Using the lsmeans Package

Russell V. Lenth
The University of Iowa
russell-lenth@uiowa.edu

September 19, 2012

1 Introduction

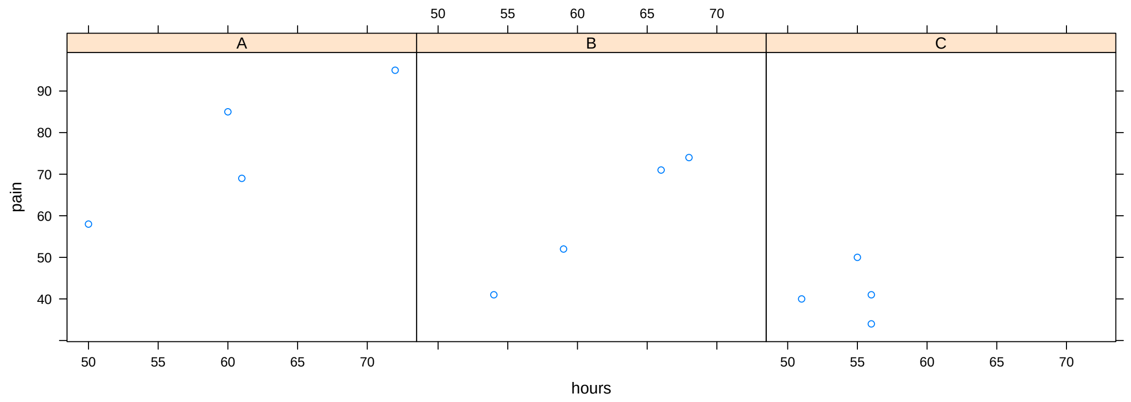
Least-squares means (or LS means), popularized by SAS, are predictions from a linear model at combinations of specified factors. SAS's documentation describes them as "predicted population margins—that is, they estimate the marginal means over a balanced population" (SAS Institute 2012). Unspecified factors and covariates are handled by summarizing the predictions over those factors and variables. This vignette gives some examples of LS means and the `lsmeans` package. Some of the finer points of LS means are explained in the context of these examples.

Like most statistical calculations, it is possible to use least-squares means inappropriately; however, they are in fact simply predictions from the model. When used with due care, they can provide useful summaries of a linear model that includes factors.

2 Analysis-of-covariance example

Oehlert (2000), p.456 gives a dataset concerning repetitive-motion pain due to typing on three types of ergonomic keyboards. Twelve subjects having repetitive-motion disorders were randomized to the keyboard types, and reported the severity of their pain on a subjective scale of 0–100 after two weeks of using the keyboard. We also recorded the time spent typing, in hours. Here are the data, and a plot.

```
R> typing = data.frame(
R>   type = rep(c("A","B","C"), each=4),
R>   hours = c(60,72,61,50, 54,68,66,59, 56,56,55,51),
R>   pain =  c(85,95,69,58, 41,74,71,52, 41,34,50,40))
R> library(lattice)
R> xyplot(pain ~ hours | type, data = typing, layout = c(3,1))
```



It appears that hours and pain are linearly related (though it's hard to know for type *C* keyboards), and that the trend line for type *A* is higher than for the other two. To test this, consider a simple covariate model that fits parallel lines to the three panels:

```
R> typing.lm = lm(pain ~ hours + type, data = typing)
```

The least-squares means resulting from this model are easily obtained by calling lsmeans with the fitted model and a formula specifying the factor of interest:

```
R> library(lsmeans)
R> lsmeans(typing.lm, ~ type)

Loading required package: Matrix
$`type lsmeans`
    lsmean       SE df lower.CL upper.CL
A 73.56518 3.640583  8 65.16998 81.96038
B 54.49529 3.722251  8 45.91176 63.07881
C 49.43953 3.943413  8 40.34600 58.53306
```

These results are the same as what are often called "adjusted means" in the analysis of covariance—predicted values for each keyboard type, when the covariate is set to its overall average value, as we now verify:

```
R> predict(typing.lm, newdata = data.frame(type = c("A","B","C"),
R>     hours = mean(typing$hours)))

       1        2        3
73.56518 54.49529 49.43953
```

The lsmeans function allows us to make predictions at other hours values. We may also obtain comparisons or contrasts among the means by specifying a keyword in the left-hand side of the formula. For example,

```
R> lsmeans(typing.lm, pairwise ~ type, at = list(hours = 55))

$`type lsmeans`
    lsmean       SE df lower.CL upper.CL
A 66.28560 4.154824  8 56.70456 75.86664
B 47.21570 4.351192  8 37.18184 57.24957
C 42.15995 3.588596  8 33.88463 50.43527


$`type pairwise differences`
       estimate       SE df t.ratio p.value
A - B 19.069896 5.081620  8 3.75272 0.01378
A - C 24.125650 5.559580  8 4.33947 0.00621
B - C  5.055754 5.719515  8 0.88395 0.66470
    p values are adjusted using the tukey method for 3 means
```

The resulting least-squares means are each about 7.3 less than the previous results, but their standard errors don't all change the same way: the first two SEs increase but the third decreases because the prediction is closer to the data in that group.

The results for the pairwise differences are the same regardless of the hours value we specify, because the hours effect cancels out when we take the differences. We confirm that the mean pain with keyboard *A* is significantly greater than it is with either of the other keyboards.

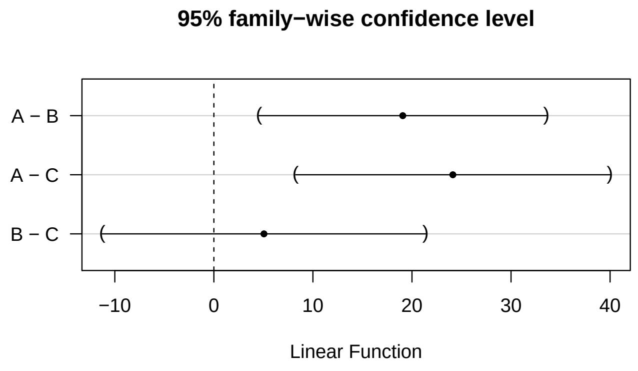
There are other choices besides pairwise. The other built-in options are revpairwise (same as pairwise but the subraction is done the other way; trt.vs.ctrl for comparing one factor level (say, a control) with each of the others, and the related trt.vs.ctrl1, and trt.vs.ctrlk for convenience in specifying which group is the control group; and poly for estimating orthogonal-polynomial contrasts, assuming equal spacing. It is possible to provide custom contrasts as well—see the documentation.

As seen in the previous output, `lsmeans` provides for adjusting the *p* values of contrasts to preserve a familywise error rate. The default for pairwise comparisons is the Tukey (HSD) method. But in covariance models, that method is only approximate. To get a more exact adjustment, we can pass the comparisons to the `glht` function in the `multcomp` package (and also pass additional arguments—in this example, none):

```
R> typing.lsm = lsmeans(typing.lm, pairwise ~ type, glhargs=list())
R> print(typing.lsm, omit=1)

$`type pairwise differences`

         Simultaneous Tests for General Linear Hypotheses

Fit: lm(formula = pain ~ hours + type, data = typing)

Linear Hypotheses:
           Estimate Std. Error t value Pr(>|t|)
A - B == 0   19.070      5.082   3.753   0.0138 *
A - C == 0   24.126      5.560   4.339   0.0061 **
B - C == 0    5.056      5.720   0.884   0.6641
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

The *p* values are slightly different, as expected. We may of course use other methods available for `glht` objects:

```
R> plot(typing.lsm[[2]])
```



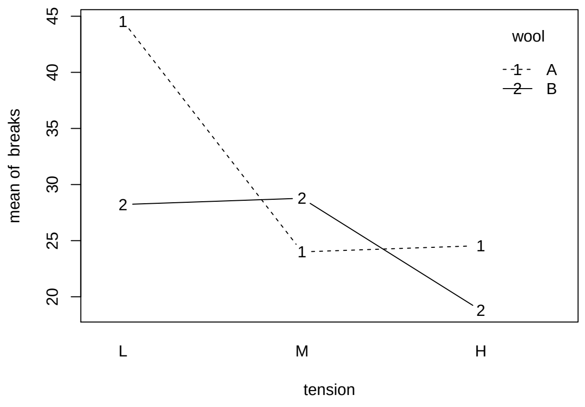
3 Two-factor example

Now consider the R-provided dataset `warpbreaks`, relating to a weaving-process experiment. This dataset (from Tukey 1977, p.82) has two factors: `wool` (two types of wool), and `tension` (low, medium, and high); and the response variable is `breaks`, the nuumber of breaks in a fixed length of yarn.

```
R> with(warpbreaks, table(wool, tension))

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

An interaction plot clearly indicates that we shouldn't consider an additive model.

```
R> with(warpbreaks, interaction.plot(tension, wool, breaks, type="b"))
```



So let us fit a model with interaction

```
R> warp.lm = lm(breaks ~ wool * tension, data = warpbreaks)
R> anova(warp.lm)

Analysis of Variance Table

Response: breaks
             Df Sum Sq Mean Sq F value    Pr(>F)
wool          1  450.7  450.67  3.7653 0.0582130 .
tension       2 2034.3 1017.13  8.4980 0.0006926 ***
wool:tension  2 1002.8  501.39  4.1891 0.0210442 *
Residuals    48 5745.1  119.69
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now we can obtain the least-squares means for the `wool`×`tension` combinations. We could request pairwise comparisons as well by specifying `pairwise ~ wool:tension`, but this will yield quite a few comparisons (15 to be exact). Often, people are satisfied with a smaller number of comparisons (or contrasts) obtained by restricting them to be at the same level of one of the factors. This can be done using the `|` symbol for conditioning. In the code below, we request comparisons of the wools at each tension, and polynomial contrasts for each wool.

```
R> print(lsmeans(warp.lm, list(pairwise ~ wool | tension,  poly ~ tension | wool)), omit=3)

$`wool:tension lsmeans`
       lsmean       SE df lower.CL upper.CL
A, L 44.55556 3.646761 48 37.22325 51.88786
B, L 28.22222 3.646761 48 20.88992 35.55453
A, M 24.00000 3.646761 48 16.66769 31.33231
B, M 28.77778 3.646761 48 21.44547 36.11008
A, H 24.55556 3.646761 48 17.22325 31.88786
B, H 18.77778 3.646761 48 11.44547 26.11008

$`wool:tension pairwise differences`
           estimate       SE df  t.ratio p.value
A - B | L 16.333333 5.157299 48  3.16703 0.00268
A - B | M -4.777778 5.157299 48 -0.92641 0.35887
A - B | H  5.777778 5.157299 48  1.12031 0.26816
    p values are adjusted using the tukey method for 2 means
```

```
$`tension:wool polynomial contrasts`
                estimate       SE df  t.ratio p.value
linear | A    -20.000000 5.157299 48 -3.87800 0.00032
quadratic | A  21.111111 8.932705 48  2.36335 0.02221
linear | B     -9.444444 5.157299 48 -1.83128 0.07327
quadratic | B -10.555556 8.932705 48 -1.18168 0.24315
    p values are not adjusted
```

(We suppressed the third element of the results because it is the same as the first, with rows rearranged.) With these data, the least-squares means are exactly equal to the cell means of the data. The main result (visually clear in the interaction plot) is that the wools differ the most when the tension is low. The signs of the polynomial contrasts indicate decrasing trends for both wools, but opposite concavities.

It is also possible to abuse `lsmeans` with a call like this:

```
R> lsmeans(warp.lm, ~ wool)   ### NOT a good idea!

$`wool lsmeans`
    lsmean       SE df lower.CL upper.CL
A 31.03704 2.105459 48 26.80373 35.27035
B 25.25926 2.105459 48 21.02595 29.49257

Warning message:
In lsmeans(warp.lm, ~wool) :
  lsmeans of wool may be misleading due to interaction with other predictor(s)
```

Each lsmean is the average of the three `tension` lsmeans at the given `wool`. As the warning indicates, the presence of the strong interaction indicates that these results are pretty meaningless. In another dataset wher an additive model would explain the data, these marginal averages, and comparisons or contrasts thereof, can nicely summarize the main effects in an interpretable way.

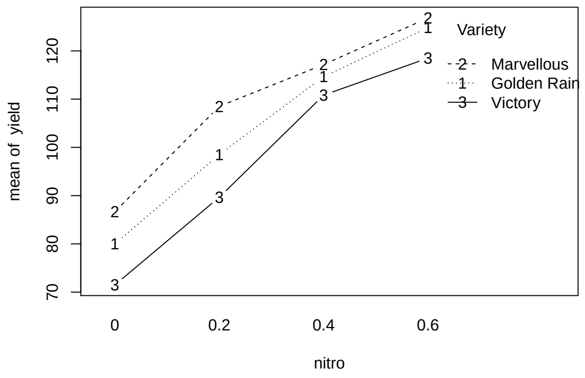
4 Split-plot example

The `nlme` package includes a famous dataset `Oats` that was used in Yates (1935) as an example of a split-plot experiment. Here is a summary of the dataset.

```
R> library(nlme)
R> summary(Oats)

 Block          Variety        nitro           yield      
 VI :12   Golden Rain:24   Min.   :0.00   Min.   : 53.0  
 V  :12   Marvellous :24   1st Qu.:0.15   1st Qu.: 86.0  
 III:12   Victory    :24   Median :0.30   Median :102.5  
 IV :12                    Mean   :0.30   Mean   :104.0  
 II :12                    3rd Qu.:0.45   3rd Qu.:121.2  
 I  :12                    Max.   :0.60   Max.   :174.0  
```

The experiment was conducted in six blocks, and each block was divided into three plots, which were randomly assigned to varieties of oats. With just `Variety` as a factor, it is a randomized complete-block experiment. However, each plot was subdivided into 4 subplots and the subplots were treated with different amounts of nitrogen. Thus, `Block` is a blocking factor, `Variety` is the whole-plot factor, and `nitro` is the split-plot factor. The response variable is `yield`, the yield of each subplot in bushels per acre. Here is an interaction plot of the data

```
R> with(Oats, interaction.plot(nitro, Variety, yield, type="b"))
```



There is not much evidence of an interaction. In this dataset, we have random factors `Block` and `Block:Variety` (which identifies the plots). So we will fit a linear mixed-effects model that accounts for these. Another technicality is that `nitro` is a numeric variable, and initially we will model it as a factor. We will use `lmer` in the `lme4` package to fit a model.

```
R> library(lme4, quietly = TRUE, warn.conflicts = FALSE)
R> Oats.lmer = lmer(yield ~ Variety + factor(nitro) + (1 | Block/Variety), data=Oats)
R> lsmeans(Oats.lmer, list(revpairwise ~ Variety,  poly ~ nitro,   ~ Variety:nitro))
```

```
Loading required package: pbkrtest
Loading required package: MASS
Loading required package: parallel
$`Variety lsmeans`
              lsmean       SE       df lower.CL upper.CL
Golden Rain 104.5000 7.797418 8.869823 86.82147 122.1785
Marvellous  109.7917 7.797418 8.869823 92.11314 127.4702
Victory      97.6250 7.797418 8.869823 79.94647 115.3035

$`Variety pairwise differences`
                          estimate       SE df  t.ratio p.value
Marvellous - Golden Rain   5.291667 7.078899 10  0.74753 0.74187
Victory - Golden Rain     -6.875000 7.078899 10 -0.97120 0.61035
Victory - Marvellous     -12.166667 7.078899 10 -1.71872 0.24583
    p values are adjusted using the tukey method for 3 means

$`nitro lsmeans`
       lsmean       SE       df  lower.CL  upper.CL
0    79.38889 7.132279 6.639194  62.33614  96.44164
0.2  98.88889 7.132279 6.639194  81.83614 115.94164
0.4 114.22222 7.132279 6.639194  97.16947 131.27497
0.6 123.38889 7.132279 6.639194 106.33614 140.44164

$`nitro polynomial contrasts`
           estimate        SE df  t.ratio p.value
linear    147.33333 13.439530 51 10.96268 0.00000
quadratic -10.33333  6.010341 51 -1.71926 0.09163
cubic      -2.00000 13.439530 51 -0.14881 0.88229
    p values are not adjusted

$`Variety:nitro lsmeans`
                    lsmean       SE       df  lower.CL  upper.CL
Golden Rain, 0    79.91667 8.220281 10.93256  61.81032  98.02301
Marvellous, 0     85.20833 8.220281 10.93256  67.10199 103.31468
```

```
Victory, 0        73.04167 8.220281 10.93256  54.93532  91.14801
Golden Rain, 0.2  99.41667 8.220281 10.93256  81.31032 117.52301
Marvellous, 0.2  104.70833 8.220281 10.93256  86.60199 122.81468
Victory, 0.2      92.54167 8.220281 10.93256  74.43532 110.64801
Golden Rain, 0.4 114.75000 8.220281 10.93256  96.64366 132.85634
Marvellous, 0.4  120.04167 8.220281 10.93256 101.93532 138.14801
Victory, 0.4     107.87500 8.220281 10.93256  89.76866 125.98134
Golden Rain, 0.6 123.91667 8.220281 10.93256 105.81032 142.02301
Marvellous, 0.6  129.20833 8.220281 10.93256 111.10199 147.31468
Victory, 0.6     117.04167 8.220281 10.93256  98.93532 135.14801
```

Unlike the warpbreaks example, the additive model makes it reasonable to look at the marginal lsmeans, which are equally-weighted marginal averages of the cell predictions in the fifth table of the output.[1]

While the default for obtaining marginal lsmeans is to weight the predictions equally, we may override this via the `fac.reduce` argument. For example, suppose that we want the `Variety` predictions when `nitro` is 0.25. We can obtain these by interpolation as follows:

```
R> lsmeans(Oats.lmer, ~ Variety, fac.reduce = function(X, lev) .75 * X[2, ] + .25 * X[3, ])

$`Variety lsmeans`
              lsmean      SE       df lower.CL upper.CL
Golden Rain 103.2500 8.01164 9.880139 85.36956 121.1304
Marvellous  108.5417 8.01164 9.880139 90.66122 126.4221
Victory      96.3750 8.01164 9.880139 78.49456 114.2554
```

(There is also a `cov.reduce` argument to change the default handling of covariates.) The polynomial contrasts for `nitro` suggest that we could substitute a quadratic trend for `nitro`; and if we do that, then there is another (probably better) way to make the above predictions:

```
R> OatsPoly.lmer = lmer(yield ~ Variety + poly(nitro, 2) + (1 | Block/Variety), data=Oats)
R> lsmeans(OatsPoly.lmer, ~ Variety, at = list(nitro = .25))

$`Variety lsmeans`
               lsmean       SE       df lower.CL upper.CL
Golden Rain 103.88437 8.002143 9.834068 86.01363 121.7551
Marvellous  109.17604 8.002143 9.834068 91.30529 127.0468
Victory      97.00937 8.002143 9.834068 79.13863 114.8801
```

These predictions are slightly higher than the interpolations mostly because they account for the downward concavity of the fitted quadratics.

5 Rank deficiencies

In many fitted models, predictions may be made at any factor combination. However, some models have rank deficiencies due to collinearity among predictors or missing factor combinations, and this can mess-up predictions. Consider the following example using a simplified model on a subset of the Oats data.

```
R> just.some = c(1,5,9,11,12,14,15,18,19,20,26,27,29,31,32,33,36)
R> wildOats.lm = lm(yield ~ Variety*factor(nitro), data=Oats, subset=just.some)
```

Now consider predictions at various factor combination. Let's first do this manually, using two different parameterizations of the model.

[1]Interestingly, SAS's implementation of least-squares means will refuse to output these cell predictions unless the interaction term is in the model.

```
R> illus = data.frame(Variety=levels(Oats$Variety), nitro=0.6)
R> illus$lsm1 = predict(wildOats.lm, newdata=illus)
R> # Another parameterization...
R> wildOats.lm2 = update(wildOats.lm, . ~ Variety*ordered(nitro))
R> illus$lsm2 = predict(wildOats.lm2, newdata=illus)
R> illus

Warning message:
In predict.lm(wildOats.lm, newdata = illus) :
  prediction from a rank-deficient fit may be misleading
Warning message:
In predict.lm(wildOats.lm2, newdata = illus) :
  prediction from a rank-deficient fit may be misleading
      Variety nitro  lsm1  lsm2
1 Golden Rain   0.6 122.5 122.5
2  Marvellous   0.6 140.0 140.0
3     Victory   0.6 145.0 227.5
```

We received warnings that there is a rank deficiency. And the predictions obtained illustrate why: they are not the same. In particular, they differ considerably for `Victory` even though they match for the other two varieties.

It happens that the first two predictions match because they are *estimable*, while the third is not. `lsmeans` takes pains to check for estimability, rather than just warning that the predictions may be misleading. Here is the `lsmeans` output for all twelve factor combinations:

```
R> lsmeans(wildOats.lm, ~ Variety*nitro)

$`Variety:nitro lsmeans`
                 lsmean       SE df lower.CL upper.CL
Golden Rain, 0     88.5 18.57514  7 44.57678 132.4232
Marvellous, 0      97.0 18.57514  7 53.07678 140.9232
Victory, 0        111.0 26.26921  7 48.88319 173.1168
Golden Rain, 0.2  108.0 26.26921  7 45.88319 170.1168
Marvellous, 0.2      NA       NA NA       NA       NA
Victory, 0.2       77.5 18.57514  7 33.57678 121.4232
Golden Rain, 0.4  107.5 18.57514  7 63.57678 151.4232
Marvellous, 0.4   118.0 26.26921  7 55.88319 180.1168
Victory, 0.4      104.5 18.57514  7 60.57678 148.4232
Golden Rain, 0.6  122.5 18.57514  7 78.57678 166.4232
Marvellous, 0.6   140.0 18.57514  7 96.07678 183.9232
Victory, 0.6         NA       NA NA       NA       NA
```

The results reflect the fact that two lsmeans are non-estimable due to empty cells.

Here's a more subtle example. Suppose we add a covariate to the `typing` dataset used earlier, and fit a model with two covariates:

```
R> typing$foo = c(63,99,66,33,45,87,81,60,51,51,48,36)
R> typing.lmfoo = lm(pain ~ hours + foo + type, data=typing)
R> lsmeans(typing.lmfoo, ~ type)

$`type lsmeans`
    lsmean       SE df lower.CL upper.CL
A 73.56518 3.640583  8 65.16998 81.96038
B 54.49529 3.722251  8 45.91176 63.07881
C 49.43953 3.943413  8 40.34600 58.53306
```

Everything looks fine, but now plug-in a different value for `hours`:

```
R> lsmeans(typing.lmfoo, pairwise ~ type, at = list(hours=65))

$`type lsmeans`
  lsmean SE df lower.CL upper.CL
A     NA NA NA       NA       NA
B     NA NA NA       NA       NA
C     NA NA NA       NA       NA

$`type pairwise differences`
      estimate       SE df t.ratio p.value
A - B 19.069896 5.081620  8 3.75272 0.01378
A - C 24.125650 5.559580  8 4.33947 0.00621
B - C  5.055754 5.719515  8 0.88395 0.66470
    p values are adjusted using the tukey method for 3 means
```

None of the lsmeans are estimable now (though, interestingly, the pairwise differences still are). This is due to linear dependence between `hours` and `foo`. If you know the linear dependence, you can make predictions:

```
R> lsmeans(typing.lmfoo, ~ type, at = list(hours=65, foo=3*(65-39)))

$`type lsmeans`
    lsmean       SE df lower.CL upper.CL
A 84.48456 3.906175  8 75.47690 93.49221
B 65.41466 3.775715  8 56.70785 74.12148
C 60.35891 5.251350  8 48.24927 72.46854
```

At the time of this writing, only `lm` and its relatives can support rank deficiencies. Also, even with `lm` objects, `glht` cannot handle rank deficiencies, so an error will occur if you include a non-null `glhargs` argument in `lsmeans`.

6 GLMM example

The dataset `cbpp` in the `lme4` package, originally from Lesnoff *et al.* (1964), provides data on the incidence of contagious bovine pleuropneumonia in 15 herds of zebu cattle in Ethiopia, collected over four time periods. These data are used as the primary example for the `glmer` function, and it is found that a model that accounts for overdisperion is advantageous; hence the addition of the `(1|obs)` in the model fitted below.

`lsmeans` may be used as in linear models to obtain marginal linear predictions for a generalized linear model or, in this case, a generalized linear mixed model. Here, we use the `trt.vs.ctrl1` contrast family to compare each period with the first, as the primary goal was to track the spread or decline of CBPP over time. We will save the results from `lsmean`, then add the inverse logits of the predictions and the estimated odds ratios for the comparisons as an aid in interpretation.

```
R> cbpp$obs = 1:nrow(cbpp)
R> cbpp.glmer = glmer(cbind(incidence, size - incidence)
R>     ~ period + (1 | herd) + (1 | obs),  family = binomial,  data = cbpp)

Number of levels of a grouping factor for the random effects
is *equal* to n, the number of observations

R> cbpp.lsm = lsmeans(cbpp.glmer, trt.vs.ctrl1 ~ period)
R> cbpp.lsm[[1]]$pred.incidence = 1 - 1 / (1 + exp(cbpp.lsm[[1]]$lsmean))
R> cbpp.lsm[[2]]$odds.ratio = exp(cbpp.lsm[[2]]$estimate)
R> cbpp.lsm
```

```
$`period lsmeans`
     lsmean        SE df asymp.LCL  asymp.UCL pred.incidence
1 -1.500292 0.2887610 NA -2.066253 -0.9343304     0.18238203
2 -2.726800 0.3809740 NA -3.473496 -1.9801052     0.06141032
3 -2.829133 0.3994052 NA -3.611953 -2.0463133     0.05577003
4 -3.366631 0.5193989 NA -4.384634 -2.3486279     0.03335476

$`period differences from control`
       estimate        SE df  z.ratio p.value odds.ratio
2 - 1 -1.226509 0.4734567 NA -2.59054 0.02851  0.2933148
3 - 1 -1.328841 0.4883951 NA -2.72083 0.01944  0.2647839
4 - 1 -1.866339 0.5905702 NA -3.16023 0.00474  0.1546889
    p values are adjusted using the sidak method for 3 tests
```

When degrees of freedom are not available, as in this case, `lsmeans` emphasizes that fact by displaying `NA` for degrees of freedom and in the column headings.

7 Contrasts

You may occasionally want to know exactly what contrast coefficients are being used, especially in the polynomial case. Contrasts are implemented in functions having names of the form *name*`.lsmc` ("lsmc" for "least-squares means contrasts"), and you can simply call that function to see the contrasts; for example,

```
R> poly.lsmc(1:4)

  linear quadratic cubic
1     -3         1    -1
2     -1        -1     3
3      1        -1    -3
4      3         1     1
```

`poly.lsmc` uses the base function `poly` plus an *ad hoc* algorithm that tries (and usually succeeds) to make integer coefficients, copmparable to what you find in published tables of orthogonal polynomial contrasts.

You may supply your own custom contrasts in two ways. One is to supply a `contr` argument in the `lsmeans` call, like this:

```
R> print(lsmeans(typing.lm, custom.comp ~ type,
R>              contr = list(custom.comp = list(A.vs.others=c(1, -.5, -.5)))),
R>     omit=1)

$`type custom.comp`
            estimate      SE df t.ratio p.value
A.vs.others 21.59777 4.49307  8 4.80691 0.00134
    p values are not adjusted
```

Each contrast family is potentially a list of several contrasts, and there are potentially more than one contrast family; so we must provide a list of lists.

The other way is to create your own `.lsmc` function, and use its base name in a formula:

```
R> inward.lsmc = function(levs, ...) {
R>     n = length(levs)
R>     result = data.frame(`grand mean` = rep(1/n, n))
R>     for (i in 1 : floor(n/2)) {
R>         x = rep(0, n)
R>         x[1:i] = 1/i
R>         x[(n-i+1):n]  = -1/i
R>         result[[paste("first", i, "vs last", i)]] = x
R>     }
R>     attr(result, "desc") = "grand mean and inward contrasts"
R>     attr(result, "adjust") = "none"
R>     result
R> }
```

Testing it, we have

```
R> inward.lsmc(1:5)

  grand.mean first 1 vs last 1 first 2 vs last 2
1        0.2                 1               0.5
2        0.2                 0               0.5
3        0.2                 0               0.0
4        0.2                 0              -0.5
5        0.2                -1              -0.5
```

. . . and an application:

```
R> print(lsmeans(Oats.lmer, inward ~ nitro), omit=1)

$`nitro grand mean and inward contrasts`
                   estimate       SE        df   t.ratio p.value
grand.mean        103.97222 6.640491  5.000417  15.65731   2e-05
first 1 vs last 1 -44.00000 4.249953 51.000000 -10.35306   0e+00
first 2 vs last 2 -29.66667 3.005170 51.000000  -9.87188   0e+00
    p values are not adjusted
```

References

Lesnoff, M., Laval, G., Bonnet, P., *et al.* (2004) Within-herd spread of contagious bovine pleuropneumonia in Ethiopian highlands, *Preventive Veterinary Medicine*, **64**, 27–40.

Oehlert, G. (2000) *A First Course in Design and Analysis of Experiments*, W. H. Freeman. This is out-of-print, but now available under a Creative Commons license via http://users.stat.umn.edu/~gary/Book.html (accessed August 23, 2012).

SAS Institute Inc. (2012) Online documentation, SAS/STAT version 9.3: Shared concepts: LSMEANS statement. http://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_introcom_a0000003362.htm (accessed August 14, 2012).

Tukey, J. W. (1977) *Exploratory Data Analysis*. Addison-Wesley.

Yates, F. (1935) Complex experiments, *Journal of the Royal Statistical Society* (Supplement), **2**, 181–247.