Tutorials
by William B. King, Ph.D.
Coastal Carolina University

I think, therefore I R.

# FACTORIAL BETWEEN SUBJECTS ANOVA

## Preliminaries

*Model Formulae*

You would probably profit from reading the Model Formulae tutorial before going on.

*Complexities*

Anyone who had a graduate school course in analysis of variance is probably still having flashbacks. Experimental designs can become extremely complex, with complications like nesting, blocking, split-plot, Latin square, etc. In this short tutorial, I will stick to the basics, which is to say I will stick to factorial designs (which involve replication in the design cells) that are completely randomized. The more complex experimental designs are handled in R for the most part by specifying the correct error structure in the model formula. There will be a taste of this in the tutorial on repeated measures.

While we're on the topic of complexity, I should add a comment about keeping experimental designs balanced (i.e., equal numbers of subjects in each cell of the design). In single-factor designs, this is not so crucial, as unbalancing the design has the primary effect of making the ANOVA more sensitive to violation of assumptions. However, when a factorial design becomes unbalanced, either because of subject attrition or because the study is observational in nature and subjects were "taken as they came," then the factors lose their independence from one another. It then makes a difference in what order the factors are entered into the model formula. One suggestion is to try entering the factors in different orders to see how much of a difference this makes to the outcome of the analysis. And that is the extent to which these problems will be discussed in this tutorial!

## Summarizing Factorial Data

For this tuturial, we will examine data on the effect of vitamin C on tooth growth in guinea pigs...

```
> data(ToothGrowth)
> str(ToothGrowth)
'data.frame':   60 obs. of  3 variables:
 $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

The variables are "len", which is tooth length, the response variable, the way in which the supplement was administered, "supp", of which the levels are "OJ" (as orange juice) and "VC" (as ascorbic acid), and "dose" of the supplement in milligrams (0.5, 1.0, and 2.0). To begin, we should note that R considers "dose" to be numerical. We will have to be absolutely certain it is treated as a factor when we construct the model formula. Otherwise, we will end up with an analysis of covariance, which is not what we want.

First, let's end all doubt that we want "dose" to be a factor...

```
> ToothGrowth$dose = factor(ToothGrowth$dose, levels=c(0.5,1.0,2.0),
+                                              labels=c("low","med","high"))
> str(ToothGrowth)
'data.frame':   60 obs. of  3 variables:
 $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: Factor w/ 3 levels "low","med","high": 1 1 1 1 1 1 1 1 1 1 ...
> ToothGrowth[seq(1,60,5),]
    len supp dose
1   4.2   VC  low
6  10.0   VC  low
11 16.5   VC  med
16 17.3   VC  med
21 23.6   VC high
26 32.5   VC high
31 15.2   OJ  low
36 10.0   OJ  low
41 19.7   OJ  med
46 25.2   OJ  med
51 25.5   OJ high
56 30.9   OJ high
```

The function factor( ) is used to declare "ToothGrowth$dose" as a factor, with levels currently coded as 0.5, 1.0, and 2.0, but now recoded with labels of "low", "med", and "high". Just to be sure, I looked at str( ) and a bit of the data frame itself. (This was as much for me as it was for you!)

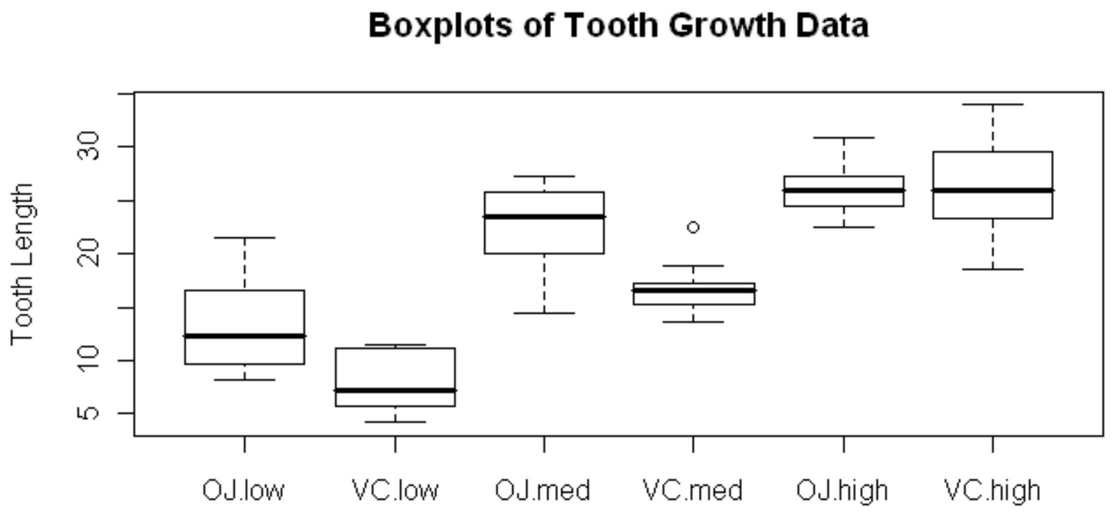
Now let's check for a balanced design using the replications( ) function...

```
> replications(len ~ supp * dose, data=ToothGrowth)
     supp      dose supp:dose
       30        20        10
> replications(len ~ supp * dose, data=ToothGrowth[1:58,])
$supp
supp
OJ VC
28 30

$dose
dose
 low  med high
  20   20   18

$supp:dose
    dose
supp low med high
  OJ  10  10    8
  VC  10  10   10
```

The fact that we got such a straightforward output when the function was run the first time tells us the design is balanced. To illustrate what would happen if the design were unbalanced, I ran the function a second time on a large subset of the data frame (dropping out the last two cases). In that example, we got complete tables of all the factors and how many replications (subjects) there were in each cell of the design. (By the way, R documentation people, this is another one I had to figure out by trial and error. The help page could use a more straightforward example!)

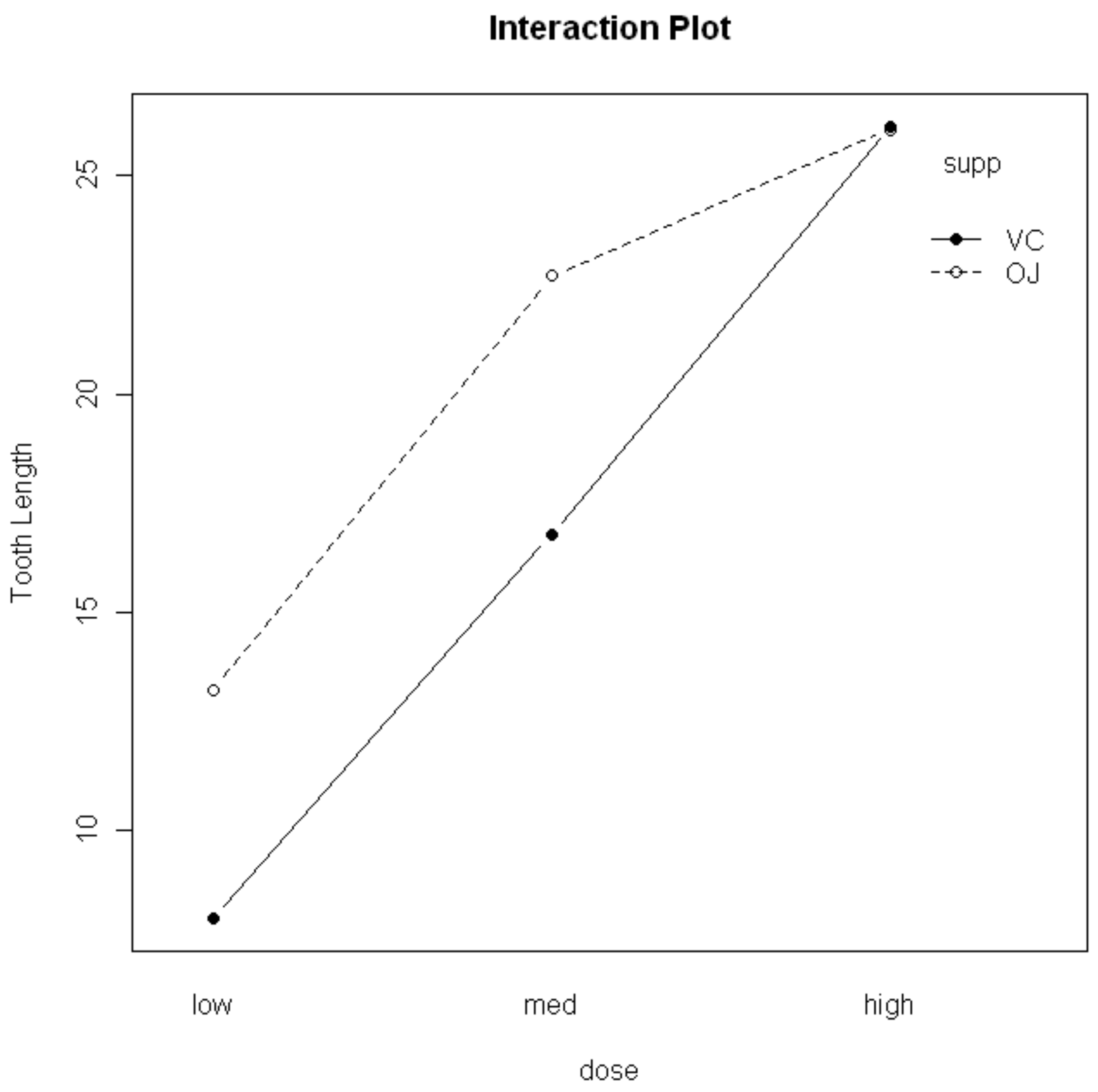
There are a number of ways the data can be summarized graphically. First, let's look at boxplots...

```
> boxplot(len ~ supp * dose, data=ToothGrowth,
+         ylab="Tooth Length", main="Boxplots of Tooth Growth Data")
```



We should make a note to check our homogeneity and normality assumptions! (How to do this is covered in other tutorials.) In the meantime, we might get a better idea of what our main effects and interactions look like if we look at a traditional interaction plot...

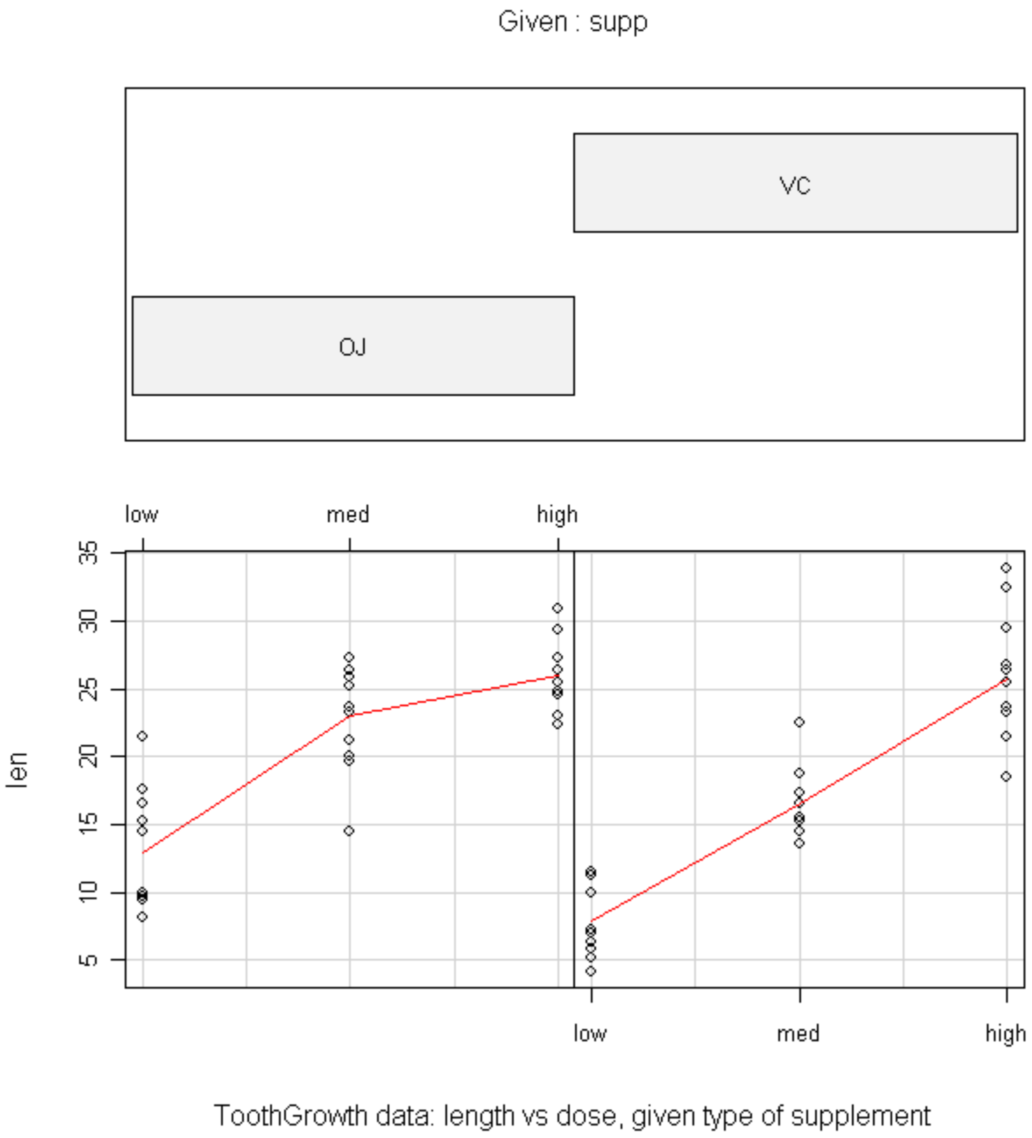
```
> with(ToothGrowth, interaction.plot(x.factor=dose, trace.factor=supp,
+                     response=len, fun=mean, type="b", legend=T,
+                     ylab="Tooth Length", main="Interaction Plot",
+                     pch=c(1,19)))
```



The syntax is straightforward, but there are a lot of options. First, "x.factor=" specifies the factor to be plotted on the horizontal axis. Second, "trace.factor=" specifies the factor to plotted as individual lines on the graph. Third, "response=" specifies the response variable. Fourth, "fun=" specifies the function to be calculated on the response variable. "Mean" is the default, so we could have left this one out. Fifth, "type=" specifies whether to plot lines, points, or both (we plotted "b"oth). We also asked for a legend, a label on the y-axis, and a main title on the graph. Finally, we specified the point characters to use. We could also have changed the line type, line color, and just about everything else about the graph.

Another type of graphic we could look at here is called a conditioning plot. I shall steal, er, ah, I mean, borrow the example right off the help page for the ToothGrowth data set...

```
> coplot(len ~ dose | supp, data=ToothGrowth, panel=panel.smooth,
+       xlab="ToothGrowth data: length vs dose, given type of supplement")
```



The formula is the tricky part here and should be read as "length by dose given supplement." The "panel=" option draws the red lines when set to "panel.smooth". The top panel can be suppressed by setting "show.given=" to FALSE. This gives a much more sensible looking plot in my humble opinion.

Numerical summaries of the data can be done with the tapply( ) function...

```
> with(ToothGrowth, tapply(len, list(supp,dose), mean))
     low   med  high
OJ 13.23 22.70 26.06
VC  7.98 16.77 26.14
> with(ToothGrowth, tapply(len, list(supp,dose), var))
      low       med      high
OJ 19.889 15.295556  7.049333
VC  7.544  6.326778 23.018222
```

Remember, in tapply( ) the first argument is the response variable, the second argument is a list of factors by which the response should be partitioned, and the third argument is the function you wish to calculate. Once again, we make note of some consternation over those cell variances.

Another way to get similar information is to use the model.tables( ) function, but the ANOVA has to be run first and saved into a model object...

```
> aov.out = aov(len ~ supp * dose, data=ToothGrowth)
```

Note: the model formula specifies "length as a function of supplement and dose with all possible interactions between the factors. Now we can look at some model tables...

```
> model.tables(aov.out, type="means", se=T)
Tables of means
Grand mean

18.81333

 supp
supp
    OJ     VC
20.663 16.963

 dose
dose
   low    med   high
10.605 19.735 26.100

 supp:dose
    dose
supp low   med   high
  OJ 13.23 22.70 26.06
  VC  7.98 16.77 26.14

Standard errors for differences of means
          supp   dose supp:dose
        0.9376 1.1484    1.6240
replic.     30     20        10
```

We have requested the various cell and marginal means along with standard errors for the differences between means. Finally, let's look at a Bartlett test for homogeneity of variance...

```
> bartlett.test(len ~ supp * dose, data=ToothGrowth)

        Bartlett test of homogeneity of variances

data:  len by supp by dose
Bartlett's K-squared = 1.4217, df = 1, p-value = 0.2331
```

It appears we have dodged the nonhomogeneity bullet!

---

**Factorial ANOVA**

We've already run the ANOVA above when we wanted to look at the model tables, but just to remind you, the function was...

```
> aov.out = aov(len ~ supp * dose, data=ToothGrowth)
```

You do not have to execute it again if you've already done it. This creates a model data object in your workspace (which we've named "aov.out") that contains a wealth of information. The extension to cases with more factors, for a completely randomized design where no Error structure needs to be specified, would be straightforward. Just list all the factors after the tilde separated by asterisks. This gives a model with all possible main effects and interactions. To leave out interactions, separate the factor names with plus signs rather than asterisks.

Before we print anything to the Console, I'm going to turn off printing of significance stars, because, frankly, I find them annoying...

```
> options(show.signif.stars=F)
```

Now we can look at the ANOVA table using either summary( ) or summary.aov( )...

```
> summary(aov.out)
            Df  Sum Sq Mean Sq F value    Pr(>F)
supp         1  205.35  205.35  15.572 0.0002312
dose         2 2426.43 1213.22  92.000 < 2.2e-16
supp:dose    2  108.32   54.16   4.107 0.0218603
Residuals   54  712.11   13.19
```

We see two significant main effects and a significant interaction. In R lingo, two or more factor names separated by colons refers to the interaction between those factors. For post hoc tests, we have the traditional Tukey HSD test...

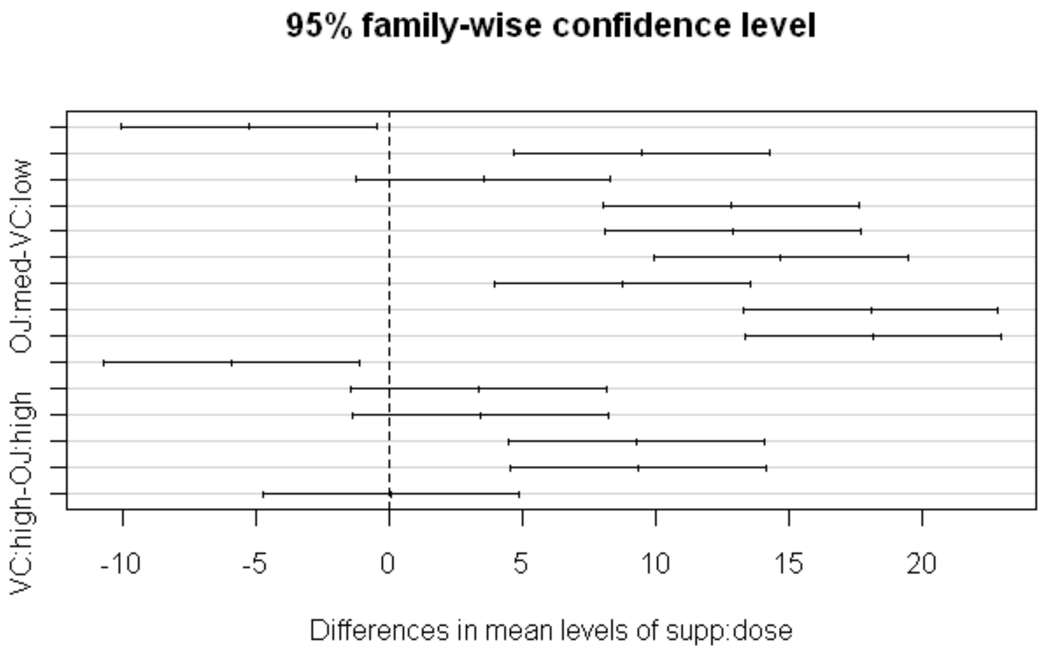
```
> TukeyHSD(aov.out)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = len ~ supp * dose, data = ToothGrowth)

$supp
      diff       lwr       upr     p adj
VC-OJ -3.7 -5.579828 -1.820172 0.0002312

$dose
           diff       lwr       upr   p adj
med-low   9.130  6.362488 11.897512 0.0e+00
high-low 15.495 12.727488 18.262512 0.0e+00
high-med  6.365  3.597488  9.132512 2.7e-06

$`supp:dose`
                 diff        lwr        upr     p adj
VC:low-OJ:low   -5.25 -10.048124 -0.4518762 0.0242521
OJ:med-OJ:low    9.47   4.671876 14.2681238 0.0000046
VC:med-OJ:low    3.54  -1.258124  8.3381238 0.2640208
OJ:high-OJ:low  12.83   8.031876 17.6281238 0.0000000
VC:high-OJ:low  12.91   8.111876 17.7081238 0.0000000
OJ:med-VC:low   14.72   9.921876 19.5181238 0.0000000
VC:med-VC:low    8.79   3.991876 13.5881238 0.0000210
OJ:high-VC:low  18.08  13.281876 22.8781238 0.0000000
VC:high-VC:low  18.16  13.361876 22.9581238 0.0000000
VC:med-OJ:med   -5.93 -10.728124 -1.1318762 0.0073930
OJ:high-OJ:med   3.36  -1.438124  8.1581238 0.3187361
VC:high-OJ:med   3.44  -1.358124  8.2381238 0.2936430
OJ:high-VC:med   9.29   4.491876 14.0881238 0.0000069
VC:high-VC:med   9.37   4.571876 14.1681238 0.0000058
VC:high-OJ:high  0.08  -4.718124  4.8781238 1.0000000
```

The degree of confidence required for the confidence intervals can be set using the "conf.level=" option. The default value is 0.95. There is also a "which=" option, which can be used to set the factors we wish to compare, in the event we don't want all possible comparisons. Feed it a character vector of factor names...

```
> TukeyHSD(aov.out, which=c("dose"), conf.level=.99)
  Tukey multiple comparisons of means
    99% family-wise confidence level

Fit: aov(formula = len ~ supp * dose, data = ToothGrowth)

$dose
           diff       lwr       upr   p adj
med-low   9.130  5.637681 12.622319 0.0e+00
high-low 15.495 12.002681 18.987319 0.0e+00
high-med  6.365  2.872681  9.857319 2.7e-06
```

You can also plot the confidence intervals from the Tukey test...

```
> plot(TukeyHSD(aov.out))
```



...although the y-labels tend to get smushed out of existence. The 15 interaction CIs are plotted here in the same order they appear in the table above.

If you want pairwise t-tests on any factor, you can have them, with a variety of different methods available for adjusting the p-value for the number of tests being done (see the help page for details)...

```
> with(ToothGrowth, pairwise.t.test(len, dose, p.adjust.method="bonferroni"))

        Pairwise comparisons using t tests with pooled SD

data:  len and dose

     low     med
```

```
med  2.0e-08 -
high 4.4e-16 4.3e-05

P value adjustment method: bonferroni
```

Enter the response variable as the first argument followed by a comma and then the grouping variable or factor. Methods available for adjusting the p-values are holm (default), hochberg, hommel, bonferonni, BH, BY, fdr, and none. (And don't for a moment think I understand what all those are!) An optional package called "multcomp" is available that implements additional post hoc multiple comparison tests.

---

## Treatment Contrasts

Treatment contrasts are explained more fully in the Oneway ANOVA tutorial. Briefly...

```
> options("contrasts")       # check to make sure we're getting treatment contrasts
$contrasts
        unordered           ordered
"contr.treatment"      "contr.poly"

> summary.lm(aov.out)

Call:
aov(formula = len ~ supp * dose, data = ToothGrowth)

Residuals:
   Min     1Q Median     3Q    Max
 -8.20  -2.72  -0.27   2.65   8.27

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       13.230      1.148  11.521 3.60e-16
suppVC            -5.250      1.624  -3.233  0.00209
dosemed            9.470      1.624   5.831 3.18e-07
dosehigh          12.830      1.624   7.900 1.43e-10
suppVC:dosemed    -0.680      2.297  -0.296  0.76831
suppVC:dosehigh    5.330      2.297   2.321  0.02411

Residual standard error: 3.631 on 54 degrees of freedom
Multiple R-squared: 0.7937,     Adjusted R-squared: 0.7746
F-statistic: 41.56 on 5 and 54 DF,  p-value: < 2.2e-16
```

Interpretation of this table can be confusing for someone (like me) who is not used to looking at ANOVA output in this form. To see what's here, refer back to the table of cell-by-cell means produced above by tapply( ). The estimated coefficient for the Intercept is the mean in the upper left corner of that table, the mean for the baseline level of "supp" combined with the baseline level of "dose". In other words, this is the mean of the OJ-low cell. We find this mean to be significantly different from zero, not an especially interesting result. The next line of the table, "suppVC", shows the difference between the means of the VC-low cell and the OJ-low cell. The standard error is for the difference between these means, so the difference is found to be significant. The "dosemed" line shows the difference between the means of the OJ-med cell and the OJ-low cell, and once again this difference is found to be statistically significant. The "dosehigh" line shows the difference in means between the OJ-high cell and the OJ-low cell, also significant. The last two lines test various elements of the interaction (good luck finding them). Frankly, I haven't yet found this table very interesting, but it's here if you want it. I don't see anything here that isn't done better by the Tukey HSD output.

Note: summary.lm( ) does not support complex error structures, so don't use it if your model formula has an Error term.

---

**More Stuff**

There are other plots available...

```
> plot(aov.out)                                   # output not shown
> plot.design(len~supp*dose, data=ToothGrowth)    # output not shown
```

The first of these gives a series of diagnostic plots similar to those obtained after a regression analysis (see for further explanation). When R says it's "Waiting to confirm page change..." in the Console, just hit Enter to see the next diagnostic plot. The second plot I don't find particularly useful.

The model object is a list with a whole lot of information in it, much of which we've already seen. To investigate it thoroughly, try this...

```
> aov.out[[1]]
> aov.out[[2]]
> aov.out[[3]]
```

...and so on. There are thirteen items (up to aov.out[[13]]) in the list.

---

**MANOVA**

There is also a function available for multivariate ANOVA when there are multiple response variables: manova( ). The basic syntax is...

```
model = manova(resp.tabl ~ IV1 * IV2 * ...)
summary(model)
summary.manova(model)
summary.aov(model)
```

The "resp.tabl" is a table in which the response variables are arranged in columns of the table. I have not used this function and will not discuss it further.

---

Return to the Table of Contents