Psych 610 Handout – R5
Two-way Between Part II: Contrasts
Equal or Balanced N only!!!

Colleen Moore & Mike Amato
University of Wisconsin - Madison

This tutorial uses Keppel & Wickens, Table 11.8, p. 223. The data are available in an excel spreadsheet in the same website where you found this document.

I. Bring in the data and prepare for the analyses you want to do.

If you have a mac, copy the data to the clipboard and tell R this:

```
> d1=read.table(pipe("pbpaste"),header=T)      # creates a data frame called "d1"
```

If you're using a PC or a mac, save your data as a text file and do one of the following:

```
> d1=read.table(file.choose(),header=T)        #opens a window for you to browse
> d1=read.table("filename.txt", header=T)      #R must be in the correct target directory
```

```
> d1            # ask R to show you the data
   errors A B
1       1 1 1
2       4 1 1
3       0 1 1
4       7 1 1
5      13 2 1
6       5 2 1
7       7 2 1
8      15 2 1
9       9 3 1
10     16 3 1
11     18 3 1
12     13 3 1
13     15 1 2
14      6 1 2
15     10 1 2
16     13 1 2
17      6 2 2
18     18 2 2
19      9 2 2
20     15 2 2
21     14 3 2
22      7 3 2
23      6 3 2
24     13 3 2
```

In previous sessions we've attached the data frame. The advantage of attaching is that we could refer directly to the columns in the data frame. But there are disadvantages to attaching, especially if you are doing transformations on your data, or want to work with more than one data frame at once. So for this session we won't attach the data frame, which means that each time you refer to a column you'll also have to specify the data frame the column is in using a dollar sign. Kind of a pain, but at least we gave it a short name.

Tell R that your factors are categorical, or else it will treat them as continuous:

```
> d1$A = factor(d1$A)        # if we had used a1,a2,a3 instead of 1,2,3 we wouldn't need to do this
> d1$B = factor(d1$B)
```

Redo the ANOVA from the previous session (handout R4), and check sure the results are the same.

```
> twoway.ex1 = aov(errors~A*B, data=d1)
```

We don't need to specify the data frame in the model equation, because we pass it as an argument to the function. So R knows where to look to find A and B.

```
> summary(twoway.ex1)
            Df Sum Sq Mean Sq F value  Pr(>F)
A            2    112  56.000  3.0545 0.07208 .
B            1     24  24.000  1.3091 0.26755
A:B          2    144  72.000  3.9273 0.03843 *
Residuals   18    330  18.333
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

II. Polynomial trend on factor A, which has 3 levels.

R can create Helmert, treatment, and polynomial contrasts for you. The contrasts get stored as attributes, or properties, of the factors they are for, so when you run your ANOVA they will get applied automatically.

```
> contrasts(d1$A) = contr.poly(3)        # makes a 3rd order polynomial contrast set for factor A
> contrasts(d1$B) = contr.poly(2)        # makes a 2nd order polynomial contrast set for factor B
```

Take a look at the contrasts:

```
> contrasts(d1$A)
               .L         .Q
1 -7.071068e-01  0.4082483
2 -9.073264e-17 -0.8164966
3  7.071068e-01  0.4082483
> contrasts(d1$B)
          .L
1 -0.7071068
2  0.7071068
```

Not exactly -1, 0, 1 like we're used to, but they do the job.

Now that we've set the contrasts on our factors, run a new ANOVA to test them. When asking for the results with summary( ), include the *split* parameter to tell it to show us the 2 contrasts on factor A.

```
> twoway.poly = aov(errors ~ A*B, data=d1)    # I named it twoway.poly
> summary(twoway.poly, split=list(A=1:2))     # R knows which data frame you're talking about.
            Df Sum Sq Mean Sq   F value  Pr(>F)
A            2    112  56.000    3.0545 0.07208 .
  A: C1      1    100 100.000    5.4545 0.03129 *   # factor A contrast 1
  A: C2      1     12  12.000    0.6545 0.42906     # factor A contrast 2
B            1     24  24.000    1.3091 0.26755
A:B          2    144  72.000    3.9273 0.03843 *
  A:B: C1    1    144 144.000    7.8545 0.01177 *   # factor A contrast 1 on AxB interaction
  A:B: C2    1      0   0.000 1.321e-31 1.00000     # factor A contrast 2 on AxB interaction
Residuals   18    330  18.333
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results show that the Linear contrast on the main effect of A is significant, F(1,18)=5.4545, p=.03129 , and also that the linear contrast in A has a significant interaction with factor B, F(1,18)=7.8545, p=.01177 .

You can test the polynomial contrast on factor B by typing the following, although since B only has 2 levels it isn't very interesting:

```
> summary(twoway.poly, split=list(B=1))
```

Do some hand-calculation to verify that our usual linear coefficients –1,0,1 yield SS=100 for A-linear, and that our usual quadratic coeff's yield SS=12.

Let's switch the A contrasts to use the ones we do in class, and see what we get.

```
> Alin = c(-1,0,1)
> Aquad = c(1,-2,1)
> contrasts(d1$A) = cbind(Alin, Aquad) # "column bind"; puts vectors together as columns
```

```
> contrasts(d1$A)
  Alin Aquad
1   -1     1
2    0    -2
3    1     1
> summary(twoway.poly2, split=list(A=1:2) )
            Df Sum Sq Mean Sq   F value  Pr(>F)
A            2    112  56.000    3.0545 0.07208 .
  A: C1      1    100 100.000    5.4545 0.03129 *
  A: C2      1     12  12.000    0.6545 0.42906
B            1     24  24.000    1.3091 0.26755
A:B          2    144  72.000    3.9273 0.03843 *
  A:B: C1    1    144 144.000    7.8545 0.01177 *
  A:B: C2    1      0   0.000 5.808e-31 1.00000
Residuals   18    330  18.333
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Warning about entering your own contrast coefficients!!
Make sure you have mutually orthogonal contrasts! To be safe, it is probably better to let R generate polynomial coefficients for you.

Thankfully, the results don't change.

Get the parameter estimates for the trends by using 'lm'. Squaring the t's below gives the F's in the table above.

```
> lmout = lm(errors ~ A*B, data=d1)
> summary(lmout)
Call:
lm(formula = errors ~ A * B, data = d1)

Residuals:
       Min         1Q     Median         3Q        Max
-6.000e+00 -3.000e+00 -2.165e-15  3.250e+00  6.000e+00

Coefficients:
             Estimate Std. Error   t value Pr(>|t|)
(Intercept)  1.000e+01  8.740e-01    11.442 1.08e-09 ***
AAlin        2.500e+00  1.070e+00     2.335   0.0313 *
AAquad      -5.000e-01  6.180e-01    -0.809   0.4291
B1           1.000e+00  8.740e-01     1.144   0.2675
AAlin:B1    -3.000e+00  1.070e+00    -2.803   0.0118 *
AAquad:B1   -4.710e-16  6.180e-01 -7.62e-16   1.0000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.282 on 18 degrees of freedom
Multiple R-squared: 0.459,      Adjusted R-squared: 0.3087
F-statistic: 3.055 on 5 and 18 DF,  p-value: 0.03610
```

The lm( ) function is really nothing new- every time you use aov( ), R actually does the analysis with lm( ) and then shows it to you in an ANOVA friendly format. We can get easily get *lmout* back into our familiar ANOVA format:

```
> anova(lmout)
Analysis of Variance Table

Response: errors
          Df Sum Sq Mean Sq F value  Pr(>F)
A          2    112  56.000  3.0545 0.07208 .
B          1     24  24.000  1.3091 0.26755
A:B        2    144  72.000  3.9273 0.03843 *
Residuals 18    330  18.333
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

III. Interaction contrast with more than 2 levels of B

Suppose we had 3 levels of factor B, and we wanted to see an interaction contrast, something like A-linear x B-linear? We would use something like the following command.

```
> summary(twoway.poly2, split=list( A=list(AL=1,AQ=2), B=list(BL=1) ) )
              Df Sum Sq Mean Sq   F value  Pr(>F)
A              2    112  56.000    3.0545 0.07208 .
  A: AL        1    100 100.000    5.4545 0.03129 *
  A: AQ        1     12  12.000    0.6545 0.42906
B              1     24  24.000    1.3091 0.26755
  B: BL        1     24  24.000    1.3091 0.26755
A:B            2    144  72.000    3.9273 0.03843 *
  A:B: AL.BL   1    144 144.000    7.8545 0.01177 *
  A:B: AQ.BL   1      0   0.000 5.808e-31 1.00000
Residuals     18    330  18.333
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Of course for this example the test doesn't mean a whole lot, because we only have 2 levels of factor B. But now you know how to test for interaction contrasts.

IV. Use Helmert contrasts to compare other treatments to control

Now Suppose that one of the 3 levels of A is a control treatment. The first Helmert contrast, 2, -1, -1, can be used to compare the other two levels of A to the control condition.

```
> contrasts(d1$A) = contr.helmert      # tell R to generate the contrasts
> contrasts(d1$A)                      # tell R to show them to you
  [,1] [,2]
1   -1   -1
2    1   -1
3    0    2
```

Notice that the automatically created contrasts are treating the final level as the control. Make sure that is correct for your data, or else manually enter the contrasts as shown above.
It's also possible to re-order an ordered factor. See R help for 'relevel' and 'reorder', as well as 'C'.

Do the ANOVA and look at the summary. Make sure to include a *split* parameter.

```
> twoway.helm1 = aov(errors~A*B, data=d1)
> summary(twoway.helm1, intercept=T, split=list(A=1:2) )
             Df Sum Sq Mean Sq  F value    Pr(>F)
(Intercept)  1   2400 2400.00 130.9091 1.083e-09 ***
A            2    112   56.00   3.0545   0.07208 .
  A: C1      1     64   64.00   3.4909   0.07807 .
  A: C2      1     48   48.00   2.6182   0.12303
B            1     24   24.00   1.3091   0.26755
A:B          2    144   72.00   3.9273   0.03843 *
  A:B: C1    1     36   36.00   1.9636   0.17813
  A:B: C2    1    108  108.00   5.8909   0.02594 *
Residuals   18    330   18.33
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

A:C2 is the contrast (-1, -1, 2). This contrast compares the 3rd level of A against the first two levels of A. It isn't significant.

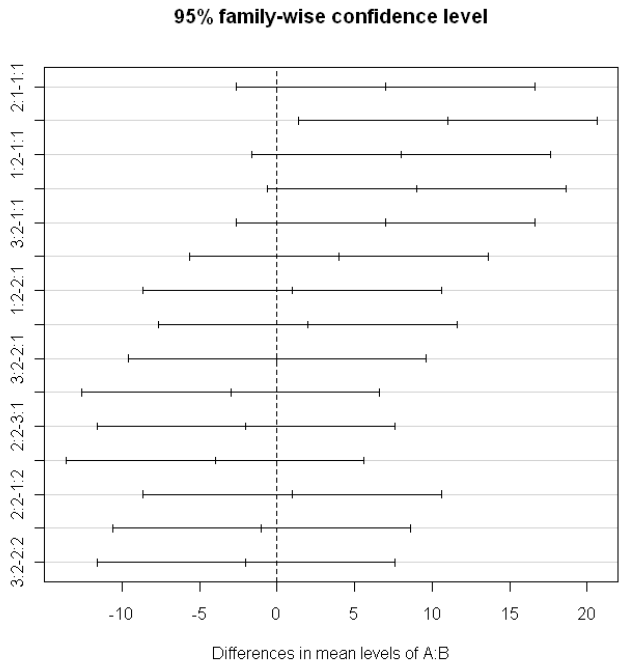
V. **Multiple comparisons** (or 'post-hoc' tests) in two-way randomized designs

A. Tukey HSD tests for all possible pairwise comparisons of cell means and for each main effect.

```
> TukeyHSD(twoway.helm1, conf.level=.95)      # don't forget that case matters!
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = errors ~ A * B, data = d1)

$A
    diff        lwr       upr     p adj
2-1    4 -1.4638549  9.463855 0.1765920
3-1    5 -0.4638549 10.463855 0.0762126
3-2    1 -4.4638549  6.463855 0.8874232

$B
    diff       lwr      upr     p adj
2-1    2 -1.672443 5.672443 0.2675471

$`A:B`
         diff         lwr       upr     p adj
2:1-1:1    7  -2.6219785 16.621979 0.2395485
3:1-1:1   11   1.3780215 20.621979 0.0198670
1:2-1:1    8  -1.6219785 17.621979 0.1372669
2:2-1:1    9  -0.6219785 18.621979 0.0745666
3:2-1:1    7  -2.6219785 16.621979 0.2395485
3:1-2:1    4  -5.6219785 13.621979 0.7700940
1:2-2:1    1  -8.6219785 10.621979 0.9993809
2:2-2:1    2  -7.6219785 11.621979 0.9841453
3:2-2:1    0  -9.6219785  9.621979 1.0000000
1:2-3:1   -3 -12.6219785  6.621979 0.9149689
2:2-3:1   -2 -11.6219785  7.621979 0.9841453
3:2-3:1   -4 -13.6219785  5.621979 0.7700940
2:2-1:2    1  -8.6219785 10.621979 0.9993809
3:2-1:2   -1 -10.6219785  8.621979 0.9993809
3:2-2:2   -2 -11.6219785  7.621979 0.9841453
```

Plot it. Much easier on the eyes.

```
> plot(TukeyHSD(twoway.helm1))
> abline(v=0,h=0)  # add vertical line at  x=0
    #more recent versions of R do this automatically
```



B. Other methods for pairwise tests can be done on just main effects, or on cell means.

Do pairwise tests on cell means for factor A, adjusting p-values with the Holm method.
So group our DV, *errors*, by factor A

```
> pairwise.t.test(d1$errors, d1$A, p.adjust.method="holm")

        Pairwise comparisons using t tests with pooled SD

data:  d1$errors and d1$A

  1    2
2 0.23 -
3 0.16 0.69

P value adjustment method: holm
```

Do pairwise tests on all cell means.
This time group *errors* by each term in the AxB interaction.

```
> pairwise.t.test(d1$errors, d1$A:d1$B)

        Pairwise comparisons using t tests with pooled SD

data:  d1$errors and d1$A:d1$B

    1:1   1:2   2:1   2:2   3:1
1:2 0.215 -     -     -     -
2:1 0.394 1.000 -     -     -
2:2 0.114 1.000 1.000 -     -
3:1 0.029 1.000 1.000 1.000 -
3:2 0.394 1.000 1.000 1.000 1.000

P value adjustment method: holm   # note that this is the default method
```

In this example, cell 1:1 differs significantly from cell 3:1, by both Tukey and Holm. The Tukey p-level is smaller, implying higher power.

Do pairwise tests on all cell means again, this time using the Bonferroni method to adjust the p-values.

```
> pairwise.t.test(d1$errors, d1$A:d1$B, p.adjust.method="bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  d1$errors and d1$A:d1$B

    1:1   1:2   2:1   2:2   3:1
1:2 0.248 -     -     -     -
2:1 0.492 1.000 -     -     -
2:2 0.122 1.000 1.000 -     -
3:1 0.029 1.000 1.000 1.000 -
3:2 0.492 1.000 1.000 1.000 1.000

P value adjustment method: bonferroni
```

Because the 1:1 vs 3:1 difference is the largest, Holm and Bonferroni adjustments are equivalent for that test. Note that the Holm adjustment has slightly lower p-values for the others that are not 1.0.