Test contrast coding in Experiment 11 (spatial oddity)

```r
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

The texture experiment design matrix consists of two fixed-effect factors: the image model (with 6 levels) and the presentation condition (two levels). Image model is coded with sequential difference coding (`contr.sdif` from MASS) and the presentation condition is coded with treatment coding (parafoveal = 0, inspection = 1). Here I test the interpretation of the fixed-effect parameters using a simplified model with only 3 levels per A factor.

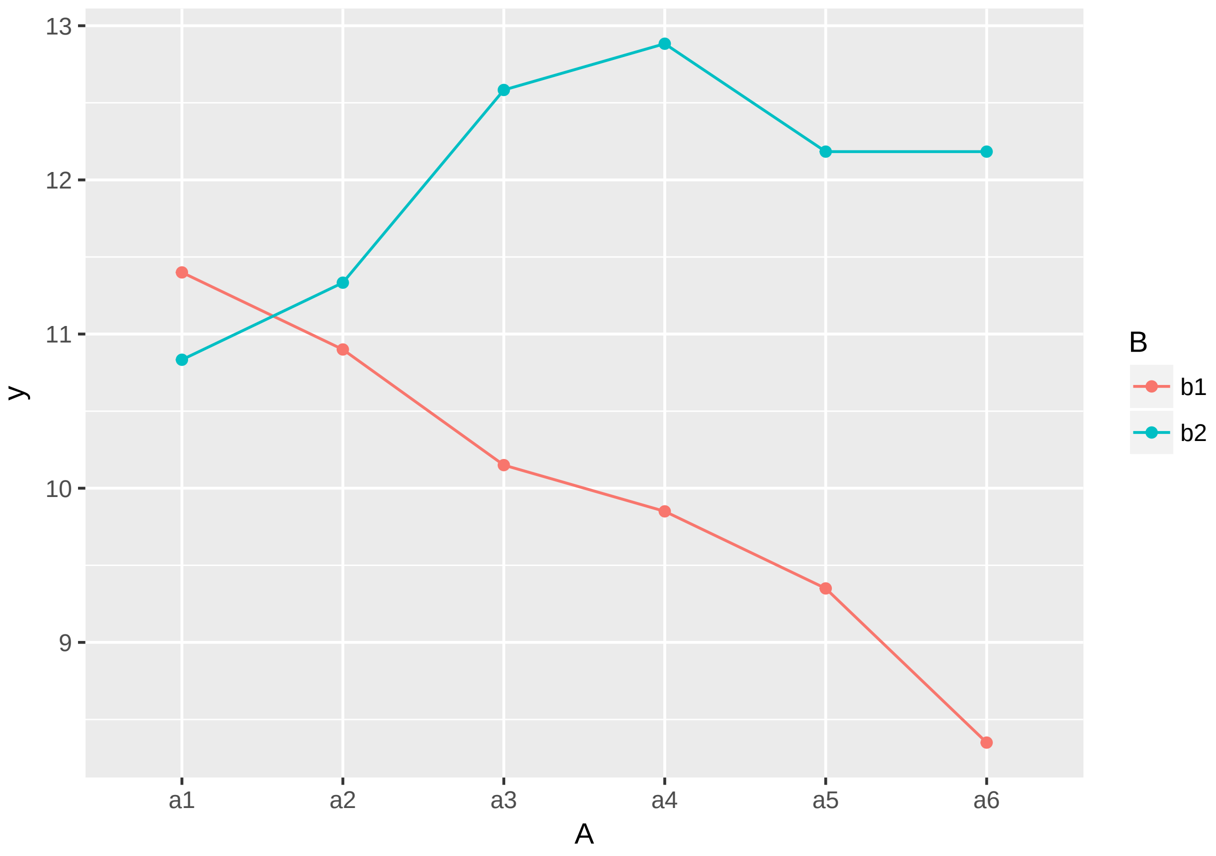
Generate data

```r
dat <- expand.grid(A = c("a1", "a2", "a3", "a4", "a5", "a6"),
                   B = c("b1", "b2"))

# set contrasts:
dat$A <- C(dat$A, MASS::contr.sdif)
dat$B <- C(dat$B, contr.treatment)

betas <- c(10,
           -0.5, -0.75, -0.3, -0.5, -1,
           2,
           1, 2, 0.6, -0.2, 1)
X <- model.matrix(~ A * B, dat)
dat$y <- as.vector(X %*% betas)

ggplot(dat, aes(x = A, y = y, colour = B)) +
  geom_point() +
  geom_line(aes(group = B)) +
  scale_y_continuous(breaks = seq(0, 20, by = 1))
```



A linear model fit recovers the betas we put in:

```
fit_gen <- lm(y ~ A * B, dat)
fit_gen_coeffs <- coefficients(fit_gen)
summary(fit_gen)
```

```
##
## Call:
## lm(formula = y ~ A * B, data = dat)
##
## Residuals:
## ALL 12 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    10.00         NA      NA       NA
## A2-1           -0.50         NA      NA       NA
## A3-2           -0.75         NA      NA       NA
## A4-3           -0.30         NA      NA       NA
## A5-4           -0.50         NA      NA       NA
## A6-5           -1.00         NA      NA       NA
## B2              2.00         NA      NA       NA
## A2-1:B2         1.00         NA      NA       NA
## A3-2:B2         2.00         NA      NA       NA
## A4-3:B2         0.60         NA      NA       NA
## A5-4:B2        -0.20         NA      NA       NA
## A6-5:B2         1.00         NA      NA       NA
##
```

```
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 11 and 0 DF,  p-value: NA
```

Test marginals

Is the beta weight for B equal to the average difference between the conditions?

```
dat %>%
  group_by(B) %>%
  summarise(marginal = mean(y))
```

```
## # A tibble: 2 x 2
##         B marginal
##   <fctr>    <dbl>
## 1     b1       10
## 2     b2       12
```

`b2` is indeed 2 higher than `b1` on average. The differences coded for the interaction terms (all 1 in this example) are indeed sequentially added to the differences coded for A when B = b1. Furthermore the mean of `b1` is 10, corresponding to the intercept term.

The contrast coding used is working as intended.

Alternative: sum coding

Using sum coding for the B factor (in our actual experiment this two-level factor corresponds to inspection condition) makes interpreting some of the group-level variances easier (see Westfall et al., 2014). However, we're interested in reporting the "simple effect" differences in the levels of A at each level of B. How does coding B as sum (-1, 1) change our ability to easily report simple effect differences of A?

```
# change B contrast to sum:
dat$B <- C(dat$B, contr.sum)  # b1 = 1, b2 = -1
fit_sum <- lm(y ~ A * B, dat)
fit_sum_coefs <- round(coefficients(fit_sum), 3)
print(fit_sum_coefs)
```

```
## (Intercept)        A2-1        A3-2        A4-3        A5-4        A6-5
##       11.00        0.00        0.25        0.00       -0.60       -0.50
##          B1     A2-1:B1     A3-2:B1     A4-3:B1     A5-4:B1     A6-5:B1
##       -1.00       -0.50       -1.00       -0.30        0.10       -0.50
```

Now,

- the intercept corresponds to the grand mean

```
mean(dat$y)
```

```
## [1] 11
```

- the B1 beta weight is half the actual magnitude, i.e. $\beta = \frac{\mu_1 - \mu_2}{2}$. This is because it codes the distance between each level and the grand mean. To get the distance between levels we would need to double the coefficient.

- the weights for the A factor code the mean difference between the difference of levels of A (averaged over B):

```
mean_A <- dat %>%
  group_by(A) %>%
  summarise(y = mean(y))

# sequential differences:
av_diffs <- diff(mean_A$y)
av_diffs
```

```
## [1]  0.00  0.25  0.00 -0.60 -0.50
```

- the interaction terms code the difference of the "simple effect" (at each level of B) from the average difference:

```
# "simple effect" differences:
diffs_b1 <- diff(dat$y[dat$B == "b1"])
diffs_b2 <- diff(dat$y[dat$B == "b2"])

diffs_b1 - av_diffs
```

```
## [1] -0.5 -1.0 -0.3  0.1 -0.5
```

```
diffs_b2 - av_diffs
```

```
## [1]  0.5  1.0  0.3 -0.1  0.5
```

Computing marginals: a few worked examples

We can then easily compute the marginal (simple) effects by adding / subtracting the interaction term to the base `sdif` term as appropriate:

- Marginal difference a2 - a1 when B = b1: $\beta_{A2-1} + \beta_{A2-1:B1}$. This gives 0 + -0.5 = -0.5. True value is -0.5.
- Marginal difference a2 - a1 when B = b2: $\beta_{A2-1} - \beta_{A2-1:B1}$. This gives 0 - -0.5 = 0.5. True value is 0.5.
- Marginal difference a3 - a2 when B = b1: $\beta_{A3-2} + \beta_{A3-2:B1}$. This gives 0.25 + -1 = -0.75. True value is -0.75.
- Marginal difference a3 - a2 when B = b2: $\beta_{A3-2} - \beta_{A3-2:B1}$. This gives 0.25 - -1 = 1.25. True value is 1.25.
- Marginal difference b2 - b1: $2 \cdot \beta_{B1}$. Then multiply by -1 to change the sign (because b1 = 1 and b2 = -1). This gives $-2\times$ -1 = 2. True value is 2.

Alternative: sdif coding

We want to check against an alternative coding that has intercept = grand mean, but whose coefs correspond to the differences we wish to test. For this we can again use contr.sdif, which will test the difference b2 - b1.

```
# change B contrast to deviance:
dat$B <- C(dat$B, MASS::contr.sdif)
fit_sdif <- lm(y ~ A * B, dat)
fit_sdif_coefs <- round(coefficients(fit_sdif), 3)
print(fit_sdif_coefs)
```

```
## (Intercept)        A2-1        A3-2        A4-3        A5-4        A6-5
##       11.00        0.00        0.25        0.00       -0.60       -0.50
##        B2-1   A2-1:B2-1   A3-2:B2-1   A4-3:B2-1   A5-4:B2-1   A6-5:B2-1
##         2.00        1.00        2.00        0.60       -0.20        1.00
```

Contrast coding matrix:

```
##     A  B        y (Intercept)  A2-1  A3-2 A4-3  A5-4  A6-5 B2-1 A2-1:B2-1
## 1  a1 b1 11.40000           1 -0.83 -0.67 -0.5 -0.33 -0.17 -0.5      0.42
## 2  a2 b1 10.90000           1  0.17 -0.67 -0.5 -0.33 -0.17 -0.5     -0.08
## 3  a3 b1 10.15000           1  0.17  0.33 -0.5 -0.33 -0.17 -0.5     -0.08
## 4  a4 b1  9.85000           1  0.17  0.33  0.5 -0.33 -0.17 -0.5     -0.08
## 5  a5 b1  9.35000           1  0.17  0.33  0.5  0.67 -0.17 -0.5     -0.08
## 6  a6 b1  8.35000           1  0.17  0.33  0.5  0.67  0.83 -0.5     -0.08
## 7  a1 b2 10.83333           1 -0.83 -0.67 -0.5 -0.33 -0.17  0.5     -0.42
## 8  a2 b2 11.33333           1  0.17 -0.67 -0.5 -0.33 -0.17  0.5      0.08
## 9  a3 b2 12.58333           1  0.17  0.33 -0.5 -0.33 -0.17  0.5      0.08
## 10 a4 b2 12.88333           1  0.17  0.33  0.5 -0.33 -0.17  0.5      0.08
## 11 a5 b2 12.18333           1  0.17  0.33  0.5  0.67 -0.17  0.5      0.08
## 12 a6 b2 12.18333           1  0.17  0.33  0.5  0.67  0.83  0.5      0.08
##    A3-2:B2-1 A4-3:B2-1 A5-4:B2-1 A6-5:B2-1
## 1       0.33      0.25      0.17      0.08
## 2       0.33      0.25      0.17      0.08
## 3      -0.17      0.25      0.17      0.08
## 4      -0.17     -0.25      0.17      0.08
## 5      -0.17     -0.25     -0.33      0.08
## 6      -0.17     -0.25     -0.33     -0.42
## 7      -0.33     -0.25     -0.17     -0.08
## 8      -0.33     -0.25     -0.17     -0.08
## 9       0.17     -0.25     -0.17     -0.08
## 10      0.17      0.25     -0.17     -0.08
## 11      0.17      0.25      0.33     -0.08
## 12      0.17      0.25      0.33      0.42
```

This is very similar to the matrix above except that the model weights for interaction terms are different (due to the multiplication with 0.5 instead of 1).

So what do these beta weights mean? Looking at the weights for the A factor alone, they are the same as the sum-coding model. So they code the mean difference between the levels of A, averaged over B. The B weight is half the value in the sum coding model, which means that it is the *actual* mean difference in the levels of B rather than half that value.

The main change from the sum model, though, is that the `A:B` interaction terms now code *double* the difference between the means (plus the opposite sign, because now the difference between b1 and b2 is oppositely-coded). So to get the marginal means for them, we need to halve the coefficients.

Computing marginals: a few worked examples

- Marginal difference a2 - a1 when B = b1: $\beta_{A2-1} - \frac{\beta_{A2-1:B2-1}}{2}$. This gives 0 - 0.5 = -0.5. True value is -0.5.
- Marginal difference a2 - a1 when B = b2: $\beta_{A2-1} + \frac{\beta_{A2-1:B2-1}}{2}$. This gives 0 + 0.5 = 0.5. True value is 0.5.
- Marginal difference a3 - a2 when B = b1: $\beta_{A3-2} - \frac{\beta_{A3-2:B2-1}}{2}$. This gives 0.25 - 1 = -0.75. True value is -0.75.
- Marginal difference a3 - a2 when B = b2: $\beta_{A3-2} + \frac{\beta_{A3-2:B2-1}}{2}$. This gives 0.25 + 1 = 1.25. True value is 1.25.
- Marginal difference b2 - b1: β_{B2-1}. This gives 2. True value is 2.

So in this case, coding B as `sdif` is pretty much the same as using sum coding (just determines how much you have to halve the coefficients). For factors with more levels, one might make more sense than the other.