Linear estimates – including LS–means (least–squares means)

Søren Højsgaard and Ulrich Halekoh

doBy version 4.6-0 as of 2018-03-05

Contents

1 Introduction

1.1 Linear functions of parameters, contrasts

A linear function of a p–dimensional parameter vector β has the form

$$C = L\beta$$

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

1.2 Tooth growth

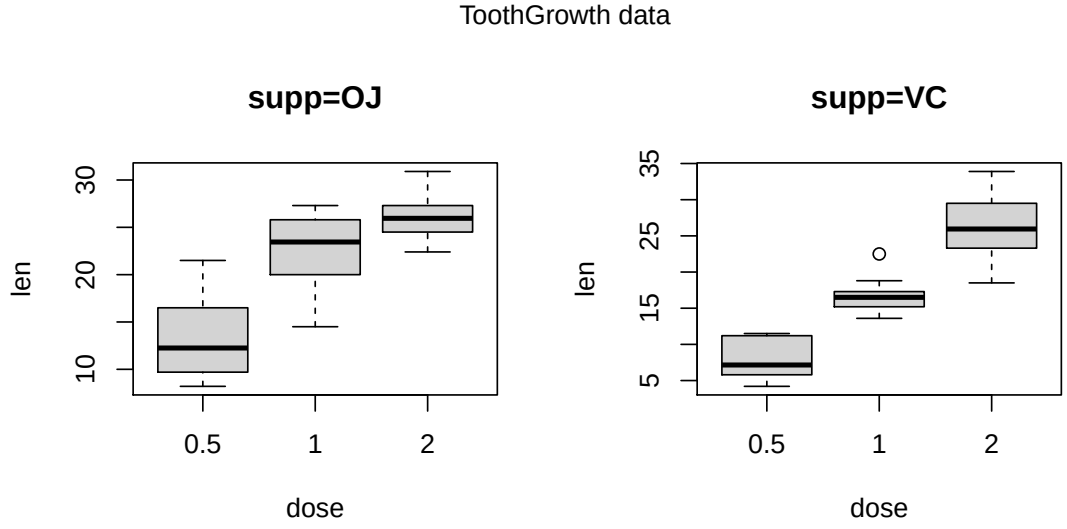
The response is the length of odontoblasts (cells responsible for tooth growth) in 60 guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, (orange juice (coded as OJ) or ascorbic acid (a form of vitamin C and (coded as VC)).

```
head(ToothGrowth, 4)
```

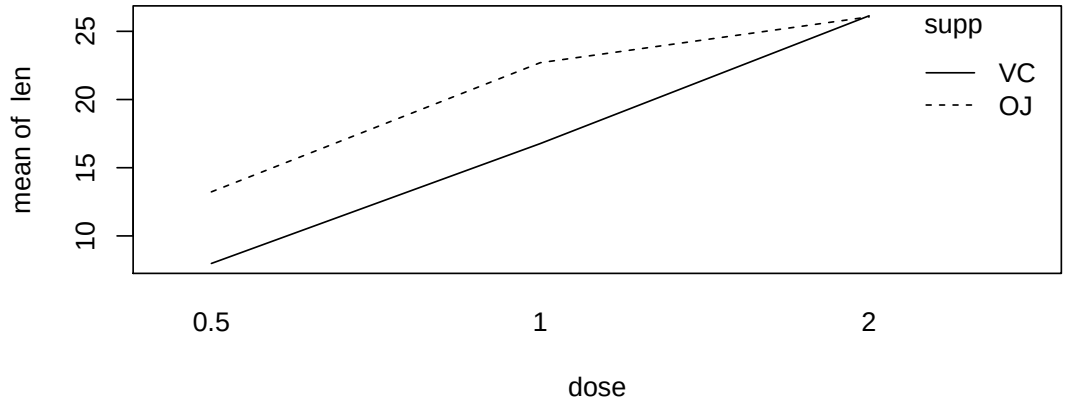
```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
```

```
ftable(xtabs(~ dose + supp, data=ToothGrowth))
```

```
##      supp OJ VC
## dose
## 0.5       10 10
## 1         10 10
## 2         10 10
```




```
with(ToothGrowth, interaction.plot(dose, supp, len))
```



The interaction plot suggests a mild interaction which is supported by a formal comparison:

```
ToothGrowth$dose <- factor(ToothGrowth$dose)
tooth1 <- lm(len ~ dose + supp, data=ToothGrowth)
tooth2 <- lm(len ~ dose * supp, data=ToothGrowth)
anova(tooth1, tooth2)

## Analysis of Variance Table
##
## Model 1: len ~ dose + supp
## Model 2: len ~ dose * supp
##   Res.Df RSS Df Sum of Sq    F Pr(>F)
## 1     56 820
## 2     54 712  2       108 4.11  0.022 *
```

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

2 Computing linear estimates

For now, we focus on the additive model:

```
tooth1


##
## Call:
## lm(formula = len ~ dose + supp, data = ToothGrowth)
##
## Coefficients:
## (Intercept)        dose1        dose2       suppVC
##       12.46         9.13        15.49        -3.70
```

The estimated length for each dose of orange juice (OJ) can be found as follows. Notice: `esticon` has been in the **doBy** package for many years; `linest` is a newer addition; `esticon` is not actively maintained but remains in **doBy** for historical reasons.

```
L <- matrix(c(1, 0, 0, 0,
              1, 1, 0, 0,
              1, 0, 1, 0), nrow=3, byrow=T)
c1 <- esticon(tooth1, L)
c1


##       beta0 Estimate Std.Error t.value     DF Pr(>|t|)  Lower Upper
## [1,]  0.000   12.455     0.988  12.603 56.000    0.000 10.475  14.4
## [2,]  0.000   21.585     0.988  21.841 56.000    0.000 19.605  23.6
## [3,]  0.000   27.950     0.988  28.281 56.000    0.000 25.970  29.9

c2 <- linest(tooth1, L)
c2


## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   12.455  0.988 56.000 12.603       0
## [2,]   21.585  0.988 56.000 21.841       0
## [3,]   27.950  0.988 56.000 28.281       0
```

We can do:

```
summary(c2)


## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   12.455  0.988 56.000 12.603       0
## [2,]   21.585  0.988 56.000 21.841       0
```

```
## [3,]   27.950  0.988 56.000 28.281       0
##
## Grid:
## NULL
##
## L:
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    1    1    0    0
## [3,]    1    0    1    0
```

```
coef(c2)
```

```
##   estimate     se df t.stat   p.value
## 1    12.46 0.9883 56  12.60 5.490e-18
## 2    21.59 0.9883 56  21.84 4.461e-29
## 3    27.95 0.9883 56  28.28 7.627e-35
```

```
confint(c2)
```

```
##   0.025 0.975
## 1 10.52 14.39
## 2 19.65 23.52
## 3 26.01 29.89
```

3 Automatic generation of L

The matrix L can be generated as follows:

```
L <- linest_matrix(tooth1, effect="dose", at=list(supp="OJ"))
```

```
## List of 8
##  $ effect        : chr "dose"
##  $ at            :List of 1
##   ..$ supp: chr "OJ"
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr "OJ"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##   ..- attr(*, "at.num")= Named list()
##  $ at.factor.name: chr "supp"
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
```

```
L
```

```
##      (Intercept) dose1 dose2 suppVC
## [1,]           1     0     0      0
## [2,]           1     1     0      0
## [3,]           1     0     1      0
```

4 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. Notice that the **lsmeans** package **?** also provides computations of LS–means, see `http://cran.r-project.org/web/packages/lsmeans/`.

4.1 LS–means on a simple example

Consider these simulated data, also shown in Fig. **??**:

```
simdat
```

```
##   treat year   y
## 1    t1    1 0.5
## 2    t1    1 1.0
## 3    t1    1 1.5
## 4    t2    1 3.0
## 5    t1    2 3.0
## 6    t2    2 4.5
## 7    t2    2 5.0
## 8    t2    2 5.5
```

```
library(ggplot2)
qplot(treat, y, data=simdat, color=year)
```

The LS–means under an additive model for the factor `treat` is the predicted outcome for each level of `treat` for an "average year":

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

```
## List of 8
##  $ effect        : chr "treat"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ treat: chr [1:2] "t1" "t2"
##   ..$ year : chr [1:2] "1" "2"
##  $ new.fact.lev  :List of 1
##   ..$ treat: chr [1:2] "t1" "t2"
##  $ vartype       :List of 2
```

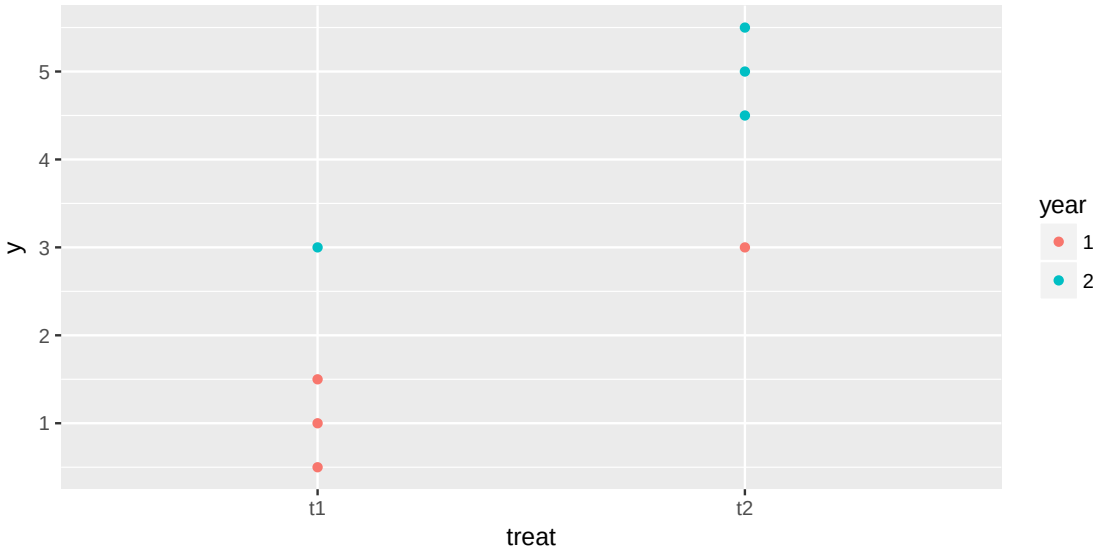


Figure 1: LSmeans - simulated data.

```
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "treat" "year"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...

lsm

## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]    2.000  0.242  5.000  8.281       0
## [2,]    4.000  0.242  5.000 16.562       0
```

Notice that the population means are

```
summaryBy(y ~ treat, data=simdat, FUN=function(x) c(m=mean(x), s=sd(x)))

##   treat  y.m  y.s
## 1    t1 1.50 1.08
## 2    t2 4.50 1.08

## or aggregate(y ~ treat, data=simdat, FUN=function(x) c(m=mean(x), s=sd(x)))
```

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".

An alternative is to think of `year` as a random effect, for example:

```
library(lme4)
lmer( y ~ treat  + (1|year), data=simdat)
```

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

- It is a "never ending" discussion whether `year` should be treated as fixed or random. For `year` to be a random effect, it should in principle come from a large population of possible years. This is certainly a dubious assumption for the years.
- If we accept to think of year as a random effect, then the variance describing this random effect will be poorly determined (because there are only two years in the study).

4.2 Additive model

Returning to the `ToothGrowth` data, orange juice and ascorbic acid are just two of many ways of supplying vitamin C (citrus and lime juice would be two alternatives). Here one can therefore argue, that it would make sense to estimate the the effect for each dose for an "average vitamin C source":

```
LSmeans(tooth1, effect="dose")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   10.605  0.856 56.000 12.391       0
## [2,]   19.735  0.856 56.000 23.058       0
## [3,]   26.100  0.856 56.000 30.495       0
```

which is the same as

```
L <- linest_matrix(tooth1, effect="dose")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
```

```
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...

L

##      (Intercept) dose1 dose2 suppVC
## [1,]           1     0     0    0.5
## [2,]           1     1     0    0.5
## [3,]           1     0     1    0.5

le <- linest(tooth1, L=L)
coef(le)

##   estimate     se df t.stat   p.value
## 1    10.61 0.8559 56  12.39 1.109e-17
## 2    19.73 0.8559 56  23.06 2.885e-30
## 3    26.10 0.8559 56  30.50 1.444e-36
```

4.3 Interaction model

For a model with interactions, the LSmeans are

```
LSmeans(tooth2, effect="dose")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   10.605  0.812 54.000 13.060       0
## [2,]   19.735  0.812 54.000 24.304       0
```

```
## [3,]   26.100  0.812 54.000 32.143       0
```

In this case, the L matrix is

```
L <- linest_matrix(tooth2, effect="dose")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...

t(L)

##              [,1] [,2] [,3]
## (Intercept)   1.0  1.0  1.0
## dose1         0.0  1.0  0.0
## dose2         0.0  0.0  1.0
## suppVC        0.5  0.5  0.5
## dose1:suppVC  0.0  0.5  0.0
## dose2:suppVC  0.0  0.0  0.5
```

5 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)
chick <- lmer(weight ~ Time * Diet + (0 + Time | Chick),
           data=ChickWeight)
coef(summary(chick))

##             Estimate Std. Error t value
## (Intercept)   33.218     1.7697 18.7701
```

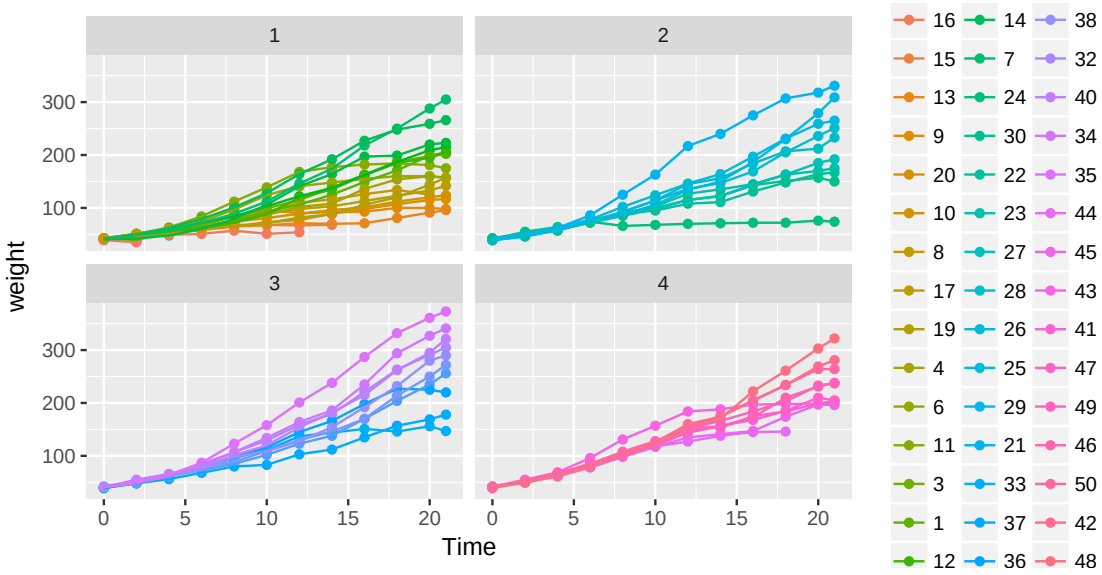


Figure 2: ChickWeight data.

```
## Time           6.339     0.6103 10.3855
## Diet2         -4.585     3.0047 -1.5258
## Diet3        -14.968     3.0047 -4.9815
## Diet4         -1.454     3.0177 -0.4818
## Time:Diet2     2.271     1.0367  2.1902
## Time:Diet3     5.084     1.0367  4.9043
## Time:Diet4     3.217     1.0377  3.1004
```

The contrast matrix for `Diet` becomes:

```
L <- linest_matrix(chick, effect="Diet")

## List of 8
##  $ effect        : chr "Diet"
##  $ at            : NULL
##  $ fact.lev      :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ vartype       :List of 2
##   ..$ numeric: chr "Time"
##   ..$ factor : chr "Diet"
##  $ cov.ave       :List of 1
##   ..$ Time: num 10.7
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr "Time"
## The general case; there are 'effect' factors or 'at' factors...

t(L)

##              [,1]  [,2]  [,3]  [,4]
```

```
## (Intercept)  1.00  1.00  1.00  1.00
## Time        10.72 10.72 10.72 10.72
## Diet2        0.00  1.00  0.00  0.00
## Diet3        0.00  0.00  1.00  0.00
## Diet4        0.00  0.00  0.00  1.00
## Time:Diet2   0.00 10.72  0.00  0.00
## Time:Diet3   0.00  0.00 10.72  0.00
## Time:Diet4   0.00  0.00  0.00 10.72
```

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

```
K1 <- linest_matrix(chick, effect="Diet", at=list(Time=1))

## List of 8
##  $ effect        : chr "Diet"
##  $ at            :List of 1
##   ..$ Time: num 1
##  $ fact.lev      :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ vartype       :List of 2
##   ..$ numeric: chr "Time"
##   ..$ factor : chr "Diet"
##  $ cov.ave       :List of 1
##   ..$ Time: num 1
##   ..- attr(*, "at.num")=List of 1
##   .. ..$ Time: num 1
##  $ at.factor.name: chr(0)
##  $ cov.ave.name  : chr "Time"
## The general case; there are 'effect' factors or 'at' factors...

t(K1)

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

The `LSmeans` for the intercepts is the predictions at `Time=0`. The `LSmeans` for the slopes becomes

```
K0 <- linest_matrix(chick, effect="Diet", at=list(Time=0))

## List of 8
```

```
##  $ effect        : chr "Diet"
##  $ at            :List of 1
##   ..$ Time: num 0
##  $ fact.lev      :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ vartype       :List of 2
##   ..$ numeric: chr "Time"
##   ..$ factor : chr "Diet"
##  $ cov.ave       :List of 1
##   ..$ Time: num 0
##   ..- attr(*, "at.num")=List of 1
##   .. ..$ Time: num 0
##  $ at.factor.name: chr(0)
##  $ cov.ave.name  : chr "Time"
## The general case; there are 'effect' factors or 'at' factors...

t(K1 - K0)

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

linest(chick, L=K1-K0)

## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]    6.339  0.610 49.855 10.383       0
## [2,]    8.609  0.838 48.282 10.273       0
## [3,]   11.423  0.838 48.282 13.631       0
## [4,]    9.556  0.839 48.565 11.386       0
```

We can cook up our own function for comparing trends:

```
LSmeans_trend <- function(object, effect, trend){

    K<-linest_matrix(object, effect=effect, at=as.list(setNames(1, trend))) -
        linest_matrix(object, effect=effect, at=as.list(setNames(0, trend)))
    linest(object, L=K)
}
LSmeans_trend(chick, effect="Diet", trend="Time")

## List of 8
##  $ effect        : chr "Diet"
##  $ at            :List of 1
```

```
##   ..$ Time: num 1
##  $ fact.lev      :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ vartype       :List of 2
##   ..$ numeric: chr "Time"
##   ..$ factor : chr "Diet"
##  $ cov.ave       :List of 1
##   ..$ Time: num 1
##   ..- attr(*, "at.num")=List of 1
##   .. ..$ Time: num 1
##  $ at.factor.name: chr(0)
##  $ cov.ave.name  : chr "Time"
## The general case; there are 'effect' factors or 'at' factors...
## List of 8
##  $ effect        : chr "Diet"
##  $ at            :List of 1
##   ..$ Time: num 0
##  $ fact.lev      :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 1
##   ..$ Diet: chr [1:4] "1" "2" "3" "4"
##  $ vartype       :List of 2
##   ..$ numeric: chr "Time"
##   ..$ factor : chr "Diet"
##  $ cov.ave       :List of 1
##   ..$ Time: num 0
##   ..- attr(*, "at.num")=List of 1
##   .. ..$ Time: num 0
##  $ at.factor.name: chr(0)
##  $ cov.ave.name  : chr "Time"
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]    6.339  0.610 49.855 10.383       0
## [2,]    8.609  0.838 48.282 10.273       0
## [3,]   11.423  0.838 48.282 13.631       0
## [4,]    9.556  0.839 48.565 11.386       0
```

6 Using (transformed) covariates

Consider the following subset of the `CO2` dataset:

```
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)

##       Type    Que        Mis
```

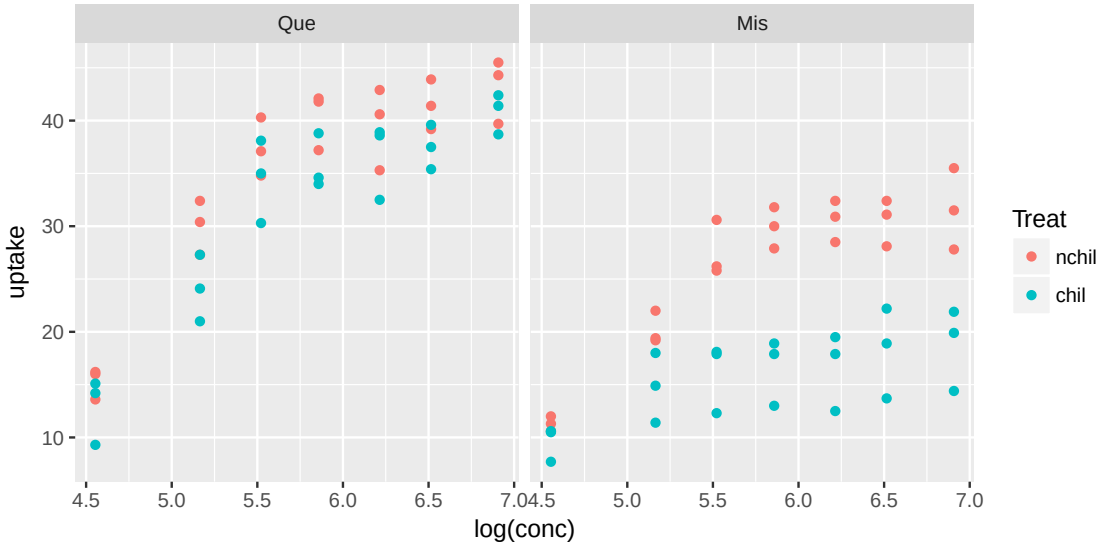


Figure 3: CO2 data

```
##       Treat nchil chil nchil chil
## Plant
## 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
## Mn1             0    0     7    0
## Mc2             0    0     0    7
## Mc3             0    0     0    7
## Mc1             0    0     0    7
```

```
qplot(x=log(conc), y=uptake, data=CO2, color=Treat, facets=~Type)
```

Below, the covariate `conc` is fixed at the average value:

```
co2.lm1 <- lm(uptake ~ conc + Type + Treat, data=CO2)
LSmeans(co2.lm1, effect="Treat")
```

```
## List of 8
##  $ effect        : chr "Treat"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ Type : chr [1:2] "Que" "Mis"
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ new.fact.lev  :List of 1
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ vartype       :List of 2
```

```
##   ..$ numeric: chr "conc"
##   ..$ factor : chr [1:2] "Type" "Treat"
##  $ cov.ave       :List of 1
##   ..$ conc: num 435
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr "conc"
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   30.643  0.956 80.000 32.066       0
## [2,]   23.783  0.956 80.000 24.888       0
```

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

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

In this case one can do

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

## List of 8
##  $ effect        : chr "Treat"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ Type : chr [1:2] "Que" "Mis"
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ new.fact.lev  :List of 1
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ vartype       :List of 2
##   ..$ numeric: chr "log.conc"
##   ..$ factor : chr [1:2] "Type" "Treat"
##  $ cov.ave       :List of 1
##   ..$ log.conc: num 5.82
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr "log.conc"
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   30.643  0.761 80.000 40.261       0
## [2,]   23.783  0.761 80.000 31.248       0
```

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

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

## List of 8
```

```
##  $ effect        : chr "Treat"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ Type : chr [1:2] "Que" "Mis"
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ new.fact.lev  :List of 1
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ vartype       :List of 2
##   ..$ numeric: chr [1:2] "conc" "I(conc^2)"
##   ..$ factor : chr [1:2] "Type" "Treat"
##  $ cov.ave       :List of 2
##   ..$ conc     : num 435
##   ..$ I(conc^2): num 275754
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr [1:2] "conc" "I(conc^2)"
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   34.543  0.982 79.000 35.191       0
## [2,]   27.683  0.982 79.000 28.202       0
```

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")

## List of 8
##  $ effect        : chr "Treat"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ Type : chr [1:2] "Que" "Mis"
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ new.fact.lev  :List of 1
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ vartype       :List of 2
##   ..$ numeric: chr [1:2] "conc" "conc2"
##   ..$ factor : chr [1:2] "Type" "Treat"
##  $ cov.ave       :List of 2
##   ..$ conc : num 435
##   ..$ conc2: num 275754
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr [1:2] "conc" "conc2"
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   30.643  0.776 79.000 39.465       0
## [2,]   23.783  0.776 79.000 30.630       0
```

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))
```

```
## List of 8
##  $ effect        : chr "Treat"
##  $ at            :List of 2
##   ..$ conc : num 10
##   ..$ conc2: num 100
##  $ fact.lev      :List of 2
##   ..$ Type : chr [1:2] "Que" "Mis"
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ new.fact.lev  :List of 1
##   ..$ Treat: chr [1:2] "nchil" "chil"
##  $ vartype       :List of 2
##   ..$ numeric: chr [1:2] "conc" "conc2"
##   ..$ factor : chr [1:2] "Type" "Treat"
##  $ cov.ave       :List of 2
##   ..$ conc : num 10
##   ..$ conc2: num 100
##   ..- attr(*, "at.num")=List of 2
##   .. ..$ conc : num 10
##   .. ..$ conc2: num 100
##  $ at.factor.name: chr(0)
##  $ cov.ave.name  : chr [1:2] "conc" "conc2"
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate    se    df t.stat p.value
## [1,]    14.74  1.70 79.00   8.66       0
## [2,]     7.88  1.70 79.00   4.63       0
```

7 Alternative models

7.1 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:

```
tooth.gam <- glm(len ~ dose + supp, family=Gamma, data=ToothGrowth)
LSmeans(tooth.gam, effect="dose", type="link")
```

```
## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
```

```
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate       se       df   t.stat p.value
## [1,]  0.09453  0.00579 56.00000 16.33340       0
## [2,]  0.05111  0.00312 56.00000 16.39673       0
## [3,]  0.03889  0.00238 56.00000 16.36460       0

LSmeans(tooth.gam, effect="dose", type="response")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   10.578  0.648 56.000 16.333       0
## [2,]   19.565  1.193 56.000 16.397       0
## [3,]   25.711  1.571 56.000 16.365       0
```

7.2 Linear mixed effects model

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

```
library(lme4)
tooth.mm <- lmer( len ~ dose  + (1|supp), data=ToothGrowth)
LSmeans(tooth1, effect="dose")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 2
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##   ..$ supp: chr [1:2] "OJ" "VC"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
```

```
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]   10.605  0.856 56.000 12.391       0
## [2,]   19.735  0.856 56.000 23.058       0
## [3,]   26.100  0.856 56.000 30.495       0
```

```
LSmeans(tooth.mm, effect="dose")
```

```
## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:2] "dose" "supp"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate    se    df t.stat p.value
## [1,]    10.61  1.98  1.31   5.36    0.08
## [2,]    19.74  1.98  1.31   9.98    0.03
## [3,]    26.10  1.98  1.31  13.20    0.02
```

Notice here that the estimates themselves identical to those of a linear model (that is not generally the case, but it is so here because data is balanced). In general the estimates are will be very similar but the standard errors are much larger under the mixed model. This comes from that there that `supp` is treated as a random effect.

```
VarCorr(tooth.mm)
```

```
##  Groups   Name        Std.Dev.
##  supp     (Intercept) 2.52
##  Residual             3.83
```

Notice that the degrees of freedom by default are adjusted using a Kenward–Roger approximation (provided that **pbkrtest** is installed). Unadjusted degrees of freedom are obtained by setting `adjust.df=FALSE`.

7.3 Generalized estimating equations

Lastly, for gee-type "models" we get

```
library(geepack)
tooth.gee <- geeglm(len ~ dose, id=supp, family=Gamma, data=ToothGrowth)
LSmeans(tooth.gee, effect="dose")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr "dose"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate       se   z.stat p.value
## [1,] 9.43e-02 1.65e-02 5.71e+00       0
## [2,] 5.07e-02 5.38e-03 9.41e+00       0
## [3,] 3.83e-02 4.15e-05 9.23e+02       0

LSmeans(tooth.gee, effect="dose", type="response")

## List of 8
##  $ effect        : chr "dose"
##  $ at            : NULL
##  $ fact.lev      :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ new.fact.lev  :List of 1
##   ..$ dose: chr [1:3] "0.5" "1" "2"
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr "dose"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...
## Coefficients:
##      estimate       se   z.stat p.value
## [1,]  10.6050   1.8562   5.7134       0
## [2,]  19.7350   2.0966   9.4130       0
## [3,]  26.1000   0.0283 922.7743       0
```

8 Miscellaneous

8.1 Example: Non–estimable contrasts

Consider this simulated dataset:

```
head(dat.nst, 4)

##   AA BB CC       y
## 1  1  1  1 -0.1371
## 2  2  1  1  0.2072
## 3  1  2  2 -0.5923
## 4  2  2  2 -1.7498

ftable(xtabs( ~ AA + BB + CC, data=dat.nst), row.vars="AA")

##    BB 1       2       3
##    CC 1 2 3 4 1 2 3 4 1 2 3 4
## AA
## 1     3 0 0 0 0 1 1 1 0 1 1 1
## 2     3 0 0 0 0 1 1 1 0 1 1 1
```

Data is highly "unbalanced": Whenever `BB=1` then `CC` is always `1`; whenever `BB` is not `1` then `CC` is never `1`. We have

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

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1102     0.6237  0.1767  0.86325
## AA2           0.4975     0.3945  1.2613  0.23584
## BB1:CC1      -0.2658     0.6832 -0.3890  0.70540
## BB2:CC2      -1.5300     0.8368 -1.8285  0.09742
## BB3:CC2      -0.6006     0.8368 -0.7177  0.48935
## BB2:CC3      -1.1407     0.8368 -1.3632  0.20271
## BB3:CC3      -0.5802     0.8368 -0.6933  0.50388
## BB2:CC4      -0.8098     0.8368 -0.9677  0.35602
```

In this case some of the `LSmeans` values are not estimable; for example:

```
lsm.BC <- LSmeans(mod.nst, effect=c("BB", "CC"))

## List of 8
##  $ effect        : chr [1:2] "BB" "CC"
##  $ at            : NULL
##  $ fact.lev      :List of 3
##   ..$ AA: chr [1:2] "1" "2"
##   ..$ BB: chr [1:3] "1" "2" "3"
##   ..$ CC: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 2
##   ..$ BB: chr [1:3] "1" "2" "3"
##   ..$ CC: chr [1:4] "1" "2" "3" "4"
```

```
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:3] "AA" "BB" "CC"
##  $ cov.ave       : Named list()
##  $ at.factor.name: NULL
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...

lsm.BC

## Coefficients:
##       estimate      se      df  t.stat p.value
##  [1,]   0.0932  0.3416 10.0000  0.2728    0.79
##  [2,]       NA      NA      NA      NA      NA
##  [3,]       NA      NA      NA      NA      NA
##  [4,]       NA      NA      NA      NA      NA
##  [5,]  -1.1710  0.5917 10.0000 -1.9791    0.08
##  [6,]  -0.2416  0.5917 10.0000 -0.4083    0.69
##  [7,]       NA      NA      NA      NA      NA
##  [8,]  -0.7817  0.5917 10.0000 -1.3212    0.22
##  [9,]  -0.2212  0.5917 10.0000 -0.3738    0.72
## [10,]       NA      NA      NA      NA      NA
## [11,]  -0.4508  0.5917 10.0000 -0.7618    0.46
## [12,]   0.3590  0.5917 10.0000  0.6067    0.56

lsm.BC2 <- LSmeans(mod.nst, effect="BB", at=list(CC=2))

## List of 8
##  $ effect        : chr "BB"
##  $ at            :List of 1
##   ..$ CC: num 2
##  $ fact.lev      :List of 3
##   ..$ AA: chr [1:2] "1" "2"
##   ..$ BB: chr [1:3] "1" "2" "3"
##   ..$ CC: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 2
##   ..$ BB: chr [1:3] "1" "2" "3"
##   ..$ CC: num 2
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:3] "AA" "BB" "CC"
##  $ cov.ave       : Named list()
##   ..- attr(*, "at.num")= Named list()
##  $ at.factor.name: chr "CC"
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...

lsm.BC2

## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]       NA     NA     NA     NA      NA
## [2,]   -1.171  0.592 10.000 -1.979    0.08
## [3,]   -0.242  0.592 10.000 -0.408    0.69
```

We describe the situation in Section ?? where we focus on `lsm.BC2`.

8.2 Handling non–estimability

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

```
X <- model.matrix( mod.nst )
Matrix::rankMatrix(X)

## [1] 8
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 3.997e-15

dim(X)

## [1] 18 14

as(X, "Matrix")

## 18 x 14 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 14 column names '(Intercept)', 'AA2', 'BB1:CC1' ...  ]]

##
## 1  1 . 1 . . . . . . . . . . .
## 2  1 1 1 . . . . . . . . . . .
## 3  1 . . . . . 1 . . . . . . .
## 4  1 1 . . . . 1 . . . . . . .
## 5  1 . . . . . . 1 . . . . . .
## 6  1 1 . . . . . 1 . . . . . .
## 7  1 . 1 . . . . . . . . . . .
## 8  1 1 1 . . . . . . . . . . .
## 9  1 . . . . . . . . 1 . . . .
## 10 1 1 . . . . . . . 1 . . . .
## 11 1 . . . . . . . . . 1 . . .
## 12 1 1 . . . . . . . . 1 . . .
## 13 1 . 1 . . . . . . . . . . .
## 14 1 1 1 . . . . . . . . . . .
## 15 1 . . . . . . . . . . . 1 .
## 16 1 1 . . . . . . . . . . 1 .
## 17 1 . . . . . . . . . . . . 1
## 18 1 1 . . . . . . . . . . . 1
```

We consider a model, i.e. an n dimensional random vector $y = (y_i)$ for which $\mathbb{E}(y) = \mu = X\beta$ and $\mathbb{C}\text{ov}(y) = V$ where X does not have full column rank We are interested in linear functions of β,

say

$$c = l^\top \beta = \sum_j l_j \beta_j.$$

```
L <- linest_matrix(mod.nst, effect="BB", at=list(CC=2))

## List of 8
##  $ effect        : chr "BB"
##  $ at            :List of 1
##   ..$ CC: num 2
##  $ fact.lev      :List of 3
##   ..$ AA: chr [1:2] "1" "2"
##   ..$ BB: chr [1:3] "1" "2" "3"
##   ..$ CC: chr [1:4] "1" "2" "3" "4"
##  $ new.fact.lev  :List of 2
##   ..$ BB: chr [1:3] "1" "2" "3"
##   ..$ CC: num 2
##  $ vartype       :List of 2
##   ..$ numeric: chr(0)
##   ..$ factor : chr [1:3] "AA" "BB" "CC"
##  $ cov.ave       : Named list()
##   ..- attr(*, "at.num")= Named list()
##  $ at.factor.name: chr "CC"
##  $ cov.ave.name  : chr(0)
## The general case; there are 'effect' factors or 'at' factors...

t(L)

##             [,1] [,2] [,3]
## (Intercept)  1.0  1.0  1.0
## AA2          0.5  0.5  0.5
## BB1:CC1      0.0  0.0  0.0
## BB2:CC1      0.0  0.0  0.0
## BB3:CC1      0.0  0.0  0.0
## BB1:CC2      1.0  0.0  0.0
## BB2:CC2      0.0  1.0  0.0
## BB3:CC2      0.0  0.0  1.0
## BB1:CC3      0.0  0.0  0.0
## BB2:CC3      0.0  0.0  0.0
## BB3:CC3      0.0  0.0  0.0
## BB1:CC4      0.0  0.0  0.0
## BB2:CC4      0.0  0.0  0.0
## BB3:CC4      0.0  0.0  0.0

linest(mod.nst, L=L)

## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]       NA     NA     NA     NA      NA
## [2,]   -1.171  0.592 10.000 -1.979    0.08
## [3,]   -0.242  0.592 10.000 -0.408    0.69
```

A least squares estimate of β is

$$\hat{\beta} = GX^\top y$$

where G is a generalized inverse of $X^\top X$. Since the generalized inverse is not unique then neither is the estimate $\hat{\beta}$. Hence $\hat{c} = l^\top \hat{\beta}$ is in general not unique.

One least squares estimate of β and one corresponding linear estimate $L\hat{\beta}$ is:

```
XtXinv <- MASS::ginv(t(X)%*%X)
bhat <- as.numeric(XtXinv %*% t(X) %*% dat.nst$y)
zapsmall(bhat)

##  [1] -0.5194  0.4975  0.3639  0.0000  0.0000  0.0000 -0.9004  0.0291  0.0000 -0.5111
## [11]  0.0495  0.0000 -0.1801  0.6297

L %*% bhat

##          [,1]
## [1,] -0.2707
## [2,] -1.1710
## [3,] -0.2416
```

For some values of l (i.e. for some rows of L) the estimate $\hat{c} = l^\top \beta$ 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 μ is a vector in the column space $C(X)$ of X. We can only learn about β through $X\beta$ so the only thing we can say something about is linear combinations $\rho^\top X\beta$. Hence we can only say something about $l^\top \beta$ if there exists ρ such that

$$l^\top \beta = \rho^\top X\beta,$$

i.e., if $l = X^\top \rho$ for some ρ, which is if l is in the column space $C(X^\top)$ of $X^\top$. This is the same as saying that l must be 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.

$$X = UDV^\top$$

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

```
S <- svd(X)
B <- S$v[, S$d < 1e-10, drop=FALSE ];
head(B) ## Basis for N(X)

##           [,1]       [,2]       [,3]       [,4]       [,5] [,6]
## [1,]  0.339176 -5.635e-04  9.968e-02 -4.350e-03 -2.274e-03    0
## [2,]  0.000000  1.193e-17 -1.110e-16  1.735e-18  4.337e-19    0
## [3,] -0.339176  5.635e-04 -9.968e-02  4.350e-03  2.274e-03    0
## [4,] -0.272743 -2.494e-01  9.244e-01 -3.167e-03 -9.422e-02    0
## [5,] -0.072691  9.176e-01  2.509e-01 -1.669e-01  2.487e-01    0
## [6,] -0.001889 -9.509e-02  5.169e-02  6.615e-01  7.421e-01    0
```

From

```
rowSums(L %*% B)

## [1]  1.790e+00  1.632e-15 -4.113e-15
```

we conclude that the first row of L is not perpendicular to all vectors in thenull space $N(X)$ whereas the two last rows of L are. Hence these two linear estimates are estimable; their value does not depend on the choice of generalized inverse:

```
lsm.BC2

## Coefficients:
##      estimate     se     df t.stat p.value
## [1,]       NA     NA     NA     NA      NA
## [2,]   -1.171  0.592 10.000 -1.979    0.08
## [3,]   -0.242  0.592 10.000 -0.408    0.69
```

8.3 Pairwise comparisons

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

```
library("multcomp")
g1 <- glht(tooth1, mcp(dose="Tukey"))
summary( g1 )

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = len ~ dose + supp, data = ToothGrowth)
##
## Linear Hypotheses:
##               Estimate Std. Error t value Pr(>|t|)
## 1 - 0.5 == 0     9.13       1.21    7.54   <1e-05 ***
## 2 - 0.5 == 0    15.49       1.21   12.80   <1e-05 ***
## 2 - 1 == 0       6.37       1.21    5.26   <1e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

The L matrix is

```
L <- g1$linfct
L

##         (Intercept) dose1 dose2 suppVC
## 1 - 0.5           0     1     0      0
```

```
## 2 - 0.5           0     0     1      0
## 2 - 1             0    -1     1      0
## attr(,"type")
## [1] "Tukey"
```

and this matrix can also be supplied to `glht`

```
glht(tooth1, linfct=L)
```