Using the lsmeans Package

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

August 27, 2012

1 Introduction

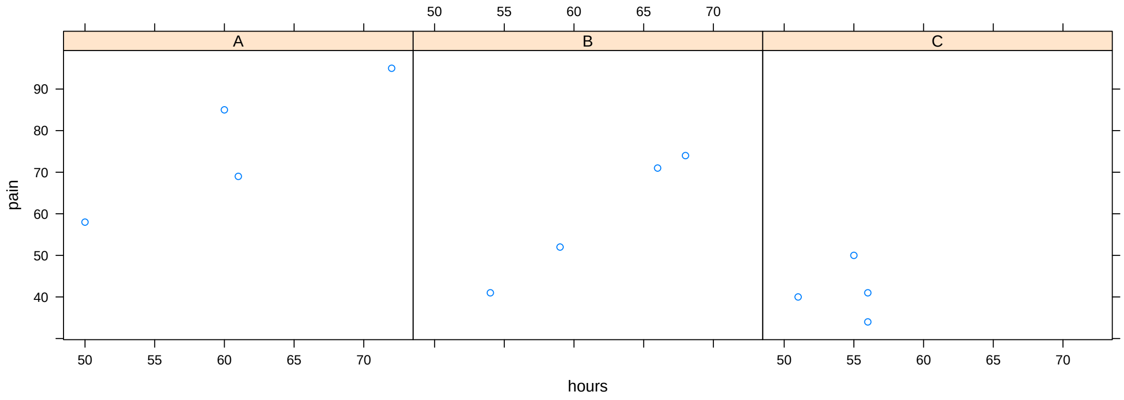
Least-squares means (or LS means), popularized by SAS, are predictions from a linear model at combinations of specified factors. SAS's documentation describes them as "predicted population margins—that is, they estimate the marginal means over a balanced population" (SAS Institute 2012). Unspecified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

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

2 Analysis-of-covariance example

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

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



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

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

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

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

$‘type lsmeans‘
  estimate       SE  t.ratio
A 73.56518 3.640583 20.20698
B 54.49529 3.722251 14.64041
C 49.43953 3.943413 12.53724
```

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

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

       1        2        3
73.56518 54.49529 49.43953
```

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

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

$‘type lsmeans‘
  estimate       SE  t.ratio
A 66.28560 4.154824 15.95389
B 47.21570 4.351192 10.85121
C 42.15995 3.588596 11.74831

$‘type pairwise differences‘
       estimate       SE  t.ratio
A - B 19.069896 5.081620 3.752720
A - C 24.125650 5.559580 4.339474
B - C  5.055754 5.719515 0.883948
```

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

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

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

3 Two-factor example

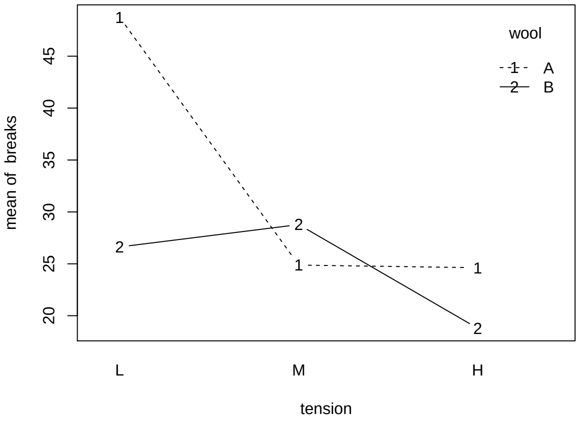
Now consider the R-provided dataset `warpbreaks`, relating to a weaving-process experiment. This dataset (from Tukey 1977, p.82) has two factors: `wool` (two types of wool), and `tension` (low, medium, and high); and the response variable is `breaks`, the nuumber of breaks in a fixed length of yarn. To make it more interesting, we'll delete some cases so that the design is unbalanced.

```
R> warp = warpbreaks[-c(1,2,3,5,8,13,21,34), ]
R> with(warp, table(wool, tension))

    tension
wool L M H
   A 4 8 8
   B 8 9 9
```

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

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



So let us fit a model with interaction

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

Analysis of Variance Table

Response: breaks
             Df Sum Sq Mean Sq F value   Pr(>F)
wool          1  271.0  270.99  2.8537 0.098947 .
tension       2 1229.6  614.78  6.4739 0.003666 **
wool:tension  2 1108.8  554.40  5.8381 0.005962 **
Residuals    40 3798.5   94.96
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
R> lsmeans(warp.lm, list(pairwise ~ wool | tension,  poly ~ tension | wool)) [-3]
```

```
$‘wool:tension lsmeans‘
     estimate       SE   t.ratio
A, L 48.75000 4.872426 10.005282
B, L 26.62500 3.445326  7.727862
A, M 24.87500 3.445326  7.219927
B, M 28.77778 3.248284  8.859378
A, H 24.62500 3.445326  7.147365
B, H 18.77778 3.248284  5.780830

$‘wool:tension pairwise differences‘
           estimate       SE    t.ratio
A - B | L 22.125000 5.967479  3.7075957
A - B | M -3.902778 4.735147 -0.8242147
A - B | H  5.847222 4.735147  1.2348554

$‘tension:wool polynomial contrasts‘
                 estimate       SE   t.ratio
linear | A    -24.125000 5.967479 -4.042746
quadratic | A  23.625000 9.115475  2.591746
linear | B     -7.847222 4.735147 -1.657229
quadratic | B -12.152778 8.039093 -1.511710
```

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

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

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

$‘wool lsmeans‘
  estimate       SE  t.ratio
A 32.75000 2.296884 14.25845
B 24.72685 1.914070 12.91847

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

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

4 Split-plot example

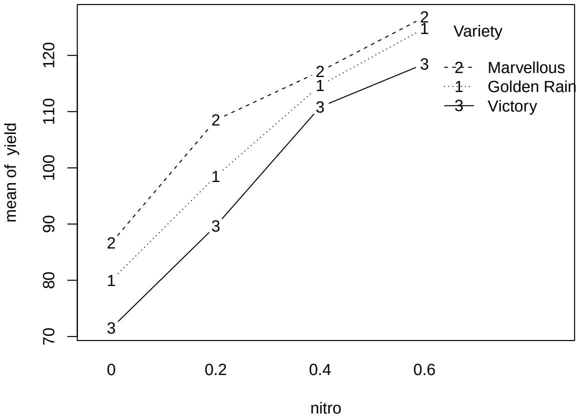
The `nlme` package includes a famous dataset `Oats` that was used in Yates (1935) as an example of a split-plot experiment. Here is a summary of the dataset.

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

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

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

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



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

```
R> Oats.lme = lme(yield ˜ Variety + factor(nitro), random = ˜1 | Block/Variety, data=Oats)
R> lsmeans(Oats.lme, list(revpairwise ˜ Variety,  poly ˜ nitro,  ˜ Variety:nitro))

$‘Variety lsmeans‘
            estimate       SE  t.ratio
Golden Rain 104.5000 7.797492 13.40174
Marvellous  109.7917 7.797492 14.08038
Victory      97.6250 7.797492 12.52005

$‘Variety pairwise differences‘
                           estimate      SE    t.ratio
Marvellous - Golden Rain   5.291667 7.07891  0.7475256
Victory - Golden Rain     -6.875000 7.07891 -0.9711947
Victory - Marvellous     -12.166667 7.07891 -1.7187204

$‘nitro lsmeans‘
     estimate       SE  t.ratio
0    79.38889 7.132357 11.13081
0.2  98.88889 7.132357 13.86483
0.4 114.22222 7.132357 16.01465
0.6 123.38889 7.132357 17.29987
```

```
$‘nitro polynomial contrasts‘
           estimate        SE    t.ratio
linear    147.33333 13.439537 10.9626791
quadratic -10.33333  6.010344 -1.7192583
cubic      -2.00000 13.439537 -0.1488146

$‘Variety:nitro lsmeans‘
                  estimate       SE   t.ratio
Golden Rain, 0    79.91667 8.220351  9.721807
Marvellous, 0     85.20833 8.220351 10.365534
Victory, 0        73.04167 8.220351  8.885468
Golden Rain, 0.2  99.41667 8.220351 12.093968
Marvellous, 0.2  104.70833 8.220351 12.737695
Victory, 0.2      92.54167 8.220351 11.257629
Golden Rain, 0.4 114.75000 8.220351 13.959257
Marvellous, 0.4  120.04167 8.220351 14.602985
Victory, 0.4     107.87500 8.220351 13.122918
Golden Rain, 0.6 123.91667 8.220351 15.074376
Marvellous, 0.6  129.20833 8.220351 15.718103
Victory, 0.6     117.04167 8.220351 14.238037
```

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

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

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

$‘Variety lsmeans‘
            estimate       SE  t.ratio
Golden Rain 103.2500 8.011712 12.88738
Marvellous  108.5417 8.011712 13.54787
Victory      96.3750 8.011712 12.02926
```

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

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

$‘Variety lsmeans‘
             estimate       SE  t.ratio
Golden Rain 103.88438 8.002227 12.98193
Marvellous  109.17604 8.002227 13.64321
Victory      97.00938 8.002227 12.12280
```

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

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

5 Empty cells

When a design is unbalanced, lsmeans are unambiguously defined in terms of predictions at the factor combinations in the model. However, if one or more combinations is actually missing, that adds some complications. Consider the following subset of the first three blocks of the `Oats` data. We will fit a model with interaction and obtain the lsmeans for the cells.

```
R> wildOats = Oats[c(2,6,8,10,15,18,19,20,21,25,26,27,31,32,33,34,35,36), ]
R> wildOats.lm = lm(yield ~ Variety*factor(nitro) + Block + Block:Variety, data=wildOats)
R> lsmeans(wildOats.lm, ~Variety:nitro)
```

```
$‘Variety:nitro lsmeans‘
                 estimate       SE  t.ratio
Golden Rain, 0         NA       NA       NA
Marvellous, 0    95.00000 5.937951 15.99879
Victory, 0       85.00000 8.092063 10.50412
Golden Rain, 0.2  96.33333 5.215362 18.47107
Marvellous, 0.2  135.00000 5.937951 22.73512
Victory, 0.2      81.00000 5.937951 13.64107
Golden Rain, 0.4 115.00000 5.215362 22.05024
Marvellous, 0.4  138.00000 8.092063 17.05375
Victory, 0.4     129.00000 5.937951 21.72467
Golden Rain, 0.6 128.66667 3.887301 33.09923
Marvellous, 0.6  130.00000 8.092063 16.06512
Victory, 0.6            NA       NA       NA
```

The results indicate `NA`s for the first and last cells. This is the default behavior of `lsmeans`, and it happens because there are special provisions in the code to identify empty cells. To see why this is necessary, consider a different parameterization of the same model, and let's compare the lsmeans for the two models with `check.cells=FALSE`, i.e., we don't take pains to check for empty cells.

```
R> wildOats.lm2 = lm(yield ~ Variety*ordered(nitro) + Block + Block:Variety, data=wildOats)
```

```
R> lsmeans(wildOats.lm, ~Variety:nitro,
R>     check.cells = FALSE)
```

```
$‘Variety:nitro lsmeans‘
                 estimate        SE   t.ratio
Golden Rain, 0    93.66667 10.284832  9.107262
Marvellous, 0     95.00000  5.937951 15.998785
Victory, 0        85.00000  8.092063 10.504120
Golden Rain, 0.2  96.33333  5.215362 18.471073
Marvellous, 0.2  135.00000  5.937951 22.735116
Victory, 0.2      81.00000  5.937951 13.641070
Golden Rain, 0.4 115.00000  5.215362 22.050243
Marvellous, 0.4  138.00000  8.092063 17.053748
Victory, 0.4     129.00000  5.937951 21.724666
Golden Rain, 0.6 128.66667  3.887301 33.099227
Marvellous, 0.6  130.00000  8.092063 16.065125
Victory, 0.6     120.00000 12.495925  9.603130
```

```
R> lsmeans(wildOats.lm2, ~Variety:nitro,
R>     check.cells = FALSE)
```

```
$‘Variety:nitro lsmeans‘
                 estimate        SE   t.ratio
Golden Rain, 0    46.66667 38.285477  1.218913
Marvellous, 0     95.00000  5.937951 15.998785
Victory, 0        85.00000  8.092063 10.504120
Golden Rain, 0.2  96.33333  5.215362 18.471073
Marvellous, 0.2  135.00000  5.937951 22.735116
Victory, 0.2      81.00000  5.937951 13.641070
Golden Rain, 0.4 115.00000  5.215362 22.050243
Marvellous, 0.4  138.00000  8.092063 17.053748
Victory, 0.4     129.00000  5.937951 21.724666
Golden Rain, 0.6 128.66667  3.887301 33.099227
Marvellous, 0.6  130.00000  8.092063 16.065125
Victory, 0.6     255.00000 42.286501  6.030293
```

Note that the lsmeans are identical for the cells that contain data, but they are drastically different for the two cells that are empty. Leaving `check.cells=TRUE` is definitely a good idea![2]

Finally, notice that if an additive model is used, the empty cells do not create a rank deficiency and are therefore not problematic:

```
R> wildOats.lma = update(wildOats.lm, . ~ . - Variety:factor(nitro))
R> wildOats.lma2 = update(wildOats.lm2, . ~ . - Variety:ordered(nitro))
R> lsmeans(wildOats.lma, ~Variety:nitro)[[1]] [c(1,2,12), ]
R> lsmeans(wildOats.lma2, ~Variety:nitro)[[1]] [c(1,2,12), ]
```

[2] This is an important distinction between SAS and R. SAS parameterizes factors using complete sets of indicators and checks for estimability. R uses a full-rank parameterization so that, by construction, everything is estimable, including inappropriate combinations.

```
               estimate        SE  t.ratio
Golden Rain, 0  81.55647 12.756235  6.39346
Marvellous, 0  104.59111  9.651565 10.83670
Victory, 0.6   121.12672 11.129134 10.88375
```

```
               estimate        SE  t.ratio
Golden Rain, 0  81.55647 12.756235  6.39346
Marvellous, 0  104.59111  9.651565 10.83670
Victory, 0.6   121.12672 11.129134 10.88375
```

The `lsmeans` library includes a function `empty.cells` that you may use to check for empty cells relative to a model:

```
R> empty.cells(~ Variety * nitro, wildOats)
[[1]]
       Variety nitro
1  Golden Rain     0
12     Victory   0.6
```

```
R> empty.cells(~ Variety + nitro, wildOats)
list()
```

6 GLMM example

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

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

```
R> library(lme4, quietly = TRUE, warn.conflicts = FALSE)
R> cbpp$obs = 1:nrow(cbpp)
R> cbpp.glmer = glmer(cbind(incidence, size - incidence)
R>     ~ period + (1 | herd) + (1 | obs),  family = binomial,  data = cbpp)

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

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

$`period lsmeans`
   estimate        SE   t.ratio pred.incidence
1 -1.500292 0.2887610 -5.195617     0.18238203
2 -2.726800 0.3809740 -7.157445     0.06141032
3 -2.829133 0.3994052 -7.083366     0.05577003
4 -3.366631 0.5193989 -6.481783     0.03335476

$`period differences from control`
       estimate        SE   t.ratio odds.ratio
2 - 1 -1.226509 0.4734567 -2.590541  0.2933148
3 - 1 -1.328841 0.4883951 -2.720833  0.2647839
4 - 1 -1.866339 0.5905702 -3.160233  0.1546889
```

In a way, the comparisons table is not needed because the results are the same as the regression coefficients under the default parameterization.

7 Miscellaneous

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

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

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

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

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

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

$‘type custom‘
            estimate      SE  t.ratio
A.vs.others 21.59777 4.49307 4.806907
```

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

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

```
R> inward.lsmc = function(levs, ...) {
R>     n = length(levs)
R>     result = data.frame(‘grand mean‘ = rep(1/n, n))
R>     for (i in 1 : floor(n/2)) {
R>         x = rep(0, n)
R>         x[1:i] = 1/i
R>         x[(n-i+1):n]  = -1/i
R>         result[[paste("first", i, "vs last", i)]] = x
R>     }
R>     attr(result, "desc") = "grand mean and inward contrasts"
R>     result
R> }
```

Testing it, we have

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

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

. . . and an application:

```
R> lsmeans(Oats.lme, inward ~ nitro) [-1]
```

```
$`nitro grand mean and inward contrasts`
                   estimate       SE   t.ratio
grand.mean        103.97222 6.640574  15.65711
first 1 vs last 1 -44.00000 4.249955 -10.35305
first 2 vs last 2 -29.66667 3.005172  -9.87187
```

References

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

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

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

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

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