Lab 4: Coding Categorical Variables

Wednesday Bushong

3/1/2017

Part I: Coding Categorical Variables

In this lab we'll be learning about dummy and contrast coding for categorical variables.

Data Information

Dependent variable: Self-Confidence, a numeric value ranging from -0.36 – 9.15; it seems that this variable has been standardized in some way already.

Predictor variable: Group, a categorical variable 4 with levels

- adult.neg: adults, receiving negative feedback
- adult.pos: adults, receiving positive feedback
- child.neg: children, receiving negative feedback
- child.pos: children, receiving negative feedback

(Note that this is really a 2x2 design with adult vs. child & negative vs. positive feedback, but we're not going to talk about categorical interactions for another week so we'll treat this as one variable.)

Let's load the data and get a feel for it:

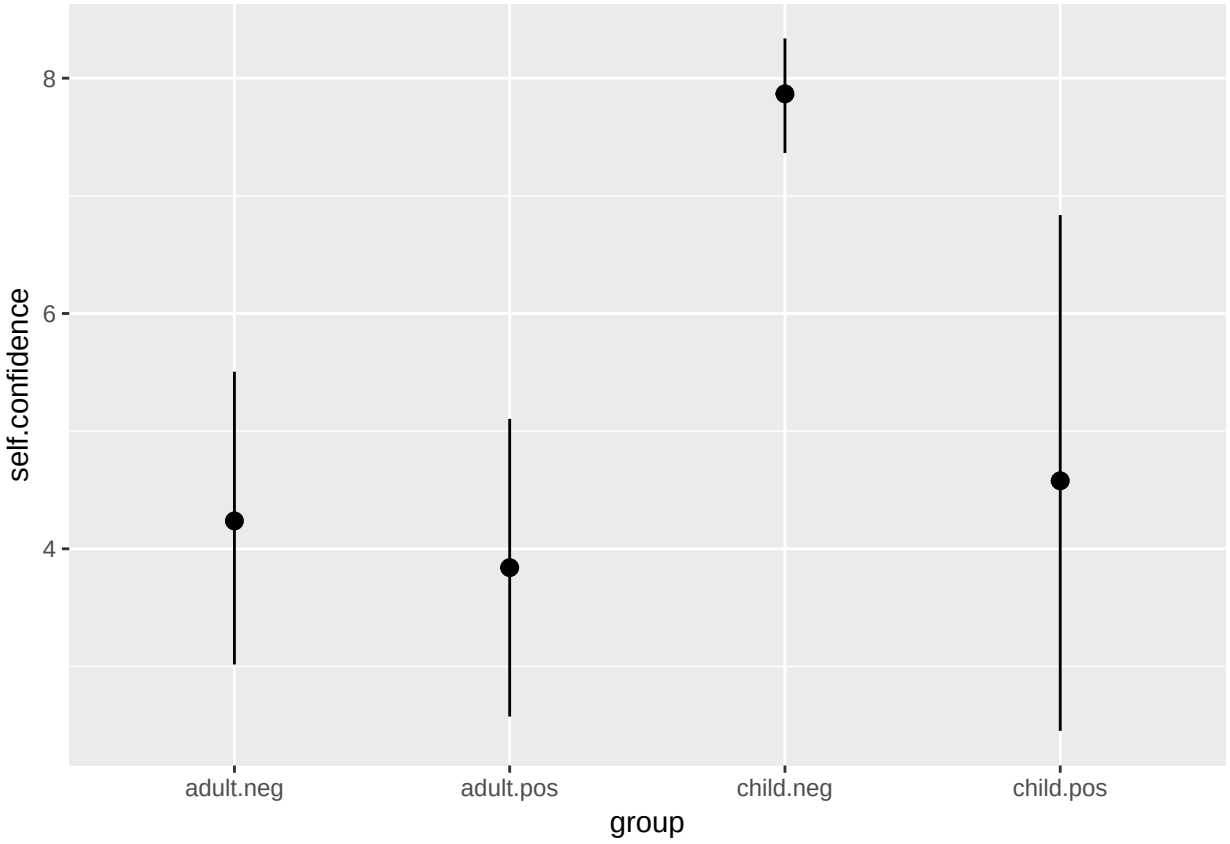
```
##       group self.confidence
## 1 adult.neg            3.90
## 2 adult.neg            2.79
## 3 adult.neg            2.26
## 4 adult.neg            2.59
## 5 adult.neg            0.47
## 6 adult.neg            5.15
```

```
##        group    self.confidence
##  adult.neg:10   Min.   :-0.360
##  adult.pos:10   1st Qu.: 2.643
##  child.neg:10   Median : 5.570
##  child.pos:10   Mean   : 5.130
##                 3rd Qu.: 7.452
##                 Max.   : 9.150
```

As we can see from the `summary()` function, we have an equal number of subjects in each of the groups, so we don't need to worry about any of our variable coding potentially being non-orthogonal if we choose to do contrast/sum coding.

First, let's visualize the data and get a feel for what the group differences might look like:

```
p <- ggplot(d, aes(x = group, y = self.confidence)) +
  stat_summary(fun.data = mean_cl_boot, geom = "pointrange")
p
```



Overview of Coding in R

One of the biggest blessings and curses of R is that every categorical variable in a data frame is defaultly assigned a coding scheme. To see how this works, let's see how R handles entering a categorical variable into a regression:

```
m <- lm(self.confidence ~ group, d)
summary(m)
```

```
##
## Call:
## lm(formula = self.confidence ~ group, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.938 -1.690 -0.257  1.558  4.232
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.2370     0.7724   5.485 3.39e-06 ***
```

```
## groupadult.pos  -0.3970     1.0924  -0.363  0.71841
## groupchild.neg   3.6300     1.0924   3.323  0.00205 **
## groupchild.pos   0.3410     1.0924   0.312  0.75672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.443 on 36 degrees of freedom
## Multiple R-squared:  0.3232, Adjusted R-squared:  0.2668
## F-statistic: 5.731 on 3 and 36 DF,  p-value: 0.002595
```

You'll notice that R isn't angry that you're only passing one variable into a regression when it has 4 levels. This is because R by default has instead created a coding scheme by default which is *implicitly* passed into the regression. That is, when you pass in "group", it passes in the 3 contrasts it has defaultly assigned. To see what those contrasts are, we can use the `contrasts()` function:

```
contrasts(d$group)
```

```
##           adult.pos child.neg child.pos
## adult.neg         0         0         0
## adult.pos         1         0         0
## child.neg         0         1         0
## child.pos         0         0         1
```

R defaultly assigns each categorical factor a dummy coding scheme, with the levels being sorted alphabetically, leaving adult.neg as the reference group.

Another convenient way to look at how all of your contrasts are being assigned is with the `model.matrix()` function. This returns, for every row in your data frame, the value of that row on each of the variables in the regression. We can check against our raw data to see how these line up.

```
head(model.matrix(m))
```

```
##   (Intercept) groupadult.pos groupchild.neg groupchild.pos
## 1           1              0              0             0
## 2           1              0              0             0
## 3           1              0              0             0
## 4           1              0              0             0
## 5           1              0              0             0
## 6           1              0              0             0
```

```
head(d)
```

```
##       group self.confidence
## 1 adult.neg            3.90
## 2 adult.neg            2.79
## 3 adult.neg            2.26
## 4 adult.neg            2.59
## 5 adult.neg            0.47
## 6 adult.neg            5.15
```

The first few rows have values of 0 for all of the dummy coded variables, because these were observations that came from the reference group, which we can confirm by seeing that first few rows of `d` are from the adult.neg group.

Dummy Coding

Let's imagine that we have the hypothesis that children who receive negative feedback have higher self-esteem than each of the other groups. How would we test that? We would want to keep the dummy coding scheme, but change the reference group. We can change contrasts in R by directly reassigning `contrasts(d$group)` to have the values we want. There is a handy function in R, `contr.treatment()`, that will create dummy codes for a specified number of levels and assign the reference group to one of those levels. Alternatively, we can manually assign contrasts by creating a matrix of our desired values.

```
## First way of dummy coding: using the contr.treatment() function
contrasts(d$group) <- contr.treatment(4, base = 3) # base specifies which level is the reference
# We can rename the contrast vectors to have more sensical names
colnames(contrasts(d$group)) <- c("childneg.vs.adultneg",
                                  "childneg.vs.adultpos",
                                  "childneg.vs.childpos")

kable(contrasts(d$group))
```

	childneg.vs.adultneg	childneg.vs.adultpos	childneg.vs.childpos
adult.neg	1	0	0
adult.pos	0	1	0
child.neg	0	0	0
child.pos	0	0	1

```
## Second way of dummy coding: manual assignment
contrasts(d$group) <- cbind(childneg.vs.adultneg = c(1, 0, 0, 0),
                            childneg.vs.adultpos = c(0, 1, 0, 0),
                            childneg.vs.childpos = c(0, 0, 0, 1))

kable(contrasts(d$group))
```

	childneg.vs.adultneg	childneg.vs.adultpos	childneg.vs.childpos
adult.neg	1	0	0
adult.pos	0	1	0
child.neg	0	0	0
child.pos	0	0	1

```
## Fit a model
m.childneg.ref <- lm(self.confidence ~ group, d)
summary(m.childneg.ref)
```

```
##
## Call:
## lm(formula = self.confidence ~ group, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.938 -1.690 -0.257  1.558  4.232
##
```

```
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  7.8670     0.7724  10.185  3.8e-12 ***
## groupchildneg.vs.adultneg  -3.6300     1.0924  -3.323 0.002053 **
## groupchildneg.vs.adultpos  -4.0270     1.0924  -3.686 0.000744 ***
## groupchildneg.vs.childpos  -3.2890     1.0924  -3.011 0.004740 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.443 on 36 degrees of freedom
## Multiple R-squared:  0.3232, Adjusted R-squared:  0.2668
## F-statistic: 5.731 on 3 and 36 DF,  p-value: 0.002595
```

Contrast Coding

Contrast coding is a bit more complicated because you need to think a lot harder about the hypotheses you want to test and what contrasts those map to, whereas with dummy coding you're always asking a pairwise-comparison type of question.

Suppose we have the following hypotheses:

- Children have higher self-confidence than adults.
- Feedback does not affect adults' self-confidence.
- Feedback does affect children's self-confidence.

How would we fill out a contrast matrix to test these questions?

```
contrasts(d$group) <- cbind(adult.vs.child = c(1, 1, -1, -1),
                            feedback.adult = c(1, -1, 0, 0),
                            feedback.child = c(0, 0, 1, -1))
# make sure to come to class to see the math of why this works out!

kable(contrasts(d$group))
```

	adult.vs.child	feedback.adult	feedback.child
adult.neg	1	1	0
adult.pos	1	-1	0
child.neg	-1	0	1
child.pos	-1	0	-1

```
m.contrastcodes1 <- lm(self.confidence ~ group, d)
summary(m.contrastcodes1)
```

```
##
## Call:
## lm(formula = self.confidence ~ group, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4.938 -1.690 -0.257  1.558  4.232
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5.1305     0.3862  13.284 1.91e-15 ***
## groupadult.vs.child  -1.0920     0.3862  -2.827  0.00762 **
## groupfeedback.adult   0.1985     0.5462   0.363  0.71841
## groupfeedback.child   1.6445     0.5462   3.011  0.00474 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.443 on 36 degrees of freedom
## Multiple R-squared:  0.3232, Adjusted R-squared:  0.2668
## F-statistic: 5.731 on 3 and 36 DF,  p-value: 0.002595
```

Checking for Orthogonality of Contrasts

R has a matrix multiplication operator, `%*%`, that we can use to compute the dot product between contrasts to check for orthogonality:

```
contrasts(d$group)[, 1] %*% contrasts(d$group)[, 2]
```

```
##      [,1]
## [1,]    0
```

```
contrasts(d$group)[, 2] %*% contrasts(d$group)[, 3]
```

```
##      [,1]
## [1,]    0
```

```
contrasts(d$group)[, 1] %*% contrasts(d$group)[, 3]
```

```
##      [,1]
## [1,]    0
```

Practice Problems

1. Dummy code d$group so that child.pos is the reference group and fit a model.

- What is the meaning of the intercept?
- From only looking at model output, what is the mean of the child.neg group?
- From only looking at model output, what is the mean of the adult group?

2. Create contrast codes for d$group that test the difference between (1) the mean of children and the mean of adults, (2) the mean of negative feedback and the mean of positive feedback, and (3) the difference between the effect of feedback for children vs. adults.

- After you've coded the contrasts and fit a model, how will you interpret each of the βs?

Part II: Type I, II, and III Sum of Squares

Let's introduce a new dataset that we'll be using next week as well:

```
d2 <- read.csv("UnbalancedInteractionData.csv")
head(d2)
```

```
##    X Self_control Gender     Grade
## 1 23     1725.860  males 7th grade
## 2 24     5852.000  males 7th grade
## 3 25     4130.750  males 7th grade
## 4 26     7385.600  males 7th grade
## 5 27     2163.206  males 7th grade
## 6 28     4548.188  males 7th grade
```

```
summary(d2)
```

```
##        X          Self_control       Gender          Grade
##  Min.   : 23.0   Min.   : 1726   females:33   10th grade:33
##  1st Qu.: 38.0   1st Qu.: 4143   males  :28   7th grade :28
##  Median : 70.0   Median : 5844
##  Mean   : 62.2   Mean   : 6166
##  3rd Qu.: 85.0   3rd Qu.: 7388
##  Max.   :100.0   Max.   :13450
```

We want to keep things simple for now, so let's reduce this data to a 2x2 problem:

```
# Contrast-code gender and grade:
contrasts(d2$Gender) <- cbind(male = c(-1, 1))
contrasts(d2$Grade) <- cbind(Grade7 = c(-1, 1))

mm <- model.matrix(Self_control ~ Gender * Grade, d2)
```

We can confirm that our contrasts are now not orthogonal by testing for orthogonality on the model matrix columns that represent our coding scheme:

```
mm[, 2] %*% mm[, 3]
```

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

```
mm[, 2] %*% mm[, 4]
```

```
##      [,1]
## [1,]   -5
```

```
mm[, 3] %*% mm[, 4]
```

```
##      [,1]
## [1,]   -5
```

Now, let's fit the actual 2x2 interaction model:

```r
m.int <- lm(Self_control ~ Gender * Grade, d2)
```

We can view the sum of squares by calling the `anova()` function:

```r
anova(m.int)
```

```
## Analysis of Variance Table
##
## Response: Self_control
##              Df    Sum Sq  Mean Sq F value    Pr(>F)
## Gender        1  93980374 93980374 17.2901 0.0001091 ***
## Grade         1  20074639 20074639  3.6933 0.0596371 .
## Gender:Grade  1    350416   350416  0.0645 0.8004819
## Residuals    57 309822737  5435487
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

But wait... what kinds of sum of squares are these?! This is another thing that R just does under the hood. This blog post on r-bloggers clarifies the issue: https://www.r-bloggers.com/anova-%E2%80%93-type-iiiiii-ss-explained/

As it turns out, `anova()` by default uses Type I SS. We can verify this by reversing the order of terms in our interaction model and see that the results change:

```r
m.int.2 <- lm(Self_control ~ Grade * Gender, d2) # enter Grade first
anova(m.int.2)
```

```
## Analysis of Variance Table
##
## Response: Self_control
##              Df    Sum Sq  Mean Sq F value    Pr(>F)
## Grade         1  15427637 15427637  2.8383   0.09751 .
## Gender        1  98627376 98627376 18.1451 7.749e-05 ***
## Grade:Gender  1    350416   350416  0.0645   0.80048
## Residuals    57 309822737  5435487
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As is often the case with R, a special package was developed for dealing with this strange, unchangeable, implicit behavior. Using the capital-A `Anova()` function from the `car` package, we can specify what type of SS we want, and confirm that the behavior is the same for both of our models:

```r
Anova(m.int, type = "II")
```

```
## Anova Table (Type II tests)
##
## Response: Self_control
##                Sum Sq Df F value    Pr(>F)
## Gender        98627376  1 18.1451 7.749e-05 ***
## Grade         20074639  1  3.6933   0.05964 .
## Gender:Grade    350416  1  0.0645   0.80048
## Residuals    309822737 57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
Anova(m.int.2, type = "II")
```

```
## Anova Table (Type II tests)
##
## Response: Self_control
##                 Sum Sq Df F value    Pr(>F)
## Grade         20074639  1  3.6933   0.05964 .
## Gender        98627376  1 18.1451 7.749e-05 ***
## Grade:Gender    350416  1  0.0645   0.80048
## Residuals    309822737 57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
Anova(m.int, type = "III")
```

```
## Anova Table (Type III tests)
##
## Response: Self_control
##                  Sum Sq Df  F value    Pr(>F)
## (Intercept)  2169884624  1 399.2071 < 2.2e-16 ***
## Gender         96726063  1  17.7953 8.909e-05 ***
## Grade          19424023  1   3.5736   0.06379 .
## Gender:Grade     350416  1   0.0645   0.80048
## Residuals     309822737 57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
Anova(m.int.2, type = "III")
```

```
## Anova Table (Type III tests)
##
## Response: Self_control
##                  Sum Sq Df  F value    Pr(>F)
## (Intercept)  2169884624  1 399.2071 < 2.2e-16 ***
## Grade          19424023  1   3.5736   0.06379 .
## Gender         96726063  1  17.7953 8.909e-05 ***
## Grade:Gender     350416  1   0.0645   0.80048
## Residuals     309822737 57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

But wait... what does the `summary()` function for model objects by default give you?

```r
summary(m.int)
```

```
##
## Call:
## lm(formula = Self_control ~ Gender * Grade, data = d2)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -4967.0 -1396.9   -91.2  1814.9  5514.2
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             6019.24     301.26  19.980  < 2e-16 ***
## Gendermale             -1270.85     301.26  -4.218 8.91e-05 ***
## GradeGrade7             -569.50     301.26  -1.890   0.0638 .
## Gendermale:GradeGrade7    76.49     301.26   0.254   0.8005
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2331 on 57 degrees of freedom
## Multiple R-squared:  0.2697, Adjusted R-squared:  0.2312
## F-statistic: 7.016 on 3 and 57 DF,  p-value: 0.0004264
```

```
summary(m.int.2)
```

```
##
## Call:
## lm(formula = Self_control ~ Grade * Gender, data = d2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4967.0 -1396.9   -91.2  1814.9  5514.2
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             6019.24     301.26  19.980  < 2e-16 ***
## GradeGrade7             -569.50     301.26  -1.890   0.0638 .
## Gendermale             -1270.85     301.26  -4.218 8.91e-05 ***
## GradeGrade7:Gendermale    76.49     301.26   0.254   0.8005
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2331 on 57 degrees of freedom
## Multiple R-squared:  0.2697, Adjusted R-squared:  0.2312
## F-statistic: 7.016 on 3 and 57 DF,  p-value: 0.0004264
```

Yep, that's right – even though the default `anova()` function gives you Type I SS, the default `summary()` function gives you Type III.