Using lsmeans

Russell V. Lenth
The University of Iowa

Abstract

Least-squares means are predictions from a linear model, or averages thereof. They are useful in the analysis of experimental data for summarizing the effects of factors, and for testing contrasts among certain marginal predictions. The **lsmeans** package provides a simple and rather comprehensive formula-based way of specifying least-squares means and contrasts thereof. It supports most R packages that fit linear or mixed models.

Keywords: least-squares means, linear models, experimental design.

1. Introduction

1.1. What are least-squares means?

Least-squares means (or LS means), are generalizations of covariate-adjusted means, and date back at least to 1976 when they were incorporated in the contributed SAS procedure named `HARVEY` (Harvey 1976). Later, they were incorporated via `LSMEANS` statements in the regular SAS releases. SAS's documentation describes them as "predicted population margins—that is, they estimate the marginal means over a balanced population" (SAS Institute Inc. 2012).

People disagree on the appropriateness of LS means. As in many statistical calculations, there are times when they are appropriate, and times when they are not. However, as long as one understands what is being calculated, one can judge its appropriateness. The main thing to remember is that LS means are simply predictions from a model over a grid of predictor values, or marginal averages thereof. More explicitly, define a set of *reference levels* for each predictor, and create a grid (call it the *reference grid*) consisting of all combinations of these. Make predictions on this grid, and compute marginal means of those predictions, if needed (usually using equal weights). For clarity, we refer to these averaged predictions as *marginal LS means.*

The default in **lsmeans** is to set the reference levels as follows: For predictors of class `factor` or `ordered`, the default reference levels are the levels of the factor. For numeric predictors, the default is to use a single reference level at the mean value of the predictor. It is possible to change the reference levels, and if this is done, it is extremely important to understand that this also alters the definition of any marginal LS means, as the averaging is done over a different set of levels.

1.2. Package overview

The **lsmeans** package (Lenth 2014) is built upon objects of class `ref.grid` which defines the grid of reference levels to use for the predictions. Such `ref.grid` objects are provided for linear models produced by most linear-models functions including `lm` and `aov` in the **stats** package; `lme` and `gls` from the **nlme** package (Pinheiro, Bates, and R-core 2013); and `lmer` and others from the **lme4** package (Bates, Maechler, Bolker, and Walker 2013). `aov` is supported only if the model does not contain an `Error()` term. Generalized linear models and GLMMs are also supported, where LS means are defined in terms of the linear predictor (before applying the link function). For `lm` objects, special provisions are included to check for estimability when the model is rank-deficient. Provisions are also made for models with a multivariate response, so that the dimensions of the response can be specified in the same way as the levels of a factor.

As explained before, LS means are predictions over the reference grid, or marginal averages thereof. These are computed by the function `lsmeans`, which works with either a `ref.grid` or a model object. The desired sets of LS means are specified using the names of the predictors, and optionally the names of "by" variables for grouping. Alternatively, these can be specified using a formula, e.g., `~ dose | treat` requests the LS means for each dose, within each treatment. `lsmeans` creates an object of class `lsmobj`, a sub-class of `ref.grid`.

The `summary` method for `ref.grid` and `lsmobj` objects computes estimates, standard errors, confidence intervals, test statistics, and P values. It also allows for groupings by one or more variables, and allows for various adjustments for multiplicity of tests.

There are several useful functions that can be used to do follow-up analyses. The most important one is `contrast`, which computes contrasts of LS means. A number of standard contrast families are provided and they can be specified by name, e.g., `"pairwise"` or `"poly"`. User-specified contrasts (or for that matter, any set of linear functions, be they contrasts or not) may be specified using a `list` of coefficients. Contrasts may also be requested directly from `lsmeans` via a `contr` argument or in the left-hand side of a formula, e.g., `poly ~ dose | treat` would request orthogonal polynomial contrasts of `dose` means at each level of each `treat`. The `contrast` function returns an `lsmobj` object; thus it is possible to do further analyses of those results, such as contrasts of contrasts.

Other useful methods for `lsmobj` objects include `test` and `confint`, which simply call `summary` with the implied portion of the statistical output; `pairs`, which calls `contrasts` for pairwise comparisons; `cld`, which provides a compact letter display of comparisons; `glht`, which passes the object to the **multcomp** package (Hothorn, Bretz, and Westfall 2013) for more exacting multiplicity adjustments; and `lsmip`, which produces an interaction-plot-like display of the LS means.

`lsmeans` works as follows. First, if given a fitted-model object, the `ref.grid` is created. This entails reconstructing the dataset used in fitting the model, by calling a `recover.data` method. Then the factor levels and other summary information is used to define the reference grid, and an `lsm.basis` method is called to assemble other needed information, such as the linear function associated with each grid point, the regression coefficients, covariance matrix, degrees of freedom information, basis for estimable functions, and so forth. New `recover.data` and `lsm.basis` methods may be written to support additional model types. The `ref.grid` object contains all needed information needed for subsequent least-squares-mean analysis, independent of the model type. In mixed models fitted by a **lme4** function, the

pbkrtest package (Halekoh and Højsgaard 2013), if installed, is used to adjust the covariance matrix and obtain degrees of freedom using the Kenward-Roger method. If degrees of freedom are not available, asymptotic results are used and labeled as such.

The `lsmeans` methods use the given specifications to obtain marginal averages of the linear predictors as needed, and the `contrast` function computes contrasts among the linear predictors. These altered sets of linear predictors define something quite similar, but more general, than a reference grid, outputted as an `lsmobj` object. The `summary` method does the statistical calculations; thus, one can re-summarize a result in a different way if needed.

There is also an `lstrends` function which uses a fitted model to obtain a difference quotient from two reference grids, and returns an `lsmobj` object. This is useful for comparing the slopes of lines in models where a covariate interacts with other predictors.

2. Some examples

Most of the remainder of this article consists of examples showing **lsmeans**'s features and how it can be used to advantage in a variety of situations.

2.1. Adjusted means in covariance models

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 we enter the data, and obtain the plot shown in Figure 1.

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

It appears that `hours` and `pain` are linearly related (though it's hard to know for keyboard C), and that the trend line for keyboard 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 + keybd, data = typing)
```

The reference levels can be discerned by calling the `ref.grid` function:

```
R> ( typing.rg <- ref.grid(typing.lm) )
```

```
'ref.grid' object with variables:
    hours = 59
    keybd = A, B, C
    pain = response variable with mean 59.167
```

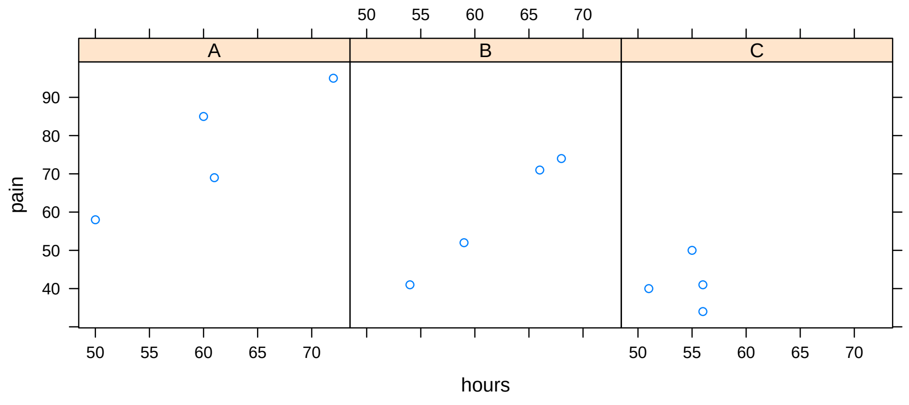

Figure 1: Display of the keyboard-pain data.

Note that only one variable has more than one level. Thus, the reference grid has only three points in it, corresponding to the three keyboards. The `summary` displays the predictions on this grid:

```
R> summary(typing.rg)
```

```
 hours keybd prediction     SE df
    59 A         73.565 3.6406  8
    59 B         54.495 3.7223  8
    59 C         49.440 3.9434  8
```

If we want `lsmeans` of the keyboard types, we get the same results, only by default, 95% confidence intervals are displayed:

```
R> ( typing.lsm <- lsmeans(typing.rg, "keybd") )
```

```
 keybd lsmean     SE df lower.CL upper.CL
 A     73.565 3.6406  8   65.170   81.960
 B     54.495 3.7223  8   45.912   63.079
 C     49.440 3.9434  8   40.346   58.533

Confidence level used: 0.95
```

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

The `cov.reduce` and `at` arguments can modify the reference grid. For example, by default, covariates are reduced to their means, but we can change this:

```
R> ref.grid(typing.lm, cov.reduce = median)
```

```
'ref.grid' object with variables:
    hours = 57.5
    keybd = A, B, C
    pain = response variable with mean 59.167
```

Or we can use `at` to create a reference grid that contains more `hours` values:

```
R> typing.rg2 <- ref.grid(typing.lm, at = list(hours = c(50,60)))
R> lsmeans(typing.rg2, c("keybd","hours"))

 keybd hours lsmean     SE df lower.CL upper.CL
 A        50 57.186 5.3185  8   44.922   69.451
 B        50 38.116 5.5940  8   25.216   51.016
 C        50 33.060 3.9434  8   23.967   42.154
 A        60 75.385 3.5944  8   67.096   83.674
 B        60 56.315 3.6406  8   47.920   64.710
 C        60 51.259 4.1093  8   41.783   60.736

Confidence level used: 0.95
```

Again, these LS means are the same as the predictions at the six points of the reference grid. However, if we specify fewer predictors, we obtain marginal averages of the predictions:

```
R> lsmeans(typing.rg2, "keybd")

 keybd lsmean     SE df lower.CL upper.CL
 A     66.286 4.1548  8   56.705   75.867
 B     47.216 4.3512  8   37.182   57.250
 C     42.160 3.5886  8   33.885   50.435

Confidence level used: 0.95

R> lsmeans(typing.rg2, "hours")

 hours lsmean     SE df lower.CL upper.CL
    50 42.788 3.8865  8   33.825   51.750
    60 60.987 2.1012  8   56.141   65.832

Confidence level used: 0.95
```

Note that the results just above for `keybd` are not the same as the results we got the first time, using `typing.rg`. This illustrates the important point that *least-squares means depend on the reference grid*. In the first case, we have predictions at the average `hours`, 59, and in the second, we have the averages of predictions at 50 and 60 hours.

2.2. Follow-up analyses

There are several followup analyses available. Using our original `typing.lsm` result, we can obtain pairwise comparisons of them:

```
R> ( typing.pairs <- pairs(typing.lsm) )

 contrast estimate     SE df t.ratio p.value
 A - B     19.0699 5.0816  8   3.753  0.0138
 A - C     24.1257 5.5596  8   4.339  0.0062
 B - C      5.0558 5.7195  8   0.884  0.6647

P value adjustment: tukey method for a family of 3 means
```

Or the same results with a compact letter display (this requires that the **multcompView** package (Graves, Piepho, Selzer, and Dorai-Raj 2012) be installed):

```
R> cld(typing.lsm, alpha = .10)

 keybd lsmean     SE df lower.CL upper.CL .group
 C     49.440 3.9434  8   40.346   58.533  1
 B     54.495 3.7223  8   45.912   63.079  1
 A     73.565 3.6406  8   65.170   81.960   2

Confidence level used: 0.95
P value adjustment: tukey method for a family of 3 means
significance level used: alpha = 0.1
```

In this display, two LS means that share at least one grouping symbol are not significantly different at the stated level. In this case, keyboard type A's predicted pain is significantly greater than either of the other two. By default, `cld` sorts the means, but this can be disabled.

Using the `contrast` function, other contrast families are available besides pairwise comparisons. For example, to obtain factor effects (differences from the grand mean), use:

```
R> contrast(typing.lsm, "eff")

 contrast estimate     SE df t.ratio p.value
 A effect  14.3985 2.9954  8   4.807  0.0040
 B effect  -4.6714 3.0941  8  -1.510  0.1695
 C effect  -9.7271 3.3569  8  -2.898  0.0299

P value adjustment: fdr method for 3 tests
```

It is possible to provide custom contrasts as well—see the documentation.

Sometimes, we want to see different analyses of the same results. For example, the above results for `pairs` had a Tukey adjustment. If you want to know what the P values are with no adjustment, just do a different summary:

```
R> summary(typing.pairs, adjust = "none")

 contrast estimate     SE df t.ratio p.value
 A - B     19.0699 5.0816  8   3.753  0.0056
 A - C     24.1257 5.5596  8   4.339  0.0025
 B - C      5.0558 5.7195  8   0.884  0.4025
```

2.3. Interfacing with multcomp

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. One must use these adjustments with caution. For example, when the standard errors are unequal, the Tukey method is only approximate, even under normality and independence assumptions. To get a more exact adjustment, we can pass the results to the `glht` function in the **multcomp** package (Hothorn *et al.* 2013):

```
R> library("multcomp")
R> typing.glht <- glht(typing.lm, typing.pairs)
R> summary(typing.glht)

         Simultaneous Tests for General Linear Hypotheses

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

Linear Hypotheses:
           Estimate Std. Error t value Pr(>|t|)
A - B == 0    19.07       5.08    3.75   0.0137 *
A - C == 0    24.13       5.56    4.34   0.0062 **
B - C == 0     5.06       5.72    0.88   0.6642
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1
(Adjusted p values reported -- single-step method)
```

These p values are exact (if the assumptions hold) and, as expected, differ slightly from those in the previous `lsmeans` output. We may of course use other methods available for `glht` objects. The plot in Figure 2 displays the comparisons in the preceding table:

```
R> plot(typing.glht)
```

We have also provided an `lsm` function that can be called within a `glht` call in a way similar to that of `mcp` as provided in the **multcomp** package. Here we display simultaneous confidence intervals for the LS means:

```
R> confint(glht(typing.lm, lsm("keybd")))

         Simultaneous Confidence Intervals

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

Quantile = 2.957
95% family-wise confidence level


Linear Hypotheses:
       Estimate lwr    upr
```

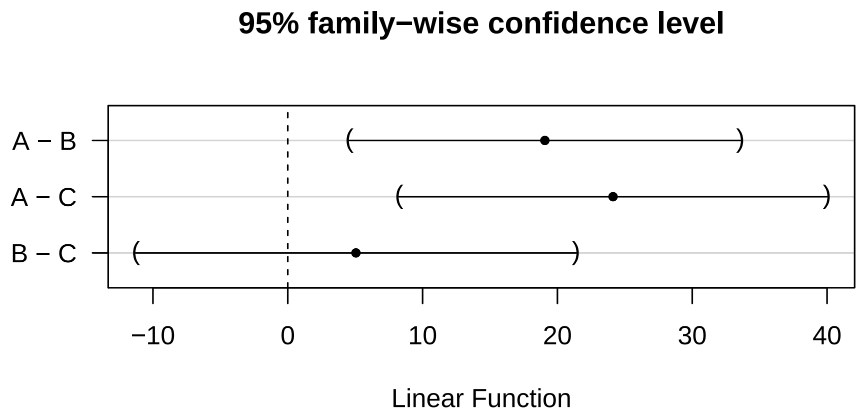

Figure 2: Graphical display of comparisons via **multcomp**

```
A == 0 73.565   62.801 84.330
B == 0 54.495   43.489 65.501
C == 0 49.440   37.780 61.100
```

The design of `lsm` is to create just one set of linear functions to hand to `glht`. It returns contrast output if specified, otherwise LS means output; so in the illustration above, the linear functions of the lsmeans themselves are used. If we had instead specified

```
R> lsm("keybd", contr="pairwise")
```

(output not shown) then the results would have been the same as shown earlier for the pairwise differences.

2.4. Fancy `lsmeans` calls

The `lsmeans` function allows for a lot of flexibility. we can call it with a fitted-model object instead of a `ref.grid`. If so, it can pass `at` and `cov.reduce` arguments to `ref.grid`. One may also specify contrasts and grouping variables. Here is an example:

```
R> lsmeans(typing.lm, specs = "keybd", by = "hours",
+     at = list(hours = c(50, 60)), contr = "trt.vs.ctrl1")

$lsmeans
hours = 50:
 keybd lsmean     SE df lower.CL upper.CL
 A     57.186 5.3185  8   44.922   69.451
 B     38.116 5.5940  8   25.216   51.016
 C     33.060 3.9434  8   23.967   42.154

hours = 60:
 keybd lsmean     SE df lower.CL upper.CL
 A      75.385 3.5944  8   67.096   83.674
 B      56.315 3.6406  8   47.920   64.710
 C      51.259 4.1093  8   41.783   60.736

Confidence level used: 0.95

$contrasts
hours = 50:
 contrast estimate     SE df t.ratio p.value
 B - A     -19.070 5.0816  8  -3.753  0.0112
 C - A     -24.126 5.5596  8  -4.339  0.0050

hours = 60:
 contrast estimate     SE df t.ratio p.value
 B - A     -19.070 5.0816  8  -3.753  0.0112
 C - A     -24.126 5.5596  8  -4.339  0.0050

P value adjustment: sidak method for 2 tests
```

The result is a `list` with two `lsmobj` objects. When a `by` variable is present, the listings are grouped accordingly, and contrasts are restricted to each group.

In addition, a formula may be used in `specs` in place of all or part of the separate `specs`, `by`, and `contr` arguments. The following (not run) are all equivalent to the above:

```
R> lsmeans(typing.lm, specs = ~ keybd, by = "hours",
+     at = list(hours = c(50, 60)), contr = "trt.vs.ctrl1")
R> lsmeans(typing.lm, specs =  ~ keybd | hours,
+     at = list(hours = c(50, 60)), contr = "trt.vs.ctrl1")
R> lsmeans(typing.lm, specs =  trt.vs.ctrl1 ~ keybd | hours,
+     at = list(hours = c(50, 60)))
```

2.5. A three-factor experiment

The `auto.noise` dataset provided with **lsmeans** contains data from a factorial experiment wherein a newly design air-pollution filter called the Octel filter is compared with a standard filter with respect to the amount of ambient noise. Besides the factor `type` for which filter is used, the experiment includes three different sizes of cars (factor `size`) and measurements from each side of the car (factor `side`). First we fit a model to the data:

```
R> noise.lm <- lm(noise ~ size*type*side, data = auto.noise)
R> anova(noise.lm)

Analysis of Variance Table

Response: noise
               Df Sum Sq Mean Sq F value  Pr(>F)
```

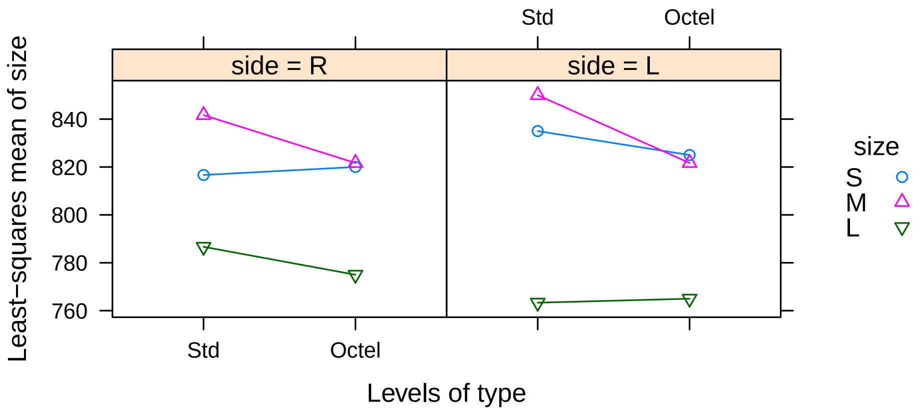

Figure 3: Three-way interaction plot for the `auto.noise` data.

```
size            2  26051    13026   893.19 < 2e-16 ***
type            1   1056     1056    72.43 1.0e-08 ***
side            1      1        1     0.05 0.82910
size:type       2    804      402    27.57 6.0e-07 ***
size:side       2   1293      647    44.33 8.7e-09 ***
type:side       1     17       17     1.19 0.28607
size:type:side  2    301      151    10.33 0.00058 ***
Residuals      24    350       15
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1
```

The default reference grid for the `lsmeans` consists of all $3 \times 2 \times 2 = 12$ factor combinations:

```
R> ref.grid(noise.lm)

'ref.grid' object with variables:
    size = S, M, L
    type = Std, Octel
    side = R, L
    noise = response variable with mean 810.14
```

The model includes all interactions, so the LS means are the cell means. The **lsmeans** package provides a convenient function `lsmip` for displaying an interaction plot. (This feature requires **lattice** (Sarkar 2013) to be installed.) Figure 3 shows separate interaction plots for each side, via

```
R> lsmip(noise.lm, size ~ type | side)
```

The left side of the formula in `lsmip` specifies which factor(s) define the different curves, and the right side specifies the factor(s) for x axis. If a `|` character is included, it separates the plot into different panels. If two or more factors are given, their factor combinations are used to create a single factor for purposes of plotting. To illustrate, some variations on the plot in Figure 3 (not shown) are as follows:

```
R> lsmip(noise.lm, size ~ type * side)  # 1 panel, 3 curves, 2*2 = 4 x values
R> lsmip(noise.lm, type * side ~ size)  # 1 panel, 2*2 = 4 curves, 3 x values
R> lsmip(noise.lm, type ~ side | size)  # 3 panels, 2 curves, 2 x values
```

The main goal of the experiment is to compare the mean noise levels for the two filters. One naïve way to do this is to simply ask for that comparison:

```
R> lsmeans(noise.lm, pairwise ~ type)
```

```
$lsmeans
 type  lsmean     SE df lower.CL upper.CL
 Std   815.56 0.9001 24   813.70   817.41
 Octel 804.72 0.9001 24   802.86   806.58

Confidence level used: 0.95

$contrasts
 contrast    estimate     SE df t.ratio p.value
 Std - Octel   10.833 1.2729 24    8.51  <.0001

Warning in lsmeans(noise.lm, pairwise ~ type) :
 lsmeans of type may be misleading due to interaction with other predictor(s)
```

`lsmeans` generates a warning message because the model includes interactions and it may not be wise to do main-effect comparisons. But whether it is wise or not, keep in mind that the LS means are marginal averages (using equal weights) of the predictions in the reference grid. So the LS mean for the `Std` filter is the average of the six predictions for which `type = Std`; and the LS mean for `Octel` is the average of the other six predictions. For a balanced experiment (which is the case here), these will be the same as the marginal means of the data:

```
R> with(auto.noise, tapply(noise, type, mean))
```

```
   Std  Octel
815.56 804.72
```

So one way to look at marginal LS means for unbalanced data is that they are estimates of the marginal means we *would* obtain, had the experiment been balanced.

Now, given the strength of the interactions, it really is not smart to compare the marginal LS means for `type`; instead, we should compare them at each combination of the other factors. This is easily done by conditioning:

```
R> lsmeans(noise.lm, pairwise ~ type | size*side)[[2]]
```

```
size = S, side = R:
 contrast    estimate    SE df t.ratio p.value
 Std - Octel  -3.3333 3.118 24  -1.069  0.2957

size = M, side = R:
 contrast    estimate    SE df t.ratio p.value
 Std - Octel  20.0000 3.118 24   6.414  <.0001

size = L, side = R:
 contrast    estimate    SE df t.ratio p.value
 Std - Octel  11.6667 3.118 24   3.742  0.0010

size = S, side = L:
 contrast    estimate    SE df t.ratio p.value
 Std - Octel  10.0000 3.118 24   3.207  0.0038

size = M, side = L:
 contrast    estimate    SE df t.ratio p.value
 Std - Octel  28.3333 3.118 24   9.087  <.0001

size = L, side = L:
 contrast    estimate    SE df t.ratio p.value
 Std - Octel  -1.6667 3.118 24  -0.535  0.5979
```

(We show only the second table of the results; the first table is the same as was shown earlier for the LS means of the three-factor combinations.) We find that in the four middle cases, the mean noise is statistically greater for the `Std` filter than the `Octel` filter. In the other two cases, the differences are nonsignificant. Note that a *separate* Tukey correction is made for each combination of the conditioning factors. Since each condition involves only two means, there is only one comparison and hence this amounts to no multiplicity correction at all. The conditioning also greatly reduces the output; if we had specified `pairwise ~ type*size*side`, we would have obtained estimates and tests of all $\binom{12}{2} = 66$ pairwise comparisons of the 12 means, and the Tukey correction would have been based on 12 means also.

2.6. 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. The dataset contains predictors `Block` (6-level factor), `Variety` (3-level factor), and `nitro` (4 unique numeric values). 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.

This experiment has 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 (Bates *et al.* 2013) to fit a model:

```
R> data(Oats, package = "nlme")
R> library("lme4")
R> Oats.lmer <- lmer(yield ~ Variety*factor(nitro) + (1|Block/Variety),
+     data = Oats)
R> anova(Oats.lmer)

Analysis of Variance Table
                      Df Sum Sq Mean Sq F value
Variety                2    526     263    1.49
factor(nitro)          3  20020    6673   37.69
Variety:factor(nitro)  6    322      54    0.30

R> lsmip(Oats.lmer, Variety ~ nitro)
```

The interaction plot is displayed in Figure 4(a).

There is not much evidence of an interaction. Let's reduce to an additive model and look at the LS means and some appropriate contrasts

```
R> Oats.add <- lmer(yield ~ Variety + factor(nitro) + (1|Block/Variety),
+     data = Oats)
R> lsmeans(Oats.add, list(revpairwise ~ Variety,  poly ~ nitro))

$` lsmeans`
 Variety      lsmean     SE   df lower.CL upper.CL
 Golden Rain 104.500 7.7975 8.87   86.821   122.18
 Marvellous  109.792 7.7975 8.87   92.113   127.47
 Victory      97.625 7.7975 8.87   79.946   115.30

Confidence level used: 0.95

$` contrasts`
 contrast                 estimate     SE df t.ratio p.value
 Marvellous - Golden Rain   5.2917 7.0789 10   0.748  0.7419
 Victory - Golden Rain     -6.8750 7.0789 10  -0.971  0.6104
 Victory - Marvellous     -12.1667 7.0789 10  -1.719  0.2458

P value adjustment: tukey method for a family of 3 means

$` lsmeans`
 nitro  lsmean     SE   df lower.CL upper.CL
   0.0  79.389 7.1324 6.64   62.336   96.442
   0.2  98.889 7.1324 6.64   81.836  115.942
   0.4 114.222 7.1324 6.64   97.169  131.276
   0.6 123.389 7.1324 6.64  106.336  140.442
```

```
Confidence level used: 0.95

$` contrasts`
 contrast  estimate      SE df t.ratio p.value
 linear     147.333 13.4395 51  10.963  <.0001
 quadratic  -10.333  6.0103 51  -1.719  0.0916
 cubic       -2.000 13.4395 51  -0.149  0.8823
```

The polynomial contrasts for `nitro` suggest that we could substitute a quadratic trend for `nitro`; so let's fit a third model where `nitro` is a quantitative predictor with a quadratic trend:

```
R> Oats.poly <- lmer(yield ~ Variety + poly(nitro, 2) + (1 | Block/Variety),
+     data=Oats)
```

If we want to see the same predictions as before, use the `at` argument to expand the reference grid:

```
R> Oats.poly.rg <- ref.grid(Oats.poly, at = list(nitro = c(0, .2, .4, .6)))
R> lsmeans(Oats.poly.rg, ~ Variety)

 Variety      lsmean     SE   df lower.CL upper.CL
 Golden Rain 104.500 7.7976 8.87   86.821   122.18
 Marvellous  109.792 7.7976 8.87   92.113   127.47
 Victory      97.625 7.7976 8.87   79.946   115.30

Confidence level used: 0.95

R> lsmeans(Oats.poly.rg, ~ nitro)

 nitro  lsmean     SE   df lower.CL upper.CL
   0.0  79.289 7.0923 6.49   62.249   96.329
   0.2  99.189 6.8379 5.62   82.178  116.199
   0.4 113.922 6.8379 5.62   96.912  130.933
   0.6 123.489 7.0923 6.49  106.449  140.529

Confidence level used: 0.95
```

(Note: With the `at` argument omitted, we would obtain different LS means for `Variety`, because they would be predictions at the average `nitro` value of 0.3 rather than the averages of four predictions.) A simpler way to get the unique values of covariates is to specify `cov.reduce = FALSE`; we show this in a call to `lsmip`, which produces the interaction plot in Figure 4(b).

```
R> lsmip(Oats.poly, Variety ~ nitro, cov.reduce = FALSE)
```

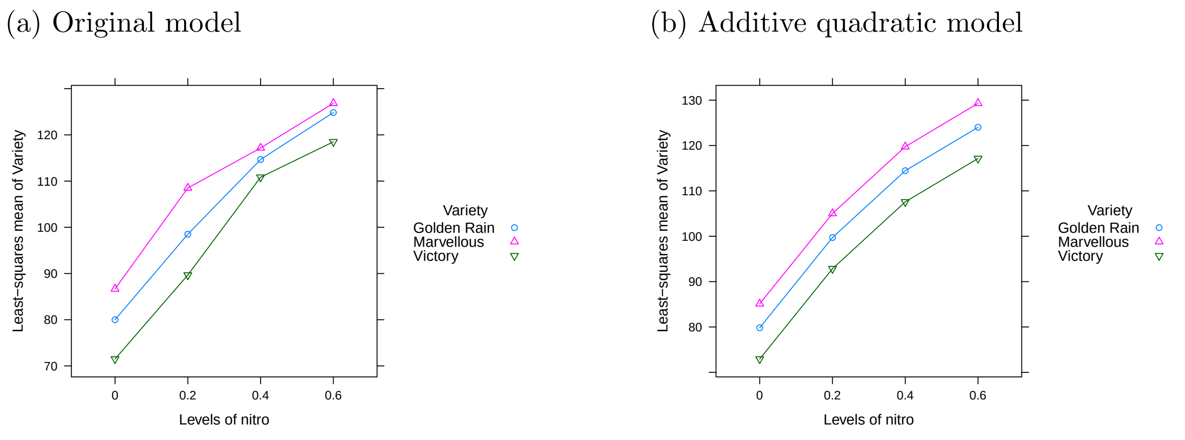

Figure 4: Interaction plots for the Oats experiment

2.7. Messy data

To illustrate some more issues, and related `lsmeans` capabilities, consider the dataset named `nutrition` that is provided with the **lsmeans** package. These data come from Milliken and Johnson (1984), and contain the results of an observational study on nutrition education. Low-income mothers are classified by race, age category, and whether or not they received food stamps (the `group` factor); and the response variable is a gain score (post minus pre scores) after completing a nutrition training program.

Consider the model that includes all main effects and two-way interactions; and let us look at the `group` by `race` LS means:

```
R> nutr.lm <- lm(gain ~ (age + group + race)^2, data = nutrition)
R> lsmip(nutr.lm, race ~ age | group)
R> lsmeans(nutr.lm, ~ group*race)
```

```
 group      race       lsmean     SE df   lower.CL upper.CL
 FoodStamps Black      4.7083 2.3681 92  0.0049714   9.4115
 NoAid      Black     -2.1904 2.4906 92 -7.1368981   2.7561
 FoodStamps Hispanic       NA     NA NA         NA       NA
 NoAid      Hispanic       NA     NA NA         NA       NA
 FoodStamps White      3.6077 1.1556 92  1.3125215   5.9028
 NoAid      White      2.2563 2.3893 92 -2.4889667   7.0016

Confidence level used: 0.95
```

Figure 5 shows the predictions from this model. One thing the `lsmeans` output illustrates is that `lsmeans` incorporates an estimability check, and returns a missing value when a prediction cannot be made uniquely. In this example, we have very few Hispanic mothers in the dataset, resulting in empty cells. This creates a rank deficiency in the fitted model and some predictors are thrown out.

We can avoid non-estimable cases by using `at` to restrict the reference levels to a smaller set:

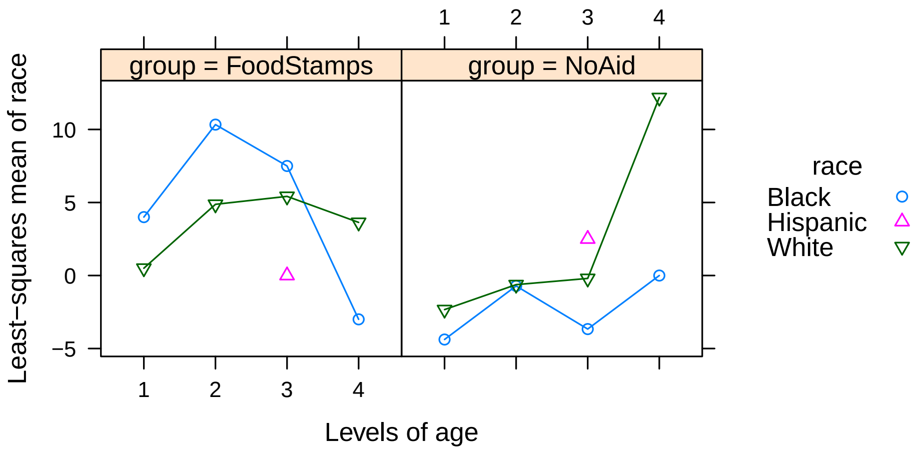

Figure 5: Predictions for the nutrition data

```
R> lsmeans(nutr.lm, ~ group*race, at = list(age = "3"))

 group     race         lsmean      SE df lower.CL upper.CL
 FoodStamps Black     7.5000e+00 2.67205 92   2.1931 12.80693
 NoAid      Black    -3.6667e+00 2.18172 92  -7.9998  0.66642
 FoodStamps Hispanic  2.1316e-14 5.34411 92 -10.6139 10.61386
 NoAid      Hispanic  2.5000e+00 3.77885 92  -5.0051 10.00513
 FoodStamps White     5.4194e+00 0.95983 92   3.5130  7.32566
 NoAid      White    -2.0000e-01 1.19498 92  -2.5733  2.17333

Confidence level used: 0.95
```

Nonetheless, the standard errors for the Hispanic mothers are enormous due to very small counts. One useful summary of the results is to narrow the scope of the reference levels to two races and the two middle age groups, where most of the data lie. However, always keep in mind that whenever we change the reference grid, we also change the definition of the LS means. Moreover, it may be more appropriate to average the two ages using weights proportional to their frequencies (23 and 64) in the data set. This may be done by changing the `fac.reduce` argument. With those ideas in mind, here are the LS means and comparisons within rows and columns:

```
R> wtavg <- function(coefs, lev) (23*coefs[1,] + 64*coefs[2,])/87
R> nutr.lsm <- lsmeans(nutr.lm, ~ group * race, fac.reduce = wtavg,
+     at = list(age=c("2","3"), race=c("Black","White")))
```

So here are the results

```
R> nutr.lsm

 group      race    lsmean     SE df lower.CL upper.CL
 FoodStamps Black  8.24896 2.9019 92   2.4856 14.01234
 NoAid      Black -2.88615 1.6914 92  -6.2455  0.47321
 FoodStamps White  5.27544 0.8649 92   3.5577  6.99322
 NoAid      White -0.31236 1.0111 92  -2.3204  1.69571

Confidence level used: 0.95
```

```
R> pairs(nutr.lsm, by = "race")

race = Black:
 contrast           estimate     SE df t.ratio p.value
 FoodStamps - NoAid  11.1351 3.5444 92   3.142  0.0023

race = White:
 contrast           estimate     SE df t.ratio p.value
 FoodStamps - NoAid   5.5878 1.3305 92   4.200  0.0001
```

```
R> pairs(nutr.lsm, by = "group")

group = FoodStamps:
 contrast      estimate     SE df t.ratio p.value
 Black - White   2.9735 3.0047 92   0.990  0.3250

group = NoAid:
 contrast      estimate     SE df t.ratio p.value
 Black - White  -2.5738 1.9706 92  -1.306  0.1948
```

The general conclusion from these analyses is that for age groups 2 and 3, the expected gains from the training are higher among families receiving food stamps. Note that this analysis is somewhat different than the results we would obtain by subsetting the data before analysis, as we are borrowing information from the other observations in estimating and testing these LS means.

2.8. Trends

The **lsmeans** package provides a function `lstrends` for estimating and comparing the slopes of fitted lines (or curves). To illustrate, consider the built-in R dataset `ChickWeight` which has data on the growths of newly hatched chicks under four different diets. The following code produces the display in Figure 6.

```
R> xyplot(weight~Time | Diet, groups = ~ Chick, data=ChickWeight, type="o",
+        layout=c(4,1))
```

Let us fit a model to these data using random slopes for each chick and allowing for a different average slope for each diet:

```
R> Chick.lmer <- lmer(weight ~ Diet * Time + (0 + Time | Chick),
+     data = ChickWeight)
```

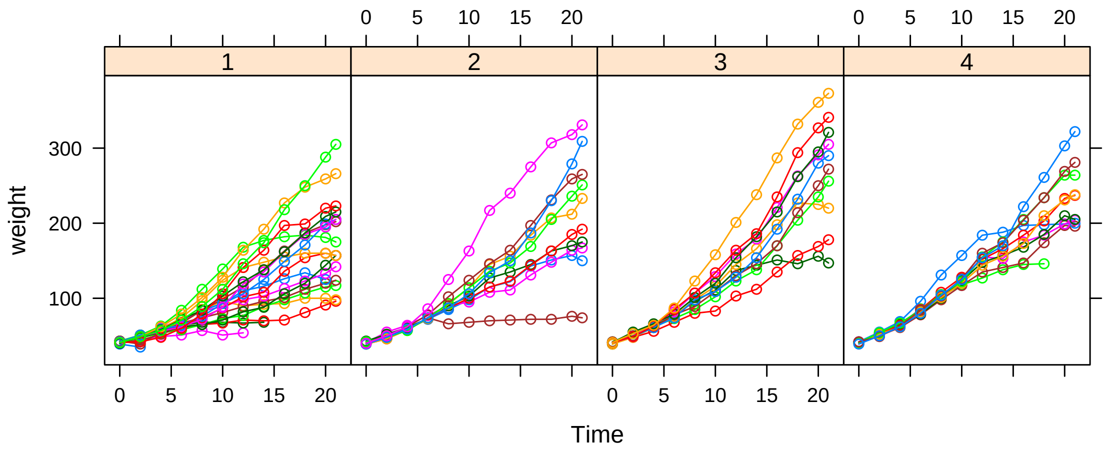

Figure 6: Growth curves of chicks, dataset `ChickWeight`.

We can then call `lsmeans` with a `trend` argument to estimate and compare the average slopes for each diet. Let's show comparisons of slopes using a compact letter display.

```
R> cld (lstrends (Chick.lmer, ~ Diet, var = "Time"))

 Diet Time.trend      SE    df lower.CL upper.CL .group
 1        6.3386 0.61050 49.85   5.1122   7.5649  1
 2        8.6091 0.83802 48.28   6.9244  10.2938  12
 4        9.5558 0.83926 48.56   7.8689  11.2428   2
 3       11.4229 0.83802 48.28   9.7382  13.1076   2

Confidence level used: 0.95
P value adjustment: tukey method for a family of 4 means
significance level used: alpha = 0.05
```

According to the Tukey HSD comparisons (with default significance level of .05), there are two groupings of slopes: Diet 1's mean slope is significantly less than 3 or 4's, Diet 2's slope is not distinguished from any other.

There is some additional trickery associated with `trend`. Consider the same model but with `Time` replaced by `log(Time + 1)`:

```
R> Chick.lmer2 <- lmer(weight ~ Diet * log(Time + 1) +
+     (0 + log(Time + 1) | Chick),  data = ChickWeight)
R> cld (lstrends (Chick.lmer2, ~ Diet, var = "log(Time + 1)"))

 Diet log(Time + 1).trend     SE     df lower.CL upper.CL .group
 1                 43.101 3.8883 122.79   35.404   50.798  1
 2                 58.493 5.3099 119.69   47.979   69.006  12
 4                 65.541 5.3365 121.74   54.977   76.106   2
 3                  75.900 5.3099 119.69   65.386   86.413   2

Confidence level used: 0.95
P value adjustment: tukey method for a family of 4 means
significance level used: alpha = 0.05
```

This compares the trends that are fitted by the model. They compare in roughly the same way, but of course the values are much higher because the transformation has compressed the scale. But we can also look at the slopes for `Time` itself:

```
R> cld (lstrends (Chick.lmer2, ~ Diet, var = "Time"))
```

```
 Diet Time.trend      SE     df lower.CL upper.CL .group
 1        3.6456 0.32889 122.79   2.9946   4.2967  1
 2        4.9475 0.44913 119.69   4.0582   5.8368  12
 4        5.5437 0.45138 121.74   4.6501   6.4373   2
 3        6.4198 0.44913 119.69   5.5306   7.3091   2

Confidence level used: 0.95
P value adjustment: tukey method for a family of 4 means
significance level used: alpha = 0.05
```

These results are somewhat comparable to those we obtained with the first model. We will get a different set of slopes at different `Times`, because the fitted trends are curved with respect to `Time`.

3. Multivariate models

The `MOats` dataset provided in the package gives the `Oats` data mentioned previously, but with a multivariate response variable `yield` with four columns representing the yields of each plot with the four levels of nitrogen. We fit a model to these data

```
R> MOats.mlm <- lm(yield ~ Block + Variety, data = MOats)
```

This model assumes an unstructured covariance matrix on each plot. Here is its reference grid:

```
R> ref.grid(MOats.mlm)
```

```
'ref.grid' object with variables:
    Block = VI, V, III, IV, II, I
    Variety = Golden Rain, Marvellous, Victory
    rep.meas = multivariate response levels: 1, 2, 3, 4
    yield = multivariate response with means:
         79.389,  98.889, 114.222, 123.389
```

The `ref.grid` function "flattens" the multivariate results by creating a pseudo-factor to account for the dimensions of the multivariate response. By default, the pseudo-factor is named `rep.meas` with integer levels. It's often better to specify a more meaningful name and levels:

```
R> MOats.rg <- ref.grid(MOats.mlm, mult.levs = list(nitro = c(0,.2,.4,.6)))
R> MOats.rg

'ref.grid' object with variables:
    Block = VI, V, III, IV, II, I
    Variety = Golden Rain, Marvellous, Victory
    nitro = multivariate response levels: 0.0, 0.2, 0.4, 0.6
    yield = multivariate response with means:
         79.389,  98.889, 114.222, 123.389
```

Now we can obtain LS means and such just as we did previously

```
R> ( MOats.lsm <- lsmeans(MOats.rg, ~ nitro | Variety) )

Variety = Golden Rain:
 nitro  lsmean     SE df lower.CL upper.CL
   0.0  80.000 5.5406 10   67.655   92.345
   0.2  98.500 6.6020 10   83.790  113.210
   0.4 114.667 8.6954 10   95.292  134.041
   0.6 124.833 7.3032 10  108.561  141.106

Variety = Marvellous:
 nitro  lsmean     SE df lower.CL upper.CL
   0.0  86.667 5.5406 10   74.321   99.012
   0.2 108.500 6.6020 10   93.790  123.210
   0.4 117.167 8.6954 10   97.792  136.541
   0.6 126.833 7.3032 10  110.561  143.106

Variety = Victory:
 nitro  lsmean     SE df lower.CL upper.CL
   0.0  71.500 5.5406 10   59.155   83.845
   0.2  89.667 6.6020 10   74.956  104.377
   0.4 110.833 8.6954 10   91.459  130.208
   0.6 118.500 7.3032 10  102.227  134.773

Confidence level used: 0.95

R> ( MOats.pcon <- contrast(MOats.lsm, "poly") )

Variety = Golden Rain:
 contrast  estimate      SE df t.ratio p.value
 linear    150.6667 24.2087 10   6.224  0.0001
 quadratic  -8.3333  9.5169 10  -0.876  0.4018
 cubic      -3.6667 31.9574 10  -0.115  0.9109

Variety = Marvellous:
 contrast  estimate      SE df t.ratio p.value
 linear    129.1667 24.2087 10   5.336  0.0003
 quadratic -12.1667  9.5169 10  -1.278  0.2300
 cubic      14.1667 31.9574 10   0.443  0.6670

Variety = Victory:
 contrast  estimate      SE df t.ratio p.value
 linear    162.1667 24.2087 10   6.699  0.0001
 quadratic -10.5000  9.5169 10  -1.103  0.2957
 cubic     -16.5000 31.9574 10  -0.516  0.6169
```

We can even obtain contrasts of contrasts to obtain interaction contrasts. In the following, we compare the polynomial contrasts among the varieties:

```
R> pairs(MOats.pcon, by = "contrast")

contrast = linear:
 contrast1                estimate     SE df t.ratio p.value
 Golden Rain - Marvellous  21.5000 34.236 10   0.628  0.8085
 Golden Rain - Victory    -11.5000 34.236 10  -0.336  0.9401
 Marvellous - Victory     -33.0000 34.236 10  -0.964  0.6147

contrast = quadratic:
 contrast1                estimate     SE df t.ratio p.value
 Golden Rain - Marvellous   3.8333 13.459 10   0.285  0.9565
 Golden Rain - Victory      2.1667 13.459 10   0.161  0.9858
 Marvellous - Victory      -1.6667 13.459 10  -0.124  0.9916

contrast = cubic:
 contrast1                estimate     SE df t.ratio p.value
 Golden Rain - Marvellous -17.8333 45.195 10  -0.395  0.9184
 Golden Rain - Victory     12.8333 45.195 10   0.284  0.9567
 Marvellous - Victory      30.6667 45.195 10   0.679  0.7809

P value adjustment: tukey method for a family of 3 means
```

3.1. GLMM example

The dataset `cbpp` in the `lme4` package, originally from Lesnoff, Laval, Bonnet, Abdicho, Workalemahu, Kifle, Peyraud, Lancelot, and Thiaucourt (2004), 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 in **lme4** 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.

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

Let us save the `summary` results from `lsmeans`, then add the inverse logits of the predictions and the estimated odds ratios for the comparisons as an aid in interpretation.

```
R> cbpp.lsm <- lsmeans(cbpp.glmer, ~ period)
R> cbpp.sum <- summary(cbpp.lsm)
R> cbpp.sum$pred.incidence <- 1 - 1 / (1 + exp(cbpp.sum$lsmean))
R> cbpp.sum

 period  lsmean      SE df  asymp.LCL asymp.UCL pred.incidence
 1      -1.5003 0.28876 NA   -2.0662  -0.93433       0.182382
 2      -2.7268 0.38097 NA   -3.4735  -1.98010       0.061411
 3      -2.8291 0.39940 NA   -3.6119  -2.04631       0.055771
 4      -3.3665 0.51939 NA   -4.3845  -2.34856       0.033358

Confidence level used: 0.95

R> cbpp.con <- summary(contrast(cbpp.lsm, "trt.vs.ctrl1"))
R> cbpp.con$odds.ratio <- exp(cbpp.con$estimate)
R> cbpp.con

 contrast estimate      SE df z.ratio p.value odds.ratio
 2 - 1     -1.2265 0.47345 NA -2.5905  0.0285    0.29332
 3 - 1     -1.3288 0.48839 NA -2.7208  0.0194    0.26479
 4 - 1     -1.8662 0.59056 NA -3.1601  0.0047    0.15470

P value adjustment: sidak method for 3 tests
P values are asymptotic
```

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.

4. Conclusions

lsmeans helps extend R's capabilities for the analysis of experimental data, especially for those users who have relied on SAS's least-squares means provisions. It goes beyond SAS in a few useful ways—for example, allowing for factor combinations even when an interaction is not in the model, and estimating trends. It provides a flexible and relatively simple way to obtain predictions from a linear model, or marginal averages thereof; and it also provides an extension of **multcomp**'s capabilities along these lines.

References

Bates D, Maechler M, Bolker B, Walker S (2013). ***lme4**: Linear Mixed-effects Models Using **Eigen** and S4.* R package version 1.1-0, URL `http://lme4.r-forge.r-project.org/`.

Graves S, Piepho HP, Selzer L, Dorai-Raj S (2012). *multcompView: Visualizations of Paired Comparisons.* R package version 0.1-5, URL `http://CRAN.R-project.org/package=multcompView`.

Halekoh U, Højsgaard S (2013). ***pbkrtest**: Parametric Bootstrap and Kenward Roger Based Methods for Mixed Model Comparison.* R package version 0.3-8, URL `http://CRAN.R-project.org/package=pbkrtest`.

Harvey W (1976). "Use of the HARVEY Procedure." In *SUGI Proceedings.* URL `http://www.sascommunity.org/sugi/SUGI76/Sugi-76-20%20Harvey.pdf`.

Hothorn T, Bretz F, Westfall P (2013). ***multcomp**: Simultaneous Inference in General Parametric Models.* R package version 1.3-1, URL `http://CRAN.R-project.org/package=multcomp`.

Lenth RV (2014). *lsmeans: Least-Squares Means.* R package version 2.00.

Lesnoff M, Laval G, Bonnet P, Abdicho S, Workalemahu A, Kifle D, Peyraud A, Lancelot R, Thiaucourt F (2004). "Within-Herd Spread of Contagious Bovine Pleuropneumonia in Ethiopian Highlands." *Preventive Veterinary Medicine*, **64**, 27–40.

Milliken GA, Johnson DE (1984). *Analysis of Messy Data – Volume I: Designed Experiments.* Van Nostrand. ISBN 0-534-02713-7.

Oehlert G (2000). *A First Course in Design and Analysis of Experiments.* W. H. Freeman.

Pinheiro J, Bates D, R-core (2013). ***nlme**: Linear and Nonlinear Mixed Effects Models.* R package version 3.1-113, URL `http://CRAN.R-project.org/package=nlme`.

Sarkar D (2013). ***lattice**: Lattice Graphics.* R package version 0.20-24, URL `http://CRAN.R-project.org/package=lattice`.

SAS Institute Inc (2012). "`LSMEANS` Statement." In *SAS/STAT(R) 9.3 User's Guide.* URL `http://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_introcom_a0000003362.htm`.

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

Affiliation:

Russell V. Lenth
Department of Statistics and Actuarial Science
241 Schaeffer Hall
The University of Iowa
Iowa City, IA 52242   USA
E-mail: `russell-lenth@uiowa.edu`
URL: `http://www.stat.uiowa.edu/~rlenth/`