<http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm>

**Lecture 5—Monday, September 13, 2010**

**Topics**

* [Fitting the ANOVA models](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#fitting)
* [Probability smear graphs](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#smear)
  + [Probability smear graphs using lattice](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#smearlattice)
  + [Probability smear graphs using base graphics](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#smearbase)
* [Displaying the predicted treatment means](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#display)
  + [Obtaining the treatment means estimated by the model](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#obtain)
  + [Graphing the treatment means using base graphics](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#basemeans)
  + [Graphing the treatment means using the lattice package—Method 1 [not done in class]](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#latticemeans1)
  + [Graphing the treatment means using the lattice package—Method 2 [not done in class]](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#latticemeans2)
* [Cited references](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#cited)
* [R code](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#Rcode)

**R functions and commands demonstrated**

* [abline](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#box) is a low-level graphics command that adds vertical lines (v= ), horizontal lines (h= ), or regression lines to a plot
* [arrows](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#arrows) draws an arrow between two points on a plot
* [as.numeric](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#basemeans) converts its argument to numeric data. We used it to obtain the numeric levels of a factor.
* [axis](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axis) is a low-level graphics function for customizing the features of the axes of the currently displayed graph. Its first argument, 1, 2, 3, or 4, specifies which axis, bottom, left, top, or right.
* [bold](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#legend) when used in combination with **expression** can be used to display text in bold face
* [box](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#box) is a low-level graphics command that draws a box around a plot
* [coef](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#coef) extracts the regression coefficients from an **lm** object
* [data.frame](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#dataframe) constructs a data frame from a collection of vectors of the same length (or matrices with the same number of rows)
* [diag](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#vcov) extracts the diagonal entries from a matrix
* [dotchart](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#dotchart) is a graphics function from base graphics for producing dot plots
* [dotplot](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#dotchart) is a graphics function from the **lattice** package for producing dot plots, a minimal ink alternative to bar plots
* [expression](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#legend) defines an R expression and is typically used to construct mathematical text expressions
* [for](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#latticemeans2) defines a for loop, a programming tool that allows a set of instructions to be executed repeatedly
* [function](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panel) is a keyword that begins the definition of a user-defined function
* [grid](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#grid) is a low-level graphics command that adds a grid of horizontal and/or vertical lines to a plot
* [if](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panellines) is used to define a conditional statement in which if a particular Boolean condition is true then a set of instructions is executed, otherwise not.
* [legend](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#legend) is a low-level graphics command that adds a legend to a base graphics plot
* [lines](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#lines) is a low-level graphics command that draws lines between plotted points on the currently active plot
* [list](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#scales) concatenates objects of diverse types together as a single object
* [mtext](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#mtext) is a low-level graphics command that can be used to add text to any one of the four margins of a graph
* [names](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#dataframe) extracts the names of the elements comprising an object
* [panel.abline](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panel) (from **lattice**) is the panel function that draws vertical lines (v= ), horizontal lines (h= ), or regression lines
* [panel.arrows](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelarrows) (from **lattice**) is the panel function that plots arrows (and error bars)
* [panel.dotplot](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#paneldotplot) (from **lattice**) is the panel function that produces a dot plot
* [panel.grid](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelgrid) (from **lattice**) is the panel function that draws a sequence of grid lines
* [panel.lines](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panellines) (from **lattice**) is the panel function that connects points with a sequence of line segments
* [panel.segments](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panel) (from **lattice**) is the panel function that connects a pair of points with a line segment
* [panel.superpose](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelarrows) (from **lattice**) is the panel function used with groups to indicate that individual groups are to be differentiated within panels. Can be used with a user-specified **panel.groups** function for customized displays
* [panel.xyplot](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panel) (from **lattice**) is the panel function that plots points
* [par](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#par) sets graphics parameters for base graphics
* [points](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#lines) is a low-level graphics command that adds individual points to the currently active plot
* [qnorm](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#qnorm) is the quantile function of a normal distribution
* [range](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axes) calculates the minimum and maximum values of its argument and returns the result as a vector
* [rbind](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#rbind) appends a second data frame or matrix to a first. The rows of the second data frame are placed after the rows of the first data frame.
* [reorder](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#reorder) is used to reorder the levels of a factor variable in the increasing order of the values of a second variable
* [rep](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#rbind) repeats a scalar or vector a specified number of time to create a new vector
* [vcov](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#vcov) when applied to a regression model (**lm** object) returns the variance-covariance matrix of the parameter estimates
* [xyplot](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#condition) (from **lattice** package) produces multi-panel scatter plot in which the different panels correspond to different values of a conditioning variable
* [|](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#condition), a vertical bar, used to separate the plotting formula from the conditioning variable in **lattice** graphics
* [...](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#dotdotdot) used in function calls so that additional arguments are passed to that function as needed when that function is called
* [%\*%](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#matrixmult) is the matrix multiplication operator in R

**R function options**

* [angle](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#arrows)= (argument to **arrows** function) specifies the angle of arrowhead edge; **angle=90** is the appropriate setting for error bars
* [at](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axis)= (argument to **axis**) is used to specify the locations of tick marks
* [auto.key](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#condition)= (argument of many **lattice** functions) is used to invoke the default settings for the key used to identify the different levels of the grouping variable in each plot. By assigning to **auto.key** a list of graphics settings some of these defaults can be overridden.
* [axes](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axes)= (argument to **plot** and other base graphics functions) can be used to prevent the *x*-axis and *y*-axis with their default settings from being drawn (by specifying **axes=F**)
* [cex](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelabline)= (argument to many graphics functions) specifies the amount of character expansion to be used for plotting symbols, **cex=1** is the default
* [cex.axis](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axis)= (argument to **axis**) sets the size of the labels at tick marks
* [code](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#arrows)= (argument to **arrows** function) specifies the type of arrow to draw; **code=3** places arrowheads on both ends
* [corner=](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#key) (setting of **key** argument of **xyplot** and other **lattice** functions) is used to specify a corner of the graph to use as the origin for specifying the absolute location of the key (legend). For instance **corner(0,0)** denotes the bottom left corner while **corner(1,1)** specifies the top right corner.
* [group.number](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#groupnumber) is a key word available in **lattice** for identifying the observations that are being used in the current group
* [groups](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#condition)= (argument to **lattice** functions) defines the grouping variable in **lattice** graphs. The effect of the different levels of a grouping variable is displayed in the same panel unlike the conditioning variable whose effect is displayed across panels.
* [key=](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#key) (argument to **xyplot** and other **lattice** functions) defines characteristics of the legend of the graph
* [labels](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axis)= (argument to **axis** and **factor**) specifies a vector of values to appear at the tick mark locations specified by **at=** in **axis** OR the [labels](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#labels) for the levels defined by the **factor** function
* [las](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axis)= (argument for many graphics text functions) controls the orientation of printed text. We chose **las=2** to display tick mark labels perpendicular to the axis on which they occur.
* [length](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#arrows)= (argument to **arrows** function) specifies the size of the arrowhead
* [lty](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelabline)= (argument to many plot functions) specifies the type of line to draw. Specifying **lty=1** yields a solid line, **lty=2** is a dashed line, and **lty=3** is a dotted line.
* [mar](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#par)= (argument to **par** function) specifies the width of the four margins of plots
* [panel](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panel)= (argument to **xyplot**) used to define a panel object that describes what is to be plotted in each panel
* [panel.groups](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelarrows)= (argument to **xyplot**) identifies a user-defined panel function that describes what to be plotted for each group in a panel
* [pch](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelabline)= stands for "print" ("plot") character and is used to designate the plotting symbol to be used in various plotting functions: **plot**, **points**, etc.
* [pt.cex](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#legend)= defines the size of plotting symbols in legends (overrides the value of **cex**)
* [scales](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#scales)= (argument to **xyplot** and other **lattice** functions) defines characteristics of the axis—the number of tick marks, position and labels of the ticks, etc.
* [space](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#condition)= (argument to **auto.key**, which is used to stet up legends for **lattice** functions) identifies where the key should appear on the outside margin of the graph. Choices include space=**"right"**, **"top"**, **"bottom"**, and **"left"**.
* [subscript](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelarrows) is a key word available in panel functions of **lattice** for identifying the observations that are being used in drawing the current panel
* [type](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#axes)= (argument to most high level plotting functions) indicates the type of graphical display to produce. We used **type='n'** to suppress the plotting of any data and **type='b'** to plot both points and line segments connecting those points. The 'n' stands for 'nothing, while 'b' stands for both.
* [xlim](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#panelabline)= (argument of all plotting functions) a vector that specifies the minimum and maximum values to display on the *x*-axis
* [ylim](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#dotdotdot)= (argument of all plotting functions) a vector that specifies the minimum and maximum values to display on the *y*-axis

**Additional R packages used**

* [effects](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#display) for the **effect** function and its **plot.eff** method to generate plots of interaction effects.
* [lattice](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#fitting) for panel graphics functions

**Fitting the ANOVA models**

R has three major graphics packages: base graphics, **lattice**, and **ggplot2**. The last one is a relatively new R package that is based on Leland Wilkinson's (of SYSTAT fame) "grammar of graphics". Today we focus on using base graphics and **lattice** to summarize ANOVA models. The plots we consider are probability smear graphs of effect estimates and interaction plots with error bars.

I return to the 3-factor ANOVA models that we considered in [lecture 2](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture2.htm#final). I reload the data, fit the full interaction model, and simplify it as much as possible.

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 the models**

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

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

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

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

In [lecture 2](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture2.htm#final) we found that there were two reasonable final models, one with two 2-factor interactions and no main effects (out3) and a second model with one two-factor interaction and one main effect (out4). We will use the first of these models, out3, to illustrate how to generate graphical displays of ANOVA models.

**Probability smear graphs**

ANOVA models are regression models in which all the regressors are dummy variables. This means that the regression coefficients are multiplying terms that are either have value zero or one. Hence all the regression coefficients are measured on the scale and either represent means or deviations from means. The upstart is that it makes sense to present the coefficient estimates together in the same graph. When confidence intervals are also included the magnitudes of the different effects can be compared and their relative importance assessed.

The coefficient estimates of a regression model can be extracted with the **coef** function.

coef(out3)

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

The intercept is the mean of the reference group while the rest of the terms correspond to effects. To ensure that we're only comparing effects, I remove the intercept from the list of estimates to plot.

ests<-coef(out3)[2:8]

ests

fac1No fac1Ru fac2S fac32 fac1No:fac2S fac1Ru:fac2S fac2S:fac32   
0.54729873 0.53698650 0.01714866 0.10757945 0.01341375 0.16350360 0.16322994

To obtain the standard errors of these estimates we can either extract them from the summary table of the model, or we can obtain the variance-covariance matrix of the parameter estimates with the **vcov** function and then extract the diagonal entries with the **diag** function.

summary(out3)$coefficients

Estimate Std. Error t value Pr(>|t|)  
(Intercept) 3.38323831 0.05245374 64.4994720 1.024057e-149  
fac1No 0.54729873 0.05837246 9.3759750 6.582616e-18  
fac1Ru 0.53698650 0.06084917 8.8248782 2.755469e-16  
fac2S 0.01714866 0.06696214 0.2560949 7.981054e-01  
fac32 0.10757945 0.04814860 2.2343214 2.641948e-02  
fac1No:fac2S 0.01341375 0.07727544 0.1735836 8.623448e-01  
fac1Ru:fac2S 0.16350360 0.08174704 2.0001164 4.665860e-02  
fac2S:fac32 0.16322994 0.06504617 2.5094473 1.277781e-02

summary(out3)$coefficients[,2]

(Intercept) fac1No fac1Ru fac2S fac32 fac1No:fac2S fac1Ru:fac2S   
0.05245374 0.05837246 0.06084917 0.06696214 0.04814860 0.07727544 0.08174704   
fac2S:fac32   
0.06504617

**#using variance-covariance matrix**

vcov(out3)

(Intercept) fac1No fac1Ru fac2S fac32 fac1No:fac2S  
(Intercept) 0.002751394 -2.018169e-03 -0.0018755968 -0.002751394 -1.313697e-03 2.018169e-03  
fac1No -0.002018169 3.407344e-03 0.0020049896 0.002018169 1.976835e-05 -3.407344e-03  
fac1Ru -0.001875597 2.004990e-03 0.0037026214 0.001875597 -2.318288e-04 -2.004990e-03  
fac2S -0.002751394 2.018169e-03 0.0018755968 0.004483928 1.313697e-03 -3.251732e-03  
fac32 -0.001313697 1.976835e-05 -0.0002318288 0.001313697 2.318288e-03 -1.976835e-05  
fac1No:fac2S 0.002018169 -3.407344e-03 -0.0020049896 -0.003251732 -1.976835e-05 5.971493e-03  
fac1Ru:fac2S 0.001875597 -2.004990e-03 -0.0037026214 -0.002970558 2.318288e-04 3.266274e-03  
fac2S:fac32 0.001313697 -1.976835e-05 0.0002318288 -0.002270055 -2.318288e-03 -2.181245e-05  
fac1Ru:fac2S fac2S:fac32  
(Intercept) 0.0018755968 1.313697e-03  
fac1No -0.0020049896 -1.976835e-05  
fac1Ru -0.0037026214 2.318288e-04  
fac2S -0.0029705578 -2.270055e-03  
fac32 0.0002318288 -2.318288e-03  
fac1No:fac2S 0.0032662738 -2.181245e-05  
fac1Ru:fac2S 0.0066825785 -5.506149e-04  
fac2S:fac32 -0.0005506149 4.231005e-03

sqrt(diag(vcov(out3)))

(Intercept) fac1No fac1Ru fac2S fac32 fac1No:fac2S fac1Ru:fac2S   
0.05245374 0.05837246 0.06084917 0.06696214 0.04814860 0.07727544 0.08174704   
fac2S:fac32   
0.06504617

sqrt(diag(vcov(out3)))[2:8]->std.errs

Using the estimates and their standard errors I next I obtain 50% and 95% confidence intervals of the effects. Technically we should use a t-distribution for this, but the sample size used in our analysis is so large that the relative difference between using a t-distribution or a normal distribution is trivial, so I use the **qnorm** function to obtain the appropriate quantiles.

lower95<-ests+qnorm(.025)\*std.errs

upper95<-ests+qnorm(.975)\*std.errs

lower50<-ests+qnorm(.25)\*std.errs

upper50<-ests+qnorm(.75)\*std.errs

Finally I assemble everything in a data frame. When I construct the data frame I include a column of labels for the effects obtained by extracting the effect names from the vector of coefficient estimates with the **names** function. I make it a factor and specify the order of the factor levels explicitly to prevent the **data.frame** function of R from putting them in alphabetical order.

var.labels<-names(ests)

new.data<-data.frame(var.labels=factor(var.labels, levels=names(ests)), ests,lower95, upper95, lower50, upper50)

Pictures that are meant to display and compare simple point estimates such as means are best done using some variation of a dot plot, rather than the "dreaded dynamite plot" that is so often favored by ecologists. (For discussions of the problems with dynamite plots see [Magnusson 2000](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#magnusson) or pp. 7–11 of [Donahue 2010](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#donahue).) A bare-bones dot plot can be obtained with the **dotchart** function of base graphics or the **dotplot** function of **lattice**. The syntax is slightly different as shown below.

dotchart(new.data$ests, labels=new.data$var.labels, xlab='ests')

library(lattice)

dotplot(var.labels~ests, data=new.data)

|  |  |
| --- | --- |
| (a) fig. 1a | (b) fig 1b |
| **Fig. 1** Default dot plots as generated by (a) the dotchart function of base graphics and (b) the dotplot function of lattice | |

To make these useful for our purposes we will need to add error bars. I illustrate how to do this for lattice and base graphics separately.

**Probability smear graphs using lattice**

Lattice graphs are generated with a single line of code. Each lattice plotting function has a default panel function that identifies the features that will be displayed in the graph. If you want additional features beyond the default you will need to construct your own panel function. The panel function must identify everything that is to be plotted including what you would otherwise get by default. When using your own panel function it doesn't really matter what basic lattice plot function you use because the default panel function for that plot function is ignored anyway.

Panel functions are begun with the keyword **panel=function(x,y)** followed by a pair of curly braces **{ }**. Special lattice panel graphing functions are then listed on separate lines within the curly braces. Each panel graphing function adds a specific feature to the graph. Typically the names of these functions begin with the word panel followed by the operation they are to perform, e.g., **panel.xyplot**, **panel.lines**, **panel.points**, etc., although recently **lattice** has been moving away from this convention.

The following lattice call uses a panel function to plot the point estimates with **panel.xyplot**, add a vertical line at 0 with **panel.abline**, and add the 95% confidence intervals with **panel.segments**. The **xlim** argument is needed to make enough room for the error bars. It uses the minimum value of the lower 95% intervals and the maximum of the 95% confidence intervals to define the plot range. I subtract and add .02 to increase this range a little bit. The *x* and *y* that appear in the declaration of the panel function correspond to the *x* and *y*-variables in the original **dotplot** call, in this case ests and var.labels respectively.

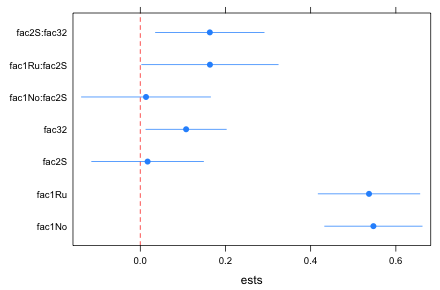
dotplot(var.labels~ests, data=new.data, xlim=c(min(new.data$lower95)-.02, max(new.data$upper95)+.02), panel=function(x,y){

panel.xyplot(x, y, pch=16, cex=1)

panel.abline(v=0, col=2, lty=2)

panel.segments(new.data$lower95, as.numeric(y), new.data$upper95, as.numeric(y), lty=1, col="#0080ff")})

* **panel.xyplot** plots the points using filled circles, pch=16.
* **panel.abline** draws a vertical line at 0, v=0, using a dashed line, lty=2, colored red, col=2.
* The syntax for **panel.segments** is panel.segments(x1,y1,x2,y2) where (x1, y1) and (x2, y2) are the two endpoints of the line segment to be drawn. Here the *y*-coordinate corresponds to the factor variable var.labels. Because the **panel.segments** function expects a number, not a factor, to plot, I convert the factor to a number with the **as.numeric** function, as.numeric(y). The color col="#0080ff" is the default blue color that **dotplot** uses for the points so I use it for the confidence intervals also. The notation "#0080ff" is standard hexadecimal notation for colors (paralleling what is used in html code). Fig. 2 shows the result.



**Fig. 2** Dot plot with error bars (95% confidence intervals)

Sometimes it is useful to sort the displayed effects so that they appear in order of increasing or decreasing value. An easy way to do that is with the **reorder** function of R.

new.data$var.labels2<-reorder(new.data$var.labels, new.data$ests)

levels(new.data$var.labels)

[1] "fac1No" "fac1Ru" "fac2S" "fac32" "fac1No:fac2S"  
[6] "fac1Ru:fac2S" "fac2S:fac32"

levels(new.data$var.labels2)

[1] "fac1No:fac2S" "fac2S" "fac32" "fac2S:fac32" "fac1Ru:fac2S"  
[6] "fac1Ru" "fac1No"

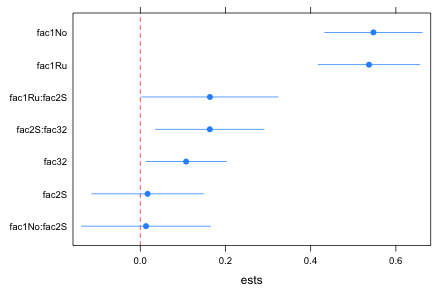
Here **reorder** has created a new factor variable I called var.labels2 in which the levels of var.labels are ordered by the increasing values of ests. In the plot I then use var.labels2 as the *y*-variable in the **dotplot** function replacing var.labels to produce Fig. 3.

dotplot(**var.labels2**~ests, data=new.data, xlim=c(min(new.data$lower95)-.02, max(new.data$upper95)+.02), panel=function(x,y){

panel.xyplot(x, y, pch=16, cex=1)

panel.abline(v=0, col=2, lty=2)

panel.segments(new.data$lower95, as.numeric(y), new.data$upper95, as.numeric(y), lty=1, col="#0080ff")})



**Fig. 3** Dot plot with observations sorted by the magnitude of the estimated effect

To obtain the classic probability smear graph I add 50% confidence intervals to the already displayed 95% confidence intervals. To make them distinguishable in the same graph, I plot the 95% confidence intervals with a thin line and the 50% confidence intervals with a thick line. I also add a more informative label to the *x*-axis (Fig. 4).

dotplot(var.labels~ests,data=new.data, xlim=c(min(new.data$lower95)-.02, max(new.data$upper95)+.02), xlab='Estimated effect', panel=function(x,y){

panel.segments(new.data$lower95, as.numeric(y), new.data$upper95, as.numeric(y), lty=1, col="black")

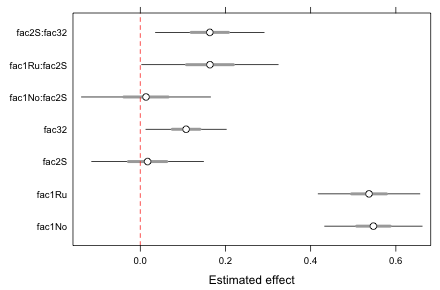
panel.segments(new.data$lower50, as.numeric(y), new.data$upper50, as.numeric(y), lty=1, col="grey60", lwd=4)

panel.xyplot(x, y, pch=16, cex=1.2, col='white')

panel.xyplot(x, y, pch=1, cex=1.1, col='black')

panel.abline(v=0, col=2, lty=2)

})



**Fig. 4** Dot plot with both 50% and 95% confidence intervals

The functions listed within the panel function are executed in the order shown. Each new feature is drawn on top of the previously drawn features. For this reason I draw the thin line first, followed by the fat line, followed by the point estimates. To make the point estimates more distinct I first draw them with a large filled white circle.

panel.xyplot(x, y, pch=16, cex=1.2, col='white')

I then superimpose on top of them a slightly smaller open circle with a black border.

panel.xyplot(x, y, pch=1, cex=1.1, col='black')

Graphs of this sort have been popularized by [Gelman and Hill (2006)](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#gelman). The rationale they give for such a picture is that it gives a range of uncertainty for the statistic. They note, p. 18, that "a 50% interval is particularly easy to interpret since the true value should be as likely to be inside as outside the interval." Another way to view the display is as an uncertainty smear. Rather than treat the line segments literally as confidence intervals with all the accompanying frequentist intellectual baggage, the diagram can be thought of as contrasting a crap shoot, the 50% confidence interval, with a sure thing, the 95% interval. A reasonable compromise should then lie somewhere between these two extremes.

Next I improve the labeling on the *y*-axis. I replace the hybrid names created by R that combines the factor name with the level with names that just specify the levels. I also take advantage of the mathematical typesetting abilities of R to include the multiplication symbol of arithmetic, ×, in the interaction terms. This is denoted %\*% in R and requires the use of the **expression** function.

**#formatted labels**

mylabs<-c('Normal', 'Ru486', 'Shrimp', 'Sibship2', expression('Normal' %\*% 'Shrimp'), expression('Ru486' %\*% 'Shrimp'), expression('Shrimp' %\*% 'Sibship2'))

To use these labels in the **dotplot** function we need to include the **scales** argument. The basic format of **scales** is

scales=list(x=list( , ,), y = list(, , ))

where *x* and *y* denote the *x*- and *y*-axes and we specify the features we want to modify on the respective axes. For instance, the following will cause the *y*-axis tick marks to be labeled using the new labels, but leave the *x*-axis alone.

scales=list(y=list(at=1:7, labels=mylabs))

Here's the complete function call.

dotplot(var.labels~ests,data=new.data, xlim=c(min(new.data$lower95)-.02, max(new.data$upper95)+.02), xlab='Estimated effect', panel=function(x,y){

panel.segments(new.data$lower95, as.numeric(y), new.data$upper95, as.numeric(y), lty=1, col="black")

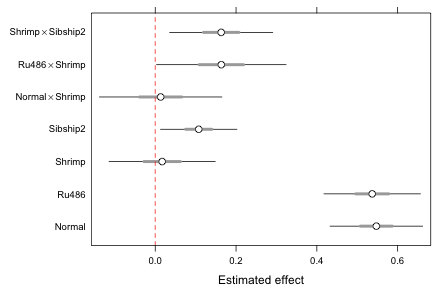
panel.segments(new.data$lower50, as.numeric(y), new.data$upper50, as.numeric(y), lty=1, col="grey60", lwd=4)

panel.xyplot(x, y, pch=16, cex=1.2, col='white')

panel.xyplot(x, y, pch=1, cex=1.1, col='black')

panel.abline(v=0, col=2, lty=2)

}, scales=list(y=list(at=1:7, labels=mylabs)))



**Fig. 5** Fig. 4 with improved labeling on the y-axis

As a final cosmetic improvement we can add grid lines. The **panel.dotplot** function adds grid lines by default so I could use this function first and let the rest of the function calls cover up what it plots.

dotplot(var.labels~ests, data=new.data, xlim=c(min(new.data$lower95)-.02, max(new.data$upper95)+.02), xlab='Estimated effect', panel=function(x,y){

panel.dotplot(x, y, pch=16, cex=1.2, col='white')

panel.segments(new.data$lower95, as.numeric(y), new.data$upper95, as.numeric(y),lty=1, col="black")

panel.segments(new.data$lower50,as.numeric(y), new.data$upper50, as.numeric(y), lty=1, col="grey60", lwd=4)

panel.xyplot(x, y, pch=16, cex=1.2, col='white')

panel.xyplot(x, y, pch=1, cex=1.1, col='black')

panel.abline(v=0, col=2, lty=2)

}, scales=list(y=list(at=1:7, labels=mylabs)))

Alternatively we can use the **panel.grid** function and specify line details for the grid ourselves. **panel.grid** has **v=** and **h=** arguments for specifying how many vertical and horizontal grid lines are desired. Putting a negative sign in front of the specified number forces the grid lines to be placed at tick marks. The call panel.grid(v=0, h=-7, lty=3) requests that no vertical grid lines be displayed, **v=0**, and that seven horizontal grid lines be drawn at the tick marks, **h=-7**. The argument lty=3 causes the grid lines to be dotted lines.

dotplot(var.labels~ests, data=new.data, xlim=c(min(new.data$lower95)-.02, max(new.data$upper95)+.02), xlab='Estimated effect', panel=function(x,y){

panel.grid(v=0, h=-7, lty=3, col='gray80')

panel.segments(new.data$lower95, as.numeric(y), new.data$upper95, as.numeric(y), lty=1, col="black")

panel.segments(new.data$lower50, as.numeric(y), new.data$upper50, as.numeric(y), lty=1, col="grey60", lwd=4)

panel.xyplot(x, y, pch=16, cex=1.2, col='white')

panel.xyplot(x, y, pch=1, cex=1.1, col='black')

panel.abline(v=0, col=2, lty=2)

}, scales=list(y=list(at=1:7, labels=mylabs)))

|  |  |
| --- | --- |
| (a) fig 6a | (b) fig 6b |
| **Fig. 6** Probability smear graphs with grid lines using (a) panel.dotplot and (b) panel.grid | |

So what do we learn from a graph such as Fig. 6?

1. The probability smear graph is a graphical representation of the coefficient summary table for the model (minus the intercept term). Using Fig. 6 we can visually carry out the significance tests reported in the summary table by determining if the displayed confidence intervals intersect the red dashed vertical line at 0. We can carry out these tests at α = .05 or at α = .50.
2. We can use Fig. 6 to assess the relative importance of the various effects in the model. This is especially useful if an appropriate reference group was chosen. In the current model the reference group is defined by hormone treatment = corticosterone, diet = detritus, and sibship = 1. It probably would have made more sense to have made hormone treatment = normal the reference group because it serves as the control in this experiment. Putting that aside for the moment we can observe the following from Fig. 6.
   1. The hormone treatment yields the biggest change in the response with normal and Ru486 yielding roughly the same effect relative to the reference group corticosterone. Because normal is the control group it tells us that Ru486 has essentially no effect while corticosterone has a very large negative effect.
   2. There is no diet effect unless it is combined with Ru486. In this case switching to the shrimp diet has a weak positive effect, enough of an effect to make the Ru486 treatment significantly different from the corticosterone group, but not enough to be significantly different from the normal group. (Technically this last test was not carried out, but it seems a reasonable conclusion based on the overlapping confidence intervals.)
   3. For tadpoles given a shrimp diet the increased response that is obtained by switching to Ru486 is of the same order of magnitude as the effect gained by switching from sibship 1 to sibship 2. This implies that the diet × Ru486 effect is no larger than effects attributable to ordinary background genetic variation.
3. If desired, we can also use the graph to estimate the effect attributable to any treatment combination of interest just by adding the corresponding displayed effects. For example, for a tadpole with sibship=1, diet=detritus, and hormone=Ru486 the effect is given by the single labeled estimate Ru486. For a tadpole with sibship=1, diet=shrimp, hormone=Ru486 we would need to add together the displayed effects for Ru486, Shrimp, and Ru486 × Shrimp. If instead we want the treatment mean (rather than effect) we then just add the effect just obtained to the model intercept value (not shown) which is 3.38.

**Probability smear graphs using base graphics**

The **lattice** package excels at generating panel displays. Because we're creating a single graph and not a panel, there is no particular reason to use the **lattice** package here. With base graphics a minimal plot is created using a higher-level graphics command, and then we add to that plot using lower level graphics commands. So the strategy is to build the graph incrementally with multiple function calls. The individual function calls correspond exactly to the different panel functions we used in generating the lattice graph, so almost everything we did there carries over.

The lower level graphics functions of base graphics require the numeric values of the effect categories for plotting, so I just number them 1 through 7.

new.data$num.labels<-1:7

When the goal is to build a modular graph, I generally use the first higher-level graphics command to set up axis limits and generate the rest of the graph using lower-level graphics commands. Here I use the **plot** function to set up the *y*-limits and *x*-limits explicitly and define labels for the axes.

plot(range(new.data[,2:6]), c(1-.3,7+.3), type='n', xlab='Effect estimates', ylab='', axes=F)

* In the first argument I use the **range** function to return the minimum and maximum *x*-limits (using the numerical columns of the data frame). The second argument sets the limits for the *y*-axis. I set these limits so that they are just beyond the plotted endpoints of 1 and 7 in order to add a little buffer around the display.

range(new.data[,2:6])  
[1] -0.1380433 0.6617067  
c(1-.3,7+.3)  
[1] 0.7 7.3

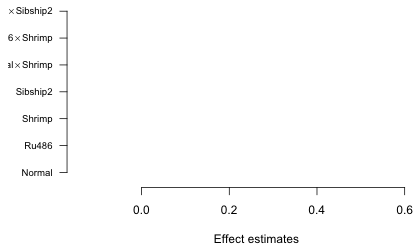
* Specifying **type='n'** , 'n' for nothing, suppresses the display of any points.
* The **axes=F** argument prevents the display of the *x*- and the *y*-axes. I include it because I want to add my own labels to the *y*-axis.

The **axis** functions are used to add axes to a plot. The first argument of **axis** is the desired axis. The axes are numbered 1 (bottom), 2 (left), 3 (top), and 4 (right). If you want to use default settings for the axis just enter the number and nothing more. To specify locations and labels for tick marks, add the **at=** and **labels=** argument just as was done in the **scales** argument of **lattice** graphs.

axis(1)

axis(2, at=1:7, labels=mylabs, las=2, cex.axis=.8)

I specify **las=2** to rotate the labels so that they are perpendicular to the axis. The **cex.axis=.8** option reduces the size of the labels to 80% of the default size. When we look at the graph this generates we see there is a problem (Fig. 7).



**Fig. 7** Graph with default margin settings

The left margin is not wide enough to accommodate the labels so they are being truncated. We need to expand the left margin beyond its default value. Margins are set with the **mar** argument of the **par** function.

par("mar")

[1] 5.1 4.1 4.1 2.1

The reported values are in line units and are given in the order bottom, left, top, right. I try doubling the left margin, but first I save the original settings so they can be restored later.

par("mar")->oldmar

par(mar=c(5.1,8.1,4.1,2.1))

plot(range(new.data[,2:6]), c(1-.3,7+.3), type='n', xlab='Effect estimates', ylab='', axes=F)

axis(1)

axis(2,at=1:7, labels=mylabs, las=2, cex.axis=.8)

box()

The **box** function with no arguments connects the axes and draws a box around the graph. Generating the rest of the graph is easy. I take the various panel functions from the previous section and remove the lead word "panel.". Where an *x* or a *y* appear as an argument I replace it with an actual variable name.

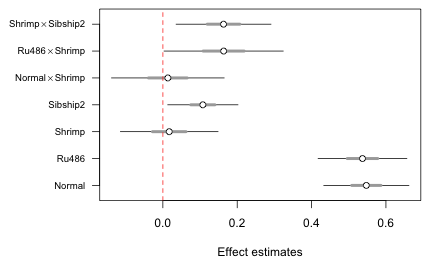
segments(new.data$lower95, new.data$num.labels, new.data$upper95, new.data$num.labels,col=1)

segments(new.data$lower50, new.data$num.labels, new.data$upper50, new.data$num.labels, col='grey60', lwd=4)

points(new.data$ests, new.data$num.labels, pch=16, cex=1.2, col='white')

points(new.data$ests, new.data$num.labels, pch=1, cex=1.1, col=1)

abline(v=0,lty=2,col=2)



**Fig. 8** Probability smear graph using base graphics

We can add grid lines, if desired, with the **grid** function. The call needed here is

grid(nx=NA, ny=NULL)

The arguments **nx** and **ny** denote the number of grid lines perpendicular to the *x*- and *y*-axes respectively. If we specify **nx=NA** we get no grid lines. If we specify **nx=NULL** we get grid lines at the tick marks. I redo the entire graph drawing the grid lines first so they don't overwrite what has already been plotted. I then reset the margins to their default values.

plot(range(new.data[,2:6]), c(1-.3,7+.3), type='n', xlab='Effect estimates', ylab='', axes=F)

axis(1)

axis(2, at=1:7, labels=mylabs, las=2, cex.axis=.8)

box()

grid(nx=NA, ny=NULL)

segments(new.data$lower95, new.data$num.labels, new.data$upper95, new.data$num.labels, col=1)

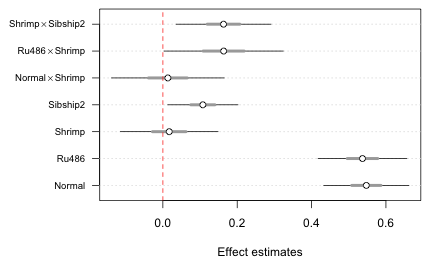
segments(new.data$lower50, new.data$num.labels, new.data$upper50, new.data$num.labels, col='grey60', lwd=4)

points(new.data$ests, new.data$num.labels, pch=16, cex=1.2, col='white')

points(new.data$ests, new.data$num.labels, pch=1, cex=1.1,col=1)

abline(v=0,lty=2,col=2)

par(mar=oldmar)



**Fig. 9** Probability smear graph with grid lines using base graphics

**Displaying the predicted treatment means**

**Obtaining the treatment means estimated by the model**

The treatment means based on a particular model are easy to calculate directly (see lecture 6) or they can be extracted with the **effect** function from the **effects** package. In the call below I request the treatment means to display the fac1:fac2 interaction the the fac3 variable is set at its first level, **fac32=0**. Then, in a second run I request the same treatment means at the second level of fac3.

effect1a<-effect('fac1:fac2', out3, given.values=c(fac32=0))

effect1b<-effect('fac1:fac2', out3, given.values=c(fac32=1))

names(effect1a)

[1] "term" "formula" "response" "variables"   
[5] "fit" "x" "model.matrix" "data"   
[9] "discrepancy" "se" "lower" "upper"   
[13] "confidence.level" "transformation"

The component **$x** identifies the levels of the factors that were used to obtain the estimated means, standard errors, and confidence interval endpoints.

effect1a$x

fac1 fac2  
1 Co D  
2 No D  
3 Ru D  
4 Co S  
5 No S  
6 Ru S

The components **$fit**, **$se**, **$lower**, and **$upper** contain the estimates, standard errors, and lower and upper endpoints of the 95% confidence intervals.

I assemble the output from the two **effect** calls in separate data frames, one for each level of fac3. In each data frame I add one more variable to indicate that fac3 level, fac3=1 or fac3=2. To generate this variable I use the **rep** function to repeat the level six times, rep(1,6) or rep(2,6). Finally I use the **rbind** function (bind by rows) to append one data frame to the other.

part1<-data.frame(effect1a$x, fac3=rep(1,6), est=effect1a$fit, se=effect1a$se, low95=effect1a$lower, up95=effect1a$upper)

part1

fac1 fac2 fac3 est se low95 up95  
2401 Co D 1 3.383238 0.05245374 3.279889 3.486587  
2411 No D 1 3.930537 0.04606953 3.839767 4.021307  
2421 Ru D 1 3.920225 0.05198867 3.817792 4.022657  
2431 Co S 1 3.400387 0.04162371 3.318376 3.482398  
2441 No S 1 3.961099 0.04277330 3.876824 4.045375  
2451 Ru S 1 4.100877 0.05022518 4.001919 4.199835

part2<-data.frame(effect1b$x, fac3=rep(2,6), est=effect1b$fit, se=effect1b$se, low95=effect1b$lower, up95=effect1b$upper)

fac.vals <-rbind(part1, part2)

fac.vals

fac1 fac2 fac3 est se low95 up95  
2401 Co D 1 3.383238 0.05245374 3.279889 3.486587  
2411 No D 1 3.930537 0.04606953 3.839767 4.021307  
2421 Ru D 1 3.920225 0.05198867 3.817792 4.022657  
2431 Co S 1 3.400387 0.04162371 3.318376 3.482398  
2441 No S 1 3.961099 0.04277330 3.876824 4.045375  
2451 Ru S 1 4.100877 0.05022518 4.001919 4.199835  
24011 Co D 2 3.490818 0.04941952 3.393447 3.588188  
24111 No D 2 4.038116 0.04304455 3.953306 4.122927  
24211 Ru D 2 4.027804 0.04393244 3.941245 4.114364  
24311 Co S 2 3.671196 0.04162371 3.589186 3.753207  
24411 No S 2 4.231909 0.04178987 4.149571 4.314247  
24511 Ru S 2 4.371686 0.04341654 4.286143 4.457229

**Graphing the treatment means using base graphics**

I elect to plot the fac1 × fac2 interaction separately for different levels of fac3. So one graph will consist of the mean profiles of fac1 separately by the levels of fac2 for one value of fac3. Then a second plot will display the same mean profiles for the second value of fac3. Because fac1 will appear on the *x*-axis I need the numerical values of the three factor levels for plotting purposes. I obtain them by using the **as.numeric** function on the factor variable.

fac.vals$fac1.num<-as.numeric(fac.vals$fac1)

fac.vals

fac1 fac2 fac3 est se low95 up95 fac1.num  
2401 Co D 1 3.383238 0.05245374 3.279889 3.486587 1  
2411 No D 1 3.930537 0.04606953 3.839767 4.021307 2  
2421 Ru D 1 3.920225 0.05198867 3.817792 4.022657 3  
2431 Co S 1 3.400387 0.04162371 3.318376 3.482398 1  
2441 No S 1 3.961099 0.04277330 3.876824 4.045375 2  
2451 Ru S 1 4.100877 0.05022518 4.001919 4.199835 3  
24011 Co D 2 3.490818 0.04941952 3.393447 3.588188 1  
24111 No D 2 4.038116 0.04304455 3.953306 4.122927 2  
24211 Ru D 2 4.027804 0.04393244 3.941245 4.114364 3  
24311 Co S 2 3.671196 0.04162371 3.589186 3.753207 1  
24411 No S 2 4.231909 0.04178987 4.149571 4.314247 2  
24511 Ru S 2 4.371686 0.04341654 4.286143 4.457229 3

As with the probability smear graph, I start by setting up the graph but plotting nothing. I use the minimum low95 and maximum up95 values to establish the *y*-limits. The plotted *x*-values are numbered 1, 2, 3, so I set the *x*-limits to begin just before a little before 1 and end just a little after 3. I then use the **axis** function to add labels to the tick marks on the *x*-axis.

plot(c(.8,3.2), range(fac.vals[,c("up95", "low95")]), type='n', xlab="Hormonal treatment", ylab='Estimated mean', axes=F)

axis(1, at=1:3, labels=levels(fac.vals$fac1))

axis(2)

box()

Next I subset the data set so that we first plot only the means corresponding to fac3 = 1.

dat1<-fac.vals[fac.vals$fac3==1,]

I want to draw the mean profile for each diet (fac2) group separately. So I next subset the data by the first diet, fac2 = 'D'.

**#first food group**

cur.dat<-dat1[dat1$fac2=='D',]

To prevent the error bars of the two mean profiles from lying on top of each other I shift one profile a little bit to the left and the other profile a little bit to the right. The variable myjitter contains the amount of the shift for the first profile.

**#amount of shift**

myjitter <- -.05

The **arrows** function is used to draw the confidence intervals. Its basic syntax is the same as the **segments** function from before. It supports additional arguments **angle**, **code**, and **length**.

* The **angle** argument controls the shape of the arrowhead; **angle=**90 yields flat ends.
* The **code** argument determines where the arrowheads should appear; **code=**3 puts arrowheads on both ends.
* The **length** argument controls the length of the arrowhead. This will typically need adjustment but for these data **length=**.05 works well.

arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05, col=1)

I follow this up with the **lines** function to draw the mean profile and the **points** function to plot the estimates at the middle of the confidence intervals. They each use the same syntax, *x*-variable followed by the *y*-variable. Just like the confidence intervals each has to shifted the same amount.

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=1, pch=16)

I next repeat these same lines of code for the second diet, fac2 = 'S', with the following changes.

* I change the subsetting level so I select observations for which fac2 = 'S'.
* I use a positive value of myjitter so that the points and error bars are moved slightly to the right.
* I change the color of the profile to red, **col=**2, and change the symbol type to open circles, **pch=**1. First I plot the points using filled white circles, **col=**'white' and **pch=**16, to cover up the drawn lines. Then I add red open circles on top.

**#second food group**

cur.dat<-dat1[dat1$fac2=='S',]

myjitter <- .05

arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05,col=2)

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=2)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col='white', pch=16, cex=1.1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=2, pch=1)

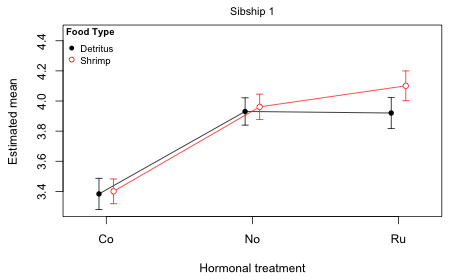
Finally I add a small header on top of the graph with the **mtext** (for margin text) function to indicate the level of fac3 that was plotted, fac3 = 1.

mtext(side=3,line=.5,'Sibship 1',cex=.9)

Lastly I add a legend to identify which diet corresponds to which mean profile.

legend('topleft', c('Detritus','Shrimp'), col=1:2, pch=c(16,1), cex=.8, pt.cex=.9, title=expression(bold('Food Type')),bty='n')

* The first argument of **legend** is the location of the legend in the plot. The nine allowable key words that identify the position are **"bottomright"**, **"bottom"**, **"bottomleft"**, **"left"**, **"topleft"**, **"top"**, **"topright"**, **"right"** and **"center"**. It is also possible to specify the location of the legend by explicitly giving the *x*- and *y*-coordinates of the top left corner of the legend.
* The second argument of **legend** is the vector of text labels that are to appear in the legend.
* I specify the remaining arguments by name. They include the colors, **col**, and symbols, **pch**, used in the plot. These are listed in an order that corresponds to the order of the text labels.
* The **cex** argument controls the size of text and symbols in the legend. It is possible to make the text and symbols different sizes in the legend. The argument **pt.cex** can be used to control the symbol size in which case **cex** controls the size of everything else.
* I add a title to the legend with the **title** argument. I make the title bold face by using a mathematical expression and the **bold** function.
* The argument **bty='n'** suppresses the drawing of a box around the legend.



**Fig. 10** Mean profile plot for sibship 1

To plot the mean profiles for sibship 2 I make only three changes to the code listed above.

1. I subset the original data frame so sibship 2 (fac3=2) is selected instead of sibship 1.
2. I change the header on top of the graph to indicate that sibship 2 was plotted.
3. I drop the legend because it already appears in the first graph.

Notice that in the initial **plot** function I use all of the data to set the *y*-limits, not just the data for sibship 2. This ensures that the scale is the same in both graphs.

plot(c(.8,3.2), range(fac.vals[,c("up95", "low95")]), type='n', xlab="Hormonal treatment", ylab='Estimated mean', axes=F)

axis(1, at=1:3, labels=levels(fac.vals$fac1))

axis(2)

box()

dat1 <- fac.vals[fac.vals$fac3==2,]

**#first food group**

cur.dat <- dat1[dat1$fac2=='D',]

**#amount of shift**

myjitter <- -.05

arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05, col=1)

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=1, pch=16)

**#second food group**

cur.dat <- dat1[dat1$fac2=='S',]

myjitter <- .05

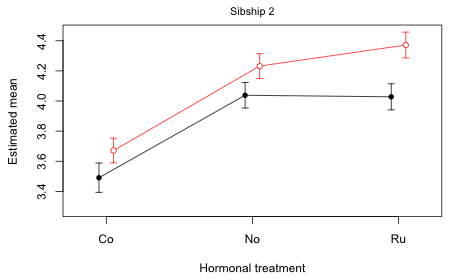
arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05, col=2)

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=2)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col='white', pch=16, cex=1.1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=2, pch=1)

mtext(side=3, line=.5, 'Sibship 2', cex=.9)



**Fig. 11** Mean profile plot for sibship 2

Finally I display both plots in the same graphics window side by side. For this I use the **mfrow** argument of **par** to divide the graphics window into one row and two columns.

par(mfrow=c(1,2))

plot(c(.8,3.2), range(fac.vals[,c("up95", "low95")]), type='n', xlab="Hormonal treatment", ylab='Estimated mean', axes=F)

axis(1, at=1:3, labels=levels(fac.vals$fac1))

axis(2)

box()

dat1 <- fac.vals[fac.vals$fac3==1,]

**#first food group**

cur.dat <- dat1[dat1$fac2=='D',]

**#amount of shift**

myjitter <- -.05

arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05, col=1)

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=1, pch=16)

**#second food group**

cur.dat<-dat1[dat1$fac2=='S',]

myjitter <- .05

arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05, col=2)

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=2)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col='white', pch=16, cex=1.1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=2, pch=1)

mtext(side=3, line=.5, 'Sibship 1', cex=.9)

legend('topleft',c('Detritus','Shrimp'), col=1:2, pch=c(16,1), cex=.8, pt.cex=.9, title=expression(bold('Food Type')), bty='n')

plot(c(.8,3.2), range(fac.vals[,c("up95", "low95")]), type='n', xlab="Hormonal treatment", ylab='Estimated mean', axes=F)

axis(1, at=1:3, labels=levels(fac.vals$fac1))

axis(2)

box()

dat1 <- fac.vals[fac.vals$fac3==2,]

**#first food group**

cur.dat <- dat1[dat1$fac2=='D',]

**#amount of shift**

myjitter <- -.05

arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05, col=1)

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=1, pch=16)

**#second food group**

cur.dat <- dat1[dat1$fac2=='S',]

myjitter <- .05

arrows(cur.dat$fac1.num+myjitter, cur.dat$low95, cur.dat$fac1.num+myjitter, cur.dat$up95, angle=90, code=3, length=.05, col=2)

lines(cur.dat$fac1.num+myjitter, cur.dat$est, col=2)

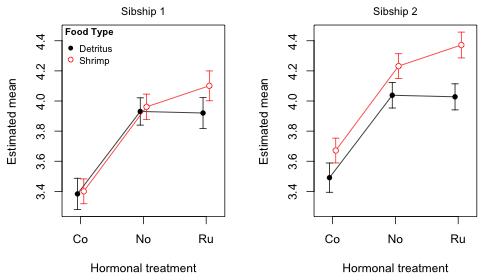
points(cur.dat$fac1.num+myjitter, cur.dat$est, col='white', pch=16, cex=1.1)

points(cur.dat$fac1.num+myjitter, cur.dat$est, col=2, pch=1)

mtext(side=3, line=.5, 'Sibship 2', cex=.9)

**#reset the graphics window**

par(mfrow=c(1,1))



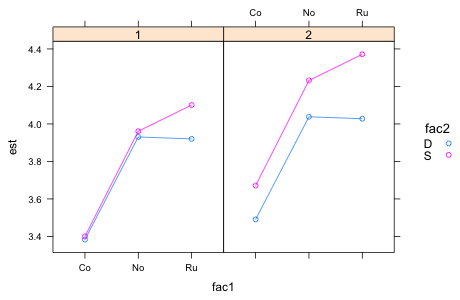
**Fig. 12** Mean profile plots for the tadpole experiment

Notice that although we have explicitly plotted only the fac1 × fac2 interaction, both interactions are clearly on display. The fac1 × fac2 interaction is obvious because in each panel the red and black profiles are not parallel, but we also can see evidence of the fac2 × fac3 interaction. If fac3 was a purely additive effect then the right panel would be identical to the left panel except for the fact that the profiles would be shifted up or down (the maine effect of fac3). Instead what we see is that the two diet profiles (fac2) are further apart in the right panel (sibship 2) than in the left panel (sibship 1). Thus sibship (fac3) is modifying the effect of diet. It has modified it sufficiently that now in the left panel the diet effect is significantly different from zero at all three hormonal treatments (the confidence intervals fail to overlap).

**Graphing the treatment means using the lattice package—Method 1 [not done in class]**

**Lattice** is the ideal package to use when the goal is to create a series of graphs that are variations on a common theme. In this case we want to generate mean profile plots separately for the levels of a third variable fac3 (sibship type). As is the norm with **lattice**, it is very easy to get a serviceable graph but much more difficult to get the exact graph you want. For example, the following function call generates separate mean profiles for each diet and separate panels for each sibship.

xyplot(est~fac1|factor(fac3), groups=fac2, data=fac.vals, type='b', auto.key=list(space="right", title='fac2', cex.title=1.1))



**Fig. 13** A serviceable lattice plot of the treatment means

* The notation est~fac1|factor(fac3) plots the estimated treatment mean against the levels of fac1 separately by the levels of fac3. The variable that appears to the right of the vertical bar, fac3, defines the panels.
* The argument **groups=fac2** generates the two profiles that appear within each panel differentiated by color.
* The argument **type='b'** causes both points and lines connecting the points to be drawn.
* The **auto.key** argument adds a legend that identifies the levels of the **groups** variable.

Unfortunately this is not the entire graph we wanted. We need to add error bars at each treatment mean and we need to have these drawn separately for each group (fac2). To accomplish this we're going to need to define our own panel function.

When the **groups** argument is included lattice calls a specific panel function called **panel.superpose**. The **panel.superpose** function in turn calls something called a **panel.groups** function that by default is **panel.xyplot** in which the points are colored separately by group. To draw the error bars we will need to create our own customized **panel.groups** function. Here's my first attempt at this function.

my.panel <- function(x, y, subscripts, col, pch,...) {

low95 <- fac.vals$low95[subscripts]

up95 <- fac.vals$up95[subscripts]

panel.xyplot(x, y, col=col, pch=pch, ...)

panel.arrows(x, low95, x, up95, angle=90, code=3, length=0.05, col=col)

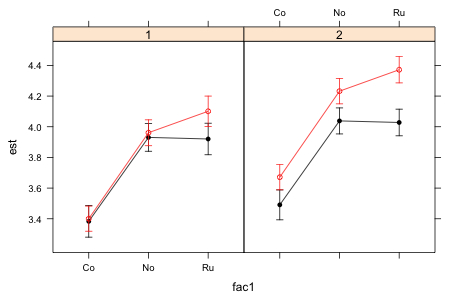
}

The **panel.xyplot** function was explained in an earlier section. The **panel.arrows** function is just the **lattice** version of the **arrows** function of base graphics.The main new feature in this code is the use of the **subscripts** argument to the panel function. When the panel function is called upon to draw a panel the variable **subscripts** contains the value of fac3 (the variable that defines the panels) for the current panel. So, in the above code the expressions fac.vals$low95[subscripts] and fac.vals$up95[subscripts] extract a vector of lower 95% endpoints and a vector of upper 95% endpoints corresponding to the level of fac3 that is currently being plotted. This is also the role the notation *x* and *y* have in the function. By default each time a panel is drawn the correct set of fac1 and est values (the *x* and *y* variables) are selected. Because low95 and up95 are not part of the original function call we have to extract the correct set of observations for the panel ourselves.

In addition I also pass to the group panel function the colors and plot characters I want it to use for the groups (the col and pch arguments in the function definition). Notice the ... notation that appears in a couple of places. It's used so that any additional arguments that **lattice** would normally needs in displaying groups get passed to the appropriate functions. Here's the complete function call using the panel function just created called "my.panel".

xyplot(est~fac1|factor(fac3), groups=fac2, data=fac.vals, type='b', col=c(1,2), pch=c(16,1), ylim=range(fac.vals[,'low95']-.1, fac.vals[,'up95']+.1), panel.groups="my.panel", panel="panel.superpose")

* **ylim=**range(fac.vals[,'low95']-.1, fac.vals[,'up95']+.1) decreases the minimum and increases the maximum *y*-limits a little bit to make room for the error bars.
* **panel.groups=**"my.panel" identifies the group panel function that I created.
* **panel=**"panel.superpose" calls the default panel function that will then call the **panel.groups** function.

  
**Fig. 14** A lattice plot with a customized group panel function

This is close to what we want except that the points still need to be jittered so that the error bars don't overlap. This can be accomplished by adding one more argument to **panel.groups** function. Just like the variable **subscripts** identifies the current panel that is being plotted, the variable **group.number** identifies the current group that is being plotted. Here's my modification of the **panel.groups** function that uses the **group.number** variable.

my.panel <- function(x, y, subscripts, col, pch, group.number, ...) {

low95 <- fac.vals$low95[subscripts]

up95 <- fac.vals$up95[subscripts]

myjitter <- c(-.05,.05)

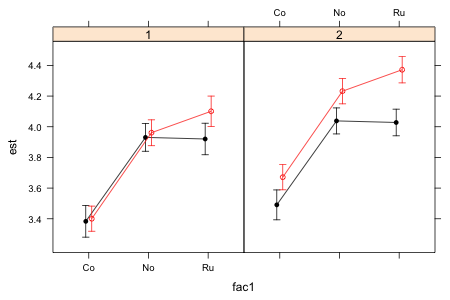
panel.xyplot(x+myjitter[group.number], y, col=col, pch=pch,...)

panel.arrows(x+myjitter[group.number], low95, x+myjitter[group.number], up95, angle=90, code=3, length=0.05, col=col)

}

The first change is that I create a vector of values I call myjitter that will be used to move the points to the left and right which I then add to the *x*-coordinate entries of the **panel.xyplot** and **panel.arrows** functions just as was done in the earlier base graphics code. The new wrinkle is that I use the **group.number** variable to select the appropriate jitter value for the group that is currently being plotted. The function call to produce the plot is unchanged.

xyplot(est~fac1|factor(fac3), groups=fac2, data=fac.vals, type='b', col=c(1,2), pch=c(16,1), ylim=range(fac.vals[,'low95']-.1, fac.vals[,'up95']+.1), panel.groups="my.panel", panel="panel.superpose")



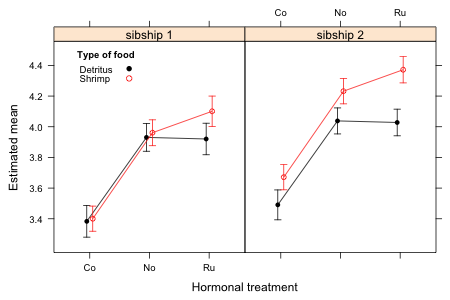
**Fig. 15** A lattice plot using the group.number variable

Finally I fix the labels on the *x*-axis, *y*-axis, and in the panels and I add a key. To fix the panel labels I use the **labels** argument of the **factor** function when declaring fac3 to be a factor. The legend is created with **key** argument.

xyplot(est~fac1| factor(fac3, levels=1:2, labels=c('sibship 1', 'sibship 2')), xlab="Hormonal treatment", ylab='Estimated mean', groups=fac2, data=fac.vals, type='b', col=c(1,2), pch=c(16,1), ylim=range(fac.vals[,'low95']-.1, fac.vals[,'up95']+.1), panel.groups="my.panel", panel="panel.superpose", key=list(x=.05, y=.75, corner=c(0,0), text=list(c('Detritus', 'Shrimp'), cex=.8), points=list(pch=c(16,1), col=1:2, cex=.9), title=expression(bold('Type of food')), cex.title=.8))

The **key** argument is mostly self-explanatory although a bit convoluted.

* The key itself is an R object called a list that is here created with the **list** function. A list object in R is one whose components can be of various types and of different lengths.
* The **text=** and **points=** arguments shown in the key are specified separately also using individual **list** functions. The elements of each list are the characteristics that I want the displayed text and symbols in the legend to have. For the **text** list I specify the vector of text that should appear in the legend and the size I want it to have. For the **points** list I specify the symbol types, colors and size.
* To position the legend inside the panels I give relative coordinates with the **x=** and **y=** arguments. Because I specified **corner=c(0,0)**, the **x=** and **y=** arguments are measured with respect to this corner, the lower left corner. The bottom of the figure is *y* = 0 and the top of the figure corresponds to *y* = 1. The left edge of the figure is *x* = 0 and the right edge of the figure corresponds to *x* = 1. By specifying **x=.05, y=.75** the legend is placed near the top left corner.

  
**Fig. 16** The final lattice plot

**Graphing the treatment means using the lattice package—Method 2 [not done in class]**

There's another way to plot separate mean profiles for each diet type (fac2) that avoids the use of the **groups** argument and the writing of special group panel functions. This method uses a **for** loop in the basic panel function. The one advantage of this approach is that it gives us more flexibility in drawing the separate graphs for each group. For instance, in the base graphics version of this plot I plotted the estimates in the mean profile for the shrimp diet twice, first using a white filled circle and the second time with an open red circle. I don't know of a way to do this with the **groups** setting because the colors that are used for the groups are defined outside of the **panel.groups** function and only one color is allowed per group. On the other hand it is possible to use different colors for the same group with a **for** loop in the main panel function. The entire code for doing this is shown below. Much of it is similar to what was used in the previous approach using a **groups** argument.

mypch<-c(16,1)

mycols<-1:2

shift<-c(-.05,.05)

xyplot(est~fac1|factor(fac3, levels=1:2, labels=c('sibship 1', 'sibship 2')), data=fac.vals, type='b', xlab='Hormonal treatment', ylab='Estimated mean', ylim=range(fac.vals[,'low95']-.1, fac.vals[,'up95']+.1), panel=function(x, y, subscripts, ...) {

**#obtain observations for current panel**

grpvals <- unique(fac.vals$fac2[subscripts])

grp <- fac.vals$fac2[subscripts]

low95 <- fac.vals$low95[subscripts]

up95 <- fac.vals$up95[subscripts]

for(i in 1:length(grpvals)) {

**#obtain observations for current group**

cur.x <- as.numeric(x[grp==grpvals[i]])+shift[i]

cur.y <- y[grp==grpvals[i]]

cur.low95 <- low95[grp==grpvals[i]]

cur.up95 <- up95[grp==grpvals[i]]

**#draw profile**

panel.lines(cur.x, cur.y, col=mycols[i])

panel.arrows(cur.x, cur.low95, cur.x, cur.up95, angle=90, code=3, length=0.05, col=mycols[i])

if(grpvals[i]=='S') panel.xyplot(cur.x, cur.y, col='white', pch=16)

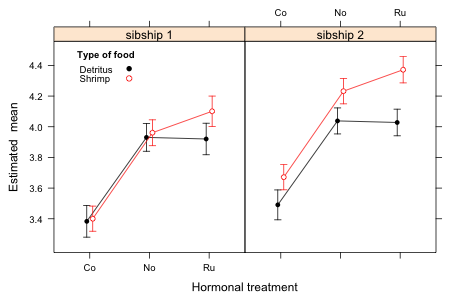
panel.xyplot(cur.x, cur.y, col=mycols[i], pch=mypch[i])

}

}, key=list(x=.05, y=.75, corner=c(0,0), text=list(c('Detritus', 'Shrimp'), cex=.8), points=list(pch=c(16,1), col=1:2, cex=.9), title=expression(bold('Type of food')), cex.title=.8))

The new features in this code are the following.

* The grp variable that is defined contains all the group labels (fac3=1 or fac3=2) for all observations that are plotted in the current panel.
* The grpvals variable on the other hand contains the unique list of group levels for the current panel. For these data grpvals = c('D', "S") and has the same value for both panels..
* I run through the **for** loop twice. This is the value of length(grpvals).
* The **panel.lines** function is used to connect the estimated means and generate the mean profile.
* Each time through the loop I first subset the data so that I'm only plotting the values for the current group. The Boolean condition grp==grpvals[i] does the selecting. The subsetted variables are named cur.x, cur.y, cur.low95, and cur.up95.
* Finally I use an **if** statement, if(grpvals[i]=='S'), to draw the filled white circles only for observations that were from the Shrimp diet.



**Fig. 17** A lattice plot that used a loop to draw the separate groups

**Cited references**

* Donahue, R. M. J. 2010. Fundamental Statistics Concepts in Presenting Data: Principles for Constructing Better Graphics. Version 1.6 <http://biostat.mc.vanderbilt.edu/wiki/pub/Main/RafeDonahue/fscipdpfcbg_currentversion.pdf>
* Gelman, Andrew and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press: New York.
* Magnusson, W. E. 2000. Error bars: are they the king's clothes? *Bulletin of the Ecological Society of America* **81**: 147–150. [http://www.esajournals.org/doi/pdf/10.1890/0012-9623%282000%29081%5B147%3AC%5D2.0.CO%3B2](http://www.esajournals.org/doi/pdf/10.1890/0012-9623(2000)081%5b147%3AC%5d2.0.CO;2)

**R Code**

A compact collection of all the R code displayed in this document appears [here](http://www.unc.edu/courses/2010fall/ecol/563/001/notes/lecture5b%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--September 18, 2010 URL: [http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm](http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture5.htm#lecture5) |