Using lsmeans

Russell V. Lenth

June 24, 2014

1 Introduction

Least-squares means (LS means for short) for a linear model are simply predictions—or marginal averages thereof—over a regular grid of predictor settings which I call the *reference grid.* They date back at least to 1976 when LS means were incorporated in the contributed SAS procedure named `HARVEY` (Harvey, 1976). Later, they were incorporated via `LSMEANS` statements in the regular SAS releases.

In simple analysis-of-covariance models, LS means are the same as covariate-adjusted means. In unbalanced factorial experiments, LS means for each factor mimic the marginal means but are adjusted to bias due to imbalance. The latter interpretation is quite similar to the "unweighted means" method for unbalanced data, as presented in old design books.

In any case, the most important things to remember are:

- LS means are computed relative to a *reference grid.*
- Once the reference grid is understood, LS means are simply predictions on this grid, or marginal averages of these predictions.

If you understand these points, then you will know what you are getting, and can judge whether or not LS means are appropriate for your analysis.

2 The reference grid

Since the reference grid is fundamental, it is our starting point. For each predictor in the model, we define a set of one or more *reference levels.* The reference grid is then the set of all combinations of reference levels. If not specified explicitly, the default reference levels are obtained as follows:

- For each predictor that is a factor, its reference levels are the unique levels of that factor.
- Each numeric predictor has just one reference level—its mean over the dataset.

So the reference grid depends on both the model and the dataset.

2.1 Example: Orange sales

To illustrate, consider the `oranges` data provided with lsmeans. This dataset has sales of two varieties of oranges (response variables `sales1` and `sales2`) at 6 stores (factor `store`), over a period of 6 days (factor `day`). The prices of the oranges (covariates `price1` and `price2`) fluctuate in the different stores and the different days. There is just one observation on each store on each day.

For starters, let's consider an additive covariance model for sales of the first variety, with the two factors and both `price1` and `price2` as covariates (since the price of the other variety could also affect sales).

```
R> library(lsmeans)
R> oranges.lm1 = lm(sales1 ~ price1 + price2 + day + store, data = oranges)
R> anova(oranges.lm1)

Analysis of Variance Table

Response: sales1
          Df Sum Sq Mean Sq F value    Pr(>F)
price1     1 516.59  516.59 29.0996 1.763e-05
price2     1  62.73   62.73  3.5334  0.072873
day        5 422.22   84.44  4.7567  0.003951
store      5 223.83   44.77  2.5217  0.058346
Residuals 23 408.31   17.75
```

The `ref.grid` function in lsmeans may be used to establish the reference grid. Here is the default one:

```
R> ( oranges.rg1 = ref.grid(oranges.lm1) )

'ref.grid' object with variables:
    price1 = 51.222
    price2 = 48.556
    day = 1, 2, 3, 4, 5, 6
    store = 1, 2, 3, 4, 5, 6
```

As outlined above, the two covariates `price1` and `price2` have their means as their sole reference level; and the two factors have their levels as reference levels. The reference grid thus consists of the $1 \times 1 \times 6 \times 6 = 36$ combinations of these reference levels. LS means are based on predictions on this reference grid, which we can obtain using `predict` or `summary`:

```
R> summary(oranges.rg1)

   price1   price2 day store prediction       SE df
 51.22222 48.55556 1   1       2.918413 2.717559 23
 51.22222 48.55556 2   1       3.848804 2.701335 23
 51.22222 48.55556 3   1      11.018569 2.534556 23
 51.22222 48.55556 4   1       6.096286 2.651370 23
 51.22222 48.55556 5   1      12.795800 2.444597 23
 51.22222 48.55556 6   1       8.748779 2.786176 23
 51.22222 48.55556 1   2       4.961475 2.377742 23
 51.22222 48.55556 2   2       5.891866 2.335579 23
 51.22222 48.55556 3   2      13.061630 2.416451 23
 51.22222 48.55556 4   2       8.139348 2.352186 23
 51.22222 48.55556 5   2      14.838862 2.466155 23
 51.22222 48.55556 6   2      10.791841 2.337599 23
 51.22222 48.55556 1   3       3.200891 2.377742 23
 51.22222 48.55556 2   3       4.131282 2.335579 23
 51.22222 48.55556 3   3      11.301047 2.416451 23
 51.22222 48.55556 4   3       6.378765 2.352186 23
 51.22222 48.55556 5   3      13.078278 2.466155 23
 51.22222 48.55556 6   3       9.031258 2.337599 23
 51.22222 48.55556 1   4       6.198757 2.363673 23
 51.22222 48.55556 2   4       7.129148 2.352186 23
 51.22222 48.55556 3   4      14.298913 2.431679 23
 51.22222 48.55556 4   4       9.376630 2.388653 23
 51.22222 48.55556 5   4      16.076144 2.519089 23
 51.22222 48.55556 6   4      12.029123 2.364688 23
 51.22222 48.55556 1   5       5.543218 2.363116 23
 51.22222 48.55556 2   5       6.473609 2.330670 23
 51.22222 48.55556 3   5      13.643374 2.363673 23
 51.22222 48.55556 4   5       8.721091 2.337599 23
 51.22222 48.55556 5   5      15.420605 2.395544 23
 51.22222 48.55556 6   5      11.373584 2.352318 23
 51.22222 48.55556 1   6      10.563739 2.366683 23
 51.22222 48.55556 2   6      11.494130 2.339254 23
 51.22222 48.55556 3   6      18.663895 2.347839 23
 51.22222 48.55556 4   6      13.741613 2.341304 23
 51.22222 48.55556 5   6      20.441126 2.370343 23
 51.22222 48.55556 6   6      16.394106 2.370539 23
```

2.2 LS means as marginal averages

The ANOVA indicates there is a significant `day` effect after adjusting for the covariates, so we might want to compare the days. The `lsmeans` function can do this:

```
R> lsmeans(oranges.rg1, "day")   ## or lsmeans(oranges.lm1, "day")

 day    lsmean       SE df  lower.CL  upper.CL
 1    5.564415 1.768083 23  1.906856  9.221974
 2    6.494807 1.728959 23  2.918183 10.071430
 3   13.664571 1.751505 23 10.041308 17.287835
 4    8.742289 1.733920 23  5.155403 12.329175
 5   15.441803 1.785809 23 11.747576 19.136029
 6   11.394782 1.766726 23  7.740031 15.049533

Results are averaged over the levels of: store
Confidence level used: 0.95
```

These results, as indicated in the annotation in the output, are in fact the averages of the predictions shown earlier, for each day, over the 6 stores. The above LS means are not the same as the marginal means of the data:

```
R> with(oranges, tapply(sales1, day, mean))

        1         2         3         4         5         6
 7.872750  7.100600 13.758600  8.042467 12.924600 11.603650
```

These unadjusted means are biased by having different `price1` and `price2` values on each day, whereas the LS means adjust for bias by using predictions at uniform `price1` and `price2` values.

Note that you may call `lsmeans` with either the reference grid or the model. If the model is given, then the first thing it does is create the reference grid; so if you already have the reference grid, as in this example, it's more efficient to make use of it.

2.3 Altering the reference grid

The `at` argument may be used to override defaults in the reference grid. You may specify this argument either in a `ref.grid` call or an `lsmeans` call; and you should specify `list` with named sets of reference levels. Here is a silly example:

```
R> lsmeans(oranges.lm1, "day", at = list(price1 = 50,
       price2 = c(40,60), day = c("2","3","4")) )

 day    lsmean       SE df  lower.CL upper.CL
 2    7.724698 1.735165 23  4.135235 11.31416
 3   14.894463 1.751037 23 11.272167 18.51676
 4    9.972180 1.766131 23  6.318660 13.62570

Results are averaged over the levels of: price2, store
Confidence level used: 0.95
```

Here, we restricted the results to three of the days, and used different prices. One possible surprise is to note that the predictions are averaged over the two `price2` values. That is because `price2` is no longer a single reference level, and we average over the levels of all factors not used to split-out the LS means. This is probably not what we want. To get separate sets of predictions for each `price2`, you need to specify it as another factor or as a `by` factor in the `lsmeans` call (we will save the result for later discussion):

```
R> org.lsm = lsmeans(oranges.lm1, "day", by ="price2",
       at = list(price1 = 50, price2 = c(40,60), day = c("2","3","4")) )
R> org.lsm

price2 = 40:
 day    lsmean       SE df  lower.CL upper.CL
 2    6.236227 1.887106 23  2.332452 10.14000
 3   13.405992 2.119376 23  9.021730 17.79025
 4    8.483710 1.866510 23  4.622540 12.34488

price2 = 60:
 day    lsmean       SE df  lower.CL upper.CL
 2    9.213169 2.109448 23  4.849443 13.57689
 3   16.382933 1.905216 23 12.441693 20.32417
 4   11.460651 2.178054 23  6.955003 15.96630

Results are averaged over the levels of: store
Confidence level used: 0.95
```

Note: We could have obtained the same results using any of these:

```
R> lsmeans(oranges.lm1, ~ day | price, at = ... )          # Ex 1
R> lsmeans(oranges.lm1, c("day","price2"), at = ... )      # Ex 2
R> lsmeans(oranges.lm1, ~ day * price, at = ... )          # Ex 3
```

Ex 1 illustrates the formula method for specifying factors, which is more compact. The | character replaces the by specification. Ex 2 and Ex 3 produce the same results, but their results are displayed as one table (with columns for day and price) rather than as two separate tables.

3 Working with the results

The ref.grid function produces an object of class "ref.grid", and the lsmeans function produces an object of class "lsmobj", which is a subclass of "ref.grid". There is really no practical difference between these two classes except for their show methods—what is displayed if you just call the functions—and the fact that an "lsmobj" is not (necessarily) a true reference grid as defined earlier in this tutorial. Let's use the str function to examine the "lsmobj" object produced just above:

```
R> str(org.lsm)

'lsmobj' object with variables:
    day = 2, 3, 4
    price2 = 40, 60
```

We no longer see the reference levels for all predictors in the model—only the levels of day and price2. These *act* like reference levels, but they do not define the reference grid upon which the predictions are based.

There are several methods for "ref.grid" (and hence also for "lsmobj") objects. One you have seen already is summary. It has a number of arguments: see its help page. In the following call, we summarize days.lsm differently than before. We will also save the object produced by summary for further discussion.

```
R> ( org.sum = summary(org.lsm, infer=c(TRUE,TRUE),
                       level=.90, adjust="bon", by = "day") )

day = 2:
 price2    lsmean       SE df  lower.CL upper.CL t.ratio p.value
     40  6.236227 1.887106 23  2.332452 10.14000   3.305  0.0062
     60  9.213169 2.109448 23  4.849443 13.57689   4.368  0.0005

day = 3:
 price2    lsmean       SE df  lower.CL upper.CL t.ratio p.value
     40 13.405992 2.119376 23  9.021730 17.79025   6.325  <.0001
     60 16.382933 1.905216 23 12.441693 20.32417   8.599  <.0001

day = 4:
 price2    lsmean       SE df  lower.CL upper.CL t.ratio p.value
     40  8.483710 1.866510 23  4.622540 12.34488   4.545  0.0003
     60 11.460651 2.178054 23  6.955003 15.96630   5.262  <.0001
```

```
Results are averaged over the levels of: store
Confidence level used: 0.9
Confidence-level adjustment: bonferroni method for 2 tests
P value adjustment: bonferroni method for 2 tests
```

The `infer` argument caused both confidence intervals and tests to be produced. The default confidence level of .95 was overridden; a Bonferroni adjustment was applied to both the intervals and the *P* values; and the tables are organized the opposite way from what we saw before.

What kind of object was produced by `summary`? Let's see:

```
R> class(org.sum)
```

```
[1] "summary.ref.grid" "data.frame"
```

The `"summary.ref.grid"` class is an extension of `"data.frame"`. It includes some attributes that, among other things, cause the messages seen in the example to appear when the object is displayed. But it can also be used as a `"data.frame"` if you just want to use the results computationally. For example, suppose we want to convert the LS means from dollars to Russian rubles:

```
R> cbind(org.sum[, 1:2], lsrubles = org.sum$lsmean * 35.7) # as of April 22, 2014
```

```
  day price2 lsrubles
1   2     40 222.6333
2   3     40 478.5939
3   4     40 302.8684
4   2     60 328.9101
5   3     60 584.8707
6   4     60 409.1452
```

Observe that, as a data frame, the summary is just one table with six rows, rather than a collection of three tables, and it contains a column for all reference variables, including any `by` variables.

Besides `str` and `summary`, there is also a `confint` method (same is `summary` with `infer=c(TRUE,FALSE)`) and a `test` method (same as `summary` with `infer=c(FALSE,TRUE)`). There is also an `update` method which may be used for changing the default display settings. For example:

```
R> org.lsm2 = update(org.lsm, by.vars = NULL, level = .99)
R> org.lsm2
```

```
 day price2    lsmean       SE df   lower.CL upper.CL
 2       40  6.236227 1.887106 23  0.9384879 11.53397
 3       40 13.405992 2.119376 23  7.4561934 19.35579
 4       40  8.483710 1.866510 23  3.2437905 13.72363
 2       60  9.213169 2.109448 23  3.2912404 15.13510
 3       60 16.382933 1.905216 23 11.0343510 21.73152
 4       60 11.460651 2.178054 23  5.3461217 17.57518

Results are averaged over the levels of: store
Confidence level used: 0.99
```

4 Contrasts and comparisons

4.1 Contrasts in general

Often, people want to do pairwise comparisons of LS means, or compute other contrasts among them. This is the purpose of the `contrast` function, which uses an `"lsmobj"` object as input. There are several standard contrast families such as `"pairwise"`, `"trt.vs.ctrl"`, and `"poly"`. In the following command, we request `"eff"` contrasts, which are differences between each mean and the overall mean:

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

price2 = 40:
 contrast  estimate       SE df t.ratio p.value
 2 effect -3.139082 1.415290 23  -2.218  0.0551
 3 effect  4.030682 1.442746 23   2.794  0.0310
 4 effect -0.891600 1.421346 23  -0.627  0.5366

price2 = 60:
 contrast  estimate       SE df t.ratio p.value
 2 effect -3.139082 1.415290 23  -2.218  0.0551
 3 effect  4.030682 1.442746 23   2.794  0.0310
 4 effect -0.891600 1.421346 23  -0.627  0.5366

Results are averaged over the levels of: store
P value adjustment: fdr method for 3 tests
```

Note that this remembers the `by` specification from before, and obtains the effects for each group. In this example, since it is an additive model, we obtain the same results in each group. This isn't wrong, it's just redundant.

Another popular method is Dunnett-style contrasts, where a particular LS mean is compared with each of the others. This is done using `"trt.vs.ctrl"`. In the following, we obtain (again) the LS means for days, and compare each with the average of the LS means on day 5 and 6.

```
R> days.lsm = lsmeans(oranges.rg1, "day")
R> contrast(days.lsm, "trt.vs.ctrl", ref = c(5,6))

 contrast       estimate       SE df t.ratio p.value
 1 - avg(5,6) -7.8538769 2.194243 23  -3.579  0.0063
 2 - avg(5,6) -6.9234858 2.127341 23  -3.255  0.0139
 3 - avg(5,6)  0.2462789 2.155529 23   0.114  0.9999
 4 - avg(5,6) -4.6760034 2.110761 23  -2.215  0.1397

Results are averaged over the levels of: store
P value adjustment: sidak method for 4 tests
```

For convenience, `"trt.vs.ctrl1"` or `"trt.vs.ctrlk"` methods are provided for use in lieu of `ref` for comparing with the first and the last LS means.

You may have noticed that by default, `lsmeans` results are displayed with confidence intervals while `contrast` results are displayed with *t* tests. You can easily override this; for example,

```
R> confint(contrast(days.lsm, "trt.vs.ctrlk"))
```

(Results not shown.)

In the above examples, a default multiplicity adjustment is determined from the contrast method. This may be overridden by adding an `adjust` argument.

4.2 Pairwise comparisons

Often, users want pairwise comparisons among the LS means. These may be obtained by specifying `"pairwise"` or `"revpairwise"` in the call to `contrast`. For group labels A, B, C, `"pairwise"` generates the comparisons $A - B, A - C, B - C$ while `"revpairwise"` generates $B - A, C - A, C - B$. As a convenience, the `pairs` method works just like `contrasts` with `"pairwise"`:

```
R> pairs(org.lsm)

price2 = 40:
 contrast  estimate       SE df t.ratio p.value
 2 - 3    -7.169765 2.479697 23  -2.891  0.0216
 2 - 4    -2.247482 2.442340 23  -0.920  0.6333
 3 - 4     4.922282 2.490068 23   1.977  0.1406

price2 = 60:
 contrast  estimate       SE df t.ratio p.value
 2 - 3    -7.169765 2.479697 23  -2.891  0.0216
 2 - 4    -2.247482 2.442340 23  -0.920  0.6333
 3 - 4     4.922282 2.490068 23   1.977  0.1406

Results are averaged over the levels of: store
P value adjustment: tukey method for a family of 3 means
```

There is also a `cld` (compact letter display) method that lists the LS means along with grouping symbols for pairwise contrasts. It requires the multcompView package (Graves *et al.*, 2012) to be installed.

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

 day    lsmean       SE df  lower.CL  upper.CL .group
 1    5.564415 1.768083 23  1.906856  9.221974  1
 2    6.494807 1.728959 23  2.918183 10.071430  1
 4    8.742289 1.733920 23  5.155403 12.329175  12
 6   11.394782 1.766726 23  7.740031 15.049533  12
 3   13.664571 1.751505 23 10.041308 17.287835   2
 5   15.441803 1.785809 23 11.747576 19.136029   2

Results are averaged over the levels of: store
Confidence level used: 0.95
P value adjustment: tukey method for a family of 6 means
significance level used: alpha = 0.1
```

Two LS means that share one or more of the same grouping symbols are not significantly different at the stated value of `alpha`, after applying the multiplicity adjustment (in this case Tukey's HSD). By default, the LS means are ordered in this display, but this may be overridden with the argument `sort=FALSE`. `cld` returns a `"summary.ref.grid"` object, not an `lsmobj`.

5 Multivariate models

The `oranges` data has two response variables. Let's try a multivariate model for predicting the sales of the two varieties of oranges, and see what we get if we call `ref.grid`:

```
R> oranges.mlm = lm(cbind(sales1,sales2) ~ price1 + price2 + day + store,
                    data = oranges)
R> ref.grid(oranges.mlm)

'ref.grid' object with variables:
    price1 = 51.222
    price2 = 48.556
    day = 1, 2, 3, 4, 5, 6
    store = 1, 2, 3, 4, 5, 6
    rep.meas = multivariate response levels: sales1, sales2
```

What happens is that the multivariate response is treated like an additional factor, by default named `rep.meas`. In turn, it can be used, to specify levels for LS means. Here we rename the multivariate response to `"variety"` and obtain `day` means (and a compact letter display for comparisons thereof) for each `variety`:

```
R> org.mlsm = lsmeans(oranges.mlm, ~ day | variety, mult.name="variety")
R> cld(org.mlsm, sort = FALSE)

variety = sales1:
 day    lsmean       SE df   lower.CL  upper.CL .group
 1    5.564415 1.768083 23  1.9068563  9.221974  1
 2    6.494807 1.728959 23  2.9181833 10.071430  12
 3   13.664571 1.751505 23 10.0413078 17.287835   23
 4    8.742289 1.733920 23  5.1554026 12.329175  123
 5   15.441803 1.785809 23 11.7475762 19.136029    3
 6   11.394782 1.766726 23  7.7400309 15.049533  123

variety = sales2:
 day    lsmean       SE df   lower.CL  upper.CL .group
 1    7.715664 2.326485 23  2.9029623 12.528365  12
 2    3.976446 2.275004 23 -0.7297584  8.682650  1
 3   16.597814 2.304671 23 11.8302400 21.365389   2
 4   11.044540 2.281532 23  6.3248316 15.764248  12
 5   14.990786 2.349808 23 10.1298371 19.851735   2
 6   12.048784 2.324699 23  7.2397770 16.857790  12

Results are averaged over the levels of: store
Confidence level used: 0.95
```

```
P value adjustment: tukey method for a family of 6 means
significance level used: alpha = 0.05
```

6 Contrasts of contrasts

With the preceding model, we might want to compare the two varieties on each day:

```
R> org.vardiff = update(pairs(org.mlsm, by = "day"), by = NULL)
```

The results (not yet shown) will comprise the six `sales1-sales2` differences, one for each day. The two `by` specifications seems odd, but the one in `pairs` specifies doing a separate comparison for each day, and the one in `update` asks that we convert it to one table with six rows, rather than 6 tables with one row each. Now, let's compare these differences to see if they vary from day to day.

```
R> cld(org.vardiff)
```

```
 contrast        day   estimate       SE df t.ratio p.value .group
 sales1 - sales2 3   -2.9332431 2.694111 23  -1.089  0.2875  1
 sales1 - sales2 4   -2.3022511 2.667062 23  -0.863  0.3969  1
 sales1 - sales2 1   -2.1512483 2.719612 23  -0.791  0.4370  1
 sales1 - sales2 6   -0.6540015 2.717524 23  -0.241  0.8120  1
 sales1 - sales2 5    0.4510165 2.746876 23   0.164  0.8710  1
 sales1 - sales2 2    2.5183608 2.659431 23   0.947  0.3535  1

Results are averaged over the levels of: store
P value adjustment: tukey method for a family of 6 means
significance level used: alpha = 0.05
```

7 Interfacing with multcomp

The multcomp package (Hothorn *et al.*, 2014) supports more exacting corrections for simultaneous inference than are available in lsmeans. Its `glht` (general linear hypothesis testing) function and associated `"glht"` class are similar in some ways to `lsmeans` and `"lsmobj"` objects, respectively. So we provide methods such as `as.glht` for working with `glht` so as to obtain "exact" inferences. To illustrate, I'll compare some simultaneous confidence intervals using the two packages. First, using a Bonferroni correction on the LS means for `day` in the `oranges` model:

```
R> confint(days.lsm, adjust = "bon")
```

```
 day    lsmean       SE df   lower.CL upper.CL
 1    5.564415 1.768083 23  0.4612605 10.66757
 2    6.494807 1.728959 23  1.5045761 11.48504
 3   13.664571 1.751505 23  8.6092668 18.71988
 4    8.742289 1.733920 23  3.7377391 13.74684
 5   15.441803 1.785809 23 10.2874881 20.59612
 6   11.394782 1.766726 23  6.2955448 16.49402

Results are averaged over the levels of: store
Confidence level used: 0.95
Confidence-level adjustment: bonferroni method for 6 tests
```

And now using multcomp:

```
R> library(multcomp)
R> confint(as.glht(days.lsm))

         Simultaneous Confidence Intervals

Fit: NULL

Quantile = 2.8605
95% family-wise confidence level


Linear Hypotheses:
       Estimate lwr     upr
1 == 0  5.5644   0.5068 10.6221
2 == 0  6.4948   1.5491 11.4405
3 == 0 13.6646   8.6543 18.6748
4 == 0  8.7423   3.7824 13.7022
5 == 0 15.4418  10.3335 20.5502
6 == 0 11.3948   6.3410 16.4485
```

The latter intervals are somewhat narrower, which is expected since the Bonferroni method is conservative.

The lsmeans package also provides an `lsm` function that can be called as the second argument of `glht`:

```
R> summary(glht(oranges.lm1, lsm("day", contr="eff")), test = adjusted("free"))

         Simultaneous Tests for General Linear Hypotheses

Fit: lm(formula = sales1 ~ price1 + price2 + day + store, data = oranges)

Linear Hypotheses:
              Estimate Std. Error t value Pr(>|t|)
1 effect == 0   -4.653      1.623  -2.867   0.0400
2 effect == 0   -3.722      1.580  -2.356   0.0989
3 effect == 0    3.447      1.605   2.149   0.1164
4 effect == 0   -1.475      1.585  -0.930   0.5848
5 effect == 0    5.225      1.642   3.182   0.0232
6 effect == 0    1.178      1.621   0.726   0.5848
(Adjusted p values reported -- free method)
```

An additional detail: If there is a `by` variable in effect, `glht` or `as.glht` returns a `list` of `glht` objects—one for each `by` level. There is a courtesy `summary` method for this `"glht.list"` class to make things a bit more user-friendly. Recall the earlier example result `org.lsm`, which contains informations for LS means for three `days` at each of two values of `price2`. Suppose we are interested in pairwise comparisons of these LS means, by `price2`. If we call

```
R> summary(as.glht(pairs(org.lsm)))
```

(results not displayed) we will obtain two `glht` objects with three contrasts each, so that the results shown will incorporate multiplicity adjustments for each family of three contrasts. If, on the other hand, we want to consider those six contrasts as one family, use

```
R> summary(as.glht(pairs(org.lsm), by = NULL))
```

... and note (look carefully at the parentheses) that this is *not* the same as

```
R> summary(as.glht(pairs(org.lsm, by = NULL)))
```

which removes the `by` grouping *before* the pairwise comparisons are generated, thus yielding $\binom{6}{2} = 15$ contrasts instead of just six.

8 A new example: Oat yields

You've probably seen enough about sales of oranges by now. To illustrate some new features, let's turn to a new example. The `Oats` dataset in the nlme (Pinheiro *et al.*, 2014) has the results of a split-plot experiment discussed in Yates (1935). The experiment was conducted on six blocks (factor `Block`). Each block was divided into three plots, which were randomized to three varieties (factor `Variety`) of oats. Each plot was divided into subplots and randomized to four levels of nitrogen (variable `nitro`). The response, `yield`, was measured once on each subplot after a suitable growing period.

We will fit a model using the `lmer` function in the lme4 package (Bates *et al.*, 2014). This will be a mixed model with random intercepts for `Block` and `Block:Variety` (which identifies the plots). I have elected to apply a logarithmic transformation to the response variable (mostly for illustration purposes, though it does produce a good fit to the data). Note that `nitro` is stored as a numeric variable, but we want to consider it as a factor in this initial model.

```
R> data("Oats", package = "nlme")
R> library("lme4")
R> Oats.lmer = lmer(log(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 0.07501 0.03751  2.0084
factor(nitro)          3 2.13505 0.71168 38.1098
Variety:factor(nitro)  6 0.04508 0.00751  0.4024
```

Apparently, the interaction is not needed. But perhaps we can simplify it further by using only a linear or quadratic trend in `nitro`. We can find out by looking at polynomial contrasts:

```
R> contrast(lsmeans(Oats.lmer, "nitro"), "poly")
```

```
NOTE: Results may be misleading due to involvement in interactions

 contrast      estimate         SE df t.ratio p.value
 linear     1.505651287 0.14404685 45  10.453  <.0001
 quadratic -0.145109968 0.06441971 45  -2.253  0.0292
 cubic      0.002731979 0.14404685 45   0.019  0.9850

Results are averaged over the levels of: Variety
```

(A message is issued when we average over predictors that interact with those that delineate the LS means. In this case, it is not a serious problem because the interaction is weak.) Both the linear and quadratic contrasts are pretty significant. All this suggests fitting an additive model where `nitro` is included as a numeric predictor with a quadratic trend.

```
R> Oats.lmer2 = lmer(log(yield) ~ Variety + poly(nitro,2) + (1|Block/Variety),
                     data = Oats)
```

The predictions from this model. Remember that `nitro` is now quantitative, and I want to see predictions at the four unique `nitro` values rather than at the average of `nitro`. This may be done using `at` as illustrated earlier, or a shortcut is to specify `cov.reduce` as `FALSE` to tell `ref.grid` to use all the unique values of numeric predictors.

```
R> Oats.lsm2 = lsmeans(Oats.lmer2, ~ nitro | Variety, cov.reduce = FALSE)
R> Oats.lsm2

Variety = Golden Rain:
 nitro   lsmean         SE    df lower.CL upper.CL
   0.0 4.354578 0.07703285 11.77 4.186367 4.522789
   0.2 4.577698 0.07453633 10.34 4.412352 4.743044
   0.4 4.728263 0.07453633 10.34 4.562917 4.893609
   0.6 4.806273 0.07703285 11.77 4.638062 4.974484

Variety = Marvellous:
 nitro   lsmean         SE    df lower.CL upper.CL
   0.0 4.412227 0.07703285 11.77 4.244016 4.580438
   0.2 4.635347 0.07453633 10.34 4.470001 4.800693
   0.4 4.785912 0.07453633 10.34 4.620566 4.951258
   0.6 4.863922 0.07703285 11.77 4.695711 5.032133

Variety = Victory:
 nitro   lsmean         SE    df lower.CL upper.CL
   0.0 4.275147 0.07703285 11.77 4.106936 4.443358
   0.2 4.498267 0.07453633 10.34 4.332921 4.663613
   0.4 4.648832 0.07453633 10.34 4.483486 4.814178
   0.6 4.726842 0.07703285 11.77 4.558631 4.895053

Confidence level used: 0.95
```

These LS means follow the same quadratic trend for each variety, but with different intercepts.

You may notice the fractional degrees of freedom in these results. These are obtained from the pbkrtest package (Halekoh and Højsgaard, 2013), if installed, and they use the Kenward-Rogers method. The degrees of freedom for the polynomial contrasts were also obtained in that way, but the results turn out to be integers.

9 Displaying LS means

The lsmeans package includes a function `lsmip` that displays predictions in an interaction-plot-like manner. It uses a formula of the form

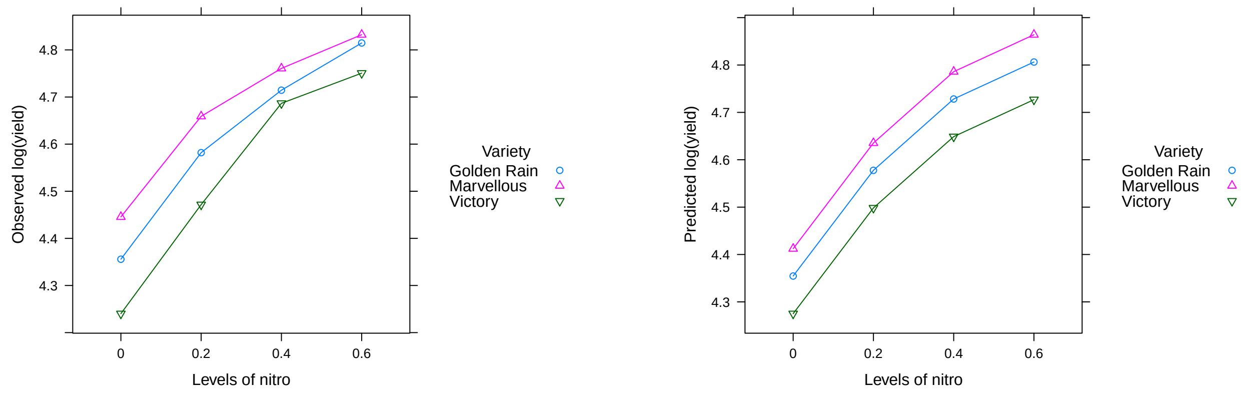

Figure 1: Interaction plots for the cell means and the fitted model, `Oats` example.

```
curve.factor(s) ~ x.factor(s) | by.factors
```

The function requires the lattice package (Sarkar, 2014) to be installed. Curve factors are those used to delineate one displayed curve from another (i.e., groups in lattice's parlance). *x* factors are those whose levels are plotted on the horizontal axis. And by factors, if present, break the plots into panels.

To illustrate, let's do a graphical comparison of the two models we have fitted to the `Oats` data.

```
R> lsmip(Oats.lmer, Variety ~ nitro, ylab = "Observed log(yield)")
```

```
R> lsmip(Oats.lsm2, Variety ~ nitro, ylab = "Predicted log(yield)")
```

The plots are shown in Figure 1. Note that the first model fits the cell means perfectly, so its plot is truly an interaction plot of the data. The other displays the parabolic trends we fitted in the revised model.

10 Transformations

Here is an interesting thing: Look at

```
R> str(Oats.lsm2)
```

```
'lsmobj' object with variables:
    nitro = 0.0, 0.2, 0.4, 0.6
    Variety = Golden Rain, Marvellous, Victory
Transformation: "log"
```

Part of the information stored with an `lsmobj ref.grid` is the transformation that was applied to the response variable. This allows us to conveniently unravel the transformation, via the `type` argument. Here are the predicted yields (as opposed to predicted log yields):

```
R> summary(Oats.lsm2, type = "response")
```

```
Variety = Golden Rain:
 nitro lsresponse       SE    df  lower.CL  upper.CL
   0.0   77.83396 5.995772 11.77  65.78336  92.09206
   0.2   97.29017 7.251652 10.34  82.46319 114.78306
   0.4  113.09895 8.429980 10.34  95.86272 133.43428
   0.6  122.27508 9.419197 11.77 103.34390 144.67418

Variety = Marvellous:
 nitro lsresponse       SE    df  lower.CL  upper.CL
   0.0   82.45287 6.351579 11.77  69.68715  97.55709
   0.2  103.06367 7.681987 10.34  87.35681 121.59464
   0.4  119.81059 8.930241 10.34 101.55150 141.35268
   0.6  129.53126 9.978162 11.77 109.47665 153.25959

Variety = Victory:
 nitro lsresponse       SE    df  lower.CL  upper.CL
   0.0   71.89068 5.537944 11.77  60.76024  85.06005
   0.2   89.86124 6.697927 10.34  76.16642 106.01840
   0.4  104.46288 7.786280 10.34  88.54278 123.24544
   0.6  112.93834 8.699962 11.77  95.45272 133.62708

Confidence level used: 0.95
```

It is important to realize that the statistical inferences are all done *before* reversing the transformation. Thus, t ratios are based on the linear predictors and will differ from those computed using the printed estimates and standard errors. Likewise, confidence intervals are computed on the linear-predictor scale, then the endpoints are back-transformed.

By the way, you may use a `type` argument in `lsmip` as well.

This kind of automatic support for transformations is available only for certain standard transformations, namely those supported by the `make.link` function in the stats package. Others require more work—see the documentation for `update` for details.

11 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 2.

```
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)
```

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

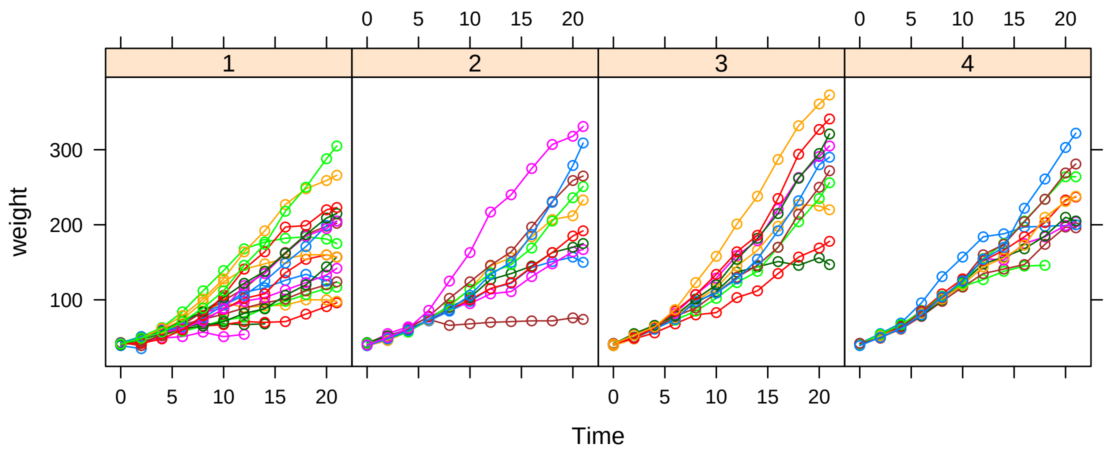

Figure 2: Growth curves of chicks, dataset `ChickWeight`.

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

 Diet Time.trend        SE    df lower.CL  upper.CL
 1      6.338556 0.6104878 49.86 5.112266  7.564845
 2      8.609136 0.8380027 48.28 6.924473 10.293800
 3     11.422871 0.8380027 48.28 9.738208 13.107534
 4      9.555825 0.8392450 48.56 7.868917 11.242734

Confidence level used: 0.95

R> cld (Chick.lst)

 Diet Time.trend        SE    df lower.CL  upper.CL .group
 1      6.338556 0.6104878 49.86 5.112266  7.564845  1
 2      8.609136 0.8380027 48.28 6.924473 10.293800  12
 4      9.555825 0.8392450 48.56 7.868917 11.242734   2
 3     11.422871 0.8380027 48.28 9.738208 13.107534   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.

Note: `lstrends` computes a difference quotient from two slightly different reference grids. Thus, you must call it with a model object, not a `ref.grid` object.

12 User preferences

lsmeans sets certain defaults, such as using .95 for the confidence coefficient, displaying intervals for `lsmeans` output and test statistics for `contrast` results, etc. As discussed before, you may use arguments in `summary` to change what's displayed, or `update` to change the defaults for a given object. But suppose you want different defaults to begin with? These can be set using the `lsm.options` statement. For example:

```
R> lsm.options(ref.grid = list(level = .90),
               lsmeans = list(),
               contrast = list(infer = c(TRUE,TRUE)))
```

This requests that any object created by `ref.grid` be set to have confidence intervals default to 90%, and that `contrast` results are displayed with both intervals and tests. No new options are set for `lsmeans` results, and the `lsmeans` part have been omitted. But even though no new defaults are set for `lsmeans`, future calls to `lsmeans` *on a model object* will be affected by this since it calls `ref.grid`; and also any contrasts from such results will "inherit" the 90% confidence level. However, calling `lsmeans` on an existing `"ref.grid"` object will inherit whatever `level` setting is stored there.

13 Two-sided formulas

In its original design, the only way to obtain contrasts and comparisons from `lsmeans` was to specify a two-sided formula, e.g., `pairwise ~ treatment`. The result is then a list of `lsmobj` objects. In its newer versions, lsmeans offers a richer family of objects that can be re-used, and dealing with a list of objects can be awkward or confusing, so I don't encourage its continued use. Nonetheless, it is still available for backward compatibility.

I'll present an example where, with one command, we obtain both the LS means and pairwise comparisons for `Variety` in the model `Oats.lmer2`:

```
R> lsmeans(Oats.lmer2, pairwise ~ Variety)

$lsmeans
 Variety       lsmean         SE    df lower.CL upper.CL
 Golden Rain 4.662050 0.07510922 10.65 4.526761 4.797338
 Marvellous  4.719699 0.07510922 10.65 4.584410 4.854987
 Victory     4.582619 0.07510922 10.65 4.447330 4.717907

Confidence level used: 0.9

$contrasts
 contrast                    estimate         SE df    lower.CL  upper.CL t.ratio p.value
 Golden Rain - Marvellous -0.05764898 0.06868444 10 -0.21647880 0.1011808  -0.839  0.6883
 Golden Rain - Victory     0.07943125 0.06868444 10 -0.07939857 0.2382611   1.156  0.5036
 Marvellous - Victory      0.13708023 0.06868444 10 -0.02174959 0.2959100   1.996  0.1636

Confidence level used: 0.9
Confidence-level adjustment: tukey method for a family of 3 means
P value adjustment: tukey method for a family of 3 means
```

This also illustrates the effect of the preceding `lsm.options` settings. I'll return to the defaults now.

```
R> lsm.options(NULL)
```

14 Messy data

To illustrate some more `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     4.708257 2.368117 92  0.7734276 8.643085
 NoAid      Black    -2.190399 2.490576 92 -6.3287039 1.947905
 FoodStamps Hispanic        NA       NA NA         NA       NA
 NoAid      Hispanic        NA       NA NA         NA       NA
 FoodStamps White     3.607680 1.155619 92  1.6875207 5.527838
 NoAid      White     2.256336 2.389273 92 -1.7136454 6.226316

Results are averaged over the levels of: age
Confidence level used: 0.9
```

Figure 3 shows the predictions from this model. One thing the 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.

Subsequent analyses might examine LS means and contrasts thereof on restricted portions of the reference grid (using `at` in the call).

15 Other types of models

15.1 Models supported by lsmeans

The lsmeans package comes with built-in support for several types of models, including these model classes for the packages:

stats : `"lm"`, `"mlm"`, `"aov"`, `"aovlist"`, `"glm"`

nlme : `"lme"`, `"gls"`

lme4 : `"lmerMod"`, `"glmerMod"`

survival : `"survreg"`, `"coxph"`

coxme : `"coxme"`

MASS : `"polr"`

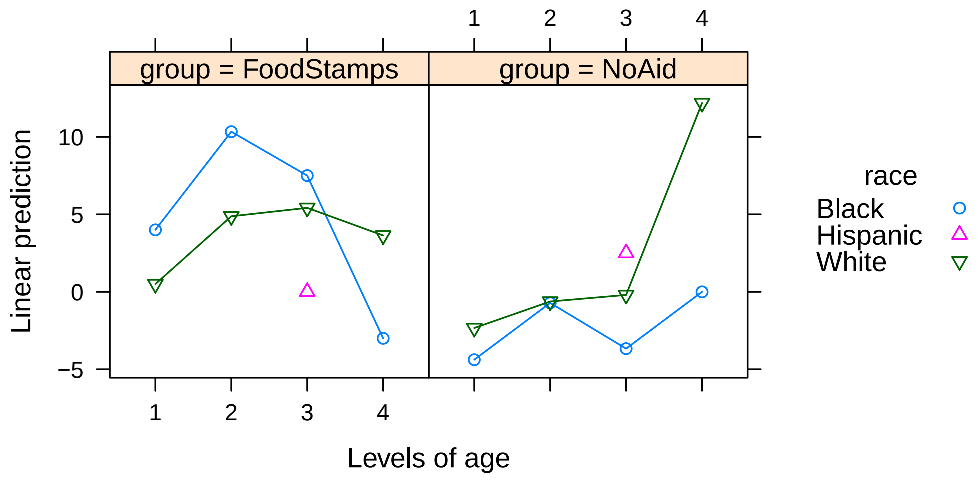

Figure 3: Predictions for the nutrition data

lsmeans support for all these models works similarly to the examples we have presented. Note that generalized linear or mixed models, and several others such as survival models, typically employ link functions such as `log` or `logit`. In all such cases, the LS means displayed are on the scale of the linear predictor, and any marginal averaging over the reference grid is performed on the linear-predictor scale. Results for `aovlist` objects are based on intra-block estimates, and should be used with caution.

15.2 Proportional-odds example

There is an interesting twist in `"polr"` objects (polytomous regression for Likert-scale data), in that an extra factor (named `"cut"` by default) is created to identify which boundary between scale positions we wish to use in predictions. An example is based on the `housing` data in the MASS package, where the response variable is satisfaction (`Sat`) on a three-point scale of low, medium, high; and predictors include `Type` (type of rental unit, four levels), `Infl` (influence on management of the unit, three levels), and `Cont` (contact with other residents, two levels). Here, we fit a (not necessarily good) model and obtain LS means for `Infl`

```
R> library(MASS)
R> housing.plr = polr(Sat ~ Infl + Type + Cont,
                      data = housing, weights = Freq)
R> ref.grid(housing.plr)

'ref.grid' object with variables:
    Infl = Low, Medium, High
    Type = Tower, Apartment, Atrium, Terrace
    Cont = Low, High
    cut = multivariate response levels: Low|Medium, Medium|High
Transformation: "logit"

R> housing.lsm = lsmeans(housing.plr, "Infl", at = list(cut = "Low|Medium"))
```

The default link function is `logit`. Look at what happens when we transform the predictions and contrasts thereof to the response scale:

```
R> summary(housing.lsm, type="response")

 Infl      cumprob         SE df asymp.LCL asymp.UCL
 Low    0.4578777 0.01997052 NA 0.4252555 0.4908651
 Medium 0.3240364 0.01776323 NA 0.2955273 0.3539141
 High   0.1888180 0.01682487 NA 0.1626818 0.2180598

Results are averaged over the levels of: Type, Cont
Confidence level used: 0.9

R> summary(pairs(housing.lsm), type="response")

 contrast      odds.ratio        SE df asymp.LCL asymp.UCL   z.ratio p.value
 Low - Medium    1.761902 0.1843879 NA  1.421327  2.184084  5.412123  <.0001
 Low - High      3.628499 0.4613860 NA  2.794987  4.710578 10.135720  <.0001
 Medium - High   2.059422 0.2559254 NA  1.595770  2.657788  5.813330  <.0001

Results are averaged over the levels of: Type, Cont
Confidence level used: 0.9
Confidence-level adjustment: tukey method for a family of 3 means
P value adjustment: tukey method for a family of 3 means
P values are asymptotic
Tests are performed on the linear-predictor scale
```

The logits are transformed to cumulative probabilities (note that a low probability means the satisfaction tends to be high, i.e., those having more influence tend to me more satisfied); and the pairwise comparisons transform to odds ratios.

Another point worth noting is that when only asymptotic tests and confidence intervals are available, degrees of freedom are set to `NA`, and test statistics and intervals are labeled differently.

15.3 Extending to more models

Developers of packages that fit models are invited and encouraged to include support for lsmeans. The help page `"extending-lsmeans"` and the vignette by the same name are provided to help make this possible.

References

Bates D, Maechler M, Bolker B, Walker S (2014). *lme4: Linear Mixed-effects Models Using Eigen and S4*. R package version 1.1-6, 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 (2014). *multcomp: Simultaneous Inference in General Parametric Models*. R package version 1.3-2, URL http://CRAN.R-project.org/package=multcomp.

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

Pinheiro J, Bates D, DebRoy S, Sarkar D, R Core Team (2014). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-117, URL http://CRAN.R-project.org/package=nlme.

Sarkar D (2014). *lattice: Lattice Graphics*. R package version 0.20-29, URL http://CRAN.R-project.org/package=lattice.

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