Stat 430 – Design of Experiments. Lab 9, Nov 19 2012

Agenda:

- 2^2 design and analysis. Linear only.
- Predictions and Filled Contour Maps
- 2^2 design and analysis. Interactions.
- 2^3 designs

2^2 design and analysis – Linear only.

Today we're going to start looking at Factorial design and a way to scale up to many variables at once. First, we have to start with factorial designs on the smallest scale possible: The 2^2 design. A factorial design is one in which every combination of levels has been observed for all the factors. For this reason, sample sizes can get very large very fast, so it's important to keep the number of levels low. That's why everything today is focussed on two-level factors.

- 2 Factors, both them of them potentially important (none are blocking factors).
- Each one at only two levels: High (+1), and Low (-1).

We code the levels as +1 and -1 so that the zero point is always between the two levels. For nominal data (categorical data with no particular ordering), 0 may not make sense, so the intercept in a linear model is assumed be a mathematical abstraction and nothing more.

Let's first consider a data set where we've looked at two factors (A and B) and every combination:

Response	A	B	Identifier
40	-	-	(1)
45	+	-	a
47	-	+	b
32	+	+	ab

The identifiers are multiplicative; they show all the factors for which that observation had the high level. Why they're called multiplicative and why they're useful will be covered in the next lab. For now they're just handy for referring to particular observations.

The effect size of each factor is the average difference observed when that factor is high rather than low. In other words: (Average at high level) – (Average at low level)

So the effect size of A is:

$A = (a + ab)/2 - ((1) + b)/2 = (45 + 32)/2 - (40 + 47)/2 = -5$

…and the effect size of B is:

B = (b + ab)/2 - ((1) + a)/2 = (47 + 32)/2 - (40 + 45)/2 = -3

This is the expected amount the response increases when moving from -1 to 1. When we put these in a linear model, the beta values are the effect sizes halved, because the beta values are the increase in response per unit increase in the explanatory value.

Beta1 = -2.5, Beta2 = -1.5

Finally, the intercept Beta0 is the global mean because our values are centered around (0,0).
Beta0 = ((1) + a + b + ab) / 4 = (40 + 45 + 47 + 32)/4 = 41

The model is:
y = beta0 + beta1*x1 + beta2*x2 + error
y = 41 -2.5*x1 – 1.5*x2 + error

Build some data (this is not in your text, but is based on the data given in part 5.1)

```
y = c(40,45,47,32)
A = rep( c(-1,1), each = 1, times=2)
B = rep( c(-1,1), each = 2, times=1)
wk09.data = data.frame(y,A,B)
```

Then we can model it. This should give us the same values

```
mod1 = lm(y ~ A + B, data=wk09.data)
mod1
```

Predictions and Filled Contour Maps

Visualizations of the data are handy. If there were levels beyond the ones we had chosen for the factors, or levels in between. Visualizations like contour maps of the response surface could show where such levels could produce optimal values of the response that we may not have found or considered in our original experiment.

Let's try predicting some of the middle values with predict.lm()

```
A = c(0,0)  # The new values to predict for factor A
B = c(0,0.5) # … and factor B
wk09.new1 = data.frame(A,B)
predict.lm(mod1,newdata=wk09.new1)
```

Let's try fitting a contour map to it, filling in the spots we don't have.
Since we’re only looking at linear factors, it shouldn’t be a surprise that the contours are just linear.

```
mod1.matrix = matrix(mod1$fitted, ncol=2)
filled.contour(mod1.matrix,x=c(-1,1), y=c(-1,1), xlim=c(-1,1),ylim=c(-1,1))
```

2^2 design and analysis – Interactions.

Now to expand the model as far as we can, which isn't much: The interaction of A and B.

If we include an interaction term, we will run out of df and won't be able to measure uncertainty. If all we care about is prediction this isn't always a problem.

As with the main effects the interaction effect size is found with (average of high level) – (average of low). What is a "high level" and "low level" of an interaction is the product of the x values that make it (a product of +1s and -1s is also +1 or -1).

Let's first consider a data set where we've looked at two factors (A and B) and every combination:

Response	A	B	**AB**	Identifier
40	-	-	**+**	(1)
45	+	-	**-**	a
47	-	+	**-**	b
32	+	+	**+**	ab

So effect size of AB is:
AB = ((1) + ab)/2 - (a + b)/2 = (40 + 32)/2 - (45 + 47)/2 = -10

…and beta12 is half of that because it's the per-unit change.

The model is now:
y = beta0 + beta1*x1 + beta2*x2 + beta12*x1*x2
y = 41 -2.5*x1 – 1.5*x2 – 5*x1*x2 + error

Let's put this into R.
A:B means include the interaction but not necessarily the main effects that it's made of.
Here it's the same as A*B because A and B are already included, but in case we didn't want them, like in the 2^3 model below, using A:B is necessary.

```
mod2 = lm(y ~ A + B + A:B, data=wk09.data)
mod2
```

If we just use the fitted values again, we may see some bending in the response surface, but it won't have any details because all we have are corner points.

```
mod2.matrix = matrix(mod2$fitted, ncol=2)
```

```
filled.contour(mod2.matrix,x=c(-1,1), y=c(-1,1), xlim=c(-1,1),ylim=c(-1,1))
```

Try other colours if blue and pink aren't working well. (heat.colors, terrain.colors)

```
filled.contour(mod2.matrix,x=c(-1,1), y=c(-1,1), xlim=c(-1,1),ylim=c(-1,1)
, color=terrain.colors)
```

Part of the problem with this analysis is that we're really only looking at the corner points. There's nothing in between to give us an idea of what's going in the space between -1 and 1. We can interpolate with the predict.lm() function and get a better picture of the surface our model is implying.

Make a 200x200 grid in the +/- 1 square:

```
points = seq(from=-1, to=1, by=0.01)
A = rep(points, each=length(points))
B = rep(points, times=length(points))
```

Use these as points for predictions

```
wk09.new2 = data.frame(A,B)
interpolations = predict.lm(mod2,newdata=wk09.new2)
```

Make a matrix of predictions and put them in the filled contour function

```
interp.matrix = matrix(interpolations, ncol=length(points))
filled.contour(interp.matrix,x=points, y=points, xlim=c(-1,1),ylim=c(-1,1))Code

filled.contour(interp.matrix,x=points, y=points, xlim=c(-1,1),ylim=c(-1,1),
color=terrain.colors)
```

2^3 designs

Let's try another, but this time with three factors, each at 2 levels. (This is jumping ahead to Chapter 6.2, but I think it flows well).

Response	A	B	C	Identifier
13	-	-	-	(1)
31	+	-	-	a
8	-	+	-	b
33	+	+	-	ab
32	-	-	+	c
15	+	-	+	ac
31	-	+	+	bc
13	+	+	+	abc

All of the main effect sizes are computed as they were in the 2^2 case:

Effect of C:

C = (c + ac + bc + abc)/4 – ((1) + a + b + ab)/4 = (32 + 15 + 31 + 13 – 13 -31 – 8 – 33)/4= 1.5.

Here is the code to enter this data into R…

```
y = c(13, 31, 8, 33, 32, 15, 31, 13)
A = rep( c(-1,1), each = 1, times=4)
B = rep( c(-1,1), each = 2, times=2)
C = rep( c(-1,1), each = 4, times=1)
wk09.data2 = data.frame(y,A,B,C)
```

…and to construct a model of the main effects.

```
mod3 = lm(y ~ A + B + C, data=wk09.data2)
mod3
```

The interaction effects behave the same way as well, including the third-order interaction effects. For anything third-order or higher, a table like the following can be very useful.

Response	A	B	C	**AB**	**ABC**	Identifier
13	-	-	-	**+**	**-**	(1)
31	+	-	-	**-**	**+**	a
8	-	+	-	**-**	**+**	b
33	+	+	-	**+**	**+**	ab
32	-	-	+	**+**	**-**	c
15	+	-	+	**-**	**-**	ac
31	-	+	+	**-**	**-**	bc
13	+	+	+	**+**	**+**	abc

Effect size of AB

AB = (13 + 33 + 32 + 13)/4 - (31 + 8 + 15 + 31)/4 = 1.5

Effect size of ABC

ABC = (31 + 8 + 33 + 13)/4 - (13 + 32 + 15 + 31)/4 = -2

In this model, notice that including the interactions doesn't change any of the estimates that were put in before. This isn't the general case for linear models, but is a result of the orthogonal way in which the experiment was set up.

```
mod4 = lm(y ~ A + B + C + A:B + A:B:C, data=wk09.data2)
mod4
```