Least-squares Means: The R Package 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 linear contrasts among predictions. The **lsmeans** package provides a simple way of obtaining least-squares means and contrasts thereof. It supports many models fitted by R core packages (as well as a few key contributed ones) that fit linear or mixed models, and provides a simple way of extending it to cover more model classes.

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

1. Introduction

Least-squares means (LS means for short) for a linear model are simply predictions—or 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 main-effects means but are adjusted for imbalance. The latter interpretation is quite similar to the "unweighted means" method for unbalanced data, as presented in old design books.

LS means are not always well understood, in part because the term itself is confusing. The most important things to remember are:

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

A user who understands these points will know what is being computed, and thus can judge whether or not LS means are appropriate for the 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  516.6   516.6  29.100 1.76e-05
price2     1   62.7    62.7   3.533  0.07287
day        5  422.2    84.4   4.757  0.00395
store      5  223.8    44.8   2.522  0.05835
Residuals 23  408.3    17.8
```

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
 51.2222 48.5556 1   1        2.91841 2.71756 23
 51.2222 48.5556 2   1        3.84880 2.70134 23
 51.2222 48.5556 3   1       11.01857 2.53456 23
 51.2222 48.5556 4   1        6.09629 2.65137 23
 51.2222 48.5556 5   1       12.79580 2.44460 23
 51.2222 48.5556 6   1        8.74878 2.78618 23
 51.2222 48.5556 1   2        4.96147 2.37774 23
 51.2222 48.5556 2   2        5.89187 2.33558 23
 51.2222 48.5556 3   2       13.06163 2.41645 23
 51.2222 48.5556 4   2        8.13935 2.35219 23
 51.2222 48.5556 5   2       14.83886 2.46615 23
 51.2222 48.5556 6   2       10.79184 2.33760 23
 51.2222 48.5556 1   3        3.20089 2.37774 23
 51.2222 48.5556 2   3        4.13128 2.33558 23
 51.2222 48.5556 3   3       11.30105 2.41645 23
 51.2222 48.5556 4   3        6.37876 2.35219 23
 51.2222 48.5556 5   3       13.07828 2.46615 23
 51.2222 48.5556 6   3        9.03126 2.33760 23
 51.2222 48.5556 1   4        6.19876 2.36367 23
 51.2222 48.5556 2   4        7.12915 2.35219 23
 51.2222 48.5556 3   4       14.29891 2.43168 23
 51.2222 48.5556 4   4        9.37663 2.38865 23
 51.2222 48.5556 5   4       16.07614 2.51909 23
 51.2222 48.5556 6   4       12.02912 2.36469 23
 51.2222 48.5556 1   5        5.54322 2.36312 23
 51.2222 48.5556 2   5        6.47361 2.33067 23
 51.2222 48.5556 3   5       13.64337 2.36367 23
 51.2222 48.5556 4   5        8.72109 2.33760 23
 51.2222 48.5556 5   5       15.42060 2.39554 23
 51.2222 48.5556 6   5       11.37358 2.35232 23
 51.2222 48.5556 1   6       10.56374 2.36668 23
 51.2222 48.5556 2   6       11.49413 2.33925 23
 51.2222 48.5556 3   6       18.66390 2.34784 23
 51.2222 48.5556 4   6       13.74161 2.34130 23
 51.2222 48.5556 5   6       20.44113 2.37034 23
 51.2222 48.5556 6   6       16.39411 2.37054 23
```

2.2. LS means as marginal averages over the reference grid

The ANOVA indicates there is a significant `day` effect after adjusting for the covariates, so we might want to do a follow-up analysis that involves comparing the days. The `lsmeans` function provides a starting point:

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

```
 day   lsmean      SE df lower.CL upper.CL
 1    5.56442 1.76808 23  1.90686  9.22197
 2    6.49481 1.72896 23  2.91818 10.07143
 3   13.66457 1.75150 23 10.04131 17.28783
 4    8.74229 1.73392 23  5.15540 12.32918
 5   15.44180 1.78581 23 11.74758 19.13603
 6   11.39478 1.76673 23  7.74003 15.04953

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 overall means for each day:

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

      1       2       3       4       5       6
7.87275 7.10060 13.75860 8.04247 12.92460 11.60365
```

These unadjusted means are not comparable with one another because they are affected by the differing `price1` and `price2` values on each day, whereas the LS means are comparable because they use predictions at uniform `price1` and `price2` values.

Note that one 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 the reference grid is already available, 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. The user may specify this argument either in a `ref.grid` call or an `lsmeans` call; and should specify a `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    7.72470 1.73517 23  4.13524  11.3142
 3   14.89446 1.75104 23 11.27217  18.5168
 4    9.97218 1.76613 23  6.31866  13.6257

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 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`, one must 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    6.23623 1.88711 23  2.33245  10.1400
 3   13.40599 2.11938 23  9.02173  17.7903
 4    8.48371 1.86651 23  4.62254  12.3449

price2 = 60:
 day   lsmean      SE df lower.CL upper.CL
 2    9.21317 2.10945 23  4.84944  13.5769
 3   16.38293 1.90522 23 12.44169  20.3242
 4   11.46065 2.17805 23  6.95500  15.9663

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 by default—and the fact that an `"lsmobj"` is not (necessarily) a true reference grid as defined earlier in this article. Let's use the `str` function to examine the `"lsmobj"` object just produced:

```
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 already seen 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  6.23623 1.88711 23  2.33245  10.1400   3.305  0.0062
     60  9.21317 2.10945 23  4.84944  13.5769   4.368  0.0005

day = 3:
 price2   lsmean      SE df lower.CL upper.CL t.ratio p.value
     40 13.40599 2.11938 23  9.02173  17.7903   6.325  <.0001
     60 16.38293 1.90522 23 12.44169  20.3242   8.599  <.0001

day = 4:
 price2   lsmean      SE df lower.CL upper.CL t.ratio p.value
     40  8.48371 1.86651 23  4.62254  12.3449   4.545  0.0003
     60 11.46065 2.17805 23  6.95500  15.9663   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 causes 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 additional messages to appear when the object is displayed. But it can also be used as a `"data.frame"` if the user just wants to use the results computationally. For example, suppose we want to convert the LS means from dollars to Russian rubles (at the July 13, 2014 exchange rate):

```
R> transform(org.sum, lsrubles = lsmean * 34.2)
```

```
  day price2   lsmean      SE df lower.CL upper.CL t.ratio     p.value lsrubles
1   2     40  6.23623 1.88711 23  2.33245  10.1400 3.30465 6.19070e-03  213.279
2   3     40 13.40599 2.11938 23  9.02173  17.7903 6.32544 3.74162e-06  458.485
3   4     40  8.48371 1.86651 23  4.62254  12.3449 4.54523 2.89206e-04  290.143
4   2     60  9.21317 2.10945 23  4.84944  13.5769 4.36757 4.50464e-04  315.090
5   3     60 16.38293 1.90522 23 12.44169  20.3242 8.59899 2.43159e-08  560.296
6   4     60 11.46065 2.17805 23  6.95500  15.9663 5.26188 4.88377e-05  391.954
```

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

Besides `str` and `summary`, there is also a `confint` method, which is the same as `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 object's 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  6.23623 1.88711 23  0.938488  11.5340
 3       40 13.40599 2.11938 23  7.456193  19.3558
 4       40  8.48371 1.86651 23  3.243791  13.7236
 2       60  9.21317 2.10945 23  3.291240  15.1351
 3       60 16.38293 1.90522 23 11.034351  21.7315
 4       60 11.46065 2.17805 23  5.346122  17.5752

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 a `"ref.grid"` or `"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, method = "eff")

price2 = 40:
 contrast estimate      SE df t.ratio p.value
 2 effect -3.13908 1.41529 23  -2.218  0.0551
 3 effect  4.03068 1.44275 23   2.794  0.0310
```

```
 4 effect -0.89160 1.42135 23  -0.627  0.5366

price2 = 60:
 contrast estimate      SE df t.ratio p.value
 2 effect -3.13908 1.41529 23  -2.218  0.0551
 3 effect  4.03068 1.44275 23   2.794  0.0310
 4 effect -0.89160 1.42135 23  -0.627  0.5366

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

Note that this preserves the `by` specification from before, and obtains the effects for each group. In this example, since it is an additive model, we obtain exactly 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) -7.853877 2.19424 23  -3.579  0.0063
 2 - avg(5,6) -6.923486 2.12734 23  -3.255  0.0139
 3 - avg(5,6)  0.246279 2.15553 23   0.114  0.9999
 4 - avg(5,6) -4.676003 2.11076 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"` and `"trt.vs.ctrlk"` methods are provided for use in lieu of `ref` for comparing with the first and the last LS means.

Note that by default, `lsmeans` results are displayed with confidence intervals while `contrast` results are displayed with t tests. One 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"` as the `method` argument 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, a `pairs` method is provided that calls `contrast` with `method="pairwise"`:

```
R> pairs(org.lsm)
```

```
price2 = 40:
 contrast estimate      SE df t.ratio p.value
 2 - 3    -7.16976 2.47970 23  -2.891  0.0216
 2 - 4    -2.24748 2.44234 23  -0.920  0.6333
 3 - 4     4.92228 2.49007 23   1.977  0.1406

price2 = 60:
 contrast estimate      SE df t.ratio p.value
 2 - 3    -7.16976 2.47970 23  -2.891  0.0216
 2 - 4    -2.24748 2.44234 23  -0.920  0.6333
 3 - 4     4.92228 2.49007 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, Piepho, Selzer, and Dorai-Raj 2012) to be installed.

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

```
 day   lsmean      SE df lower.CL upper.CL .group
 1    5.56442 1.76808 23  1.90686  9.22197  1
 2    6.49481 1.72896 23  2.91818 10.07143  1
 4    8.74229 1.73392 23  5.15540 12.32918  12
 6   11.39478 1.76673 23  7.74003 15.04953  12
 3   13.66457 1.75150 23 10.04131 17.28783   2
 5   15.44180 1.78581 23 11.74758 19.13603   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    5.56442 1.76808 23  1.906856  9.22197  1
 2    6.49481 1.72896 23  2.918183 10.07143  12
 3   13.66457 1.75150 23 10.041308 17.28783   23
 4    8.74229 1.73392 23  5.155403 12.32918  123
 5   15.44180 1.78581 23 11.747576 19.13603    3
 6   11.39478 1.76673 23  7.740031 15.04953  123

variety = sales2:
 day   lsmean      SE df  lower.CL upper.CL .group
 1    7.71566 2.32649 23  2.902962 12.52836  12
 2    3.97645 2.27500 23 -0.729758  8.68265  1
 3   16.59781 2.30467 23 11.830240 21.36539   2
 4   11.04454 2.28153 23  6.324832 15.76425  12
 5   14.99079 2.34981 23 10.129837 19.85174   2
 6   12.04878 2.32470 23  7.239777 16.85779  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   -2.933243 2.69411 23  -1.089  0.2875  1
 sales1 - sales2 4   -2.302251 2.66706 23  -0.863  0.3969  1
 sales1 - sales2 1   -2.151248 2.71961 23  -0.791  0.4370  1
 sales1 - sales2 6   -0.654002 2.71752 23  -0.241  0.8120  1
 sales1 - sales2 5    0.451016 2.74688 23   0.164  0.8710  1
 sales1 - sales2 2    2.518361 2.65943 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
```

There is little evidence of variety differences, nor that these differences vary from day to day.

7. Interfacing with multcomp

The **multcomp** package (Hothorn, Bretz, and Westfall 2013) 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, let's 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    5.56442 1.76808 23  0.46126  10.6676
 2    6.49481 1.72896 23  1.50458  11.4850
 3   13.66457 1.75150 23  8.60927  18.7199
 4    8.74229 1.73392 23  3.73774  13.7468
 5   15.44180 1.78581 23 10.28749  20.5961
 6   11.39478 1.76673 23  6.29554  16.4940
```

```
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 = 2.861
95% family-wise confidence level


Linear Hypotheses:
          Estimate lwr    upr
1, 6 == 0  5.564    0.507 10.622
2, 6 == 0  6.495    1.549 11.441
3, 6 == 0 13.665    8.654 18.675
4, 6 == 0  8.742    3.782 13.702
5, 6 == 0 15.442   10.333 20.550
6, 6 == 0 11.395    6.341 16.449
```

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    -4.65       1.62   -2.87    0.040
2 effect == 0    -3.72       1.58   -2.36    0.099
3 effect == 0     3.45       1.60    2.15    0.116
4 effect == 0    -1.47       1.59   -0.93    0.585
5 effect == 0     5.22       1.64    3.18    0.023
6 effect == 0     1.18       1.62    0.73    0.585
(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 information 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

Orange-sales illustrations are probably getting tiresome. To illustrate some new features, let's turn to a new example. The `Oats` dataset in the **nlme** package (Pinheiro, Bates, and R-core 2013) 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, Maechler, Bolker, and Walker 2013). This will be a mixed model with random intercepts for `Block` and `Block:Variety` (which identifies the plots). A logarithmic transformation is applied 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 0.0750  0.0375   2.008
factor(nitro)         3 2.1350  0.7117  38.110
Variety:factor(nitro) 6 0.0451  0.0075   0.402
```

Apparently, the interaction is not needed. But perhaps we can further simplify the model 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     1.50565129 0.1440469 45  10.453  <.0001
 quadratic -0.14510997 0.0644197 45  -2.253  0.0292
 cubic      0.00273198 0.1440469 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)
```

Remember that `nitro` is now used as a quantitative predictor. But for comparing with the previous model, we 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`, which tells `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 4.35458 0.0770328 11.77  4.18637  4.52279
   0.2 4.57770 0.0745363 10.34  4.41235  4.74304
   0.4 4.72826 0.0745363 10.34  4.56292  4.89361
   0.6 4.80627 0.0770328 11.77  4.63806  4.97448

Variety = Marvellous:
 nitro  lsmean        SE    df lower.CL upper.CL
   0.0 4.41223 0.0770328 11.77  4.24402  4.58044
   0.2 4.63535 0.0745363 10.34  4.47000  4.80069
   0.4 4.78591 0.0745363 10.34  4.62057  4.95126
   0.6 4.86392 0.0770328 11.77  4.69571  5.03213

Variety = Victory:
```

```
 nitro  lsmean        SE    df lower.CL upper.CL
   0.0 4.27515 0.0770328 11.77  4.10694  4.44336
   0.2 4.49827 0.0745363 10.34  4.33292  4.66361
   0.4 4.64883 0.0745363 10.34  4.48349  4.81418
   0.6 4.72684 0.0770328 11.77  4.55863  4.89505

Confidence level used: 0.95
```

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

Fractional degrees of freedom are displayed in these results. These are obtained from the **pbkrtest** package (Halekoh and Højsgaard 2013), and they use the Kenward-Rogers method. (The degrees of freedom for the polynomial contrasts were also obtained from **pbkrtest**, but the results turn out to be integers.)

9. Displaying LS means graphically

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

```
curve.factors ~ x.factors | by.factors
```

The function requires the **lattice** package (Sarkar 2013) to be installed. In the above formula, `curve.factors` specifies factor(s) 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.

The help page for `lsmip` gives examples involving several factors, showing the flexibility we have in combining factors or using them to create multi-panel plots.

10. Transformations

When a transformation or link function is used in fitting a model, `ref.grid` (also called by `lsmeans`) stores that information in the returned object, as seen in this example:

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

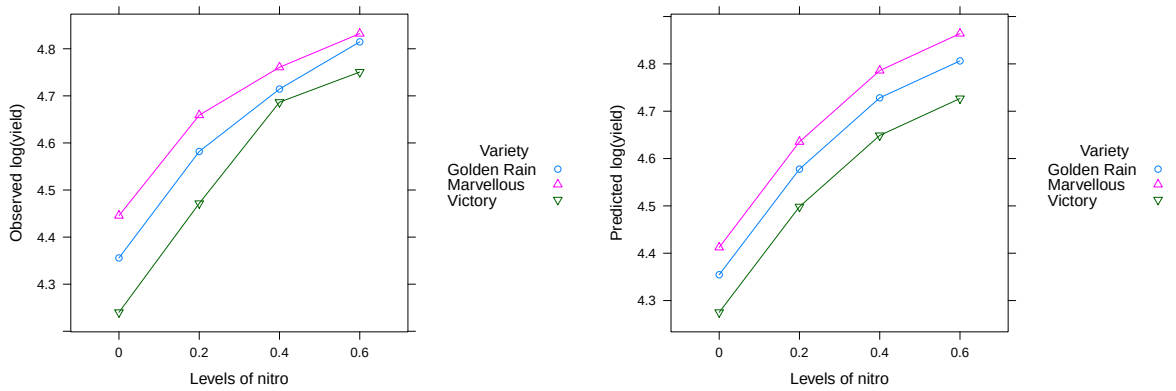

Figure 1: Interaction plots for the cell means and the fitted model, `Oats` example.

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

This allows us to conveniently unravel the transformation, via the `type` argument in `summary` or related functions such as `lsmip` and `predict`. Here are the predicted yields for (as opposed to predicted log yields) for the polynomial model:

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

Variety = Golden Rain:
 nitro lsresponse      SE    df lower.CL upper.CL
   0.0    77.8340 5.99577 11.77  65.7834  92.0921
   0.2    97.2902 7.25165 10.34  82.4632 114.7831
   0.4   113.0989 8.42998 10.34  95.8627 133.4343
   0.6   122.2751 9.41920 11.77 103.3439 144.6742

Variety = Marvellous:
 nitro lsresponse      SE    df lower.CL upper.CL
   0.0    82.4529 6.35158 11.77  69.6871  97.5571
   0.2   103.0637 7.68199 10.34  87.3568 121.5946
   0.4   119.8106 8.93024 10.34 101.5515 141.3527
   0.6   129.5313 9.97816 11.77 109.4766 153.2596

Variety = Victory:
 nitro lsresponse      SE    df lower.CL upper.CL
   0.0    71.8907 5.53794 11.77  60.7602  85.0601
   0.2    89.8612 6.69793 10.34  76.1664 106.0184
   0.4   104.4629 7.78628 10.34  88.5428 123.2454
   0.6   112.9383 8.69996 11.77  95.4527 133.6271
```

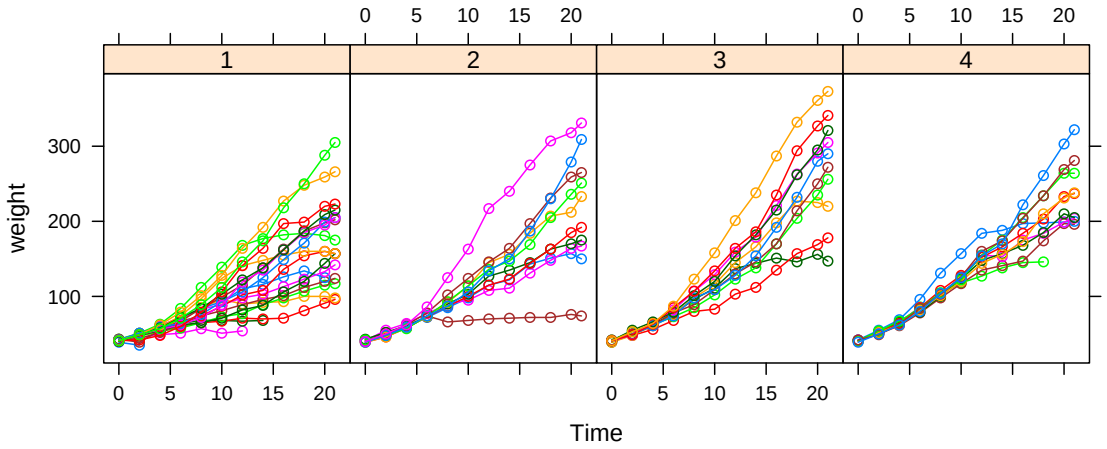

Figure 2: Growth curves of chicks, dataset `ChickWeight`.

```
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.

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.

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

```
 Diet Time.trend       SE    df lower.CL upper.CL
 1       6.33856 0.610488 49.86  5.11227  7.56484
 2       8.60914 0.838003 48.28  6.92447 10.29380
 3      11.42287 0.838003 48.28  9.73821 13.10753
 4       9.55583 0.839245 48.56  7.86892 11.24273

Confidence level used: 0.95
```

Here we obtain estimates and pairwise comparisons of the slopes using a compact letter display.

```
R> cld (Chick.lst)
```

```
 Diet Time.trend       SE    df lower.CL upper.CL .group
 1       6.33856 0.610488 49.86  5.11227  7.56484  1
 2       8.60914 0.838003 48.28  6.92447 10.29380  12
 4       9.55583 0.839245 48.56  7.86892 11.24273   2
 3      11.42287 0.838003 48.28  9.73821 13.10753   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 based on two slightly different reference grids. Thus, it must be called with a model object, not a `ref.grid` object.

12. User preferences

lsmeans sets certain defaults for displaying results—for example, using .95 for the confidence coefficient, and showing intervals for `lsmeans` output and test statistics for `contrast` results. As discussed before, one may use arguments in `summary` to change what is 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 levels 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 could have been omitted. These options are stored with objects created by `ref.grid`, `lsmeans`, and `contrast`. For example, even though no new defaults are set for `lsmeans`, future calls to `lsmeans` *on a model object* will be

displayed with 90% confidence intervals, because `lsmeans` calls `ref.grid`. However, calling `lsmeans` on an existing `"ref.grid"` object will inherit that object's setting.

13. Two-sided formulas

In its original design, the only way to obtain contrasts and comparisons in **lsmeans** was to specify a two-sided formula, e.g., `pairwise ~ treatment`, in the `lsmeans` call. 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 its continued use is not encouraged. Nonetheless, it is still available for backward compatibility.

Here is 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 4.66205 0.0751092 10.65  4.52676  4.79734
 Marvellous  4.71970 0.0751092 10.65  4.58441  4.85499
 Victory     4.58262 0.0751092 10.65  4.44733  4.71791

Confidence level used: 0.9

$contrasts
 contrast                   estimate        SE df   lower.CL upper.CL t.ratio p.value
 Golden Rain - Marvellous -0.0576490 0.0686844 10 -0.2164788 0.101181  -0.839  0.6883
 Golden Rain - Victory     0.0794312 0.0686844 10 -0.0793986 0.238261   1.156  0.5036
 Marvellous - Victory      0.1370802 0.0686844 10 -0.0217496 0.295910   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 example also illustrates the effect of the preceding `lsm.options` settings. Let us now return to the default display for contrast results.

```
R> lsm.options(ref.grid = NULL, contrast = 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 (1992), 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. A Type-II (hierarchical) analysis-of-variance table is also shown.

```
R> nutr.lm <- lm(gain ~ (age + group + race)^2, data = nutrition)
R> library("car")
R> Anova(nutr.lm)

Anova Table (Type II tests)

Response: gain
           Sum Sq Df F value  Pr(>F)
age          82.4  3   0.961   0.414
group       658.1  1  23.044 6.1e-06
race         11.2  2   0.196   0.823
age:group    91.6  3   1.069   0.366
age:race     87.3  3   1.019   0.388
group:race  113.7  2   1.991   0.142
Residuals  2627.5 92
```

One main effect (`group`) is quite significant, and there is possibly an interaction with `race`. Let us look at the `group` by `race` LS means:

```
R> lsmip(nutr.lm, race ~ age | group)
R> lsmeans(nutr.lm, ~ group*race)

 group      race       lsmean      SE df    lower.CL upper.CL
 FoodStamps Black     4.70826 2.36812 92  0.00497136  9.41154
 NoAid      Black    -2.19040 2.49058 92 -7.13689810  2.75610
 FoodStamps Hispanic       NA      NA NA          NA       NA
 NoAid      Hispanic       NA      NA NA          NA       NA
 FoodStamps White     3.60768 1.15562 92  1.31252147  5.90284
 NoAid      White     2.25634 2.38927 92 -2.48896668  7.00164

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

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.

We can avoid non-estimable cases by using `at` to restrict the reference levels to a smaller set. A useful summary of the results might be obtained by narrowing 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 in the data set. The simplest way to do this is to add a `weights` argument.[1] With those ideas in mind, here are the LS means and comparisons within rows and columns:

[1] It may also be done by specifying a custom function in the `fac.reduce` argument, but for simple weighting, `weights` is simpler.

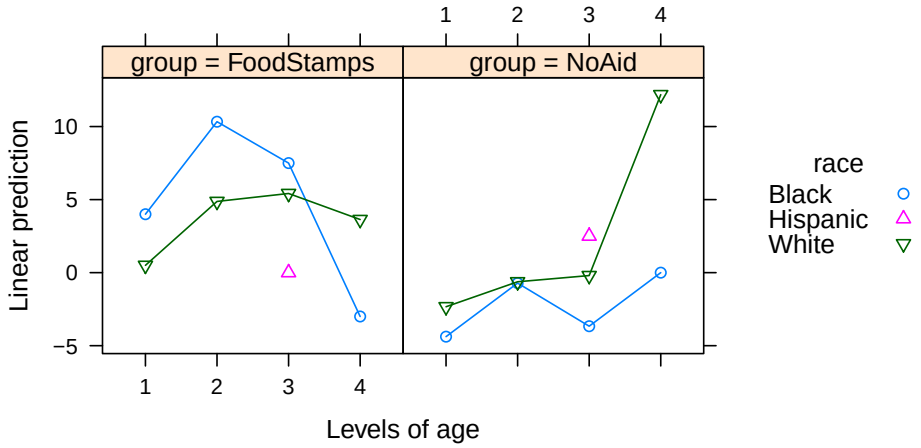

Figure 3: Predictions for the nutrition data

```
R> nutr.lsm <- lsmeans(nutr.lm, ~ group * race, weights = "proportional",
       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  8.275710 2.917880 92  2.48055 14.070871
 NoAid      Black -2.858277 1.678104 92 -6.19114  0.474582
 FoodStamps White  5.270305 0.868032 92  3.54632  6.994292
 NoAid      White -0.316369 1.010292 92 -2.32290  1.690158

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

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

 contrast           race  estimate      SE df t.ratio p.value
 FoodStamps - NoAid Black 11.13399 3.55123 92   3.135  0.0023
 FoodStamps - NoAid White  5.58667 1.33198 92   4.194  0.0001

Results are averaged over the levels of: age

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

 contrast      group      estimate      SE df t.ratio p.value
 Black - White FoodStamps  3.00540 3.01639 92   0.996  0.3217
 Black - White NoAid      -2.54191 1.95876 92  -1.298  0.1976

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

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.

14.1. More on weighting

The `weights` argument can be a vector of numerical weights (it has to be of the right length), or one of four text values: `"equal"` (weight the predictions equally when averaging them, the default), `"proportional"` (weight them proportionally to the observed frequencies of the factor combinations being averaged over), `"outer"` (weight according to the outer products of the one-factor marginal counts), or `"cells"` (weight each mean differently, according to the frequencies of the predictions being averaged). Compare the results with these schemes:

```
R> lsmeans(nutr.lm, "race", weights = "equal")

 race      lsmean      SE df  lower.CL upper.CL
 Black    1.25893 1.64600 92 -2.010175  4.52803
 Hispanic      NA      NA NA        NA       NA
 White    2.93201 1.34661 92  0.257519  5.60650

Results are averaged over the levels of: age, group
Confidence level used: 0.95

R> lsmeans(nutr.lm, "race", weights = "prop")

 race      lsmean      SE df  lower.CL upper.CL
 Black    1.92655 1.39403 92 -0.842112  4.69522
 Hispanic      NA      NA NA        NA       NA
 White    2.52282 0.60446 92  1.322310  3.72333

Results are averaged over the levels of: age, group
Confidence level used: 0.95
```

```
R> lsmeans(nutr.lm, "race", weights = "outer")

 race      lsmean       SE df  lower.CL upper.CL
 Black    2.54667 1.431362 92 -0.296136  5.38948
 Hispanic      NA       NA NA        NA       NA
 White    3.14294 0.749415 92  1.654536  4.63134

Results are averaged over the levels of: age, group
Confidence level used: 0.95

R> lsmeans(nutr.lm, "race", weights = "cells")

 race       lsmean       SE df lower.CL upper.CL
 Black    0.380952 1.166180 92 -1.93518  2.69709
 Hispanic 1.666667 3.085422 92 -4.46125  7.79458
 White    2.795181 0.586592 92  1.63016  3.96020

Results are averaged over the levels of: age, group
Confidence level used: 0.95
```

Note there are four different sets of answers. The `"equal"` weighting is self-explanatory. But what's the distinction between `"proportional"` and `"outer"`? To clarify, consider:

```
R> temp = lsmeans(nutr.lm, c("group","race"), weights = "prop")
R> lsmeans(temp, "race", weights = "prop")

 race      lsmean       SE df  lower.CL upper.CL
 Black    2.54667 1.431362 92 -0.296136  5.38948
 Hispanic      NA       NA NA        NA       NA
 White    3.14294 0.749415 92  1.654536  4.63134

Results are averaged over the levels of: age, group
Confidence level used: 0.95
```

The previous results using `"outer"` weights are the same as those using `"proportional"` weights on one factor at a time. Thus, if only one factor is being averaged over, `"outer"` and

`"proportional"` are the same. Another way to look at it is that outer weights are like the expected counts in a chi-square test; each factor is weighted independently of the others.

The results for `"cells"` weights stand out because everything is estimable—that's because the empty cells in the data were given weight zero. These results are the same as the unadjusted means:

```
R> with(nutrition, tapply(gain, race, mean))

   Black Hispanic    White
0.380952 1.666667 2.795181
```

14.2. Alternative covariate adjustments

Urquhart (1982) reports data on slaughter weights of animals that entered a feedlot as yearling calves. The animals came from 11 different herds, and each animal was randomized to one of three diets. In addition, the weight of each yearling at entry was recorded. The `feedlot` dataset provided in **lsmeans** contains these results. From the feedlot operator's perspective, both diets and herds are fixed effects. Let us fit a factorial model with slaughter weight `swt` as the response and entry weight `ewt` as a covariate.

```
R> feedlot.lm <- lm(swt ~ ewt + herd * diet, data = feedlot)
R> Anova(feedlot.lm)

Anova Table (Type II tests)

Response: swt
          Sum Sq Df F value   Pr(>F)
ewt        66728  1  20.842 5.62e-05
herd       46885 10   1.464    0.193
diet       13427  2   2.097    0.138
herd:diet  61698 17   1.134    0.363
Residuals 115257 36
```

The interaction tesrm doesn't make much of a contribution here, so we will work with an additive model instead (which also ameliorates some non-estimability issues due to missing cells).

```
R> feedlot.add <- update(feedlot.lm, . ~ . - herd:diet)
```

Here are the `LS~means` for the herds, and a compact letter display for comparisons thereof:

```
R> cld(lsmeans(feedlot.add, "herd"))

 herd   lsmean      SE df lower.CL upper.CL .group
 9     985.972 33.4665 53  918.847  1053.10  1
 36    989.519 32.9014 53  923.527  1055.51  1
```

```
 24    989.668 22.8401 53  943.857  1035.48  1
 33   1002.922 27.7087 53  947.345  1058.50  1
 32   1008.557 21.5982 53  965.236  1051.88  1
 16   1013.836 35.7958 53  942.039  1085.63  1
 31   1032.030 26.1786 53  979.522  1084.54  1
 35   1035.855 22.4000 53  990.926  1080.78  1
 34   1041.464 22.3438 53  996.648  1086.28  1
 19   1058.210 22.2438 53 1013.594  1102.83  1
 3    1065.074 26.2413 53 1012.441  1117.71  1

Results are averaged over the levels of: diet
Confidence level used: 0.95
P value adjustment: tukey method for a family of 11 means
significance level used: alpha = 0.05
```

No herds are found to be different—not a surprise given that the P value for `herd` is about the same as for the original model. However, these predictions are made at the same entry weight for every herd. This is *not* the right thing to do here, because the herds differ in genetic makeup, the way they were fed and managed, and so forth—which affect the yearlings' entry weights. This is an example where a treatment affects a covariate. Each herd should have its own reference value for entry weight. This is done in `lsmeans` by providing a formula in the `cov.reduce` argument. The formula `ewt ~ herd` indicates that the reference grid should be constructed using the predicted value of `ewt`, based on a linear model with `herd` as the predictor. Here are the results:

```
R> cld(lsmeans(feedlot.add, "herd", cov.reduce = ewt ~ herd))

 herd   lsmean      SE df lower.CL upper.CL .group
 9     856.137 26.6693 53  802.646  909.629  1
 16    943.262 34.0558 53  874.955 1011.570  12
 32    959.998 20.4765 53  918.928 1001.069  12
 24    984.512 22.7985 53  938.785 1030.240   2
 3     993.465 23.8366 53  945.654 1041.275   23
 34   1010.555 21.8867 53  966.656 1054.454   23
 19   1058.382 22.2438 53 1013.767 1102.997   234
 33   1072.005 26.2225 53 1019.409 1124.601   234
 35   1092.972 20.5827 53 1051.688 1134.255    34
 31   1105.465 23.8366 53 1057.654 1153.275    34
 36   1126.980 25.9739 53 1074.883 1179.077     4

Results are averaged over the levels of: diet
Confidence level used: 0.95
P value adjustment: tukey method for a family of 11 means
significance level used: alpha = 0.05
```

What a world of difference! We now see many significant differences in the comparisons. By the way, another approach would be to simply omit `ewt` from the model, to prevent making

inappropriate adjustments in the traditional analysis. With such a model (not shown), the predictions are similar to those above; however, their standard errors are substantially higher, because—as seen in the ANOVA table—the covariate explains a lot of the variation.

Another use of formulas in `cov.reduce` is to create representative values of some covariates when others are specified in `at`. For example, suppose there are three covariates x_1, x_2, x_3 in a model, and we want to see predictions at a few different values of x_1. We might use

```
R> rg <- ref.grid(my.model, at = list(x1 = c(5,10,15)),
                  cov.reduce = list(x2 ~ x1,  x3 ~ x1 + x2))
```

(When more than one formula is given, they are processed in the order given.) The values used for x_2 and x_3 will depend on x_1 and should in some sense be more realistic values of those covariates as x_1 varies than would be the overall means of x_2 and x_3. Of course, it would be important to display the values used—available as `rg@grid`—when reporting the analysis.

15. Other types of models

15.1. Models supported by lsmeans

The **lsmeans** package comes with built-in support for several packages and model classes:

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

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

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

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

coxme : `"coxme"`

MASS : `"polr"`

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 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. The example here 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 | cut)
```

The default link function is `logit`. Now we transform the predictions and contrasts thereof to the response scale:

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

cut = Low|Medium:
 Infl    cumprob        SE df asymp.LCL asymp.UCL
 Low    0.457878 0.0199705 NA  0.419070  0.497204
 Medium 0.324036 0.0177632 NA  0.290234  0.359781
 High   0.188818 0.0168249 NA  0.158020  0.224021

cut = Medium|High:
 Infl    cumprob        SE df asymp.LCL asymp.UCL
 Low    0.734574 0.0164491 NA  0.701108  0.765545
 Medium 0.611010 0.0189302 NA  0.573327  0.647412
 High   0.432695 0.0255170 NA  0.383521  0.483232

Results are averaged over the levels of: Type, Cont
Confidence level used: 0.95

R> summary(pairs(housing.lsm), type = "response") [1:3, ]

cut = Low|Medium:
 contrast      odds.ratio       SE df  z.ratio p.value
 Low - Medium     1.76190 0.184388 NA  5.41212  <.0001
 Low - High       3.62850 0.461386 NA 10.13572  <.0001
 Medium - High    2.05942 0.255925 NA  5.81333  <.0001

Results are averaged over the levels of: Type, Cont
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 cumulative probability means a low number of people are dissatisfied; thus, we find that those having more influence tend to me more satisfied). Note also that the pairwise comparisons transform to odds ratios. Only the first three rows of the comparisons table are shown because the results for `cut = Medium|High` will be identical.

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

The functions `ref.grid` and `lsmeans` work by first reconstructing the dataset (so that the reference grid can be identified) and extracting needed information about the model, such as the regression coefficients, covariance matrix, and the linear functions associated with each point in the reference grid. For a fitted model of class, say, `"modelobj"`, these tasks are accomplished by defining S3 methods `recover.data.modelobj` and `lsm.basis.modelobj`. The help page `"extending-lsmeans"` and the vignette by the same name provide details and examples.

Developers of packages that fit models are encouraged to include support for **lsmeans** by incorporating (and exporting) `recover.data` and `lsm.basis` methods for their model classes.

16. Discussion

The design goal of **lsmeans** is primarily to provide the functionality of the `LSMEANS` statement in various SAS procedures. Thus its emphasis is on tabular results which, of course, may also be used as data for further analysis or graphics. By design, it can be extended with relative ease to additional model classes. A unique capability of **lsmeans** is its explicit reliance on the concept of a reference grid, which I feel is a useful approach for understanding what is being computed.

Some **lsmeans** capabilities exceed those of SAS, including the `lstrends` capability, more flexibility in organizing the output, and more built-in contrast families. In addition, SAS does not allow LS means for factor combinations when the model does not include the interaction of those factors; or creating a grid of covariate values using `at`.

There are a few other R packages that provide capabilities that overlap with those of **lsmeans**. The **effects** package (Fox 2003; Fox and Hong 2009) can compute LS means. However, for an unbalanced dataset, it does not use equal weights, but rather it appears to use "outer" weights, as described in Section 14.1. Also, it does not check estimability, so some results could be questionable. The emphasis of **effects** is on graphical rather than tabular displays. It has special strengths for curve-fitting models such as splines. In contrast, **lsmeans**'s strengths are more in the area of factorial models where one wants traditional summaries in the form of estimates, contrasts, and interaction plots.

The **doBy** package (Højsgaard, Halekoh, Robison-Cox, Wright, and Leidi 2013) provides an `LSmeans` function that has some of the capabilities of `lsmeans`, but it produces a data frame rather than a reusable object. In earlier versions of the package, this function was named `popMeans`. The package also has an `LSmatrix` function to obtain the linear functions needed

to obtain LS means.

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/.

Fox J (2003). "Effect Displays in R for Generalised Linear Models." *Journal of Statistical Software*, **8**(15), 1–27. URL http://www.jstatsoft.org/v08/i15/.

Fox J, Hong J (2009). "Effect Displays in R for Multinomial and Proportional-Odds Logit Models: Extensions to the **effects** Package." *Journal of Statistical Software*, **32**(1), 1–24. URL http://www.jstatsoft.org/v32/i01/.

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.

Højsgaard S, Halekoh U, Robison-Cox J, Wright K, Leidi AA (2013). ***doBy****: Groupwise summary statistics, LSmeans, general linear contrasts, various utilities*. R package version 4.5-10, URL http://CRAN.R-project.org/package=doBy.

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.

Milliken GA, Johnson DE (1992). *Analysis of Messy Data – Volume I: Designed Experiments*. Chapman & Hall/CRC. ISBN 0-412-99081-4.

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.

Urquhart NS (1982). "Adjustment in covariates when one factor affects the covariate." *Biometrics*, **38**, 651–660.

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

Affiliation:

Russell V. Lenth, Professor Emeritus
Department of Statistics and Actuarial Science
The University of Iowa
Iowa City, IA 52242 USA
E-mail: russell-lenth@uiowa.edu