1 Introduction

1.1 Linear functions of parameters, contrasts

A linear function of a $p$–dimensional parameter vector $\beta$ has the form

$$C = K\beta$$
where $K$ is a $q \times p$ matrix. The corresponding linear estimate is $\hat{C} = K\hat{\beta}$. A linear hypothesis has the form $H_0 : K\beta = m$ for some $q$ dimensional vector $m$.

## 1.2 Least-squares means (LS–means)

A special type of linear estimates is the so called least–squares means (or LS–means). Other names for these estimates include population means and marginal means. Consider an imaginary field experiment analyzed with model of the form

```r
> lm( y ~ treat + block + year)
```

where `treat` is a treatment factor, `block` is a blocking factor and `year` is the year (a factor) where the experiment is repeated over several years. This model specifies the conditional mean $\mathbb{E}(Y|\text{treat, block, year})$. One may be interested in predictions of the form $\mathbb{E}(Y|\text{treat})$. This quantity can not formally be found from the model. However, it is tempting to average the fitted values of $\mathbb{E}(Y|\text{treat, block, year})$ across the levels of `block` and `year` and think of this average as $\mathbb{E}(Y|\text{treat})$. This average is precisely what is called the LS–means. If the experiment is balanced then this average is identical to the average of the observations when stratified according to `treat`.

An alternative is to think of `block` and `year` as random effects, for example:

```r
> library(lme4)
> lmer( y ~ treat + (1|block) + (1|year))
```

In this case one would directly obtain $\mathbb{E}(Y|\text{treat})$ from the model. However, there are at least two reasons why one may be hesitant to consider such a random effects model.

- Suppose there are three blocks and the experiment is repeated over three consecutive years. This means that the random effects are likely to be estimated with a large uncertainty (the estimates will have only two degrees of freedom).

- Furthermore, treating `block` and `year` as random effects means they should in principle come from a large population of possible blocks and years. This may or may not be reasonable for the blocks, but it is certainly a dubious assumption for the years.

Below we describe `LSmeans` as implemented in the `doBy` package. Notice that the `lsmeans` package also provides computations of LS–means, see [http://cran.r-project.org/web/packages/lsmeans/](http://cran.r-project.org/web/packages/lsmeans/)

## 2 LS–means for linear models

### 2.1 LS–means – a first example

Consider these simulated data

```r
> simdat
  treat year y
1  t1  1  0.5
2  t1  1  1.0
3  t1  1  1.5
4  t2  1  3.0
```
shown in the figure below.

```r
> library(ggplot2)
> qplot(treat, y, data=simdat, color=year, size=I(3))
```

The LS–means under an additive model for the factor treat is

```r
> msim <- lm(y ~ treat + year, data=simdat)
> LSmeans( msim, effect="treat")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>treat</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.2415</td>
<td>5</td>
<td>8.281</td>
<td>4.192e-04</td>
<td>t1</td>
</tr>
<tr>
<td>4</td>
<td>0.2415</td>
<td>5</td>
<td>16.562</td>
<td>1.465e-05</td>
<td>t2</td>
</tr>
</tbody>
</table>
```

whereas the population means are

```r
> summaryBy(y~treat, data=simdat)

<table>
<thead>
<tr>
<th>treat</th>
<th>y.mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>1.5</td>
</tr>
<tr>
<td>t2</td>
<td>4.5</td>
</tr>
</tbody>
</table>
```

Had data been balanced (same number of observations for each combination of treat and year) the results would have been the same. An argument in favor of the LS–means is that these figures better represent what one would expect on in an “average year”.

### 2.2 Example: Warpbreaks

```r
> summary( warpbreaks )

<table>
<thead>
<tr>
<th>breaks</th>
<th>wool</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>:10.0</td>
<td>A:27</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>:18.2</td>
<td>B:27</td>
</tr>
<tr>
<td>Median</td>
<td>:26.0</td>
<td>H:18</td>
</tr>
<tr>
<td>Mean</td>
<td>:28.1</td>
<td></td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>:34.0</td>
<td></td>
</tr>
<tr>
<td>Max.</td>
<td>:70.0</td>
<td></td>
</tr>
</tbody>
</table>
```

```r
> head( warpbreaks, 4 )
```

---

3
breaks wool tension
1  26  A  L
2  30  A  L
3  54  A  L
4  25  A  L

> ftable(xtabs(~ wool + tension, data=warpbreaks))

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

> (warp.lm <- lm(breaks ~ wool + tension, data=warpbreaks))

Call:
  lm(formula = breaks ~ wool + tension, data = warpbreaks)

Coefficients:
  (Intercept) woolB tensionM tensionH
  39.28    -5.78    -10.00   -14.72

The fitted values are:
> uni <- unique(warpbreaks[,2:3])
> prd <- cbind(breaks=predict(warp.lm, newdata=uni), uni); prd

            breaks wool tension
breaks wool tension
1  39.28  A  L
10 29.28  A  M
19 24.56  A  H
28 33.50  B  L
37 23.50  B  M
46 18.78  B  H
2.3 The LS–means

We may be interested in making predictions of the number of breaks for each level of tension for any type or an average type of wool. The idea behind LS–means is to average the predictions above over the two wool types. These quantities are the LSmeans for the effect tension.

This is done with:

```
> LSmeans(warp.lm, effect="tension")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.39</td>
<td>2.738</td>
<td>50</td>
<td>13.289</td>
<td>4.948e-18</td>
<td>L</td>
</tr>
<tr>
<td>26.39</td>
<td>2.738</td>
<td>50</td>
<td>9.637</td>
<td>5.489e-13</td>
<td>M</td>
</tr>
<tr>
<td>21.67</td>
<td>2.738</td>
<td>50</td>
<td>7.913</td>
<td>2.269e-10</td>
<td>H</td>
</tr>
</tbody>
</table>
```

The term LSmeans comes from that these quantities are the same as the least squares main effects of tension when data is balanced:

```
> doBy::summaryBy(breaks ~ tension, data=warpbreaks)

tension breaks.mean
1 L  36.39
2 M  26.39
3 H  21.67
```

When data is not balanced these quantities are in general not the same.

2.4 LS–means for models with interactions

Consider a model with interaction:

```
> warp.lm2 <- update(warp.lm, .~.+wool:tension)
> coef( summary( warp.lm2 ))

Estimate Std. Error t value Pr(>|t|)
(Intercept) 44.56 3.647 12.218 2.426e-16
woolB -16.33 5.157 -3.167 2.677e-03
tensionM -20.56 5.157 -3.986 2.281e-04
tensionH -20.00 5.157 -3.878 3.199e-04
woolB:tensionM 21.11 7.294 2.895 5.698e-03
woolB:tensionH 10.56 7.294 1.447 1.543e-01
```

In this case the contrast matrix becomes:

```
> K2 <- LSmatrix(warp.lm2, effect="tension"); K2

      (Intercept) woolB tensionM tensionH woolB:tensionM woolB:tensionH
[1,]    1 0.5      0      0       0.0      0.0
[2,]    1 0.5      1      0       0.5      0.0
[3,]    1 0.5      0      1       0.0      0.5
```

```
> linest(warp.lm2, K=K2)

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.39</td>
<td>2.579</td>
<td>48</td>
<td>14.112</td>
<td>1.055e-18</td>
<td>L</td>
</tr>
<tr>
<td>26.39</td>
<td>2.579</td>
<td>48</td>
<td>10.234</td>
<td>1.183e-13</td>
<td>M</td>
</tr>
<tr>
<td>21.67</td>
<td>2.579</td>
<td>48</td>
<td>8.402</td>
<td>5.468e-11</td>
<td>H</td>
</tr>
</tbody>
</table>
```
3 Using the at= argument

> library(ggplot2)
> ChickWeight$Diet <- factor(ChickWeight$Diet)
> qplot(Time, weight, data=ChickWeight, colour=Chick, facets="Diet, 
geom=c("point","line"))

Consider random regression model:

> library(lme4)
> rr <- lmer(weight ~ Time*Diet + (0+Time|Chick), data=ChickWeight)
> coef(summary(rr))

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>33.218</td>
<td>1.7697</td>
</tr>
<tr>
<td>Time</td>
<td>6.339</td>
<td>0.6103</td>
</tr>
<tr>
<td>Diet2</td>
<td>-4.585</td>
<td>3.0047</td>
</tr>
<tr>
<td>Diet3</td>
<td>-14.968</td>
<td>3.0047</td>
</tr>
<tr>
<td>Diet4</td>
<td>-1.454</td>
<td>3.0177</td>
</tr>
<tr>
<td>Time:Diet2</td>
<td>2.271</td>
<td>1.0367</td>
</tr>
<tr>
<td>Time:Diet3</td>
<td>5.084</td>
<td>1.0367</td>
</tr>
<tr>
<td>Time:Diet4</td>
<td>3.217</td>
<td>1.0377</td>
</tr>
</tbody>
</table>

The contrast matrix for Diet becomes:

> LSmatrix(rr, effect="Diet")

<table>
<thead>
<tr>
<th>(Intercept)</th>
<th>Time</th>
<th>Diet2</th>
<th>Diet3</th>
<th>Diet4</th>
<th>Time:Diet2</th>
<th>Time:Diet3</th>
<th>Time:Diet4</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>10.72</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>[2,]</td>
<td>1</td>
<td>10.72</td>
<td>1</td>
<td>0</td>
<td>10.72</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>[3,]</td>
<td>1</td>
<td>10.72</td>
<td>0</td>
<td>1</td>
<td>0.00</td>
<td>10.72</td>
<td>0.00</td>
</tr>
<tr>
<td>[4,]</td>
<td>1</td>
<td>10.72</td>
<td>0</td>
<td>0</td>
<td>1.00</td>
<td>0.00</td>
<td>10.72</td>
</tr>
</tbody>
</table>

The value of Time is by default taken to be the average of that variable. Hence the LSmatrix is the predicted weight for each diet at that specific point of time. We can consider other points of time with

> K1 <- LSmatrix(rr, effect="Diet", at=list(Time=1)); K1

<table>
<thead>
<tr>
<th>(Intercept)</th>
<th>Time</th>
<th>Diet2</th>
<th>Diet3</th>
<th>Diet4</th>
<th>Time:Diet2</th>
<th>Time:Diet3</th>
<th>Time:Diet4</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,]</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[2,]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[3,]</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>[4,]</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
The LSmeans for the intercepts is the predictions at Time=0. The LSmeans for the slopes becomes

```r
K0 <- LSmatrix(rr, effect="Diet", at=list(Time=0))
K1-K0
```

(Intercept) Time Diet2 Diet3 Diet4 Time:Diet2 Time:Diet3 Time:Diet4
[1,]  0  1  0  0  0  0  0  0
[2,]  0  1  0  0  0  1  0  0
[3,]  0  1  0  0  0  0  1  0
[4,]  0  1  0  0  0  0  0  1

```r
> LSmeans(rr, K=K1-K0)
```

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>Diet</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.339</td>
<td>0.6105</td>
<td>49.86</td>
<td>10.38</td>
<td>4.632e-14</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8.609</td>
<td>0.8380</td>
<td>48.28</td>
<td>10.27</td>
<td>9.705e-14</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11.423</td>
<td>0.8380</td>
<td>48.28</td>
<td>13.63</td>
<td>3.588e-18</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9.556</td>
<td>0.8392</td>
<td>48.56</td>
<td>11.39</td>
<td>2.584e-15</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

We can cook up our own function for comparing trends:

```r
> LSmeans_trend <- function(object, effect, trend){
  K<-LSmatrix(object, effect=effect, at=as.list(setNames(1, trend))) -
  LSmatrix(object, effect=effect, at=as.list(setNames(0, trend)))
  LSmeans(object, K=K)
}

> LSmeans_trend(rr, effect="Diet", trend="Time")
```

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>Diet</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.339</td>
<td>0.6105</td>
<td>49.86</td>
<td>10.38</td>
<td>4.632e-14</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8.609</td>
<td>0.8380</td>
<td>48.28</td>
<td>10.27</td>
<td>9.705e-14</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11.423</td>
<td>0.8380</td>
<td>48.28</td>
<td>13.63</td>
<td>3.588e-18</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9.556</td>
<td>0.8392</td>
<td>48.56</td>
<td>11.39</td>
<td>2.584e-15</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

4 Using (transformed) covariates

Consider the following subset of the CO2 dataset:

```r
> data(CO2)
> CO2 <- transform(CO2, Treat=Treatment, Treatment=NULL)
> levels(CO2$Treat) <- c("nchil","chil")
> levels(CO2$Type) <- c("Que","Mis")
> ftable(xtabs(~ Plant + Type + Treat, data=CO2), col.vars=2:3)
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Que</th>
<th>Mis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treat</td>
<td>nchil</td>
<td>chil</td>
</tr>
</tbody>
</table>

Plants
- Qn1: 7 0 0 0
- Qn2: 7 0 0 0
- Qn3: 7 0 0 0
- Qc1: 0 7 0 0
- Qc3: 0 7 0 0
- Qc2: 0 7 0 0
- Mn3: 0 0 7 0
- Mn2: 0 0 7 0
Below, the covariate conc is fixed at the average value:

```r
> co2.lm1 <- lm(uptake ~ conc + Type + Treat, data=CO2)
> LSmeans(co2.lm1, effect="Treat")
estimate      se    df  t.stat  p.value Treat conc
 1  30.64 0.9556  80 32.07 2.010e-47 nchil 435
 2  23.78 0.9556  80 24.89 2.037e-39 chil 435
```

If we use log(conc) instead we will get an error when calculating LS–means:

```r
> co2.lm <- lm(uptake ~ log(conc) + Type + Treat, data=CO2)
> LSmeans(co2.lm, effect="Treat")
```

In this case one can do

```r
> co2.lm2 <- lm(uptake ~ log.conc + Type + Treat,
                   data=transform(CO2, log.conc=log(conc)))
> LSmeans(co2.lm2, effect="Treat")
```

This also highlights what is computed: The average of the log of conc; not the log of the average of conc.

In a similar spirit consider

```r
> co2.lm3 <- lm(uptake ~ conc + I(conc^2) + Type + Treat, data=CO2)
> LSmeans(co2.lm3, effect="Treat")
```

Above I(conc^2) is the average of the squared values of conc; not the square of the average of conc, cfr. the following.
> co2.lm4 <- lm(uptake ~ conc + conc2 + Type + Treat, data=transform(CO2, conc2=conc^2))
> LSmeans(co2.lm4, effect="Treat")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>Treat</th>
<th>conc</th>
<th>conc2</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.64</td>
<td>0.78</td>
<td>79</td>
<td>39.46</td>
<td>9.318e-54</td>
<td>nchil</td>
<td>435</td>
<td>275754</td>
</tr>
<tr>
<td>23.78</td>
<td>0.78</td>
<td>79</td>
<td>30.63</td>
<td>1.356e-45</td>
<td>chil</td>
<td>435</td>
<td>275754</td>
</tr>
</tbody>
</table>

If we want to evaluate the LS–means at conc=10 then we can do:

> LSmeans(co2.lm4, effect="Treat", at=list(conc=10, conc2=100))

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>Treat</th>
<th>conc</th>
<th>conc2</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.735</td>
<td>1.70</td>
<td>79</td>
<td>8.662</td>
<td>4.456e-13</td>
<td>nchil</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>7.876</td>
<td>1.70</td>
<td>79</td>
<td>4.630</td>
<td>1.417e-05</td>
<td>chil</td>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>

5 Alternative models

5.1 Generalized linear models

We can calculate LS–means for e.g. a Poisson or a gamma model. Default is that the calculation is calculated on the scale of the linear predictor. However, if we think of LS–means as a prediction on the linear scale one may argue that it can also make sense to transform this prediction to the response scale:

> warp.poi <- glm(breaks ~ wool + tension, family=poisson, data=warpbreaks)
> LSmeans(warp.poi, effect="tension", type="link")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>z.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.589</td>
<td>0.039</td>
<td>91.64</td>
<td>0</td>
<td>L</td>
</tr>
<tr>
<td>3.268</td>
<td>0.046</td>
<td>71.10</td>
<td>0</td>
<td>M</td>
</tr>
<tr>
<td>3.070</td>
<td>0.051</td>
<td>60.55</td>
<td>0</td>
<td>H</td>
</tr>
</tbody>
</table>

> LSmeans(warp.poi, effect="tension", type="response")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>z.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.20</td>
<td>1.42</td>
<td>91.64</td>
<td>0</td>
<td>L</td>
</tr>
<tr>
<td>26.25</td>
<td>1.22</td>
<td>71.10</td>
<td>0</td>
<td>M</td>
</tr>
<tr>
<td>21.55</td>
<td>1.10</td>
<td>60.55</td>
<td>0</td>
<td>H</td>
</tr>
</tbody>
</table>

> warp.qpoi <- glm(breaks ~ wool + tension, family=quasipoisson, data=warpbreaks)
> LSmeans(warp.qpoi, effect="tension", type="link")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>z.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.589</td>
<td>0.081</td>
<td>44.39</td>
<td>0.000e+00</td>
<td>L</td>
</tr>
<tr>
<td>3.268</td>
<td>0.095</td>
<td>34.44</td>
<td>6.093e-260</td>
<td>M</td>
</tr>
<tr>
<td>3.070</td>
<td>0.105</td>
<td>29.33</td>
<td>3.883e-189</td>
<td>H</td>
</tr>
</tbody>
</table>

> LSmeans(warp.qpoi, effect="tension", type="response")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>z.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.20</td>
<td>2.93</td>
<td>44.39</td>
<td>0.000e+00</td>
<td>L</td>
</tr>
<tr>
<td>26.25</td>
<td>2.49</td>
<td>34.44</td>
<td>6.093e-260</td>
<td>M</td>
</tr>
<tr>
<td>21.55</td>
<td>2.26</td>
<td>29.33</td>
<td>3.883e-189</td>
<td>H</td>
</tr>
</tbody>
</table>

For comparison with the linear model, we use identity link
> warp.gam <- glm(breaks ~ wool + tension, family=Gamma(link=identity),
   data=warpbreaks)
> LSmeans(warp.gam, effect="tension", type="link")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.66</td>
<td>3.222</td>
<td>50</td>
<td>11.07</td>
<td>4.766e-15</td>
<td>L</td>
</tr>
<tr>
<td>27.12</td>
<td>2.448</td>
<td>50</td>
<td>11.08</td>
<td>4.543e-15</td>
<td>M</td>
</tr>
<tr>
<td>21.53</td>
<td>1.944</td>
<td>50</td>
<td>11.08</td>
<td>4.629e-15</td>
<td>H</td>
</tr>
</tbody>
</table>

Notice that the linear estimates are practically the same as for the linear model, but the standard errors are smaller and hence the confidence intervals are narrower.

An alternative is to fit a quasi Poisson “model”

> warp.poi3 <- glm(breaks ~ wool + tension, family=quasipoisson(link=identity),
   data=warpbreaks)
> LSmeans(warp.poi3, effect="tension")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>z.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.00</td>
<td>2.950</td>
<td>12.204</td>
<td>2.965e-34</td>
<td>L</td>
</tr>
<tr>
<td>26.83</td>
<td>2.544</td>
<td>10.546</td>
<td>5.316e-26</td>
<td>M</td>
</tr>
<tr>
<td>21.62</td>
<td>2.281</td>
<td>9.475</td>
<td>2.657e-21</td>
<td>H</td>
</tr>
</tbody>
</table>

5.2 Linear mixed effects model

For the sake of illustration we treat wool as a random effect:

> library(lme4)
> warp.mm <- lmer(breaks ~ tension + (1|wool), data=warpbreaks)
> LSmeans(warp.mm, effect="tension")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.39</td>
<td>3.653</td>
<td>2.538</td>
<td>9.961</td>
<td>0.004230</td>
<td>L</td>
</tr>
<tr>
<td>26.39</td>
<td>3.653</td>
<td>2.538</td>
<td>7.224</td>
<td>0.009354</td>
<td>M</td>
</tr>
<tr>
<td>21.67</td>
<td>3.653</td>
<td>2.538</td>
<td>5.931</td>
<td>0.015093</td>
<td>H</td>
</tr>
</tbody>
</table>

Notice here that the estimates themselves are very similar to those above but the standard errors are much larger. This comes from the fact that wool is treated as a random effect.

> VarCorr(warp.mm)

Groups   Name Std.Dev.
wool     (Intercept) 3.42
Residual 11.62

Notice that the degrees of freedom by default are adjusted using a Kenward–Rogier approximation (provided that pbkrtest is installed). Unadjusted degrees of freedom are obtained with

> LSmeans(warp.mm, effect="tension", adjust.df=FALSE)

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.39</td>
<td>3.653</td>
<td>49</td>
<td>9.961</td>
<td>2.288e-13</td>
<td>L</td>
</tr>
<tr>
<td>26.39</td>
<td>3.653</td>
<td>49</td>
<td>7.224</td>
<td>2.986e-09</td>
<td>M</td>
</tr>
<tr>
<td>21.67</td>
<td>3.653</td>
<td>49</td>
<td>5.931</td>
<td>2.986e-07</td>
<td>H</td>
</tr>
</tbody>
</table>
5.3 Generalized estimating equations

Lastly, for gee-type “models” we get

```r
> library(geepack)
> warp.gee <- geeglm(breaks ~ tension, id=wool, family=poisson, data=warpbreaks)
> LSmeans(warp.gee, effect="tension")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>z.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.594</td>
<td>0.15869</td>
<td>22.65</td>
<td>1.427e-113</td>
<td>L</td>
</tr>
<tr>
<td>3.273</td>
<td>0.06401</td>
<td>51.13</td>
<td>0.000e+00</td>
<td>M</td>
</tr>
<tr>
<td>3.076</td>
<td>0.09428</td>
<td>32.62</td>
<td>1.903e-233</td>
<td>H</td>
</tr>
</tbody>
</table>

> LSmeans(warp.gee, effect="tension", type="response")

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>z.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.39</td>
<td>5.775</td>
<td>22.65</td>
<td>1.427e-113</td>
<td>L</td>
</tr>
<tr>
<td>26.39</td>
<td>1.689</td>
<td>51.13</td>
<td>0.000e+00</td>
<td>M</td>
</tr>
<tr>
<td>21.67</td>
<td>2.043</td>
<td>32.62</td>
<td>1.903e-233</td>
<td>H</td>
</tr>
</tbody>
</table>
```

6 Miscellaneous

6.1 Under the hood

Under the hood, `LSmeans()` generates a contrast matrix

```r
> K <- LSmatrix(warp.lm, effect="tension"); K

(Intercept) woolB tensionM tensionH
[1,] 1 0.5 0 0
[2,] 1 0.5 1 0
[3,] 1 0.5 0 1
```

and passes this matrix onto `linest()`:

```r
> linest( warp.lm, K=K )

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>tension</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.39</td>
<td>2.738</td>
<td>50</td>
<td>13.289</td>
<td>4.948e-18</td>
<td>L</td>
</tr>
<tr>
<td>26.39</td>
<td>2.738</td>
<td>50</td>
<td>9.637</td>
<td>5.489e-13</td>
<td>M</td>
</tr>
<tr>
<td>21.67</td>
<td>2.738</td>
<td>50</td>
<td>7.913</td>
<td>2.269e-10</td>
<td>H</td>
</tr>
</tbody>
</table>
```

6.2 Example: Non–estimable contrasts

Consider this highly unbalanced simulated dataset:

```r
> head(dat.nst)

<table>
<thead>
<tr>
<th>AA</th>
<th>BB</th>
<th>CC</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-0.52701</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-0.72391</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>-1.33544</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>-0.02428</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>-0.57596</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
<td>-1.45776</td>
</tr>
</tbody>
</table>
```
> ftable(xtabs(~ AA + BB + CC, data=dat.nst))

<table>
<thead>
<tr>
<th>CC</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2 1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

We have

> mod.nst <- lm(y ~ AA + BB : CC, data=dat.nst)
> coef(mod.nst)

(Intercept)       AA2 BB1:CC1 BB2:CC1 BB3:CC1 BB1:CC2
  -0.43358    -0.02948    0.43235      NA      NA
  -0.23154    -0.56854      NA   0.76186   0.89603      NA
BB2:CC4 BB3:CC4
  1.36700      NA

In this case some of the LSmeans values are not estimable (see Section 6.3 for details):

> LSmeans(mod.nst, effect=c("BB", "CC"))

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>BB</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.01597</td>
<td>0.3569</td>
<td>10</td>
<td>-0.04475</td>
<td>0.9652</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>-0.67986</td>
<td>0.6182</td>
<td>10</td>
<td>-1.09965</td>
<td>0.2973</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>-1.01666</td>
<td>0.6182</td>
<td>10</td>
<td>-1.64475</td>
<td>0.1310</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0.31355</td>
<td>0.6182</td>
<td>10</td>
<td>0.50715</td>
<td>0.6230</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>0.44771</td>
<td>0.6182</td>
<td>10</td>
<td>0.72416</td>
<td>0.4856</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0.91868</td>
<td>0.6182</td>
<td>10</td>
<td>1.48594</td>
<td>0.1681</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>-0.44832</td>
<td>0.6182</td>
<td>10</td>
<td>-0.72515</td>
<td>0.4850</td>
<td>3</td>
</tr>
</tbody>
</table>

### 6.3 Handling non-estimability

The model matrix for the model in Section 6.2 does not have full column rank and therefore not all values are calculated by LSmeans().

> X <- model.matrix(mod.nst); as(X,"Matrix")

18 x 14 sparse Matrix of class "dgCMatrix"

1 1 . . . . . . . . . . . .
2 1 1 1 . . . . . . . . . .
3 1 . . . . 1 . . . . . .
4 1 1 . . . . 1 . . . . .
5 1 . . . . 1 . . . . . .
6 1 1 . . . . 1 . . . . .
We consider a linear normal model, i.e. an \( n \) dimensional random vector \( y = (y_i) \) for which \( \mathbb{E}(y) = \mu = X \beta \) and \( \text{Cov}(y) = \sigma^2 I \) where \( X \) does not have full column rank. We are interested in linear functions of \( \beta \), say

\[
c = k^T \beta = \sum_j k_j \beta_j.
\]

\[
> \text{K <- LSmatrix(mod.nst, effect="BB", at=list(CC=2));K}
\]

[1,] 1 0.5 0 0 0 1 0 0 0
[2,] 1 0.5 0 0 0 0 1 0 0
[3,] 1 0.5 0 0 0 0 0 1 0

BB2:CC3 BB3:CC3 BB1:CC4 BB2:CC4 BB3:CC4
[1,] 0 0 0 0 0
[2,] 0 0 0 0 0
[3,] 0 0 0 0 0

\[
> \text{LSmeans(mod.nst, K=K)}
\]

<table>
<thead>
<tr>
<th>estimate</th>
<th>se</th>
<th>df</th>
<th>t.stat</th>
<th>p.value</th>
<th>BB</th>
<th>CC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>-0.6799</td>
<td>0.6182</td>
<td>10</td>
<td>-1.100</td>
<td>0.2973</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>-1.0169</td>
<td>0.6182</td>
<td>10</td>
<td>-1.645</td>
<td>0.1310</td>
<td>3</td>
</tr>
</tbody>
</table>

A least squares estimate of \( \beta \) is

\[
\hat{\beta} = GX^T y
\]

where \( G \) is a generalized inverse of \( X^T X \). Since the generalized inverse is not unique then neither is the estimate \( \hat{\beta} \). One least squares estimate of \( \beta \) is

\[
> \text{XtXinv <- MASS::ginv(t(X)%*%X)}
\]

\[
> \text{bhat <- as.numeric(XtXinv %*% t(X) %*% dat.nst$y)}
\]

\[
> \text{zapsmall(bhat)}
\]

\[
[1] \ -0.0472 \ -0.0295 \ 0.0460 \ 0.0000 \ 0.0000 \ 0.0000 \ -0.6179 \ -0.9549 \ 0.0000
\]

\[
[10] \ 0.3755 \ 0.5097 \ 0.0000 \ 0.9807 \ -0.3863
\]

Hence \( \hat{c} = k^T \hat{\beta} \) is in general not unique.

\[
> \text{K \%*% bhat}
\]

\[
[,1]
\]

\[
[1,] \ -0.06198
[2,] \ -0.67986
[3,] \ -1.01686
\]
However, for some values of \(k\), the estimate \(\hat{c}\) is unique (i.e. it does not depend on the choice of generalized inverse). Such linear functions are said to be estimable and can be described as follows:

All we specify with \(\mu = X\beta\) is that \(\mu\) is a vector in the linear subspace \(L = C(X)\) where \(C(X)\) denotes the column space of \(X\). We can only learn about \(\beta\) through \(X\beta\) so the only thing we can say something about is linear combinations \(\rho^T X\beta\). Hence we can only say something about \(k^T \beta\) if there exists \(\rho\) such that \(k^T \beta = \rho^T X\beta\), i.e., if \(k = X^T \rho\), that is, if \(k\) is in the column space \(C(X^T)\) of \(X^T\). That is, if \(k\) is perpendicular to all vectors in the null space \(N(X)\) of \(X\). To check this, we find a basis \(B\) for \(N(X)\). This can be done in many ways, for example via a singular value decomposition of \(X\), i.e.

\[
X = UDV^T
\]

A basis for \(N(X)\) is given by those columns of \(V\) that corresponds to zeros on the diagonal of \(D\).

```r
S <- svd(X)
> names(S)
[1] "d" "u" "v"

B <- S$v[, S$d < 1e-10, drop = FALSE ]; zapsmall(B) ## Basis for N(X)
[1,]  -0.3535  0.0067  -0.0001  -0.0016  -0.0023  0.0000
[2,]   0.0000   0.0000  0.00000   0.0000   0.0000   0.0000
[3,]   0.3535  -0.0067  0.00016  0.00160  0.00230  0.0000
[4,]  -0.0202  -0.8619   0.0965  0.13100  0.47980  0.0000
[5,]   0.0039   0.2046  0.48078  -0.71410  0.46580  0.0000
[6,]  -0.0018  -0.4633  0.06278  -0.54460  -0.69630  0.0000
[7,]   0.3535  -0.0067  0.00016  0.00160  0.00230  0.0000
[8,]   0.3535  -0.0067  0.00016  0.00160  0.00230  0.0000
[9,]   0.0000   0.0000  0.00000  0.00000   0.00000  0.0000
[10,]  0.3535  -0.0067  0.00016  0.00160  0.00230  0.0000
[11,]  0.3535  -0.0067  0.00016  0.00160  0.00230  0.0000
[12,]  0.0000   0.0000  0.00000  0.00000   0.00000  0.0000
[13,]  0.3535  -0.0067  0.00016  0.00160  0.00230  0.0000
[14,]  0.3535  -0.0067  0.00016  0.00160  0.00230  0.0000

> zapsmall( rowSums(K %*% B) )
[1]  -1.994  0.000  0.000
```

### 6.4 Pairwise comparisons

We will just mention that for certain other linear estimates, the matrix \(K\) can be generated automatically using `glht()` from the `multcomp` package. For example, pairwise comparisons of all levels of `tension` can be obtained with

```r
> library("multcomp")
> g1 <- glht(warp.lm, mcp(tension="Tukey"))
> summary( g1 )

Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts

> 14
Fit: \texttt{lm(formula = breaks \sim wool + tension, data = warpbreaks)}

Linear Hypotheses:

\begin{tabular}{lrrrr}
& Estimate & Std. Error & t value & Pr(>|t|) \\
M - L == 0 & -10.00 & 3.87 & -2.58 & 0.0335 * \\
H - L == 0 & -14.72 & 3.87 & -3.80 & 0.0011 ** \\
H - M == 0 & -4.72 & 3.87 & -1.22 & 0.4474 \\
\end{tabular}

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Adjusted p values reported -- single-step method)

The $K$ matrix generated in this case is:

```r
> K1 <- g1$linfct; K1

(Intercept) woolB tensionM tensionH
M - L 0 0 1 0
H - L 0 0 0 1
H - M 0 0 -1 1
attr(,"type")
[1] "Tukey"
```