**Lecture 3—Wednesday, September 1, 2010**

**Topics**

* [Graphical summaries of ANOVA results](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#graphical)
  + [Graphing a model with a 2-factor interaction and a separate main effect](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#graph1)
  + [Graphing a model with two 2-factor interactions](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#graph2)
* [Multiple regression with both categorical and continuous predictors](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#multiple)
  + [The additive model—analysis of covariance](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#additive)
  + [Interactions between categorical and continuous predictors](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#interactions)
* [R code used in lecture 3](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#Rcode)

**Graphical summaries of ANOVA results**

Last time we considered a three-factorial analysis of variance model. Because we were dealing with experimental data, we started with the full three-factor interaction model and tried to drop the higher order interaction terms that were not statistically significant while adhering to the principle of marginality. The code below recreates the data and refits all of the relevant models from last time.

temp.dat <- read.table("http://www.unc.edu/courses/2010fall/ecol/563/001/data/lecture2/tadpoles.txt", header=T)

prelim<-as.character(temp.dat$var)

treatment<-substr(prelim,6,9)

response<-as.numeric(substr(prelim,10,nchar(prelim)))

treatment1<-factor(treatment)

**#create factors**

fac1<-factor(substr(treatment1,1,2))

fac2<-factor(substr(treatment1,3,3))

fac3<-factor(substr(treatment1,4,4))

**#fit models**

out1<-lm(response~fac1\*fac2\*fac3)

out2<-update(out1,.~.-fac1:fac2:fac3)

out3<-update(out2,.~.-fac1:fac3)

out4<-update(out3,.~.-fac1:fac2)

Although neither the fac1 × fac2 × fac3 three-factor interaction nor the fac1 × fac3 two-factor interaction proved to be statistically significant, our conclusions about the fac1 × fac2 interaction ended up being more ambiguous. If we add the fac2 × fac3 interaction first, then the fac1 × fac2 interaction is not statistically significant (*p* = 0.08).

**#fac2:fac3 interaction added first**

anova(out4)

Analysis of Variance Table

Response: response  
Df Sum Sq Mean Sq F value Pr(>F)   
fac1 2 18.4339 9.2169 151.1291 < 2.2e-16 \*\*\*  
fac2 1 1.5013 1.5013 24.6162 1.350e-06 \*\*\*  
fac3 1 2.2771 2.2771 37.3375 4.137e-09 \*\*\*  
fac2:fac3 1 0.4700 0.4700 7.7069 0.005948 \*\*   
Residuals 233 14.2100 0.0610   
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

**#test fac1:fac2 with fac2:fac3 already in the model**

anova(out4, out3)

Analysis of Variance Table

Model 1: response ~ fac1 + fac2 + fac3 + fac2:fac3  
Model 2: response ~ fac1 + fac2 + fac3 + fac1:fac2 + fac2:fac3  
Res.Df RSS Df Sum of Sq F Pr(>F)   
1 233 14.210   
2 231 13.908 2 0.30173 2.5057 0.08384 .  
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

On the other hand, if we add fac1 × fac2 as the first two-factor interaction, it is statistically significant (*p* = 0.04). Then when the fac2 × fac3 is added to this model, it too is statistically significant (*p* = 0.01).

anova(out3)

Analysis of Variance Table

Response: response  
Df Sum Sq Mean Sq F value Pr(>F)   
fac1 2 18.4339 9.2169 153.0824 < 2.2e-16 \*\*\*  
fac2 1 1.5013 1.5013 24.9343 1.169e-06 \*\*\*  
fac3 1 2.2771 2.2771 37.8201 3.382e-09 \*\*\*  
fac1:fac2 2 0.3926 0.1963 3.2603 0.04015 \*   
fac2:fac3 1 0.3792 0.3792 6.2973 0.01278 \*   
Residuals 231 13.9083 0.0602   
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Given the arbitrariness of the α = .05 cut-off, an argument can be made for either model out3 or out4. In model out3 we have a fac1 × fac2 as well as a fac2 × fac3 interaction so there are no directly interpretable main effects. In model out4 we have a significant fac2 × fac3 interaction as well as a significant main effect due to fac1. I generate a graphical summary of each model in turn.

**Graphing a model with a 2-factor interaction and a separate main effect [model out4]**

Using the R **effects** package we've obtained informative displays of two-factor interactions in the form of mean profile plots. The **effects** package uses the plotting capabilities of the **lattice** package. In the **lattice** package we can specify viewports, arbitrary rectangular regions within the plotting area, in which to produce plots. This makes it possible to place separate plots side-by-side.

In model out4 we have a significant fac2 × fac3 interaction and a significant fac1 main effect. The absence of any interactions with fac1 means that no matter what the value of fac1, the basic fac2 × fac3 interaction remains the same. If we plot the fac1 × fac3 interaction as a profile plot for a fixed value of fac1 and then change the value fac1, the effect is to just move the profile plot up or down. So, one strategy is to plot the basic fac2 × fac3 interaction separately for the three different values of fac1 and then place the plots side-by-side using the same scale for each. This displays the nature of the fac2 × fac3 interaction (it's the same in each plot), as well as the main effect of fac1, indicated by the amount the interaction profile plot is shifted up or down as fac1 is varied.

Specifying a value for a variable such as fac1 that is not participating in the effect of interest, here fac2 × fac3, is done with the **given.values** argument of the **effect** function. To use it we have to provide values for the fac1 regressors using their names as they are given in the regression model output. From the output we see that the dummy regressors for fac1 are labeled fac1No and fac1Ru. These are dummy variables that indicate x4 = No and x4 = Ru respectively.

coef(out4)

(Intercept) fac1No fac1Ru fac2S fac32 fac2S:fac32   
3.35409005 0.55945800 0.62461127 0.06247182 0.09988437 0.18046122

To make the plots I proceed as follows.

**#make first graph: fac1 = Co**

plot(effect('fac2:fac3',out4, given.values=c(fac1No=0, fac1Ru=0)), multiline=T, nrow=1, ncol=3, col=1, more=T)

* **given.values** is an argument to the **effect** function. To get the first level of fac1 each of the dummy regressors is set to 0.
* The **nrow=** and **ncol=** options set up the dimensions of the viewport. In this case I define it to consist of one row and three columns.
* The argument **col=1** specifies that current graph should occupy column 1 of the viewport.
* The **more=T** argument indicates that another graph will follow so the viewport should stay open.

To obtain the second plot corresponding to fac1=No I change the value of the first dummy regressor fac1No from 0 to 1. I also switch the viewport to column 2 with **col=2**.

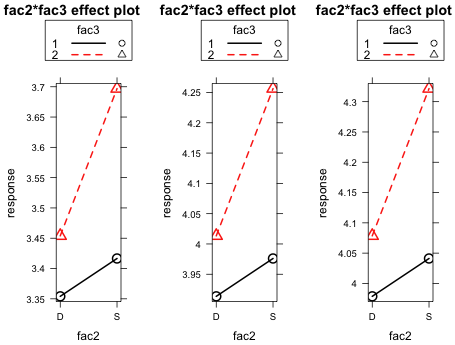
**#make second graph: fac1 = No**

plot(effect('fac2:fac3',out4, given.values=c(fac1No=1, fac1Ru=0)), multiline=T, nrow=1, ncol=3, col=2, more=T)

Finally I make the last graph corresponding to fac1=Ru. For this the dummy variable assignments are **fac1No=0** and **fac1Ru=1**. I plot this graph in column 3 using the argument **col=3**. I also specify **more=F** to close the viewport.

**#make third graph: fac1 = Ru**

plot(effect('fac2:fac3',out4, given.values=c(fac1No=0, fac1Ru=1)), multiline=T, nrow=1, ncol=3, col=3, more=F)



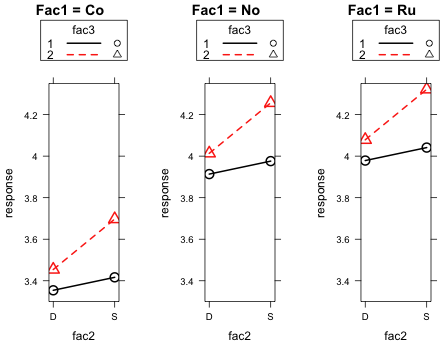
**Fig. 1** Profile plots of the fac2 x fac3 interaction for different values of fac1

The graphs look identical because R has adjusted the scale in each. We need to set the scale ourselves so that the same scale is used in every plot. This can be accomplished by setting the *y*-limits explicitly with the argument **ylim=c(3.3, 4.35)**. Also we to change the title of the graphs so that they indicate the level of fac1 being used. The title of graphs is set with the **main=** argument. I redo the graph with these changes.

plot(effect('fac2:fac3',out4, given.values=c(fac1No=0, fac1Ru=0)), multiline=T, nrow=1, ncol=3, col=1, more=T, ylim=c(3.3, 4.35), main='Fac1 = Co')

plot(effect('fac2:fac3',out4, given.values=c(fac1No=1, fac1Ru=0)), multiline=T, nrow=1, ncol=3, col=2, more=T, ylim=c(3.3, 4.35), main='Fac1 = No')

plot(effect('fac2:fac3',out4, given.values=c(fac1No=0, fac1Ru=1)), multiline=T, nrow=1, ncol=3, col=3, more=F, ylim=c(3.3, 4.35), main='Fac1 = Ru')



**Fig. 2** Profile plots of the fac2 x fac3 interaction for different values of fac1

In Fig. 2 the nature of the two-factor interaction is clear. The response increases when we switch from fac2 = D to fac2 = S, but the increase is greater when fac3 is at level 2 than when it is at level 1. In addition we can see the main effect of fac1. Here fac1 = No serves as the control group. Switching to fac1 = Co decreases the response relative to the control, while switching to fac1 = Ru has essentially no effect.

**Graphing a model with two 2-factor interactions [model out3]**

As with the previous model I specify a viewport in order to place the profile plot of each two-factor interaction side-by-side in the same graphics window. To facilitate comparisons I use the **x.var** and **z.var** arguments to force the factor that is present in both interactions, fac1, to be plotted on the *x*-axis.

**#make first graph**

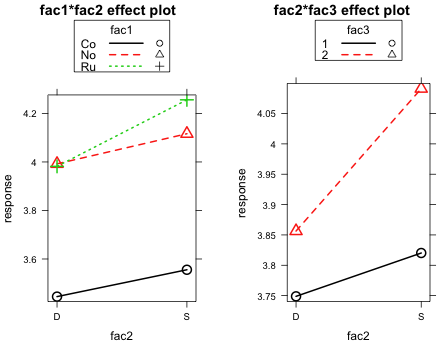
plot(effect('fac1:fac2',out3),multiline=T, z.var='fac1', x.var='fac2', nrow=1, ncol=2, col=1, more=T)

* I specify **z.var** and **x.var** specifically so that the same variable appears on the *x*-axis in both this plot and the next.
* The **nrow=** and **ncol=** options set up the dimensions of the viewport. In this case I specify that it consist of one row and two columns.
* The argument **col=1** specifies that current graph should occupy column 1 of the viewport.
* The **more=T** argument indicates that another graph will follow so the viewport should remain open.

**#make second graph**

plot(effect('fac2:fac3',out3), multiline=T, nrow=1, ncol=2, col=2, more=F)

* I don't bother to specify **x.var** and **z.var** here because **x.var='fac2'** turns out to be the default.
* I use **nrow=1** and **ncol=2** as before. This time I specify **col=2** to indicate that the current graph should occupy column 2.
* I specify **more=F** to close the viewport.

  
**Fig. 3** Profile plots of the two significant interactions in the model

While these graphs are really meant to demonstrate the nature of the two-factor interactions, it's worth pondering what the individual values of the plotted points represent. For instance in the left graph six values are plotted but we know there are twelve treatment means in the experiment. The variable that is not part of the displayed interaction, fac3, has two categories and we know that six of the twelve treatments have fac3 = 1 and six have fac3 = 2. So, how do we end up with these six plotted values? If we assign the output of the **effect** function to a named object the **$fit** component of this object contains the values that are being plotted.

effect.12<-effect('fac1:fac2',out3)

effect.12$fit

[,1]  
2401 3.444905  
2411 3.992204  
2421 3.981892  
2431 3.555621  
2441 4.116333  
2451 4.256111

The first value, 3.4449, is the plotted value in Fig. 3 corresponding to fac2 = D and fac1 = Co. If we look at the coefficient estimates from the regression model we see the following.

coef(out3)

(Intercept) fac1No fac1Ru fac2S fac32 fac1No:fac2S   
3.38323831 0.54729873 0.53698650 0.01714866 0.10757945 0.01341375   
fac1Ru:fac2S fac2S:fac32   
0.16350360 0.16322994

The intercept of the regression model is the estimate of the mean when fac1 = Co, fac2 = D, and fac3 = 1. The coefficient labeled fac32 is the amount the mean changes when we switch from fac3 = 1 to fac3 = 2. If we add the estimate of the intercept to this estimated coefficient we obtain the estimated mean when fac1 = Co, fac2 = D, and fac3 = 2.

coef(out3)[1]

(Intercept)   
3.383238

coef(out3)[1]+coef(out3)[5]

(Intercept)   
3.490818

As we can see neither of these values corresponds to what's being plotted. The code below counts the number of non-missing values for each level of fac3.

tapply(response, fac3, function(x) sum(!is.na(x)))

1 2   
102 137

So it turns out there are 102 observations in the data set for which fac3 = 1 and 137 for which fac3 = 2. If we use these numbers to take a weighted mean of the two fac3 means we calculated above for fac1 = Co and fac2 = D, we obtain the same value that the **effect** function is reporting.

(102\*coef(out3)[1]+137\*(coef(out3)[1]+coef(out3)[5]))/(102+137)

(Intercept)   
3.444905

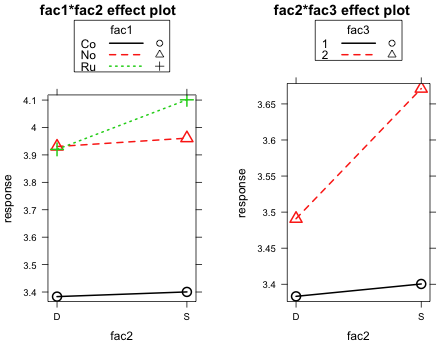
So the plotted points in Fig. 3 are the treatment means averaged over the levels of fac3 but weighted by the corresponding sample sizes. These weights might be interpretable if we were dealing with a true survey sample, but the different samples sizes arose due to attrition of subjects and other vagaries of the experiment.

To obtain more interpretable values we need to tell the **effect** function what to do about fac3, the factor that is missing from the interaction. As we saw in the previous section, we can assign values to terms not included in an effect by using the **given.values** argument of **effect**. For instance to get **effect** to generate the treatment means that correspond to level 1 of fac3 I would need to use **given.values=c(fac32=0**). Notice that the syntax still requires us to enter this single value as a vector using the **c** function even though it is just a scalar.

plot(effect('fac1:fac2',out3, given.values=c(fac32=0)),multiline=T, z.var='fac1', x.var='fac2', nrow=1, ncol=2, col=1, more=T)

I can obtain the same baseline mean profile in the second plot by forcing fac2 = Co. In terms of the model regressors this corresponds to specifying **fac1No=0** and **fac1Ru=0** in the **given.values** argument.

plot(effect('fac2:fac3',out3, given.values=c(fac1No=0,fac1Ru=0)), multiline=T, nrow=1, ncol=2, col=2 ,more=F)

  
**Fig. 4** Same as Fig. 3 except the plotted points now represent individual treatment means

Finally we need to force both graphs to have roughly the same scale by specifying the *y*-limits explicitly with the **ylim** argument of **plot**. We can only get approximately the same scale here because the legends at the top of the graphs are not the same size and this alters the size of the plotting region we get. Unfortunately the **effects** package does not allow us to control placement of the legend ourselves. I also change the title of the plots so that it indicates the value that was used for the variable that is not part of the displayed interaction.

plot(effect('fac1:fac2',out3, given.values=c(fac32=0)),multiline=T, z.var='fac1', x.var='fac2', nrow=1, ncol=2, col=1, more=T, ylim=c(3.35,4.15), main='fac1:fac2 when fac3=1')

plot(effect('fac2:fac3',out3, given.values=c(fac1No=0,fac1Ru=0)), multiline=T, nrow=1, ncol=2,col=2,more=F, ylim=c(3.35,4.15), main='fac2:fac3 when fac1=Co')

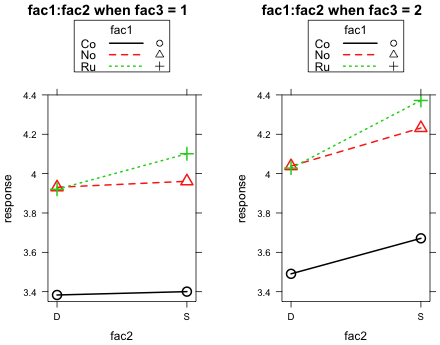
|  |
| --- |
| **fig 2** |
| **Fig. 5** Same as Fig. 3 except y-axes are scaled roughly the same in the two plots. |

The bottom mean profile is the same in each plot and corresponds to the combination fac1=Co and fac3=1. This makes it easier to assess the relative importance of the interactions being displayed in the two graphs.

It's worth noting that depending on which value we choose for the omitted variable, the graph changes. The change we get is not merely a vertical translation of the profiles. For instance if we focus on the left plot in which the fac1:fac2 interaction is shown, we will get a different picture depending on whether we choose fac3=1 or fac3=2. This is because there is a fac2:fac3 interaction present in the model too. Changing fac3 changes the effect that fac2 has on the response. In Fig. 6 I generate the left-hand plot of Fig. 5 but I use two different values of fac3.

plot(effect('fac1:fac2',out3, given.values=c(fac32=0)),multiline=T, z.var='fac1', x.var='fac2', nrow=1, ncol=2, col=1, more=T, ylim=c(3.35,4.4), main='fac1:fac2 when fac3 = 1')

plot(effect('fac1:fac2',out3, given.values=c(fac32=1)),multiline=T, z.var='fac1', x.var='fac2', nrow=1, ncol=2, col=2, more=F, ylim=c(3.35,4.4), main='fac1:fac2 when fac3 = 2')



**Fig. 6** The fac1:fac2 interaction at two different values of fac3

This is the kind of picture we obtained in lecture 2 when we were trying to assess whether a three-factor interaction was present. The rule we used was that if the nature of the displayed two-factor interaction changes as we vary the level of a third variable, that is evidence for a three-factor interaction. In Fig. 6 we see that although the picture does change when the level of fac3 is changed (because a fac2 × fac3 interaction is present), the basic nature of the two-factor interaction between fac1 and fac2 does not change. In both plots the No and Co profiles are parallel to each other; the Ru and No profiles nearly coincide but the Ru profile has a slightly larger mean when fac2 = S. Hence, as we concluded in lecture 2, a 3-factor interaction does not appear to be present.

**Multiple regression with both continuous and categorical predictors**

The inclusion of categorical and continuous predictors in the same regression model introduces some new ideas particularly in interpreting interaction terms. I recreate the simulated data that we used in lecture 1.

set.seed(10)

x1<-runif(90)

x2<-rbinom(90,10,.5)

x3<-rgamma(90,.1,.1)

x4<-rep(c('A','B'),c(45,45))

x5<-rep(c('L','M','H'),30)

**#noise for response**

set.seed(20)

epsilon<-rnorm(90,0,2)

**#remove outlier**

x4<-x4[x3<25]

epsilon<-epsilon[x3<25]

x3<-x3[x3<25]

x4cat<-factor(x4)

**#create dummy variables to generate response**

z1<-(x4=='B')

w1<-(x5=='M')

w2<-(x5=='H')

All of the unique features of this regression model occur when we have just a one categorical and one continuous predictor, so I restrict myself to this scenario. For this I use the categorical variable x4 and the continuous variable x3 to generate two different response variables. In one, y1, both variables contribute to the response in an additive fashion (the additive model) and in the other, y2, the two predictors interact (the interaction model).

**#additive model**

y1<- 10+5\*z1+3\*x3+epsilon

**#interaction model**

y2<- 10+5\*z1+3\*x3-2\*z1\*x3+epsilon

**Additive model—analysis of covariance**

I fit the additive model first. The variable x4cat is the factor version of the categorical variable x4 and has two levels, A and B.

lm(y1~x4cat+x3)->model1

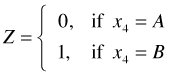
levels(x4cat)

[1] "A" "B"

contrasts(x4cat)

B  
A 0  
B 1

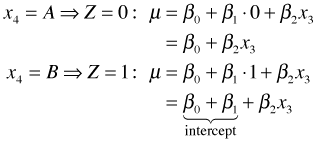
The dummy variable corresponding to x4cat is coded as follows.



The additive regression model for the mean is the following.

additive model

This last equation can be treated as two different regression equations depending on whether Z = 0 or Z = 1.



These two equations represent two lines that have the same slope, β2, but different intercepts. I write an R function with two arguments, x3 and Z, that returns the mean predicted from this regression model and then use it to draw the two regression lines. In the code below the **coef** function of R extracts the regression coefficients and the **curve** function draws the lines. The **curve** function requires a function of a single variable *x* as its first argument, so in each case I explicitly specify a value for Z and let x3 correspond to *x*. I use the **range** function of R to obtain the minimum and maximum values of x3 in the data set in order to limit the extent of the plotted regression lines.

model1.func<-function(x3,z) coef(model1)[1]+ coef(model1)[2]\*z+ coef(model1)[3]\*x3

**#plot data first: color the points according to x4 categories**

plot(y1~x3,col=z1+1)

**#model when x4 = A**

curve(model1.func(x,0),from=range(x3)[1], to=range(x3)[2], add=T)

**#model when x4 = B**

curve(model1.func(x,1),from=range(x3)[1], to=range(x3)[2], add=T, col=2, lty=2)

|  |
| --- |
| **fig. 7** |
| **Fig. 7** Additive model |

R supports mathematical typesetting with its **expression** function. I use it to include subscripts in the *x*-axis and *y*-axis labels of Fig. 7 and I also add a legend to the graph to identify the regression lines.

plot(y1~x3,col=z1+1, ylab=expression(y[1]), xlab=expression(x[3]))

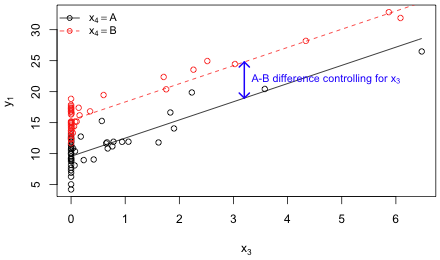
curve(model1.func(x,0),from=range(x3)[1], to=range(x3)[2], add=T)

curve(model1.func(x,1),from=range(x3)[1], to=range(x3)[2], add=T, col=2, lty=2)

legend('topleft', c(expression(x[4] == 'A'), expression(x[4]=='B')), pch=1, col=1:2, lty=c(1,2), cex=.9, bty='n')

arrows(3.2, model1.func(3.2,0), 3.2, model1.func(3.2,1), code=3, angle=45, length=.1, col=4, lwd=2)

text(3.2, (model1.func(3.2,0)+model1.func(3.2,1))/2, expression(paste('A-B difference controlling for ',x[3])), pos=4, cex=.9, col=4)

  
**Fig. 8** Analysis of covariance model

If the focus of the analysis is on the categorical variable and the continuous variable is merely a nuisance variable, then the additive model we've just fit is also called an analysis of covariance model. In analysis of covariance the goal is to test for group differences while controlling for the effect of a covariate. The covariate is a continuous variable that is also related to the response and has a distribution that is different in the two groups (as defined by the levels of the categorical variable). Analysis of covariance is a way to statistically control for a confounding variable that was not controlled for in the experimental design. The blue arrow in Fig. 8 displays the estimated group difference μB – μA from the analysis of covariance model. This difference is given by the coefficient β1 in the regression model. Because the regression lines are parallel the concept of a mean group difference is well-defined. The difference between the groups (distance between the lines) is the same regardless of the value of chosen for the confounding variable x3.

**Interactions between continuous and categorical predictors**

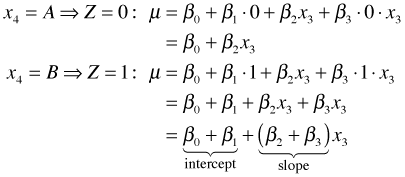
I next fit a model in which the continuous and categorical predictors interact. In building the model I follow the principle of marginality and include the individual predictors separately also.

lm(y2~x4cat\*x3) -> model2

The regression model for the mean is the following.

interaction

This single equation represents two different regression equations depending on whether Z = 0 or Z = 1.



The interaction regression model yields two lines, one for each category of x4. Unlike the additive model these lines that have different intercepts and different slopes. Significance tests for the effect coefficients β1 and β3 have very clear interpretations in the interaction model. A test of H0: β3 = 0 is a test of whether the slopes are different in the two groups and a test of H0: β1 = 0 is a test of whether the intercepts are different.

I write an R function with arguments x3 and Z that returns the predicted mean from the interaction regression model. I then use it to draw the regression lines for the two groups, A and B.

model2.func<-function(x3,z) coef(model2)[1]+ coef(model2)[2]\*z+ coef(model2)[3]\*x3+ coef(model2)[4]\*x3\*z

plot(y2~x3,col=z1+1, ylab=expression(y[2]), xlab=expression(x[3]))

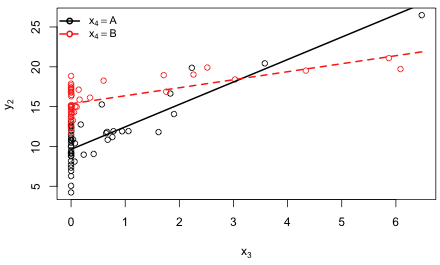
**# Group A**

curve(model2.func(x,0), from=range(x3)[1], to=range(x3)[2], add=T, lwd=2)

**# Group B**

curve(model2.func(x,1), from=range(x3)[1], to=range(x3)[2], add=T, col=2,lty=2, lwd=2)

legend('topleft', c(expression(x[4] == 'A'),expression(x[4]=='B')), pch=1, lwd=2, col=1:2, lty=c(1,2), cex=.9, bty='n', pt.cex=1)

  
**Fig. 9** Regression model with an interaction between a continuous and a categorical predictor

Including a categorical variable with more than two categories in a regression model along with a continuous predictor introduces no new ideas. Instead of having just two different lines we have one line for each category. Models that have more than one categorical predictor and a single continuous predictor have very similar interpretations.

**R Code**

A compact collection of all the R code used in this document appears [here](http://www.unc.edu/courses/2010fall/ecol/563/001/notes/lecture3%20Rcode.txt).

[Course Home Page](http://www.unc.edu/courses/2010fall/ecol/563/001/index.html)

|  |
| --- |
| Jack Weiss *Phone:* (919) 962-5930 *E-Mail:* jack\_weiss@unc.edu *Address:* Curriculum in Ecology, Box 3275, University of North Carolina, Chapel Hill, 27516 Copyright © 2010 Last Revised--Sept 7, 2010 URL: [http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture3.htm#lecture3) |