Contrasts

Bryce Bartlett

April 9, 2018

Introduction

This is a test and validation document for testing an example of a multifactor ANOVA using an "effects" coding scheme. It proceeds in two parts. First is to simulate data with an arbitrary set of betas. Second is to estimate effects using an effects coding scheme. Third is to validate the meaning of the effects coding.

Effects coding is a special modification of dummy variable coding where the intercept is the grand mean, and each effect estimate provides the additive effect relative to the mean.

Given a polytomous explanatory variable (such as race/ethnicity, gender, occupation, etc.), the most common is dummy variable coding relative to an (excluded) reference category. In this case, the factor is expanded to a dummy variable series with a reference that is omitted.

This is the example of dummy variable coding for a factor variable with coding of "a", "b", or "c", with "c" as the reference category.

```
###contr.treatment is the standard dummy variable coding scheme in R, which uses the first category as
dummy = contr.SAS(3)
#contr.sum is R's version of effects coding
sum = contr.sum(3)
dimnames(dummy) = dimnames(sum) =
  list(letters[1:3],paste0('dummy.',letters[1:2]))

kable(dummy)
```

	dummy.a	dummy.b
a	1	0
b	0	1
c	0	0

This coding leads to effects interpreted relative to the omitted category. Another common coding is "effects" coding. In this case, the omitted category as a value of -1 across all dummy variable series.

```
kable(sum)
```

	dummy.a	dummy.b
a	1	0
b	0	1
c	-1	-1

In this instance, the intercept is the unweighted mean of the outcome variable, and the estimated effects are the deviance of each group from the unweighted mean. Changing the coding schemas does not effect fit statistics; it is just an alternative "look" at the effects (Hardy 1993, 64–75).

An identical effect of the omitted category is easy to calculate—it is just -1 times the sum of all other categories. The effect coding is identified as a "sum" contrast in R, because the "sum" of all effects (absent the intercept) is constrained to 0.

The remainder of the document illustrates an application of the effects coding scheme.

Step 1: Arbitrarily Generate data

This generates two factors (eff and eff1 generate arbitrary continuous outcomes y and y1). These numbers are not strictly recoverable; there are an infinite number of solutions that can create the same outcome variables. These are only identifiable uniquely if something is fixed (that is why dummy variables are estimated with an omitted category or some other constraint).

Note that I do this 1,000 times as a "simulation experiment".

```
simdat = function(...){

  f1=rep(factor(c('a','b','c')),10)
  f2=rep(factor(c('d','e','f')),10)

  eff = matrix(c(1.25,-1,2,-1,1,7),1,6)

  eff2 = matrix(c(1.25,-1,2,0,0,0),1,6)

  dat = expand.grid(f1,f2); colnames(dat) = c('f1','f2')

  #coding for generating effects
  xhat =   cbind(model.matrix(~-1+f1,data=dat),
             model.matrix(~-1+f2,data=dat))

  #full set
  dat$yhat =xhat %*% t(eff)
  dat$y = rnorm(nrow(dat),mean=dat$yhat,sd=1.5)

  #factor 1 only
  dat$yhat2 = xhat %*% t(eff2)
  dat$y2 = rnorm(nrow(dat),mean=dat$yhat2,sd=1.5)

  return(dat)

}

dats = lapply(1:1000,FUN=simdat)
```

2. Effect coding model.

I calculate the grand mean and estimate a model with effects coding as below.

```
#grand mean
M=unlist(lapply(dats,function(dat) mean(dat$y)))

#model
t.sum = lapply(dats,function(dat)
                lm(y~f1+f2,data=dat,
                   contrasts=list(f1='contr.sum',f2='contr.sum'))
)
```

Here is an example of the results from an effects coded regression.

```
library(stargazer)
```

```
## Warning: package 'stargazer' was built under R version 3.4.3
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.1. https://CRAN.R-project.org/package=stargazer
```

```
stargazer(t.sum[[1]],type='text')
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                                   y
## ------------------------------------------------
## f11                            0.534***
##                                 (0.069)
##
## f12                            -1.786***
##                                 (0.069)
##
## f21                            -3.277***
##                                 (0.069)
##
## f22                            -1.455***
##                                 (0.069)
##
## Constant                       3.116***
##                                 (0.049)
##
## ------------------------------------------------
## Observations                      900
## R2                               0.864
## Adjusted R2                      0.863
## Residual Std. Error       1.460 (df = 895)
## F Statistic          1,418.269*** (df = 4; 895)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

Using the results of the regression, I calculate the expected means of the variables as described above. Specifically, the intercept plus the estimate is the expected value of the mean. For the omitted category, the estimate is -1 times the sum of all other categories.

```
#calculate mean levels across all categoies from the
sm2 = do.call(rbind,lapply(t.sum,coef))
mean.code = data.frame(
  a=sm2[,1]+sm2[,2],
  b=sm2[,1]+sm2[,3],
  c=sm2[,1]+ -1*(sm2[,2]+sm2[,3]),
  d=sm2[,1]+sm2[,4],
  e=sm2[,1]+sm2[,5],
  f=sm2[,1]+-1*(sm2[,4]+sm2[,5]))
```

```
mm =apply(mean.code,2,mean)

kable(mm)
```

a	3.583677
b	1.336493
c	4.338274
d	-0.242470
e	1.748962
f	7.751952

When I compare this relative to the observed group-level means, I confirm that it is correct.

```
mns.1 = lapply(dats,function(dat){
  return(dat %>%
           group_by(f1) %>%
           summarize(mean=mean(y)) %>%
           select(mean))
  })

mns.2 = lapply(dats,function(dat){
  return(dat %>%
           group_by(f2) %>%
           summarize(mean=mean(y)) %>%
           select(mean))
  })

c.means = c(apply(do.call(cbind,mns.1),1,mean),
apply(do.call(cbind,mns.2),1,mean))

names(c.means) = letters[1:6]

dd = rbind(c.means,mm)
rownames(dd) = c('Sliced Means','Means from Calculated from Regression')
kable(dd)
```

	a	b	c	d	e	f
Sliced Means	3.583677	1.336493	4.338274	-0.24247	1.748962	7.751952
Means from Calculated from Regression	3.583677	1.336493	4.338274	-0.24247	1.748962	7.751952

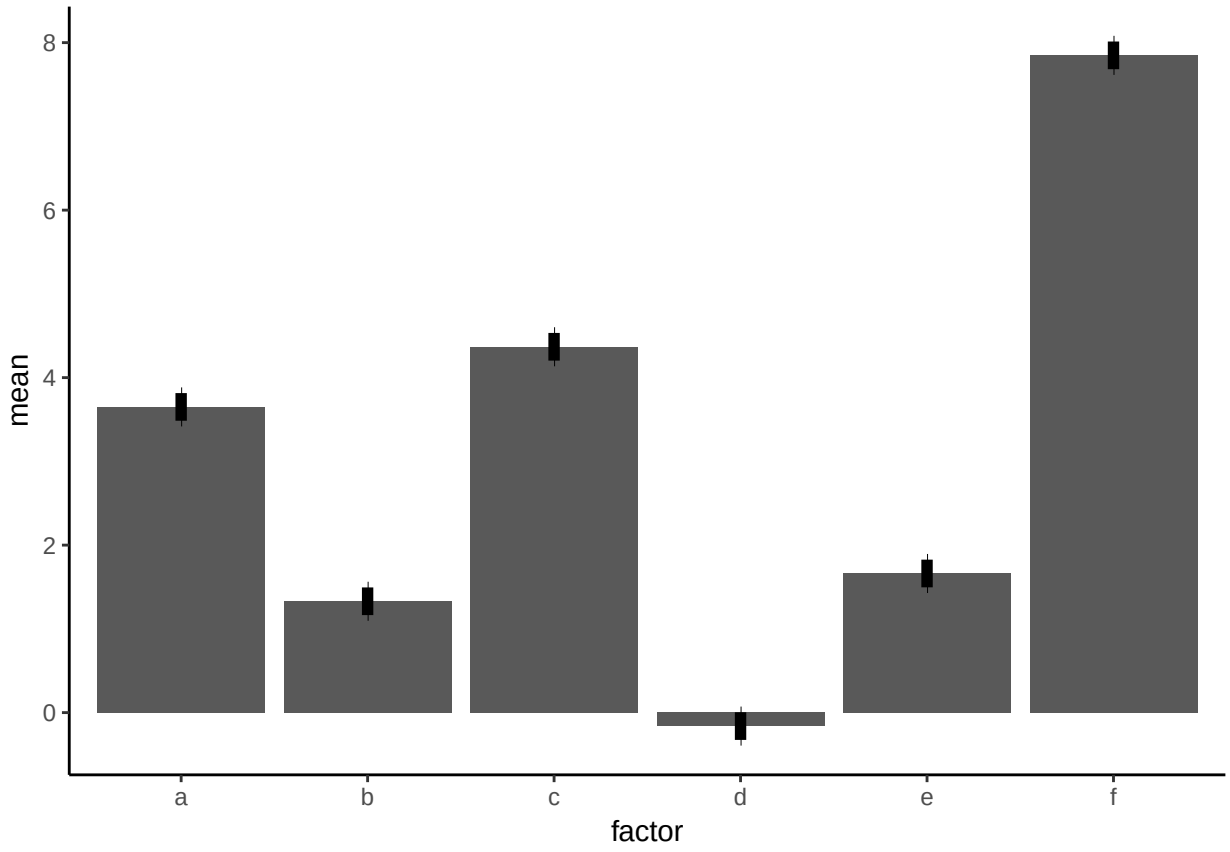
Finally, using the msm package, it is straightforward to calculate standard errors from the effects coding model.

```
#https://stats.idre.ucla.edu/r/faq/how-can-i-estimate-the-standard-error-of-transformed-regression-para

#collect betas from one model
beta = coef(t.sum[[1]])
#collect variance-covariance matrix
vc = vcov(t.sum[[1]])

#S.E. of mean (with predictions at 1 (all others at 0))
#unit tests
```

```
#deltamethod(~x1+x2,beta,vc) #se for factor 1
#deltamethod(~x1-x2-x3,beta,vc) #se for omitted factor
#sqrt(t(c(1,1)) %*% vc[1:2,1:2] %*% c(1,1)) #se for factor 1
#sqrt(t(c(1,-1,-1)) %*% vc[1:3,1:3] %*% c(1,-1,-1)) #se for omitted factor

#f.sum = cbind(rep(1,3),contr.sum(3))
#pred = f.sum %*% beta[1:3]
#sqrt(diag(f.sum %*% vc[1:3,1:3] %*% t(f.sum)))

#set up x matrix
xm = cbind(contr.sum(3),matrix(0,3,2))
xm = rbind(xm,cbind(matrix(0,3,2),contr.sum(3)))
xm = cbind(rep(1,6),xm)

pred = data.frame(
  mean=xm %*% beta,
  se = sqrt(diag((xm %*% vc %*% t(xm))))
)
```

```
## Warning in data.row.names(row.names, rowsi, i): some row.names duplicated:
## 4,5,6 --> row.names NOT used
```

```
pred$factor = factor(letters[1:6])

library(ggplot2)

ggplot(pred,aes(x=factor,y=mean,ymax=mean+1.96*se,ymin=mean-1.96*se))+
  geom_bar(stat='identity') +
  geom_errorbar(width=0,size=2) +
  theme_classic()
```



Citations

Hardy, Melissa A. 1993. *Regression with Dummy Variables.* Sage University Papers Series, no. 07-093. Newbury Park: Sage Publications.