Psych 610 Handout – R3 Colleen Moore & Mike Amato
Post-Hoc Tests in One-Way Between Group Designs University of Wisconsin - Madison

```
This handout uses data from Myers, Fundamentals of Experimental Design, 2nd Ed,
p. 109. The study has 6 groups. The data are on the course website in an excel
file named "MyersDataOneWay.xls".
```

1. Bring in the Data and look at descriptives

You have multiple options for getting your data into R.

```
> your.data = read.table(file.choose(), header=T)   # navigate to your data file and click it
```
or
```
> your.data = read.table(pipe("pbpaste"),header=T)  #paste from clipboard on a mac only
```

Take your pick, then check to make sure it read in properly by typing the name of the data frame you just created.

```
> your.data
    y group
1   7     1
2  33     1
3  26     1
4  27     1
5  21     1
6   6     1
7  14     1
8  19     1
9   6     2
10 11     2
11 11     2
12 18     2
13 14     2
14 18     2
15 19     2
16 14     2
17  9     3
18 12     3
19  6     3
20 24     3
21  7     3
22 10     3
23  1     3
 ...<rows continue>...
```

```
...<continued>...
24 10     3
25 42     4
26 25     4
27  8     4
28 28     4
29 30     4
30 22     4
31 17     4
32 32     4
33 28     5
34  6     5
35  1     5
36 15     5
37  9     5
38 15     5
39  2     5
40 37     5
41 13     6
42 18     6
43 23     6
44  1     6
45  3     6
46  4     6
47  6     6
48  2     6
```

Attach your data, and tell R that *group* is a categorical factor.

```
> attach(your.data)
> group = as.factor(group)
```

Calculate the group means, and store them in vector *mm*

```
> mm = tapply(y, group, mean)
> mm
     1      2      3      4      5      6
19.125 13.875  9.875 25.500 14.125  8.750
```
.

Use *tapply( )* to make two more vectors for the group lengths and standard deviations. Then tell R to display those vectors. And do it all with one line of code.

semicolon separates commands

```
> nn=tapply(y,group,length); sdsd=tapply(y,group,sd); nn; sdsd
1 2 3 4 5 6
8 8 8 8 8 8
        1         2         3         4         5         6
 9.642725  4.454131  6.621124 10.226017 12.699128  8.241879
```

2. Do the Anova

Are the means reliably different across the groups? Find out!

```
> aov.ex1 = aov(y~group)                    #calculate the anova
> summary(aov.ex1, intercept=T)             #show the anova summary table
            Df   Sum Sq  Mean Sq   F value     Pr(>F)
(Intercept)  1  11102.1  11102.1  135.8251  9.586e-15 ***
group        5   1554.9    311.0    3.8046   0.006228 **
Residuals   42   3433.0     81.7
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Yes! We have a significant treatment effect.
At this point it would be a good idea to graph the means, do the box plots, graph the residuals, and do other things described in the "Making Friends with Your Data" R handout.

3. Pairwise tests of mean differences

3A. Run unadjusted pair-wise t-tests for all the groups. The default setting in R for this test is to adjust p-levels as a post-hoc using the Holm method, so to get un-adjusted p-levels for this exercise you need to tell it not to do that.

```
> pairwise.t.test(y, group, p.adjust="none", pool.sd = T)

   Pairwise comparisons using t tests with pooled SD

data:  y and group

  1       2       3       4       5

2 0.25204 -       -       -       -
3 0.04702 0.38127 -       -       -
4 0.16583 0.01375 0.00127 -       -
5 0.27499 0.95616 0.35251 0.01577 -
6 0.02679 0.26334 0.80468 0.00061
0.24110

P value adjustment method: none
```

`p.adjust="none"`	The default in R is to adjust p-values with the Holm method. For this exercise you need to tell it not to do that.
`pool.sd = T`	This is unnecessary, since R’s default is to assume homogeneity of variance. If that assumption weren’t true for your data, you could use *pool.sd = F*. Try that out and see what happens.

If we don’t make any adjustments to our p-values, we find that 6 of the 15 differences appear significant. That would be some pretty shady statistics though.

3B. Now control the family-wise Type I error by adjusting the p-values using various techniques.

```
> pairwise.t.test(y, group, p.adjust="bonferroni", pool.sd = T)

   Pairwise comparisons using t tests with pooled SD

data:  y and group

  1      2      3      4      5
2 1.0000 -      -      -      -
3 0.7054 1.0000 -      -      -
4 1.0000 0.2063 0.0190 -      -
5 1.0000 1.0000 1.0000 0.2365 -
6 0.4018 1.0000 1.0000 0.0092 1.0000

P value adjustment method: bonferroni
```

Making the conservative Bonferroni adjustment to 15 pairwise comparisons, only two differences remain significant. If we do it again using Holm's method, we can see that Holm's method is more powerful than Bonferroni's.

```
> pairwise.t.test(y, group, p.adjust="holm", pool.sd = T)

   Pairwise comparisons using t tests with pooled SD

data:  y and group

  1      2      3      4      5
2 1.0000 -      -      -      -
3 0.4702 1.0000 -      -      -
4 1.0000 0.1788 0.0177 -      -
5 1.0000 1.0000 1.0000 0.1892 -
6 0.2947 1.0000 1.0000 0.0092 1.0000

P value adjustment method: holm
```

Since we are using the default parameters for this test, you would get the same result with:
`pairwise.t.test(y,group)`

With the Holm method the same two are significant, but their adjusted p-values are smaller.

The "BH" and "BY" methods are for adjusting for the "False Discovery Rate". It is not a true control of family-wise error, but something different. BH stands for Benjamini & Hochberg, and BY is for Benjamini & Yekutieli. See the references in the R help section on "p.adjust". Let's see how they come out.

```
> pairwise.t.test(y, group, p.adjust="BH", pool.sd = T)

   Pairwise comparisons using t tests with pooled SD

data:  y and group

  1      2      3      4      5
2 0.3750 -      -      -      -
3 0.1176 0.4399 -      -      -
4 0.3553 0.0591 0.0095 -      -
5 0.3750 0.9562 0.4399 0.0591 -
6 0.0804 0.3750 0.8622 0.0092 0.3750

P value adjustment method: BH
```

```
> pairwise.t.test(y, group, p.adjust="BY", pool.sd = T)

   Pairwise comparisons using t tests with pooled SD

data:  y and group

  1     2     3     4     5
2 1.000 -     -     -     -
3 0.390 1.000 -     -     -
4 1.000 0.196 0.031 -     -
5 1.000 1.000 1.000 0.196 -
6 0.267 1.000 1.000 0.030 1.000

P value adjustment method: BY
```

Based on the False Discovery Rate methods, we still find the same pair of means meeting the .05 criterion. However, the BH method suggests that other pairs are possibly showing a trend toward a significant difference.

3C. Tukey tests on all possible pairs

Remember how up in step 2 we first calculated the ANOVA and called it “aov.ex1”, then in a separate step asked R to show us a summary of aov.ex1? Well the way you use the TukeyHSD( ) function is similar to the summary function. It takes the variable from the original ANOVA calculation as one of its arguments.

```
> TukeyHSD(aov.ex1, conf.level=.95)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = y ~ group)

$group
       diff        lwr       upr     p adj
2-1  -5.250 -18.744686  8.244686 0.8522904
3-1  -9.250 -22.744686  4.244686 0.3346360
4-1   6.375  -7.119686 19.869686 0.7207042
5-1  -5.000 -18.494686  8.494686 0.8761170
6-1 -10.375 -23.869686  3.119686 0.2188268
3-2  -4.000 -17.494686  9.494686 0.9480751
4-2  11.625  -1.869686 25.119686 0.1271135
5-2   0.250 -13.244686 13.744686 0.9999999
6-2  -5.125 -18.619686  8.369686 0.8644848
4-3  15.625   2.130314 29.119686 0.0149617
5-3   4.250  -9.244686 17.744686 0.9336134
6-3  -1.125 -14.619686 12.369686 0.9998597
5-4 -11.375 -24.869686  2.119686 0.1424939
6-4 -16.750 -30.244686 -3.255314 0.0075302
6-5  -5.375 -18.869686  8.119686 0.8395496
```

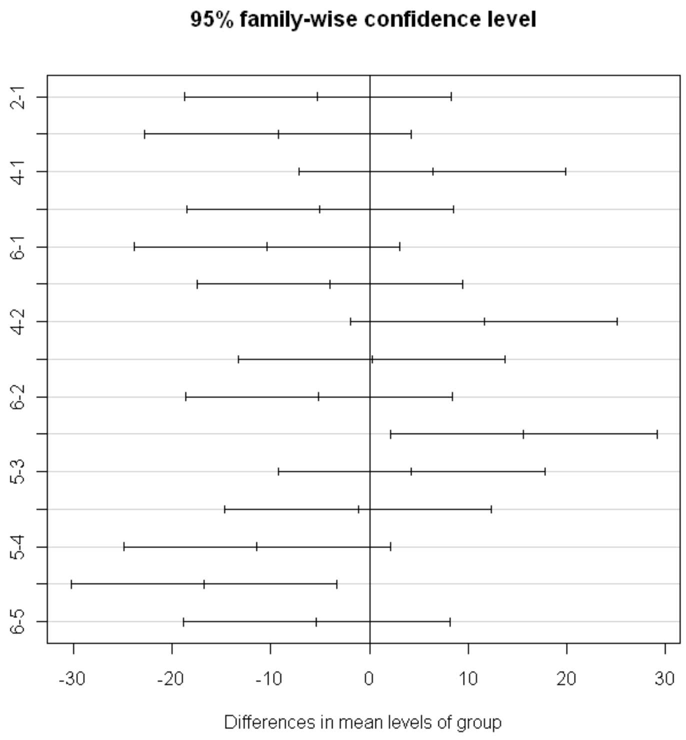
Instead of using the variable “aov.ex1” we could just embed the calculation directly:

> TukeyHSD(aov(y~group), conf.level=.95)

This will give you all possible pairwise tests and the p values, and 95% confidence intervals. The default confidence level is 95%, but we entered it anyway. You could easily change it to some other value. Compare the p-values to the Holm and Bonferroni adjusted p's. The smaller the adjusted p, the better the power. For this example only the same two pairs are showing up as significantly different, regardless of the post-hoc correction we use.

The TukeyHSD function additionally gives us the confidence interval upper and lower bounds, and the difference between the sample means. You can plot the Tukey confidence intervals:

```
> plot(TukeyHSD(aov(y~group), conf.level=.95))
```



Replace that junky looking dashed vertical line at x=0 by drawing a snazzy solid vertical line:
> abline(v=0)

4. Calculate Tukey and Scheffe tests by programming the commands yourself.

This is kind of fun because we use R's built-in distributions to find the F's and q's that we need. We can get all the information we need from the summary of the overall ANOVA.

```
> summary(aov.ex1)
            Df Sum Sq Mean Sq F value   Pr(>F)
group        5 1554.9   311.0  3.8046 0.006228 **
Residuals   42 3433.0    81.7
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

4A. Tukey method programmed

Do a pairwise test on group1 vs group2.

```
> numgroups = 6              # tell R how many group means you have
> dferror = 42               # tell R your df for error
> mserror = 81.7             # tell R your MSerror
> coeff = c(1,-1,0,0,0,0)    # enter your contrast coefficients
> psi = sum(coeff*mm)        # calculate the difference between the means (mm is from page1)
> psi
[1] 5.25

# Calculate the standard error of the difference. Take the absolute values of the coeff in order to only add in the
two means we want to find the difference for.
> sediff = sqrt( (mserror/2) * sum((1/nn)*abs(coeff)) )
> sediff
[1] 3.1957

#  Calculate q for your test, the studentized range statistic
> calcq = psi/sediff
> calcq
[1] 1.642832
```

Compare the value of calcq to the table value of the studentized range statistic for 6 groups and dferror = 42.

```
> qtukey(.95, numgroups, dferror)
[1] 4.221779
```

Is 1.642832 greater than or equal to 4.221779? No, it isn’t, so the difference between group1 and group2 is non-significant. We can ask for the probability that calcq is exceeded by chance:

```
> ptukey(calcq,numgroups,dferror,nranges=1,lower.tail=F)
[1] 0.8521682
```

That number should match the p-value from the Tukey test of group 1 vs 2 that R calculated for us earlier (in part 3C). It’s slightly off because we rounded MSerror, but it’s pretty close. We should also expect our hand calculations of the confidence interval to be off by a similar amount:

```
> lowerbound = psi-qtukey(.95,numgroups,dferror)*sediff
> upperbound = psi+qtukey(.95,numgroups,dferror)*sediff
> lowerbound; upperbound
[1] -8.241541
[1] 18.74154
```

Yup, looks like a rounding issue. Note also that the signs are flipped because of the contrast we used. No big deal.

4B. Scheffe method programmed

Remember, Scheffe is best to use for complex contrasts, or when you are doing a lot of contrasts. I have yet to find a Scheffe method that anyone else has built for us in R.

```
> dfnum=5; dferror=42; mserror=81.74    # enter values of df and mserror
> coeff=c(5,-1,-1,-1,-1,-1)             # this contrast compares the first group to the other 5
> psi=sum(coeff*mm); psi                # calculate psi
[1] 23.5
> sscoeff=sum(coeff*coeff/nn)           # calculate the sum of the squared contrast coefficients
> mspsi=(psi*psi)/sscoeff; mspsi        # find the mean squared for the contrast
[1] 147.2667
> contrastF=mspsi/mserror; contrastF    # calculate the F for the contrast
[1] 1.801648
> tableF=qf(.05,dfnum,dferror,lower.tail=F)       # find the table F
[1] 2.437693
> critvalue=dfnum*tableF; critvalue  # convert tableF to ScheffeF by multiplying by numerator df
[1] 12.18846
```

So comparing the contrast F to the critical value, we find that this contrast is nonsignificant by the Scheffe method.
We can make Scheffe confidence intervals:

```
> lowbound=psi-sqrt(critvalue)*sqrt(mserror*sscoeff)
> upperbound=psi+ sqrt(critvalue)*sqrt(mserror*sscoeff)
> confinterval=c(lowbound,upperbound); confinterval
[1] -37.62339  84.62339                    # this would be useful for a graph
```

4C. Fisher-Hayter method

Similar to the Tukey contrast above, just find the qtukey and ptukey with dfnum instead of number of groups:

```
>  qtukey(.95,dfnum,dferror)
[1] 4.030232
> ptukey(calcq,dfnum,dferror,nranges=1,lower.tail=F)
[1] 0.77272
```

No matter how we slice it, there is just no significant difference between group1 and group2.