Contrasts

Ben Bolker

19 April 2021

Contrasts are the way that R (and other statistical software) sets up tests of differences between different groups in an experimental or observational study. Equivalently, they are the way to define the parameters of a linear model that involves categorical predictors.

There are lots of ways to use built-in R functions to define different contrasts, but sometimes we want to define our own custom contrasts. In any case, understanding how to set up your own contrast matrix helps you understand how the built-in functions work.

R's default set of definitions is called *treatment contrasts*. For example, in a linear model with a single categorical predictor (equivalent to a one-way ANOVA), the parameters β_i would be defined as:

$$\begin{aligned}
\beta_0 = &\quad \mu_1 &&= \text{intercept} = \text{predicted value of group 1} \\
\beta_1 = &\quad \mu_2 - \mu_1 &&= \text{pred. value of group 2} - \text{pred. value of group 1} \\
\beta_2 = &\quad \mu_3 - \mu_1 &&= \text{pred. value of group 3} - \text{pred. value of group 1}
\end{aligned} \tag{1}$$

and so on.

This is equivalent to

$$\begin{aligned}
\text{predicted value of group 1} = &\quad \mu_1 = &&\beta_0 \\
\text{predicted value of group 2} = &\quad \mu_2 = &&\beta_0 + \beta_1 \\
\text{predicted value of group 3} = &\quad \mu_3 = &&\beta_0 + \beta_2 \\
&\ldots
\end{aligned} \tag{2}$$

In the treatment contrast case it's reasonably straightforward to see how to get from the first set of equations (defining the parameters, or β values, in terms of differences between predicted values (μ), or group means) to the second set (defining the predicted values in terms of the parameters), but it's not always so straightforward. (If you don't see it immediately, try adding the first two equations in eq. 1 to get the second equation in eq. 2.)

The first set (parameters in terms of group differences) is the more natural way to think about which comparisons we want to test statistically; unfortunately, the second set (group differences in terms of parameters) is the way that R wants us to tell it which comparisons to make.

However, we can write down the second set of equations as the product of a *contrast matrix* $\boldsymbol{C}$ and the parameter vector $\vec{\beta}\ =$

$(\beta_0, \beta_1, \beta_2, \ldots)$:

$$\boldsymbol{C}\vec{\beta} = \begin{pmatrix} 1 & 0 & 0 & \ldots \\ 1 & 1 & 0 & \ldots \\ 1 & 0 & 1 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \ldots \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \ldots \end{pmatrix} \quad . \tag{3}$$

Now we can do some linear algebra (or ask R to do it) in order to go from the human-friendly to the R-friendly specification of contrasts. Specifically, if we denote the vector of predicted values as $\vec{\mu}$, then R wants us to specify the contrast matrix $\boldsymbol{C}$ such that

$$\vec{\mu} = \boldsymbol{C}\vec{\beta}. \tag{4}$$

We need to solve this equation for $\vec{\beta}$, so that we can specify $\vec{\beta}$ in terms of linear relationships among the $\vec{\mu}$ values. Mathematically, we solve equation 4 by multiplying both sides by the inverse of $\boldsymbol{C}$:

$$\begin{aligned} \boldsymbol{C}^{-1}\vec{\mu} &= \boldsymbol{C}^{-1}\boldsymbol{C}\vec{\beta} \\ &= \vec{\beta}. \end{aligned} \tag{5}$$

If we take the matrix $\boldsymbol{C}$ from eq. 3 above and invert it (`solve()` does matrix inversion in R):

```
Cmat = matrix(c(1,0,0,
                1,1,0,
                1,0,1), nrow=3, byrow=TRUE)
solve(Cmat)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]   -1    1    0
## [3,]   -1    0    1
```

Comparing these values to the relations in eq. 1, we can see that we have successfully recovered $\boldsymbol{C}^{-1}$ such that

$$\boldsymbol{C}^{-1}\vec{\mu} = \begin{pmatrix} 1 & 0 & 0 & \ldots \\ -1 & 1 & 0 & \ldots \\ -1 & 0 & 1 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \ldots \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \ldots \end{pmatrix} \tag{6}$$

A more complex example

A common situation is that we want to test whether the combination of two treatments has more or less effect than either treatment alone. This is different from the typical setup for a two-way interaction, discussed below, where the null hypothesis is that the combination of the two treatments has an additive effect. This occurred in

McKeon et al. (2012), where the authors compared the effects of crabs and shrimp and their combination in protecting coral against starfish predation.

There are four treatments: we know the amount of predation in the control treatment (μ_0), crabs-only treatment (μ_C), shrimp-only treatment (μ_S), and crabs-plus-shrimp treatment (μ_{CS}). Suppose we want to parameterize this model in terms of

- the overall mean predation level: $(\mu_0 + \mu_C + \mu_S + \mu_{CS})/4$
- the average effect of symbionts, i.e. the difference between the control (μ_0) and the average of the symbiont treatments $(\mu_C + \mu_S + \mu_{CS})/3$
- the difference between crabs and shrimp, $\mu_C - \mu_S$
- the difference between the combined-symbiont treatment, μ_{CS}, and the average of the single-symbiont treatments, $(\mu_C + \mu_S)/2$

Reminder about matrix-by-vector multiplication: the value of the i^{th} element is the sum of the products of row i of $\boldsymbol{C}$ with the vector. So, for example, the first element (which we want to be the mean of all treatment means) is $\mu_0 c_{11} + \mu_C c_{12} + \mu_S c_{13} + \mu_{CS} c_{14}$. If we want the first parameter (β_0) to equal the mean of the μ values, then we need to set $c_{11} = c_{12} = c_{13} = c_{14} = 1/4$.

The signs are set up to allow for the fact that we want to quantify the *decrease* in predation under symbiont prediction.

Define cc_inv as follows:

```
##            ttt
## contrast    none C    S    CS
##   intercept  1/4  1/4  1/4  1/4
##   avg_symb     1 -1/3 -1/3 -1/3
##   C.vs.S       0    1   -1    0
##   twosymb      0  1/2  1/2   -1
```

This is what we get for $\boldsymbol{C}$ by inverting $\boldsymbol{C}^{-1}$ (solve(cc_inv)) (with a little bit of cosmetic stuff):

```
##      intercept avg_symb C.vs.S twosymb
## none    1       3/4        0      0
## C       1      -1/4      1/2    1/3
## S       1      -1/4     -1/2    1/3
## CS      1      -1/4        0   -2/3
```

Let's make an example: there will be only one value per treatment because we're lazy (and so the R^2 will be 1.0), but we'll see if this actually does what we expect. When we use the contrast matrix, we drop the first column (which is all ones) since R will add an intercept automatically.

```
dd <- data.frame(
  ttt=c("none","C","S","CS"),
  pred=c(5,2,3,1))
dd$ttt <- factor(dd$ttt, levels=dd$ttt) ## make sure factor is in order
## drop first column (~ttt-1 doesn't work the way we want it to)
lm1 <- lm(pred~ttt,contrasts=list(ttt=cc[,-1]),data=dd)
fractions(coef(lm1))
```

```
## (Intercept) tttavg_symb   tttC.vs.S  ttttwosymb
##        11/4           3          -1         3/2
```

This does work as expected; the values are [mean] 11/4 (=(5+2+3+1)/4), [symbiont effect] 3 (=5-(2+3+1)/3), [crab vs shrimp] -1 (=2-3), [extra symbiont effect] 3/2 = (2+3)/2-1.

Crawley (2002) gives another custom-contrast example, but he pretty much just shows C without much discussion of how one would derive it.

Categorical predictors: contrasts

Independent contrasts.

The *contrast matrix* determines what a given row of the design matrix (for level i of a categorical variable) looks like.

If we have a vector of predicted values $\bar{y}$, the contrast matrix is essentially defined as

$$\bar{y} = C\beta$$

Set contrasts in general via `options()` or per-factor via `contrasts()`, or within the model statement, e.g.

```
d <- data.frame(f=factor(rep(c("a","b"), each=3)), y=c(1,1,1,3,3,3))
coef(lm(y~f,data=d))
```

```
## (Intercept)          fb
##           1           2
```

```
coef(lm(y~f,data=d,contrasts=list(f="contr.sum")))
```

```
## (Intercept)          f1
##           2          -1
```

Or:

```
contrasts(d$f) <- "contr.sum"
## or (slightly dangerous because it sets the options
## *globally*, sometimes leading to confusion)
options(contrasts=c("contr.sum","contr.poly"))
```

Reordering factors: `levels`, `reorder`, `relevel`

```
levels(relevel(d$f,"b"))

## [1] "b" "a"

levels(with(d,reorder(f,y,mean)))

## [1] "a" "b"
```

In general requesting a contrast for an n-level factor gets us only an $n \times (n-1)$ matrix: the first column is an implicit intercept (all-1) column.

Treatment contrasts (default: "dummy", "corner-point")

First level of factor (often alphabetical!) is the default intercept/baseline for `contr.treatment` (default): `contr.SAS` uses the *last* level of the factor (which is SAS's default). You can specify a baseline via `contr.treatment(n,base=b)`, but it may make more sense to relevel the factor to put the baseline (typically control) treatment first. The full contrast matrix is not orthogonal (i.e. $\boldsymbol{C}^T\boldsymbol{C}$ is not diagonal: we want $C_i^T C_j = 0$ whenever $i \neq j$).

TODO explain more about what orthogonality means and why we would care

```
(cc <- contr.treatment(4))

##   2 3 4
## 1 0 0 0
## 2 1 0 0
## 3 0 1 0
## 4 0 0 1
```

The comparisons between treatments and the baseline are all orthogonal to each other,

```
is_orthog <- function(x) {
    xsq <- t(x) %*% x
    return(all(xsq-diag(diag(xsq))==0)  && ## off-diagonals are zero
        all(diag(xsq)!=0))
}

is_orthog(cc)

## [1] TRUE

cc <- cbind(1,cc) ## add intercept column
is_orthog(cc)

## [1] FALSE
```

If we want to know the *meaning* of $\boldsymbol{\beta}$, it's easiest to invert:

$$\beta = C^{-1}\bar{y}$$

```
solve(cc)
```

```
##    1 2 3 4
##    1 0 0 0
## 2 -1 1 0 0
## 3 -1 0 1 0
## 4 -1 0 0 1
```

Example (from Gotelli and Ellison (2004)):

```
ants <- data.frame(
    place=rep(c("field","forest"),c(6,4)),
    colonies=c(12, 9, 12, 10,
               9, 6, 4, 6, 7, 10))
```

```
mean(ants$colonies[ants$place=="field"])
```

```
## [1] 9.666667
```

```
mean(ants$colonies[ants$place=="forest"])
```

```
## [1] 6.75
```

```
pr <- function(m) printCoefmat(coef(summary(m)),digits=3,signif.stars=FALSE)
pr(lm1 <- lm(colonies~place,data=ants))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    9.667      0.958   10.09    8e-06
## placeforest   -2.917      1.515   -1.92     0.09
```

The `(Intercept)` row refers to β_1, which is the mean density in the "field" sites ("field" comes before "forest"). The `placeforest` row tells us we are looking at the effect of the `place` variable on the `forest` level, i.e. the difference between the "forest" and "field" sites. (The only ways we could know that "field" is the baseline site are (1) to remember, or look at `levels(ants$place)` or (2) to notice which level is *missing* from the list of parameter estimates.)

Helmert

In this case the full matrix (intercept and all comparisons) is orthogonal (which is why Helmert were the default contrasts in R's ancestor, S-PLUS), but the comparisons are less intuitive.

```
(cc <- cbind(1,contr.helmert(4)))
```

```
##   [,1] [,2] [,3] [,4]
## 1    1   -1   -1   -1
## 2    1    1   -1   -1
## 3    1    0    2   -1
## 4    1    0    0    3
```

```
is_orthog(cc)
```

```
MASS::fractions(solve(cc))
```

```
##      1     2     3     4
## [1,]   1/4   1/4   1/4   1/4
## [2,]  -1/2   1/2     0     0
## [3,]  -1/6  -1/6   1/3     0
## [4,] -1/12 -1/12 -1/12   1/4
```

β_1=mean; β_2=contrast between levels 1 and 2; β_3=contrast between levels 1 & 2 and level 3; etc..

```
cfun <- function(contr) {
    pr(update(lm1,contrasts=list(place=contr)))
}
cfun("contr.helmert")
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    8.208      0.758   10.83  4.7e-06
## place1        -1.458      0.758   -1.92     0.09
```

Sum-to-zero

What if I want to compare the values with the mean (Schielzeth 2010)?

Sum-to-zero contrasts *not* orthogonal (??)

```
cc <- contr.sum(4)
is_orthog(cc)
```

```
## [1] FALSE
```

```
(cc <- cbind(1,contr.sum(4)))
```

```
##   [,1] [,2] [,3] [,4]
## 1    1    1    0    0
## 2    1    0    1    0
## 3    1    0    0    1
## 4    1   -1   -1   -1
```

```
is_orthog(cc)
```

```
## [1] FALSE
```

```
MASS::fractions(solve(cc))
```

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

β_1=mean; β_2=level 1 vs average of levels 2-4; β_3=level 2 vs. average of levels 1,3, 4; β_4=level 3 vs. average of levels 1,2, 4

Note that we don't have a contrast directly involving level 4.

```
cfun("contr.sum")
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    8.208      0.758   10.83  4.7e-06
## place1         1.458      0.758    1.92     0.09
```

Same as Helmert contrasts in this example, except for the sign of `place1`.

No-intercept

When we specify a formula with `-1` or `+0` (with default treatment contrasts) we get an identity matrix for the contrasts: each level has its own parameter.

```
pr(update(lm1,.~.-1))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## placefield     9.667      0.958   10.09    8e-06
## placeforest    6.750      1.174    5.75  0.00043
```

Sometimes clearer (and we get confidence intervals etc. on the predictions for each level), but the hypotheses tested are rarely interesting (is the mean of each level equal to zero?)

More generally, if you want to compute the group means, you can

- Use the `predict` function:

```
predict(lm1,newdata=data.frame(place=c("field","forest")),interval="confidence")
```

- Use `effects::allEffects`:

```
summary(allEffects(lm1))
```

- Use `emmeans::emmeans`:

```
emmeans(lm1,spec=~place)
```

Forward difference contrasts:

```
(cc <- cbind(mean=1,MASS::contr.sdif(4)))
```

```
##   mean   2-1  3-2   4-3
## 1    1 -0.75 -0.5 -0.25
## 2    1  0.25 -0.5 -0.25
## 3    1  0.25  0.5 -0.25
## 4    1  0.25  0.5  0.75
```

```
MASS::fractions(solve(cc))
```

```
##      1   2   3   4
## mean 1/4 1/4 1/4 1/4
## 2-1   -1   1   0   0
## 3-2    0  -1   1   0
## 4-3    0   0  -1   1
```

```
## not orthogonal at all
```

Exercise How would you modify this contrast so the intercept is the value of the first level, rather than the mean?

Interactions

Interactions as *differences in differences*

Interpretation problems/marginality principle (Venables 1998,schielzeth_simple_2010)

```
head(d <- expand.grid(F=LETTERS[1:3],f=letters[1:3]))
```

```
##   F f
## 1 A a
## 2 B a
## 3 C a
## 4 A b
## 5 B b
## 6 C b
```

```
m0 <- model.matrix(~F*f,d)
ff <- solve(m0)
colnames(ff) <- apply(d,1,paste,collapse=".")
ff["FB",] ## contrast between (A,a) and (B,a)
```

```
## A.a B.a C.a A.b B.b C.b A.c B.c C.c
##  -1   1   0   0   0   0   0   0   0
```

```
ff["fb",] ## contrast between (A,a) and (A,b)
```

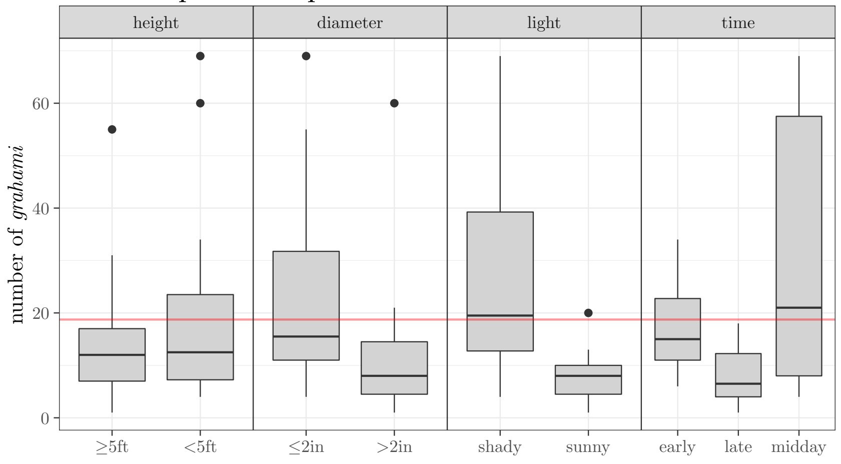
```
## A.a B.a C.a A.b B.b C.b A.c B.c C.c
##  -1   0   0   1   0   0   0   0   0
```

```
old.opts <- options(contrasts=c("contr.sum","contr.poly"))
m <- model.matrix(~F*f,d)
ff <- solve(m)*9
colnames(ff) <- apply(d,1,paste,collapse=".")
ff["F1",] ## contrast between (A,.) and (grand mean)

## A.a B.a C.a A.b B.b C.b A.c B.c C.c
##   2  -1  -1   2  -1  -1   2  -1  -1

ff["f1",] ## contrast between (a,.) and (grand mean)

## A.a B.a C.a A.b B.b C.b A.c B.c C.c
##   2   2   2  -1  -1  -1  -1  -1  -1

options(old.opts) ## reset
```

Exercise: How would you construct a version of 'contr.sum} where the first, not the last, level is aliased/dropped?

Things get slightly more interesting/complicated when we have more than two levels of a categorical variable. I'll look at some data on lizard perching behaviour, from the 'brglm} package (and before that from McCullagh and Nelder (1989), ultimately from Schoener (1970). I'm going to ignore the fact that these data might best be fitted with generalized linear models.

A quick look at the data: response is number of *Anolis grahami* lizards found on perches in particular conditions.



For a moment we're going to just look at the `time` variable. If we leave the factors as is (alphabetical) then β_1="early", β_2="late"-"early", β_3="midday"-"early". At the very least, it probably makes sense to change the order of the levels:

```
lizards$time <- factor(lizards$time,levels=c("early","midday","late"))
```

All this does (since we haven't changed the baseline factor) is swap the definitions of β_2 and β_3.

In a linear model, we could also use sum-to-zero contrasts:

```
pr(lm(grahami~time,data=lizards,contrasts=list(time=contr.sum)))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    19.30       3.53    5.47  2.4e-05
## time1          -1.67       4.93   -0.34     0.74
## time2          12.85       5.10    2.52     0.02
```

Now the `(Intercept)` parameter is the overall mean: `time1` and `time2` are the deviations of the first ("early") and second ("midday") groups from the overall mean. (The names are useless: the `car` package offers a slightly better alternative called `contr.Sum`). There are other ways to change the contrasts (i.e., use the `contrasts()` function to change the contrasts for a particular variable permanently, or use `options(contrasts=c("contr.sum","contr.poly"))` to change the contrasts for *all* variables), but the way shown above may be the most transparent.

There are other options for contrasts such as `MASS::contr.sdif()`, which gives the successive differences between levels.

```
pr(lm(grahami~time,data=lizards,contrasts=list(time=contr.sdif)))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    19.30       3.53    5.47  2.4e-05
## time2-1        14.52       8.74    1.66    0.112
## time3-2       -24.02       8.74   -2.75    0.012
```

You might have particular contrasts in mind (e.g. "control" vs. all other treatments, then "low" vs "high" within treatments), in which case it is probably worth learning how to set contrasts. (We will talk about testing *all pairwise differences later*, when we discuss multiple comparisons. This approach is probably not as useful as it is common.)

Multiple treatments and interactions

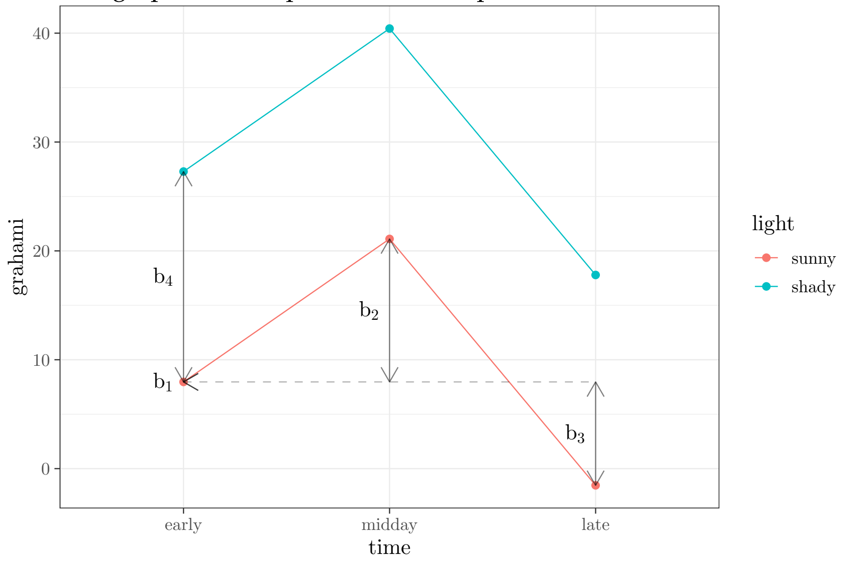
Additive model

Let's consider the `light` variable in addition to `time`.

```
pr(lmTL1 <- lm(grahami~time+light,data=lizards))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.96       5.63    1.41   0.1735
## timemidday     13.14       7.11    1.85   0.0801
## timelate       -9.50       6.85   -1.39   0.1817
## lightshady     19.33       5.73    3.37   0.0032
```

Here's a graphical interpretation of the parameters:

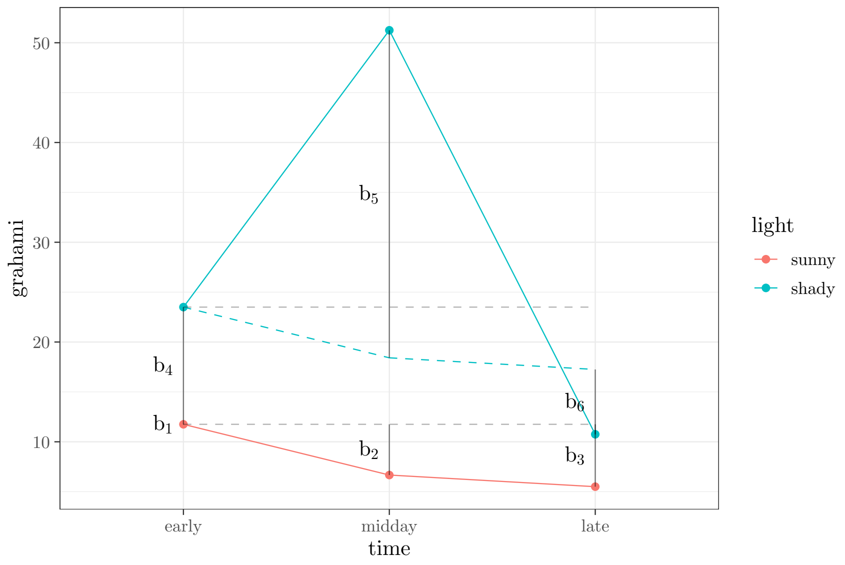


β_1 is the intercept ("early","sunny"); β_2 and β_3 are the differences from the baseline level ("early") of the *first* variable (`time`) in the *baseline* level of the other parameter(s) (`light`="shady"); β_4 is the difference from the baseline level ("sunny") of the *second* variable (`light`) in the *baseline* level of `time` ("early").

Now let's look at an interaction model:

```
pr(lmTL2 <- lm(grahami~time*light,data=lizards))
```

```
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)               11.75       5.38    2.19   0.0431
## timemidday                -5.08       8.21   -0.62   0.5441
## timelate                  -6.25       7.60   -0.82   0.4224
## lightshady                11.75       7.60    1.55   0.1406
## timemidday:lightshady     32.83      11.19    2.93   0.0093
## timelate:lightshady       -6.50      10.75   -0.60   0.5534
```



Parameters β_1 to β_4 have the same meanings as before. Now we also have β_5 and β_6, labelled “timemidday:lightshady” and “timelate:lightshady”, which describe the difference between the expected mean value of these treatment combinations based on the additive model (which are $\beta_1 + \beta_2 + \beta_4$ and $\beta_1 + \beta_3 + \beta_4$ respectively) and their actual values.

Now re-do this for sum-to-zero contrasts . . . the fits are easy:

```
pr(lmTL1S <- update(lmTL1,contrasts=list(time=contr.sum,light=contr.sum)))

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    18.84       2.87    6.57  2.7e-06
## time1          -1.21       4.01   -0.30   0.7654
## time2          11.93       4.15    2.87   0.0097
## light1         -9.66       2.87   -3.37   0.0032

pr(lmTL2S <- update(lmTL2,contrasts=list(time=contr.sum,light=contr.sum)))

##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    18.236      2.255    8.09  3.1e-07
## time1          -0.611      3.146   -0.19  0.84830
## time2          10.722      3.271    3.28  0.00444
## light1        -10.264      2.255   -4.55  0.00028
## time1:light1    4.389      3.146    1.39  0.18100
## time2:light1  -12.028      3.271   -3.68  0.00187
```

(The intercept doesn’t stay exactly the same when we add the interaction because the data are unbalanced: try `with(lizards,table(light,time))`)

“Effects” contrasts

What if we want the equivalent of coding a binary response as (-0.5,0.5) (so the effects sum to zero, but the difference is

```
coef(lm(grahami~light, data=lizards))
```

```
## (Intercept)  lightshady
##    8.090909   20.409091
```

```
coef(lm(grahami~light, data=lizards, contrasts=list(light=contr.sum)))
```

```
## (Intercept)      light1
##    18.29545   -10.20455
```

```
coef(lm(grahami~light, data=lizards,
        contrasts=list(light=matrix(c(0.5,-0.5)))))
```

```
## (Intercept)      light1
##    18.29545   -20.40909
```

```
with(lizards,mean(grahami[light=="sunny"])-mean(grahami[light=="shady"]))
```

```
## [1] -20.40909
```

```
## equivalent of sum-to-zero but with (A-B) rather than (A
inv_cc <- matrix(c(0.5,0.5,-0.5,0.5),nrow=2,byrow=TRUE)
solve(inv_cc)
```

```
##      [,1] [,2]
## [1,]    1   -1
## [2,]    1    1
```

Ways to set contrasts

There are *too many* ways to set contrasts in R

- **global options**: `options(contrasts=c("contr.sum", "contr.poly"))` (the first element is for (regular) *unordered* factors, the second for *ordered* factors)
- **on the fly in a model**, via the `contrasts` argument. `contrasts` is documented in `?model.matrix`, under `contrasts.arg`:

  > a list, whose entries are values (numeric matrices, `function`s or character strings naming functions) . . .

For example, `lm(..., contrasts=list(factorA="contr.sum",factorB="contr.treatment")` or `list(factorA=contr.sum, factorB=contr.treatment)` or `list(factorA=contr.sum(2), factorB=contr.sum(4))` (where the numeric values are the number of levels of the factors: since contrast functions return a matrix, this works too . . .

- **by setting the contrast attribute of a variable**: you can say e.g. `contrasts(dd$factor) <- cc` where `cc` is

  either a numeric matrix ... or the (quoted) name of a function which computes such matrices

  so something like `contrasts(dd$factorA) <- "contr.sum"` is probably best, but you could use `contr.sum(2)` instead.

- via the `C()` function, which does more or less the same thing on the fly, e.g. using `C(factorA, "contr.sum")` in a model is the same as specifying `contrasts=list(factorA, "contr.sum")`. Once again there are **three** ways to specify the contrasts (as a function (`contr.treatment`), as a string (`"contr.treatment"`), or as a matrix (`contr.treatment(2)`).

  I think that makes 1 + 3 + 2 + 3 = 9 = **too many** ways.

Other refs

- Schad et al. (2018); Gelman (2008)
- `http://sas-and-r.blogspot.com/2010/10/example-89-contrasts.html`
- see also: `gmodels::fit.contrast`, `rms::contrast.rms` for on-the-fly contrasts
- `http://www.ats.ucla.edu/stat/r/library/contrast_coding.htm`

References

Crawley, Michael J. 2002. *Statistical Computing: An Introduction to Data Analysis Using S-PLUS*. John Wiley & Sons.

Gelman, Andrew. 2008. "Scaling Regression Inputs by Dividing by Two Standard Deviations." *Statistics in Medicine* 27 (15): 2865–73. `https://doi.org/10.1002/sim.3107`.

Gotelli, Nicholas J., and Aaron M. Ellison. 2004. *A Primer of Ecological Statistics*. Sunderland, MA: Sinauer.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. London: Chapman; Hall.

McKeon, C. Seabird, Adrian Stier, Shelby McIlroy, and Benjamin Bolker. 2012. "Multiple Defender Effects: Synergistic Coral Defense by Mutualist Crustaceans." *Oecologia* 169 (4): 1095–1103. `https://doi.org/10.1007/s00442-012-2275-2`.

Schad, Daniel J., Sven Hohenstein, Shravan Vasishth, and Reinhold Kliegl. 2018. "How to Capitalize on a Priori Contrasts in Linear (Mixed) Models: A Tutorial." *arXiv:1807.10451 [Stat]*, July. `http://arxiv.org/abs/1807.10451`.

Schielzeth, Holger. 2010. "Simple Means to Improve the Interpretability of Regression Coefficients." *Methods in Ecology and Evolution* 1: 103–13. https://doi.org/10.1111/j.2041-210X.2010.00012.x.

Schoener, Thomas W. 1970. "Nonsynchronous Spatial Overlap of Lizards in Patchy Habitats." *Ecology* 51 (3): 408–18. https://doi.org/10.2307/1935376.

Venables, W. N. 1998. "Exegeses on Linear Models." In *1998 International S-PLUS User Conference*. Washington, DC. http://www.stats.ox.ac.uk/pub/MASS3/Exegeses.pdf.