Linear estimates and LS–means in the `doBy` package

Søren Højsgaard and Ulrich Halekoh

doBy version 4.6.12 as of 2022-02-06

Contents

1 Introduction

1.1 Linear functions of parameters

A linear function of a p–dimensional parameter vector β has the form

$$C = L\beta$$

where L is a $q \times p$ matrix which we call the *Linear Estimate Matrix* of simply *LE-matrix*. The corresponding linear estimate is $\hat{C} = L\hat{\beta}$. A *linear hypothesis* has the form $H_0 : L\beta = m$ for some q dimensional vector m.

1.2 LSmeans (population means, marginal means)

A special type of linear estimates is the so called *least-squares means* (or *LS-means*). Other names for these estimates include *population means* and *marginal means*. Consider an imaginary field experiment analyzed with model of the form

```
> lm( y ~ treat + block + year)
```

where `treat` is a treatment factor, `block` is a blocking factor and `year` is the year (a factor) where the experiment is repeated over several years. This model specifies the conditional mean $\mathbb{E}(Y|\texttt{treat}, \texttt{block}, \texttt{year})$. One may be interested in predictions of the form $\mathbb{E}(Y|\texttt{treat})$. This quantity can not formally be found from the model. However, it is tempting to average the fitted values of $\mathbb{E}(Y|\texttt{treat}, \texttt{block}, \texttt{year})$ across the levels of `block` and `year` and think of this average as $\mathbb{E}(Y|\texttt{treat})$. This average is precisely what is called the LS–means. If the experiment is balanced then this average is identical to the average of the observations when stratified according to `treat`.

An alternative is to think of `block` and `year` as random effects, for example:

```
> library(lme4)
> lmer( y ~ treat + (1|block) + (1|year))
```

In this case one would directly obtain $\mathbb{E}(Y|\texttt{treat})$ from the model. However, there are at least two reasons why one may be hesitant to consider such a random effects model.

- Suppose there are three blocks and the experiment is repeated over three consecutive years. This means that the random effects are likely to be estimated with a large uncertainty (the estimates will have only two degrees of freedom).
- Furthermore, treating `block` and `year` as random effects means they should in principle come from a large population of possible blocks and years. This may or may not be reasonable for the blocks, but it is a questionable assumption for the years.

1.3 A simulated dataset

In the following sections we consider these data:

```
> library(doBy)
> set.seed(141164)
> 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  0.73516 1.26234 -0.8019
## 2  2 1 1  0.18606 0.02870 -0.6705
## 3  3 1 1 -0.10532 1.09523 -1.3990
## 4  1 2 1 -0.31093 0.02323 -1.2826
## 5  2 2 1 -0.96994 2.72459  1.3790
## 6  3 2 1  0.59921 1.09027  0.7911
## 7  1 3 1 -0.02469 0.35388 -0.6327
## 8  2 3 1  0.38602 0.95420  0.6125
## 9  3 3 1  0.83019 2.33996  0.8216
```

```
## 10 1 1 2 -1.19164 0.06174  0.8128
```

Consider the additive model

$$y_i = \beta_0 + \beta^1_{A(i)} + \beta^2_{B(i)} + \beta^3_{C(i)} + \beta^4 x_i + e_i \tag{1}$$

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

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

## (Intercept)          A2          A3          B2          B3          C2           x
##     0.05759    -0.47363    -0.38526    -0.17882     0.27992    -0.37859     0.31206
```

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

1.4 LSmeans: details

LSmeans, population means and marginal means are used synonymously in the literature. These quantities are a special kind of contrasts as defined in Section 1.1. 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 $\mathbb{E}(y|A, B, C, x)$. Sometimes one is interested in quantities like $\mathbb{E}(y|A)$. This quantity can not formally be found because of the presences of unless B, C and x. However, suppose that A is a treatment of main interest, B is a blocking factor, C represents days on which the experiment was carried out and x denotes e.g. temperature for the specific block on the specific date. Then it is tempting to fix x at its average value $\bar{x}$ and average $\mathbb{E}(y|A, B, C, \bar{x})$ 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

$$\beta^0 + \frac{1}{3}(\beta^2_{B2} + \beta^2_{B3}) + \frac{1}{2}(\beta^3_{C2}) + \beta^4 \bar{x}. \tag{3}$$

This quantity can be estimated as (if $\bar{x} = 1.242$) :

```
> w <- c(1, 0, 0, 1/3, 1/3, 1/2, 1.242)
> sum(coef(mm) * w)

## [1] 0.2896
```

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

```
> L <- matrix(c(1, 0, 0, 1/3, 1/3, 1/2, 1.242,
                1, 1, 0, 1/3, 1/3, 1/2, 1.242,
                1, 0, 1, 1/3, 1/3, 1/2, 1.242), nr=3, byrow=TRUE)
> L %*% coef(mm)
```

```
##          [,1]
## [1,]  0.28958
## [2,] -0.18405
## [3,] -0.09568
```

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.

1.5 Using LEmatrix and LSmeans

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 LEmatrix function provides an automated way of generating such matrices. The above `W` matrix is constructed by

```
> L <- LE_matrix(mm, effect='A')
> L

##      (Intercept) A2 A3     B2     B3  C2     x
## [1,]           1  0  0 0.3333 0.3333 0.5 1.242
## [2,]           1  1  0 0.3333 0.3333 0.5 1.242
## [3,]           1  0  1 0.3333 0.3333 0.5 1.242
```

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 LSmeans function is simply a wrapper around first a call to LEmatrix followed by a call to (by default) `linest()`:

```
> linest(mm, L=L)

##      estimate std.error statistic      df p.value
## [1,]   0.2895    0.3182    0.9099 11.0000    0.38
## [2,]  -0.1841    0.3195   -0.5763 11.0000    0.58
## [3,]  -0.0957    0.3188   -0.3003 11.0000    0.77

> LSM <- LSmeans(mm, effect='A')
> LSM

##      estimate std.error statistic      df p.value
## [1,]   0.2895    0.3182    0.9099 11.0000    0.38
## [2,]  -0.1841    0.3195   -0.5763 11.0000    0.58
## [3,]  -0.0957    0.3188   -0.3003 11.0000    0.77
```

1.6 Special details

1.6.1 Summary

More details about how the matrix was constructed is provided by the `summary()` method; for example:

```
> summary(LSM)

## Coefficients:
##      estimate std.error statistic      df p.value
```

```
## [1,]   0.2895    0.3182    0.9099 11.0000    0.38
## [2,]  -0.1841    0.3195   -0.5763 11.0000    0.58
## [3,]  -0.0957    0.3188   -0.3003 11.0000    0.77
##
## Grid:
##   A     x
## 1 1 1.242
## 2 2 1.242
## 3 3 1.242
##
## L:
##      (Intercept) A2 A3     B2     B3  C2     x
## [1,]           1  0  0 0.3333 0.3333 0.5 1.242
## [2,]           1  1  0 0.3333 0.3333 0.5 1.242
## [3,]           1  0  1 0.3333 0.3333 0.5 1.242
```

1.6.2 Specification of `effect`

The `effect` argument can be specified in two ways:

```
> ## 1) a vector of characters and
> LE_matrix(mm, effect= c("A", "C"))

##      (Intercept) A2 A3     B2     B3 C2     x
## [1,]           1  0  0 0.3333 0.3333  0 1.242
## [2,]           1  1  0 0.3333 0.3333  0 1.242
## [3,]           1  0  1 0.3333 0.3333  0 1.242
## [4,]           1  0  0 0.3333 0.3333  1 1.242
## [5,]           1  1  0 0.3333 0.3333  1 1.242
## [6,]           1  0  1 0.3333 0.3333  1 1.242

> ## 2) a right hand sided formula:
> ## LE_matrix(mm, effect= ~ A + C)
```

1.6.3 Fixing values in the `at` list

We can fix values of the explanatory variables using the `at=` argument: For example:

```
> LE_matrix(mm, at=list(A=1, x=2))

##      (Intercept) A2 A3     B2     B3  C2 x
## [1,]           1  0  0 0.3333 0.3333 0.5 2
```

1.6.4 Combining `effect` and `at`

```
> LE_matrix(mm, effect= ~ A + C, at=list(A=1))

##      (Intercept) A2 A3     B2     B3 C2     x
## [1,]           1  0  0 0.3333 0.3333  0 1.242
## [2,]           1  0  0 0.3333 0.3333  1 1.242
```

1.6.5 Total average

Omitting `effect` as in

```
> LE_matrix(mm)

##      (Intercept)     A2     A3     B2     B3  C2     x
## [1,]           1 0.3333 0.3333 0.3333 0.3333 0.5 1.242

> LSmeans(mm)

##      estimate std.error statistic       df p.value
## [1,]  0.00321   0.18365   0.01751 11.00000    0.99
```

gives the "total average".

1.6.6 Log transformed covariates

Unless otherwise specified, numerical covariates are fixed at their average value, for example for `x` below:

If we use `log(x)` instead we will get an error when calculating LS–means. Instead one must modify data:

```
> mm2 <- lm(y ~ A + B + C + log(x), data=dd)
> ## LSmeans(mm2, effect="A) ## Will fail
> mm2 <- lm(y ~ A + B + C + log.x,
            data=transform(dd, log.x=log(x)))
> LSmeans(mm2, effect=~A)

##      estimate std.error statistic     df p.value
## [1,]    0.315     0.372     0.846 11.000    0.42
## [2,]   -0.254     0.373    -0.682 11.000    0.51
## [3,]   -0.051     0.371    -0.138 11.000    0.89

> LSmeans(mm2, effect=~A)$L

##      (Intercept) A2 A3     B2     B3  C2  log.x
## [1,]           1  0  0 0.3333 0.3333 0.5 -0.993
## [2,]           1  1  0 0.3333 0.3333 0.5 -0.993
## [3,]           1  0  1 0.3333 0.3333 0.5 -0.993
```

This also highlights what is computed: The average of the log of `x`; not the log of the average of `x`.

1.6.7 Powers of covariates

Similar phenomenon for powers of covariates:

```
> mm2 <- lm(y ~ A + B + C + x + I(x^2), data=dd)
> LSmeans(mm2, effect=~A)

##      estimate std.error statistic     df p.value
## [1,]   -0.331     0.430    -0.770 10.000    0.46
## [2,]   -0.396     0.306    -1.292 10.000    0.23
## [3,]   -0.137     0.286    -0.477 10.000    0.64

> LSmeans(mm2, effect=~A)$L
```

```
##      (Intercept) A2 A3     B2     B3  C2     x I(x^2)
## [1,]           1  0  0 0.3333 0.3333 0.5 1.242  1.542
## [2,]           1  1  0 0.3333 0.3333 0.5 1.242  1.542
## [3,]           1  0  1 0.3333 0.3333 0.5 1.242  1.542
```

Above `I(x^2)` is the average of the squared values of `x`; not the square of the average of `x`, cfr. the following.

```
> mm2 <- lm(y ~ A + B + C + x + x2,
            data=transform(dd, x2=x^2))
> LSmeans(mm2, effect=~A)

##      estimate std.error statistic      df p.value
## [1,]  -0.0398    0.3321   -0.1200 10.0000    0.91
## [2,]  -0.1049    0.2890   -0.3631 10.0000    0.72
## [3,]   0.1544    0.3135    0.4925 10.0000    0.63

> LSmeans(mm2, effect=~A)$L

##      (Intercept) A2 A3     B2     B3  C2     x    x2
## [1,]           1  0  0 0.3333 0.3333 0.5 1.242 3.596
## [2,]           1  1  0 0.3333 0.3333 0.5 1.242 3.596
## [3,]           1  0  1 0.3333 0.3333 0.5 1.242 3.596
```

2 Miscellaneous

2.1 Excerpt of the CO2 data

For illustration we use subsets of the CO2 data:

```
> data(CO2)
> CO2 <- subset(CO2, conc < 300) ## OK
> CO2.bal <- CO2
> rownames(CO2.bal) <- NULL
```

Data is perfectly balanced. We also use an unbalanced subset of the data

```
> CO2.ubal <- CO2.bal[-c(1, 5, 12, 17, 18, 19, 20, 28),]
> CO2.ubal %>% head

##   Plant   Type  Treatment conc uptake
## 2   Qn1 Quebec nonchilled  175   30.4
## 3   Qn1 Quebec nonchilled  250   34.8
## 4   Qn2 Quebec nonchilled   95   13.6
## 6   Qn2 Quebec nonchilled  250   37.1
## 7   Qn3 Quebec nonchilled   95   16.2
## 8   Qn3 Quebec nonchilled  175   32.4

> xtabs(~Type + Treatment + conc, data=CO2.ubal) %>%
      ftable(row.vars = "Type")

##             Treatment nonchilled         chilled
##             conc              95 175 250      95 175 250
## Type
## Quebec                         2   2   3       3   2   1
## Mississippi                    2   2   3       2   3   3
```

2.2 Two linear models for CO2 data

```
> library(broom)
> form.add <- uptake ~ conc + Treatment + Type
> form.int <- uptake ~ conc * Treatment + Type
> fm1.bal <- lm(form.add, data=CO2.bal)
> fm1.ubal <- lm(form.add, data=CO2.ubal)
```

2.3 Linear estimates and LSmeans

Common task 1: Consider this task: Estimate the value of the response uptake for each combination of Type and Treatment . This can be obtained, for example, as follows:

```
> LSmeans(fm1.bal, effect=~Type+Treatment, at=list(conc=100))

##      estimate std.error statistic    df p.value
## [1,]    20.30      1.30     15.56 32.00       0
## [2,]    11.19      1.30      8.58 32.00       0
## [3,]    15.33      1.30     11.75 32.00       0
## [4,]     6.22      1.30      4.77 32.00       0

> LSmeans(fm1.ubal, effect=~Type+Treatment, at=list(conc=100))

##      estimate std.error statistic    df p.value
## [1,]    21.11      1.53     13.79 24.00       0
## [2,]    11.68      1.66      7.05 24.00       0
## [3,]    15.19      1.48     10.26 24.00       0
## [4,]     5.76      1.48      3.88 24.00       0
```

Common task 2: Estimate the value of the response uptake when for each level of Treatment. Formally this question can not be answered under the model because the model gives the conditional mean of uptake given conc, Type and Treatment. There is, so to speak, no way of getting rid of Type. There are different workarounds for this situation: We can average over Type and fix conc at its average. Alternatively, we can fit model without Type effect and repeat the steps above.

```
> LSmeans(fm1.bal, effect=~Treatment)

##      estimate std.error statistic     df p.value
## [1,]   23.622     0.885    26.682 32.000       0
## [2,]   18.656     0.885    21.072 32.000       0

> LSmeans(fm1.ubal, effect=~Treatment)

##      estimate std.error statistic    df p.value
## [1,]    24.30      1.03     23.60 24.00       0
## [2,]    18.38      1.04     17.73 24.00       0
```

```
> ## Fit model without Type
> fm2.bal <- update(fm1.bal, .~. - Type)
> fm2.ubal <- update(fm1.ubal, .~. - Type)
>
> LSmeans(fm2.bal, effect=~Treatment)

##      estimate std.error statistic    df p.value
## [1,]    23.62      1.42     16.63 33.00       0
```

```
## [2,]    18.66      1.42     13.13 33.00       0
```

```
> LSmeans(fm2.ubal, effect=~Treatment)
```

```
##      estimate std.error statistic    df p.value
## [1,]    24.38      1.66     14.68 25.00       0
## [2,]    17.62      1.66     10.61 25.00       0
```

Notice that the predictions (estimates) are identical for the balanced dataset but not for the unbalanced dataset. Notice also that the standard errors of the predictions are somewhat larger when fitting a model without `Type` effect. This is to be expected as the variation due to `Type` goes into the residual.

2.4 Generalized linear models

We can calculate LS–means for e.g. a Poisson or a gamma model. Default is that the calculation is calculated on the scale of the linear predictor.

```
> fm.glm <- glm(form.add, family=Gamma, data=CO2.ubal)
> LSmeans(fm.glm, effect=~Treatment, type="link")
```

```
##      estimate std.error statistic p.value
## [1,]  0.04879   0.00287  17.01566       0
## [2,]  0.05993   0.00343  17.46004       0
```

2.5 Generalized estimating equations

For gee-type models we get

```
> library(geepack)
> fm.gee <- geeglm(uptake ~ conc + Treatment + Type,
                   id=Plant, family=Gamma, data=CO2.ubal)
> LSmeans(fm.gee, effect=~Treatment)
```

```
##      estimate std.error statistic p.value
## [1,]  0.04879   0.00228  21.39458       0
## [2,]  0.05993   0.00382  15.69861       0
```

2.6 Linear mixed effects model

For illustration we treat `Plant` as a random effect:

```
> library(lme4)
```

```
## Loading required package:  Matrix
```

```
> fm.mix <- lmer(uptake ~ conc + Treatment + Type + (1|Plant), data=CO2.ubal)
```

```
## boundary (singular) fit:  see ?isSingular
```

```
> LSmeans(fm.mix, effect=~Treatment)
```

```
##      estimate std.error statistic    df p.value
## [1,]    24.30      1.05     23.04  7.37       0
## [2,]    18.38      1.06     17.30  7.62       0
```

Notice that the degrees of freedom by default are adjusted using a Kenward–Roger approximation. Unadjusted degrees of freedom are obtained by setting `adjust.df=FALSE`.

2.7 Pairwise comparisons

We will just mention that for certain other linear estimates, the matrix L can be generated automatically using `glht()` from the **multcomp** package. For example, pairwise comparisons of all levels of `dose` can be obtained with

```
> library("multcomp")

## Loading required package:  mvtnorm

## Loading required package:  survival

## Loading required package:  TH.data

## Loading required package:  MASS

##
## Attaching package:  'TH.data'

## The following object is masked from 'package:MASS':
##
##     geyser

> g1 <- glht(fm2.ubal, mcp(Treatment="Tukey"))
> tidy(g1)

## # A tibble: 1 x 7
##   term      contrast             null.value estimate std.error statistic adj.p.value
##   <chr>     <chr>                     <dbl>    <dbl>     <dbl>     <dbl>       <dbl>
## 1 Treatment chilled - nonchilled          0    -6.76      2.36     -2.87     0.00829
```

The L matrix is

```
> L <- g1$linfct
> L

##                       (Intercept) conc Treatmentchilled
## chilled - nonchilled           0    0                1
## attr(,"type")
## [1] "Tukey"

> linest(fm2.ubal, L=L)

##                       estimate std.error statistic    df p.value
## chilled - nonchilled    -6.76      2.36     -2.87 25.00    0.01
```