

LS-means (least-squares means) and other linear estimates

Søren Højsgaard and Ulrich Halekoh

doBy version 4.5-15 as of 2016-03-31

Contents

1	Introduction	1
1.1	Linear functions of parameters, contrasts	1
1.2	Least-squares means (LS-means)	2
2	LS-means for linear models	2
2.1	LS-means – a first example	2
2.2	Example: Warpbreaks	3
2.3	The LS-means	5
2.4	LS-means for models with interactions	5
3	Using the at= argument	6
4	Using (transformed) covariates	7
5	Alternative models	9
5.1	Generalized linear models	9
5.2	Linear mixed effects model	10
5.3	Generalized estimating equations	11
6	Miscellaneous	11
6.1	Under the hood	11
6.2	Example: Non-estimable contrasts	11
6.3	Handling non-estimability	12
6.4	Pairwise comparisons	14

1 Introduction

1.1 Linear functions of parameters, contrasts

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

$$C = K\beta$$

where K is a $q \times p$ matrix. The corresponding linear estimate is $\hat{C} = K\hat{\beta}$. A linear hypothesis has the form $H_0 : K\beta = m$ for some q dimensional vector m .

1.2 Least-squares means (LS-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 $\mathbb{E}(Y|\text{treat}, \text{block}, \text{year})$. One may be interested in predictions of the form $\mathbb{E}(Y|\text{treat})$. This quantity can not formally be found from the model. However, it is tempting to average the fitted values of $\mathbb{E}(Y|\text{treat}, \text{block}, \text{year})$ across the levels of **block** and **year** and think of this average as $\mathbb{E}(Y|\text{treat})$. 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 $\mathbb{E}(Y|\text{treat})$ 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 certainly a dubious assumption for the years.

Below we describe LSmeans as implemented in the **doBy** package. Notice that the **lsmeans** package Lenth (2013) also provides computations of LS-means, see <http://cran.r-project.org/web/packages/lsmeans/>.

2 LS-means for linear models

2.1 LS-means – a first example

Consider these simulated data

```
> simdat
  treat year  y
1   t1    1 0.5
2   t1    1 1.0
3   t1    1 1.5
```

```

4   t2   1 3.0
5   t1   2 3.0
6   t2   2 4.5
7   t2   2 5.0
8   t2   2 5.5

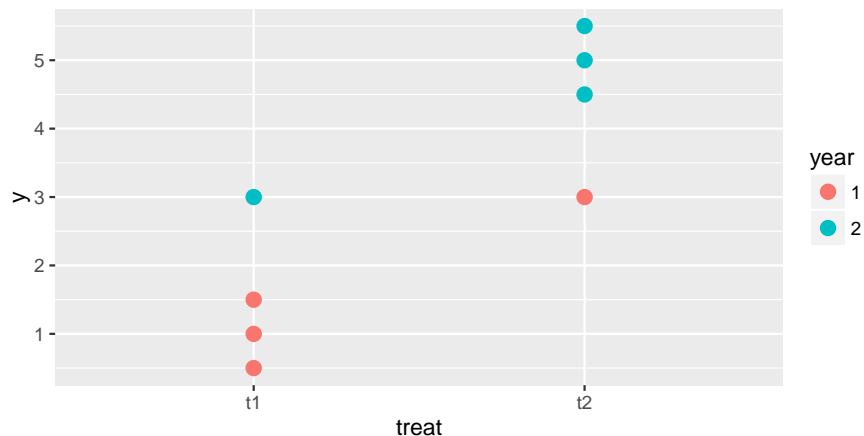
```

shown in the figure below.

```

> library(ggplot2)
> qplot(treat, y, data=simdat, color=year, size=I(3))

```



The LS-means under an additive model for the factor `treat` is

```

> msim <- lm(y ~ treat + year, data=simdat)
> LSmeans( msim, effect="treat")

```

	estimate	se	df	t.stat	p.value	treat
1	2	0.2415	5	8.281	4.192e-04	t1
2	4	0.2415	5	16.562	1.465e-05	t2

whereas the population means are

```

> summaryBy(y~treat, data=simdat)

```

treat	y.mean
1 t1	1.5
2 t2	4.5

Had data been balanced (same number of observations for each combination of `treat` and `year`) the results would have been the same. An argument in favor of the LS-means is that these figures better represent what one would expect on in an “average year”.

2.2 Example: Warpbreaks

```

> summary( warpbreaks )

```

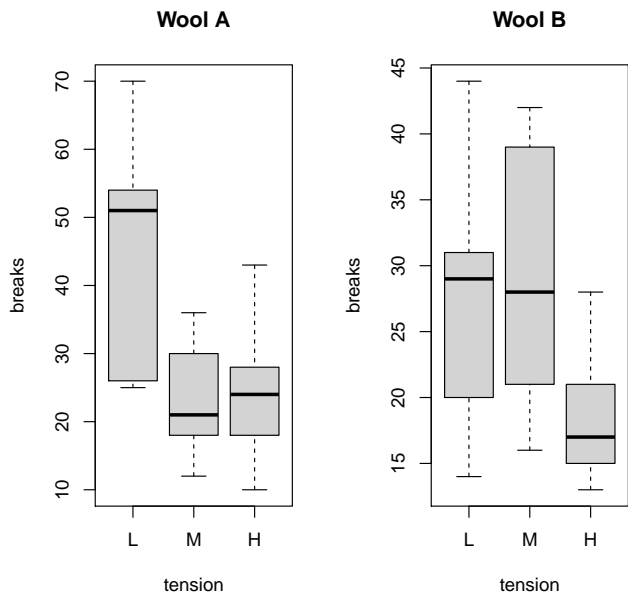
breaks	wool	tension
Min. :10.0	A:27	L:18
1st Qu.:18.2	B:27	M:18
Median :26.0		H:18
Mean :28.1		
3rd Qu.:34.0		
Max. :70.0		

```
> head( warpbreaks, 4 )
  breaks wool tension
1     26   A      L
2     30   A      L
3     54   A      L
4     25   A      L

> ftable(xtabs( ~ wool + tension, data=warpbreaks))
```

```
      tension L M H
wool
A           9 9 9
B           9 9 9
```

warpbreaks data



```
> (warp.lm <- lm(breaks ~ wool + tension, data=warpbreaks))
```

```
Call:
lm(formula = breaks ~ wool + tension, data = warpbreaks)
```

```
Coefficients:
(Intercept)      woolB      tensionM      tensionH
    39.28         -5.78        -10.00        -14.72
```

The fitted values are:

```
> uni <- unique(warpbreaks[,2:3])
> prd <- cbind(breaks=predict(warp.lm, newdata=uni), uni); prd
```

```
  breaks wool tension
1  39.28   A      L
10 29.28   A      M
19 24.56   A      H
28 33.50   B      L
37 23.50   B      M
46 18.78   B      H
```

2.3 The LS-means

We may be interested in making predictions of the number of breaks for each level of `tension` for *any* type or an *average* type of `wool`. The idea behind LS-means is to average the predictions above over the two wool types. These quantities are the LSmeans for the effect `tension`.

This is done with:

```
> LSmeans(warp.lm, effect="tension")
  estimate    se df t.stat  p.value tension
1    36.39 2.738 50 13.289 4.948e-18      L
2    26.39 2.738 50  9.637 5.489e-13      M
3    21.67 2.738 50  7.913 2.269e-10      H
```

The term LSmeans comes from that these quantities are the same as the least squares main effects of `tension` when data is balanced:

```
> doBy::summaryBy(breaks ~ tension, data=warpbreaks)
  tension breaks.mean
1      L          36.39
2      M          26.39
3      H          21.67
```

When data is not balanced these quantities are in general not the same.

2.4 LS-means for models with interactions

Consider a model with interaction:

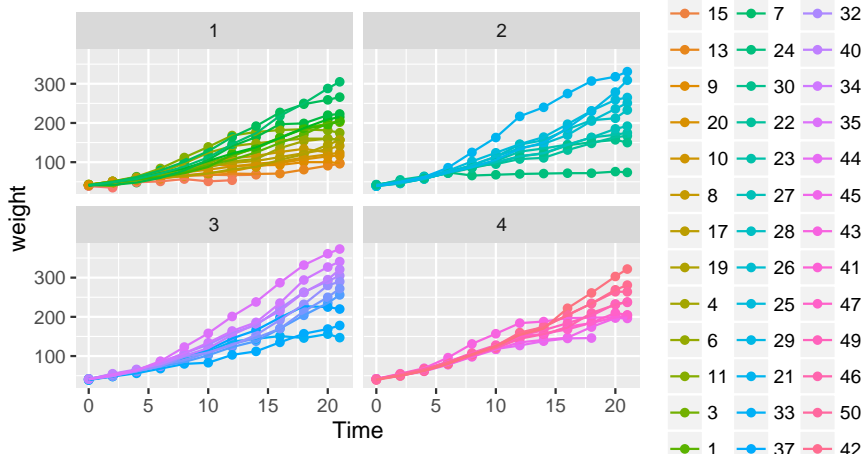
```
> warp.lm2 <- update(warp.lm, .~.+wool:tension)
> coef( summary( warp.lm2 ))
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    44.56      3.647  12.218 2.426e-16
woolB          -16.33      5.157  -3.167 2.677e-03
tensionM       -20.56      5.157  -3.986 2.281e-04
tensionH       -20.00      5.157  -3.878 3.199e-04
woolB:tensionM  21.11      7.294   2.895 5.698e-03
woolB:tensionH  10.56      7.294   1.447 1.543e-01
```

In this case the contrast matrix becomes:

```
> K2 <- LSmatrix(warp.lm2, effect="tension"); K2
  (Intercept) woolB tensionM tensionH woolB:tensionM woolB:tensionH
[1,]         1  0.5         0         0           0.0           0.0
[2,]         1  0.5         1         0           0.5           0.0
[3,]         1  0.5         0         1           0.0           0.5
> linest(warp.lm2, K=K2)
  estimate    se df t.stat  p.value tension
1    36.39 2.579 48 14.112 1.055e-18      L
2    26.39 2.579 48 10.234 1.183e-13      M
3    21.67 2.579 48  8.402 5.468e-11      H
```

3 Using the at= argument

```
> library(ggplot2)
> ChickWeight$Diet <- factor(ChickWeight$Diet)
> qplot(Time, weight, data=ChickWeight, colour=Chick, facets=~Diet,
        geom=c("point", "line"))
```



Consider random regression model:

```
> library(lme4)
> rr <- lmer(weight~Time*Diet + (0+Time|Chick), data=ChickWeight)
> coef(summary(rr))
```

	Estimate	Std. Error	t value
(Intercept)	33.218	1.7697	18.7701
Time	6.339	0.6103	10.3855
Diet2	-4.585	3.0047	-1.5258
Diet3	-14.968	3.0047	-4.9815
Diet4	-1.454	3.0177	-0.4818
Time:Diet2	2.271	1.0367	2.1902
Time:Diet3	5.084	1.0367	4.9043
Time:Diet4	3.217	1.0377	3.1004

The contrast matrix for Diet becomes:

```
> LSmatrix(rr, effect="Diet")
```

	(Intercept)	Time	Diet2	Diet3	Diet4	Time:Diet2	Time:Diet3	Time:Diet4
[1,]	1	10.72	0	0	0	0.00	0.00	0.00
[2,]	1	10.72	1	0	0	10.72	0.00	0.00
[3,]	1	10.72	0	1	0	0.00	10.72	0.00
[4,]	1	10.72	0	0	1	0.00	0.00	10.72

The value of Time is by default taken to be the average of that variable. Hence the LSmeans is the predicted weight for each diet at that specific point of time. We can consider other points of time with

```
> K1 <- LSmatrix(rr, effect="Diet", at=list(Time=1)); K1
```

	(Intercept)	Time	Diet2	Diet3	Diet4	Time:Diet2	Time:Diet3	Time:Diet4
[1,]	1	1	0	0	0	0	0	0
[2,]	1	1	1	0	0	1	0	0
[3,]	1	1	0	1	0	0	1	0
[4,]	1	1	0	0	1	0	0	1

The LSmeans for the intercepts is the predictions at Time=0. The LSmeans for the slopes becomes

```
> K0 <- LSmatrix(rr, effect="Diet", at=list(Time=0))
> K1-K0
      (Intercept) Time Diet2 Diet3 Diet4 Time:Diet2 Time:Diet3 Time:Diet4
[1,]           0    1    0    0    0           0           0           0
[2,]           0    1    0    0    0           1           0           0
[3,]           0    1    0    0    0           0           1           0
[4,]           0    1    0    0    0           0           0           1
```

```
> LSmeans(rr, K=K1-K0)
  estimate    se    df t.stat  p.value Diet Time
1   6.339 0.6105 49.86 10.38 4.632e-14  1  1
2   8.609 0.8380 48.28 10.27 9.705e-14  2  1
3  11.423 0.8380 48.28 13.63 3.588e-18  3  1
4   9.556 0.8392 48.56 11.39 2.584e-15  4  1
```

We can cook up our own function for comparing trends:

```
> LSmeans_trend <- function(object, effect, trend){
  K<-LSmatrix(object, effect=effect, at=as.list(setNames(1, trend))) -
    LSmatrix(object, effect=effect, at=as.list(setNames(0, trend)))
  LSmeans(object, K=K)
}
> LSmeans_trend(rr, effect="Diet", trend="Time")
  estimate    se    df t.stat  p.value Diet Time
1   6.339 0.6105 49.86 10.38 4.632e-14  1  1
2   8.609 0.8380 48.28 10.27 9.705e-14  2  1
3  11.423 0.8380 48.28 13.63 3.588e-18  3  1
4   9.556 0.8392 48.56 11.39 2.584e-15  4  1
```

4 Using (transformed) covariates

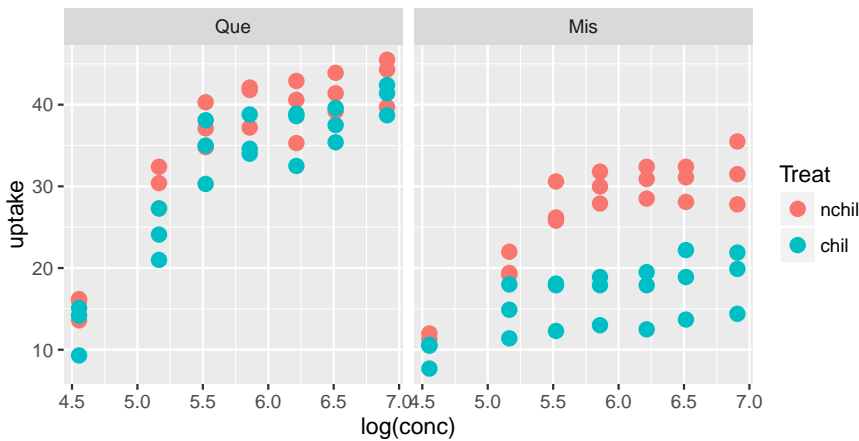
Consider the following subset of the CO2 dataset:

```
> data(CO2)
> CO2 <- transform(CO2, Treat=Treatment, Treatment=NULL)
> levels(CO2$Treat) <- c("nchil","chil")
> levels(CO2$Type) <- c("Que","Mis")
> ftable(xtabs( ~ Plant + Type + Treat, data=CO2), col.vars=2:3)
```

Plant	Type		Treat	
	Que	Mis	nchil	chil
Qn1	7	0	0	0
Qn2	7	0	0	0
Qn3	7	0	0	0
Qc1	0	7	0	0
Qc3	0	7	0	0
Qc2	0	7	0	0
Mn3	0	0	7	0
Mn2	0	0	7	0

```
Mn1      0    0    7    0
Mc2      0    0    0    7
Mc3      0    0    0    7
Mc1      0    0    0    7
```

```
> qplot(x=log(conc), y=uptake, data=C02, color=Treat, facets=~Type, size=I(3))
```



Below, the covariate `conc` is fixed at the average value:

```
> co2.lm1 <- lm(uptake ~ conc + Type + Treat, data=C02)
> LSmeans(co2.lm1, effect="Treat")
```

	estimate	se	df	t.stat	p.value	Treat	conc
1	30.64	0.9556	80	32.07	2.010e-47	nchil	435
2	23.78	0.9556	80	24.89	2.037e-39	chil	435

If we use `log(conc)` instead we will get an error when calculating LS-means:

```
> co2.lm <- lm(uptake ~ log(conc) + Type + Treat, data=C02)
> LSmeans(co2.lm, effect="Treat")
```

In this case one can do

```
> co2.lm2 <- lm(uptake ~ log.conc + Type + Treat,
               data=transform(C02, log.conc=log(conc)))
> LSmeans(co2.lm2, effect="Treat")
```

	estimate	se	df	t.stat	p.value	Treat	log.conc
1	30.64	0.7611	80	40.26	7.169e-55	nchil	5.819
2	23.78	0.7611	80	31.25	1.366e-46	chil	5.819

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

In a similar spirit consider

```
> co2.lm3 <- lm(uptake ~ conc + I(conc^2) + Type + Treat, data=C02)
> LSmeans(co2.lm3, effect="Treat")
```

	estimate	se	df	t.stat	p.value	Treat	conc	I(conc^2)
1	34.54	0.9816	79	35.19	4.926e-50	nchil	435	275754
2	27.68	0.9816	79	28.20	5.382e-43	chil	435	275754

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


```

> co2.lm4 <- lm(uptake ~ conc + conc2 + Type + Treat, data=
               transform(CO2, conc2=conc^2))
> LSmeans(co2.lm4, effect="Treat")

```

	estimate	se	df	t.stat	p.value	Treat	conc	conc2
1	30.64	0.7765	79	39.46	9.318e-54	nchil	435	275754
2	23.78	0.7765	79	30.63	1.356e-45	chil	435	275754

If we want to evaluate the LS-means at `conc=10` then we can do:

```

> LSmeans(co2.lm4, effect="Treat", at=list(conc=10, conc2=100))

```

	estimate	se	df	t.stat	p.value	Treat	conc	conc2
1	14.735	1.701	79	8.662	4.456e-13	nchil	10	100
2	7.876	1.701	79	4.630	1.417e-05	chil	10	100

5 Alternative models

5.1 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. However, if we think of LS-means as a prediction on the linear scale one may argue that it can also make sense to transform this prediction to the response scale:

```

> warp.poi <- glm(breaks ~ wool + tension, family=poisson, data=warpbreaks)
> LSmeans(warp.poi, effect="tension", type="link")

```

	estimate	se	z.stat	p.value	tension
1	3.589	0.03916	91.64	0	L
2	3.268	0.04596	71.10	0	M
3	3.070	0.05071	60.55	0	H

```

> LSmeans(warp.poi, effect="tension", type="response")

```

	estimate	se	z.stat	p.value	tension
1	36.20	1.418	91.64	0	L
2	26.25	1.206	71.10	0	M
3	21.55	1.093	60.55	0	H

```

> warp.qpoi <- glm(breaks ~ wool + tension, family=quasipoisson, data=warpbreaks)
> LSmeans(warp.qpoi, effect="tension", type="link")

```

	estimate	se	z.stat	p.value	tension
1	3.589	0.08085	44.39	0.000e+00	L
2	3.268	0.09488	34.44	6.093e-260	M
3	3.070	0.10467	29.33	3.883e-189	H

```

> LSmeans(warp.qpoi, effect="tension", type="response")

```

	estimate	se	z.stat	p.value	tension
1	36.20	2.926	44.39	0.000e+00	L
2	26.25	2.490	34.44	6.093e-260	M
3	21.55	2.256	29.33	3.883e-189	H

For comparison with the linear model, we use identity link

```

> warp.gam <- glm(breaks ~ wool + tension, family=Gamma(link=identity),
                 data=warpbreaks)
> LSmeans(warp.gam, effect="tension", type="link")

```

	estimate	se	df	t.stat	p.value	tension
1	35.66	3.222	50	11.07	4.766e-15	L
2	27.12	2.448	50	11.08	4.543e-15	M
3	21.53	1.944	50	11.08	4.629e-15	H

Notice that the linear estimates are practically the same as for the linear model, but the standard errors are smaller and hence the confidence intervals are narrower.

An alternative is to fit a quasi Poisson “model”

```

> warp.poi3 <- glm(breaks ~ wool + tension, family=quasipoisson(link=identity),
                  data=warpbreaks)
> LSmeans(warp.poi3, effect="tension")

```

	estimate	se	z.stat	p.value	tension
1	36.00	2.950	12.204	2.965e-34	L
2	26.83	2.544	10.546	5.316e-26	M
3	21.62	2.281	9.475	2.657e-21	H

5.2 Linear mixed effects model

For the sake of illustration we treat `wool` as a random effect:

```

> library(lme4)
> warp.mm <- lmer(breaks ~ tension + (1|wool), data=warpbreaks)
> LSmeans(warp.mm, effect="tension")

```

	estimate	se	df	t.stat	p.value	tension
1	36.39	3.653	2.538	9.961	0.004230	L
2	26.39	3.653	2.538	7.224	0.009354	M
3	21.67	3.653	2.538	5.931	0.015093	H

Notice here that the estimates themselves are very similar to those above but the standard errors are much larger. This comes from that there that `wool` is treated as a random effect.

```

> VarCorr(warp.mm)

```

Groups	Name	Std.Dev.
wool	(Intercept)	3.42
Residual		11.62

Notice that the degrees of freedom by default are adjusted using a Kenward–Roger approximation (provided that `pbkrtest` is installed). Unadjusted degrees of freedom are obtained with

```

> LSmeans(warp.mm, effect="tension", adjust.df=FALSE)

```

	estimate	se	df	t.stat	p.value	tension
1	36.39	3.653	49	9.961	2.288e-13	L
2	26.39	3.653	49	7.224	2.986e-09	M
3	21.67	3.653	49	5.931	2.986e-07	H

5.3 Generalized estimating equations

Lastly, for gee-type “models” we get

```
> library(geepack)
> warp.gee <- geeglm(breaks ~ tension, id=wool, family=poisson, data=warpbreaks)
> LSmeans(warp.gee, effect="tension")
  estimate      se z.stat    p.value tension
1    3.594 0.15869  22.65 1.427e-113      L
2    3.273 0.06401  51.13 0.000e+00      M
3    3.076 0.09428  32.62 1.903e-233      H

> LSmeans(warp.gee, effect="tension", type="response")
  estimate      se z.stat    p.value tension
1   36.39 5.775  22.65 1.427e-113      L
2   26.39 1.689  51.13 0.000e+00      M
3   21.67 2.043  32.62 1.903e-233      H
```

6 Miscellaneous

6.1 Under the hood

Under the hood, LSmeans() generates a contrast matrix

```
> K <- LSMatrix(warp.lm, effect="tension"); K
      (Intercept) woolB tensionM tensionH
[1,]           1  0.5         0         0
[2,]           1  0.5         1         0
[3,]           1  0.5         0         1
```

and passes this matrix onto linest():

```
> linest( warp.lm, K=K )
  estimate      se df t.stat    p.value tension
1   36.39 2.738 50 13.289 4.948e-18      L
2   26.39 2.738 50  9.637 5.489e-13      M
3   21.67 2.738 50  7.913 2.269e-10      H
```

6.2 Example: Non-estimable contrasts

Consider this highly unbalanced simulated dataset:

```
> head(dat.nst)
  AA BB CC      y
1  1  1  1 -0.3385
2  2  1  1 -1.2434
3  1  2  2  0.8397
4  2  2  2 -0.7839
5  1  3  2 -0.2641
6  2  3  2  0.3046
```

```
> ftable(xtabs( ~ AA + BB + CC, data=dat.nst))
      CC 1 2 3 4
AA BB
1  1    3 0 0 0
   2    0 1 1 1
   3    0 1 1 1
2  1    3 0 0 0
   2    0 1 1 1
   3    0 1 1 1
```

We have

```
> mod.nst <- lm(y ~ AA + BB : CC, data=dat.nst)
> coef( mod.nst )
(Intercept)      AA2      BB1:CC1      BB2:CC1      BB3:CC1      BB1:CC2
-0.74141      0.44935      0.03128           NA           NA           NA
BB2:CC2      BB3:CC2      BB1:CC3      BB2:CC3      BB3:CC3      BB1:CC4
0.54462      0.53702           NA      -0.41070      0.75253           NA
BB2:CC4      BB3:CC4
1.08095           NA
```

In this case some of the LSmeans values are not estimable (see Section 6.3 for details):

```
> LSmeans(mod.nst, effect=c("BB", "CC"))
  estimate      se df   t.stat p.value BB CC
1 -0.48545 0.3440 10 -1.41124 0.1885 1 1
2      NA      NA NA      NA      NA 2 1
3      NA      NA NA      NA      NA 3 1
4      NA      NA NA      NA      NA 1 2
5  0.02789 0.5958 10  0.04681 0.9636 2 2
6  0.02029 0.5958 10  0.03405 0.9735 3 2
7      NA      NA NA      NA      NA 1 3
8 -0.92743 0.5958 10 -1.55661 0.1506 2 3
9  0.23580 0.5958 10  0.39576 0.7006 3 3
10      NA      NA NA      NA      NA 1 4
11  0.56422 0.5958 10  0.94698 0.3660 2 4
12 -0.51673 0.5958 10 -0.86728 0.4061 3 4
```

6.3 Handling non-estimability

The model matrix for the model in Section 6.2 does not have full column rank and therefore not all values are calculated by LSmeans().

```
> X <- model.matrix( mod.nst ); as(X,"Matrix")
18 x 14 sparse Matrix of class "dgCMatrix"

1 1 . 1 . . . . . . . . . . . . . .
2 1 1 1 . . . . . . . . . . . . . .
3 1 . . . . . 1 . . . . . . . . . .
4 1 1 . . . . . 1 . . . . . . . . . .
5 1 . . . . . . 1 . . . . . . . . . .
6 1 1 . . . . . . 1 . . . . . . . . . .
```

```

7  1 . 1 . . . . . . . . . .
8  1 1 1 . . . . . . . . . .
9  1 . . . . . . . . . . 1 . . . .
10 1 1 . . . . . . . . . . 1 . . . .
11 1 . . . . . . . . . . 1 . . . .
12 1 1 . . . . . . . . . . 1 . . . .
13 1 . 1 . . . . . . . . . . . . . .
14 1 1 1 . . . . . . . . . . . . . .
15 1 . . . . . . . . . . . . . . 1 .
16 1 1 . . . . . . . . . . . . . . 1 .
17 1 . . . . . . . . . . . . . . . 1
18 1 1 . . . . . . . . . . . . . . . 1

```

We consider a linear normal model, i.e. an n dimensional random vector $y = (y_i)$ for which $\mathbb{E}(y) = \mu = X\beta$ and $\text{Cov}(y) = \sigma^2 I$ where X does not have full column rank We are interested in linear functions of β , say

$$c = k^\top \beta = \sum_j k_j \beta_j.$$

```

> K <- LSmatrix(mod.nst, effect="BB", at=list(CC=2));K
      (Intercept) AA2 BB1:CC1 BB2:CC1 BB3:CC1 BB1:CC2 BB2:CC2 BB3:CC2 BB1:CC3
[1,]           1 0.5      0      0      0      1      0      0      0
[2,]           1 0.5      0      0      0      0      1      0      0
[3,]           1 0.5      0      0      0      0      0      1      0
      BB2:CC3 BB3:CC3 BB1:CC4 BB2:CC4 BB3:CC4
[1,]         0      0      0      0      0
[2,]         0      0      0      0      0
[3,]         0      0      0      0      0

```

```

> LSmeans(mod.nst, K=K)
  estimate      se df  t.stat p.value BB CC
1         NA      NA NA      NA      NA  1  2
2  0.02789  0.5958 10  0.04681  0.9636  2  2
3  0.02029  0.5958 10  0.03405  0.9735  3  2

```

A least squares estimate of β is

$$\hat{\beta} = GX^\top y$$

where G is a generalized inverse of $X^\top X$. Since the generalized inverse is not unique then neither is the estimate $\hat{\beta}$. One least squares estimate of β is

```

> XtXinv <- MASS::ginv(t(X)%*%X)
> bhat <- as.numeric(XtXinv %*% t(X) %*% dat.nst$y)
> zapsmall(bhat)
 [1] -0.3318  0.4493 -0.3784  0.0000  0.0000  0.0000  0.1350  0.1274  0.0000
[10] -0.8203  0.3429  0.0000  0.6713 -0.4096

```

Hence $\hat{c} = k^\top \hat{\beta}$ is in general not unique.

```

> K %*% bhat
      [,1]
[1,] -0.10709
[2,]  0.02789
[3,]  0.02029

```

However, for some values of k , the estimate \hat{c} is unique (i.e. it does not depend on the choice of generalized inverse). Such linear functions are said to be estimable and can be described as follows:

All we specify with $\mu = X\beta$ is that μ is a vector in the linear subspace $L = C(X)$ where $C(X)$ denotes the column space of X . We can only learn about β through $X\beta$ so the only thing we can say something about is linear combinations $\rho^\top X\beta$. Hence we can only say something about $k^\top \beta$ if there exists ρ such that $k^\top \beta = \rho^\top X\beta$, i.e., if $k = X^\top \rho$, that is, if k is in the column space $C(X^\top)$ of X^\top . That is, if k is perpendicular to all vectors in the null space $N(X)$ of X . To check this, we find a basis B for $N(X)$. This can be done in many ways, for example via a singular value decomposition of X , i.e.

$$X = UDV^\top$$

A basis for $N(X)$ is given by those columns of V that corresponds to zeros on the diagonal of D .

```
> S<-svd(X)
> names(S)

[1] "d" "u" "v"

> B<-S$v[, S$d<1e-10, drop=FALSE ]; zapsmall(B) ## Basis for N(X)

      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] -0.3401 -0.0063  0.0033 -0.0161 -0.0951  0
[2,]  0.0000  0.0000  0.0000  0.0000  0.0000  0
[3,]  0.3401  0.0063 -0.0033  0.0161  0.0951  0
[4,] -0.1790  0.0779  0.3570 -0.5385  0.7379  0
[5,]  0.2069  0.0159  0.3519 -0.6750 -0.6144  0
[6,] -0.0065  0.9897  0.0688  0.1105 -0.0587  0
[7,]  0.3401  0.0063 -0.0033  0.0161  0.0951  0
[8,]  0.3401  0.0063 -0.0033  0.0161  0.0951  0
[9,] -0.0006  0.1175 -0.8625 -0.4900  0.0472  0
[10,]  0.3401  0.0063 -0.0033  0.0161  0.0951  0
[11,]  0.3401  0.0063 -0.0033  0.0161  0.0951  0
[12,]  0.0000  0.0000  0.0000  0.0000  0.0000 -1
[13,]  0.3401  0.0063 -0.0033  0.0161  0.0951  0
[14,]  0.3401  0.0063 -0.0033  0.0161  0.0951  0

> zapsmall( rowSums(K%*%B) )

[1] 0.6496 0.0000 0.0000
```

6.4 Pairwise comparisons

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

```
> library("multcomp")
> g1 <- glht(warp.lm, mcp(tension="Tukey"))
> summary( g1 )
```

Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts

```
Fit: lm(formula = breaks ~ wool + tension, data = warpbreaks)
```

```
Linear Hypotheses:
```

	Estimate	Std. Error	t value	Pr(> t)	
M - L == 0	-10.00	3.87	-2.58	0.0336	*
H - L == 0	-14.72	3.87	-3.80	0.0011	**
H - M == 0	-4.72	3.87	-1.22	0.4475	

```
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
(Adjusted p values reported -- single-step method)
```

The K matrix generated in this case is:

```
> K1 <- g1$linfct; K1  
      (Intercept) woolB tensionM tensionH  
M - L           0     0         1         0  
H - L           0     0         0         1  
H - M           0     0        -1         1  
attr("type")  
[1] "Tukey"
```

References

Russell V. Lenth. **lsmeans**: *Least-squares means*, 2013. URL <http://CRAN.R-project.org/package=lsmeans>. R package version 1.06-06.