Brief Graphing Demo for Factorial ANOVA

Graduate Statistics

January 27, 2016

The following vignette illustrates a few elementary techniques for performing factorial ANOVA and graphing mean differences. This document is far from exhaustive and should be considered to be a brief example rather than a complete tutorial. Particularly, *this document avoids any discussion of the validity of the results yielded by this particular analysis*.

Income, Sex, and Education in 1988 Chile

0.1 Constructing codes

The following dataset is included in John Fox's `car` package. See `?car::Chile` for more information. We will examine `income` (in pesos) as a function of `sex` (a factor with levels: 'F', female; 'M', male), `education` (a factor with levels: 'P', Primary; 'PS', Post-secondary; 'S', Secondary), and their interaction. To do this, we construct the following set of codes:

	Female			Male		
	Primary	Secondary	Post-secondary	Primary	Secondary	Post-secondary
X1	+1	+1	+1	-1	-1	-1
X2	-2	+1	+1	-2	+1	+1
X3	0	-1	+1	0	-1	+1
X4	-2	+1	+1	+2	-1	-1
X5	0	-1	+1	0	+1	-1

Now in R. Make sure you have the `car` package installed.

```
main <- car::Chile # assign the data we're interested in to a data.frame called 'main'

X1 <- (main$sex == "F")*1 + (main$sex == "M")*-1
X1 <- (main$sex == "F")*2 - 1
# quicker alternative but less readable
X2 <- (main$education == "P")*-2 + (main$education != "P")*1
# note the handyness of `!` here
X3 <- (main$education == "S")*-1 + (main$education == "PS")*1
# note that we don't need to worry about the Primary group. They get zeros automatically!
X4 <- X1 * X2
X5 <- X1 * X3
```

To ensure our codes will be interpretable, we need to satisfy two criteria: 1) each code is orthogonal to every other code[1]; 2) each code sums to zero. We can accomplish this easily using R and a little bit of matrix algebra. First, we note that if the rows of a matrix are orthogonal, post-multiplying it by its transpose will

[1] i.e., their pairwise dot products are zero

result in a *diagonal* matrix. That is, all of the entries off the leading diagonal ↘ will be zero[2]:

$$\begin{pmatrix} \cdots & X_1 & \cdots \\ \cdots & X_2 & \cdots \\ \cdots & X_3 & \cdots \\ \cdots & X_4 & \cdots \\ \cdots & X_5 & \cdots \end{pmatrix} \cdot \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ X_1 & X_2 & X_3 & X_4 & X_5 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} X_1 \cdot X_1 & X_1 \cdot X_2 & X_1 \cdot X_3 & X_1 \cdot X_4 & X_1 \cdot X_5 \\ X_2 \cdot X_1 & X_2 \cdot X_2 & X_2 \cdot X_3 & X_2 \cdot X_4 & X_2 \cdot X_5 \\ X_3 \cdot X_1 & X_3 \cdot X_2 & X_3 \cdot X_3 & X_3 \cdot X_4 & X_3 \cdot X_5 \\ X_4 \cdot X_1 & X_4 \cdot X_2 & X_4 \cdot X_3 & X_4 \cdot X_4 & X_4 \cdot X_5 \\ X_5 \cdot X_1 & X_5 \cdot X_2 & X_5 \cdot X_3 & X_5 \cdot X_4 & X_5 \cdot X_5 \end{pmatrix}$$

In R, we do this using the matrix multiplication function %*% and the transpose function t().

```
codes <- matrix(c(
                1,1,1,-1,-1,-1,
                -2,1,1,-2,1,1,
                0,-1,1,0,-1,1,
                -2,1,1,2,-1,-1,
                0,-1,1,0,1,-1), byrow = T, nrow = 5)
codes %*% t(codes)

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    6    0    0    0    0
## [2,]    0   12    0    0    0
## [3,]    0    0    4    0    0
## [4,]    0    0    0   12    0
## [5,]    0    0    0    0    4
```

Ensuring the rows sum to zero is simple:

```
rowSums(codes)

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

0.2 Performing the analysis

All of the single degree-of-freedom tests are part of the standard linear model output:

```
modA <- lm(main$income ~ X1 + X2 + X3 + X4 + X5)
summary(modA)

##
## Call:
## lm(formula = main$income ~ X1 + X2 + X3 + X4 + X5)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -67904 -20324  -3963   -324 181037
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 40759.41     760.66  53.584   <2e-16 ***
## X1             49.55     760.66   0.065    0.948
## X2          11464.25     492.86  23.261   <2e-16 ***
## X3          16352.67    1003.53  16.295   <2e-16 ***
## X4            590.65     492.86   1.198    0.231
```

[2]Matrix multiplication takes the dot products rows of the first matrix with columns of the second matrix; each entry off the diagonal is the dot product of two codes.

```
## X5           1187.12    1003.53   1.183    0.237
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35370 on 2587 degrees of freedom
##   (107 observations deleted due to missingness)
## Multiple R-squared:  0.2012,Adjusted R-squared:  0.1997
## F-statistic: 130.3 on 5 and 2587 DF,  p-value: < 2.2e-16
```

We can use the `anova()` function to perform two degree-of-freedom tests. Here we perform an "omnibus" test of the effects of education:

```
modC <- lm(main$income ~ X1 + X4 + X5)
anova(modC, modA)

## Analysis of Variance Table
##
## Model 1: main$income ~ X1 + X4 + X5
## Model 2: main$income ~ X1 + X2 + X3 + X4 + X5
##   Res.Df        RSS Df Sum of Sq      F    Pr(>F)
## 1   2589 4.0400e+12
## 2   2587 3.2368e+12  2 8.032e+11 320.97 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

0.3 Graphing the data

The `tapply()` function is an incredibly useful function for obtaining cell means. It takes three primary arguments: `X` is the vector were interested in; in this case it's `income`. `INDEX` is a list of factors which we want to "cross"; in this case, `education` and `sex`. `FUN` is the function we'll apply; in this case it will be the `mean()` function. Because we have missing data, we'll have to provide the additonal argument (to `mean()`), `na.rm = TRUE`. And to increase readability, we'll wrap the whole shebang inside a `with()` to avoid repeatedly referencing the data frame.

```
cellMeans <- with(main,
tapply(income, list(sex, education), mean, na.rm = TRUE)
)
```

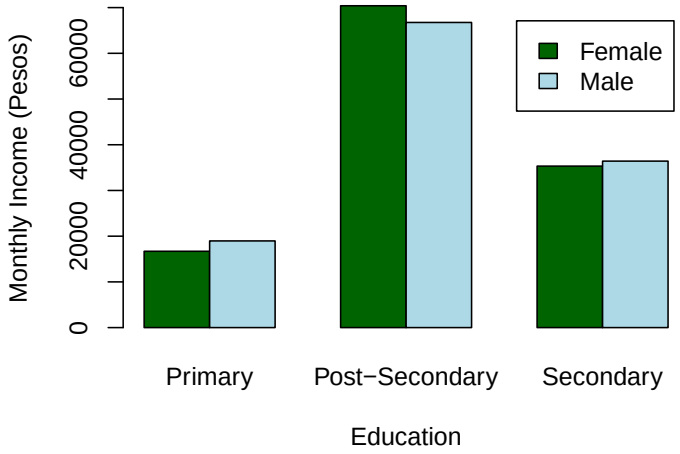
We can also use `tapply()` to find standard deviations. `table()`, which is really `tapply()` in disguise, returns cell counts. However, we're going to have to be careful to exclude observations for which `income` is missing. This will allow us to figure out standard errors (remember your formulae).

```
cellSDs <- with(main,
tapply(income, list(sex, education), sd, na.rm = TRUE)
)

cellNs <- with(main[!(is.na(main$income)), ],
table(sex, education)
)

cellNs

##    education
```

```
## sex   P  PS   S
##   F 590 192 540
##   M 482 253 536

cellSEs <- cellSDs / sqrt(cellNs)
```

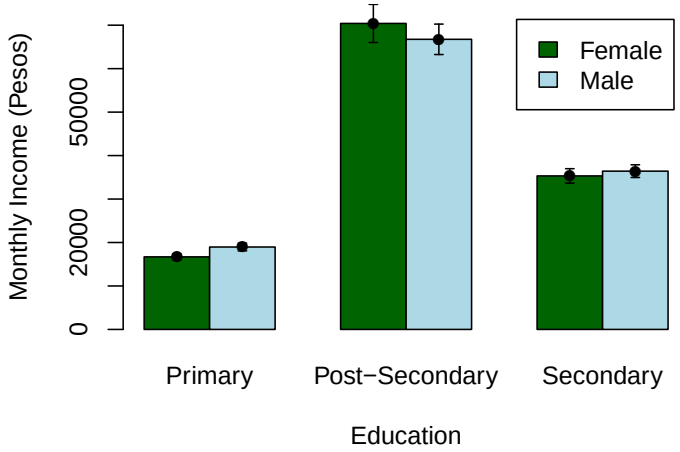
Now we can make a barplot. See if you can figure out what the code below means (see `?barplot`).

```
barplot(cellMeans, beside = TRUE, col = c("darkgreen","lightblue"),
        names.arg = c("Primary", "Post-Secondary", "Secondary"),
        legend = c("Female", "Male"), ylab = "Monthly Income (Pesos)",
        xlab = "Education")
```



You can add error bars in base R if you're a purist or if you don't know any better, but I think it's easier to use the `Hmisc` package (you'll probably need to install it).

```
xPositions <- barplot(cellMeans, beside = TRUE, col = c("darkgreen","lightblue"),
        names.arg = c("Primary", "Post-Secondary", "Secondary"),
        legend = c("Female", "Male"), ylab = "Monthly Income (Pesos)",
        xlab = "Education", ylim = c(0, max(cellMeans + cellSEs)))

Hmisc::errbar(xPositions, cellMeans, cellMeans + cellSEs, cellMeans - cellSEs, add = T)
```



Again, see if you can figure out how this works. Learning to read the documentation is likely the most important skill to develop if you want to become competent with R.

Note: this document was created with `knitr` and LyX.