 -0.79528804  1.05006859  0.74361773  0.78521887  0.09011896
       C1:x        C2:x
 0.10462121  0.53540908
```

In this case we get

```
> 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 1.301764    0
[2,]           1  1  0 0.3333333 0.3333333  0 1.301764    0
[3,]           1  0  1 0.3333333 0.3333333  0 1.301764    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:

```
> 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
```

### 5.4 Using transformed covariates

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

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

(Intercept)          A2          A3          B2          B3          C2
-0.59135782 -0.71082211  0.85508250  0.70605884  1.19987094  0.35549300
  C1:log(x)   C2:log(x)
 0.12561952  0.03272904
```

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

```
> dd <- transform(dd, log.x = log(x))
> mm3 <- lm(y~A+B+C+C:log.x, data=dd)
> popMatrix(mm3,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 -1.344774        0
[2,]           1  1  0 0.3333333 0.3333333  0 -1.344774        0
[3,]           1  0  1 0.3333333 0.3333333  0 -1.344774        0
```

## 6 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()`:

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

         General Linear Hypotheses

Linear Hypotheses:
       Estimate
1 == 0 -0.05322
2 == 0 -0.73577
3 == 0  0.83409
```

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

```
> 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 -0.05322    0.56810  -0.094    0.927
2 == 0 -0.73577    0.56810  -1.295    0.220
3 == 0  0.83409    0.56810   1.468    0.168
(Univariate p values reported)

> 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 -0.05322 -1.29100  1.18457
2 == 0 -0.73577 -1.97355  0.50202
3 == 0  0.83409 -0.40370  2.07187
```

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

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

```
> 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 -0.05322    0.56810  -0.094    1.000
2 == 0 -0.73577    0.56810  -1.295    0.492
3 == 0  0.83409    0.56810   1.468    0.394
(Adjusted p values reported -- single-step method)

> confint(g)

         Simultaneous Confidence Intervals

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

Quantile = 2.7322
95% family-wise confidence level


Linear Hypotheses:
       Estimate lwr      upr
1 == 0 -0.05322 -1.60536  1.49893
2 == 0 -0.73577 -2.28791  0.81637
3 == 0  0.83409 -0.71806  2.38623
```