```
> moedata <- data.frame(moe=c(308,278,428,398,426,331,
           214,534,433,512,231,320,272,158,376,503,322,220),
  size=factor(rep(c(.015,.025),9)),
  species=factor(rep(c("aspen","birch","maple"),each=6)))
```

These data are from Moore and McCabe (1999) problem 13.11. An experiment studies the modulus of elasticity of wood chips made from three different species of wood cut to two different sizes. Two sizes (.015 inch thick or .025 inch thick). Three species, aspen, birch, and maple.

```
> moedata
   moe  size species
1  308 0.015   aspen
2  278 0.025   aspen
3  428 0.015   aspen
. . .
```

```
> meansa <- with(moedata, tapply(moe,list(species,size),mean));meansa
```

The function `tapply()` allows us to get summaries of its first argument by the levels of the factors listed in its second argument. The result is a vector (for one factor), a matrix (for two factors), and so on with dimensions corresponding to the numbers of levels of the factors in the list. The third argument is the function you want applied to the groups after the data have been split, in this case, the mean. These are the means that we will plot in interaction plots.

```
         0.015    0.025
aspen 387.3333 335.6667
birch 292.6667 455.3333
maple 323.3333 293.6667
```

```
> meansb <- with(moedata, tapply(moe,list(size,species),mean));meansb
```

We can do it the other way too.

```
         aspen    birch    maple
0.015 387.3333 292.6667 323.3333
0.025 335.6667 455.3333 293.6667
```

```
> t(meansa)
```

Of course, one is just the transpose of the other.

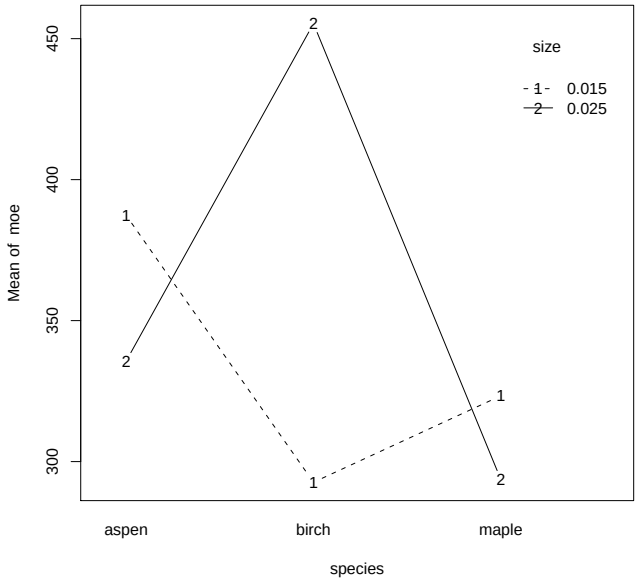
```
         aspen    birch    maple
0.015 387.3333 292.6667 323.3333
0.025 335.6667 455.3333 293.6667
```

```
> with(moedata, interactplot(species,size,moe))
```

You can get interaction plots using the `interactplot` function in the Stat5303 package. Base R includes a interaction.plot function, and the gplots package includes a plotmeans function. `interactplot` is based on interaction.plot but contains some of the features of plotmeans.

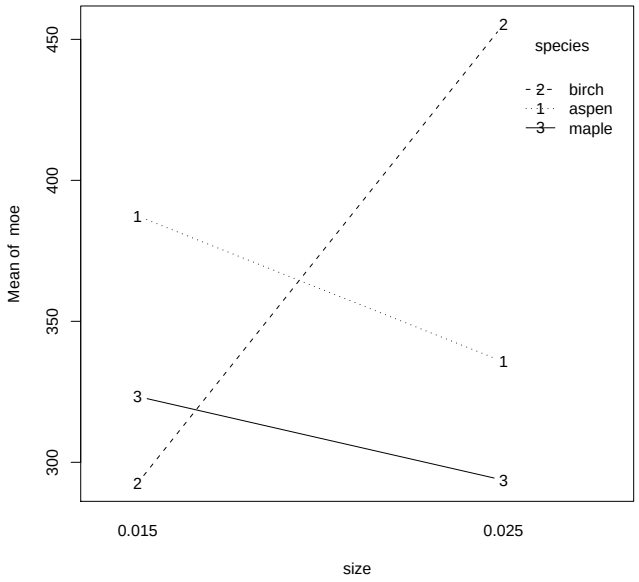
The way it works is that the first argument is the factor used along the horizontal axis, the second argument defines the traces that get drawn, and the third argument gives the response to be averaged. There are a *lot* more optional arguments, some of which we will eventually use.

This *appears* to be a whopping big interaction. Birch seems to act differently than aspen or maple.



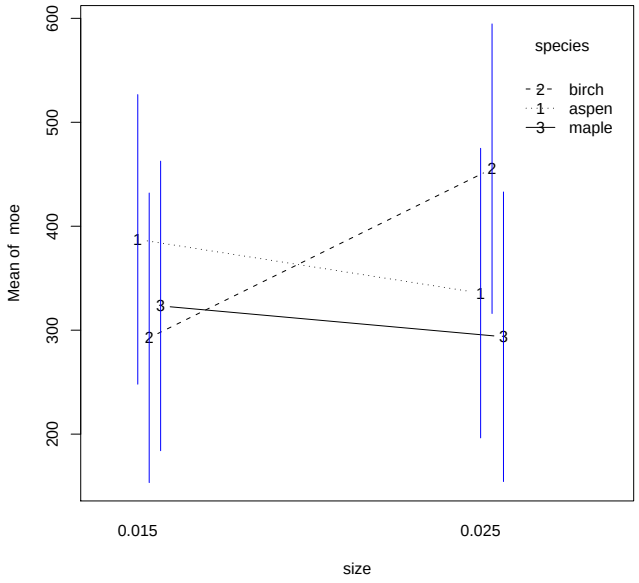

```
> with(moedata, interactplot(size,species,moe))
```

Interaction plots can look very differently depending on which variable is put on the x-axis and which is used to label the lines. Here we reverse the order, and I find this plot more understandable than the first one.



```
> with(moedata, interactplot(size,species,moe,confidence=.95))
```

You can also add confidence intervals for each mean. By default we use a pooled estimate of error computed by pooling the variances within each of the groups (essentially the MSE if we had done the corresponding ANOVA). If you use `pooled=FALSE`, you get confidence intervals based on the variance in each group. (These will have fewer degrees of freedom, and the t-coeffiencts will be much greater because of that.)
In this example we see that the apparent interaction was really just random variation.




```
> irondata <- data.frame(iron.in.liver=c(.70,.93,2.11,1.28,1.87,2.53),
  diet=factor(rep(c("skim","whey","casein"),2)),
  cu=factor(rep(c("control","deficient"),rep(3,2))))
```

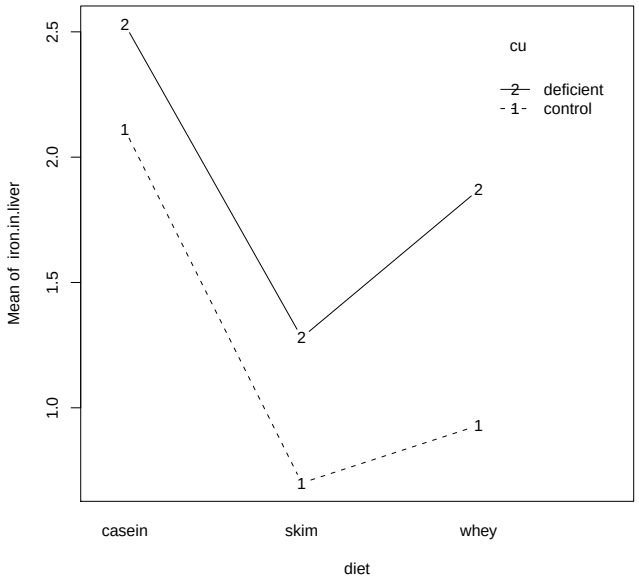
These are the data shown in Table 8.5, with the copper and diet factors.

```
> irondata
  iron.in.liver   diet        cu
1          0.70   skim   control
2          0.93   whey   control
3          2.11 casein   control
4          1.28   skim deficient
5          1.87   whey deficient
6          2.53 casein deficient
```

```
> with(irondata, interactplot(diet,cu,iron.in.liver))
```

Not much interaction, but of course we don't have a measure of error for comparison.

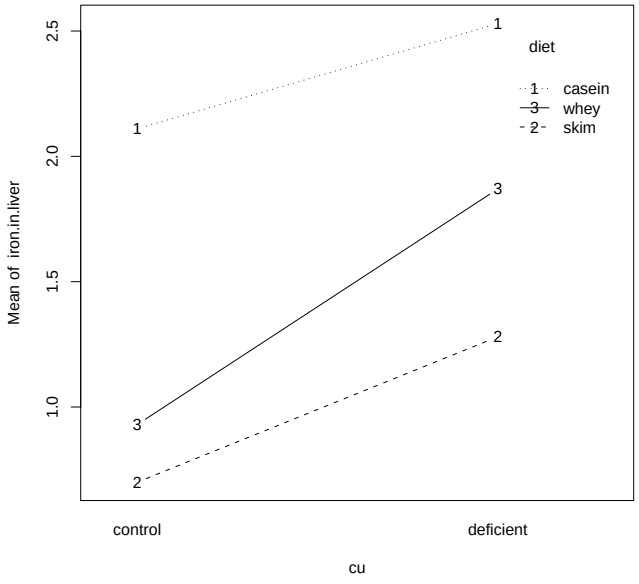
```
Warning message:
In interactplot(diet, cu, iron.in.liver) :
  no degrees of freedom for estimating error
```




```
> with(irondata, interactplot(cu,diet,iron.in.liver))
```

Not much different this way

```
Warning message:
In interactplot(cu, diet, iron.in.liver) :
  no degrees of freedom for estimating error
```



```
> chickdata <- data.frame(weight=c(584,489,453,616,606,621),
  ca=factor(c(1,2,3,1,2,3)),
  p=factor(c(1,1,1,2,2,2)))
```

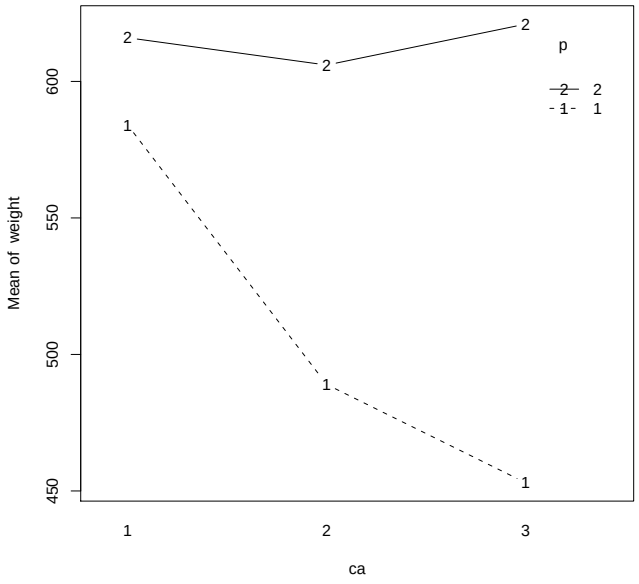
Nelson et al chick weight data, just group means.

```
> chickdata
  weight ca p
1    584  1 1
2    489  2 1
3    453  3 1
4    616  1 2
5    606  2 2
6    621  3 2
```

```
> with(chickdata, interactplot(ca,p,weight))
```

For phosphorus 1, means change with levels of calcium, but they are fairly steady for phosphorus 2.

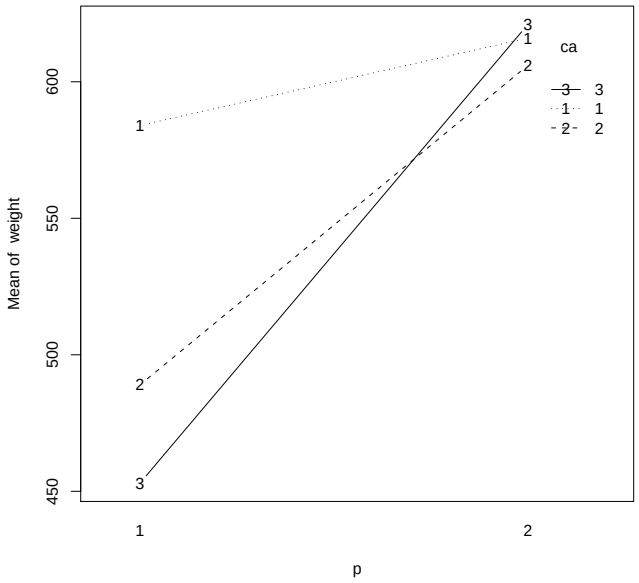
```
Warning message:
In interactplot(ca, p, weight) : no degrees of freedom for estimating error
```



> **with(chickdata, interactplot(p,ca,weight))**

This is the same information the other way. I find the first plot easier to interpret.

```
Warning message:
In interactplot(p, ca, weight) : no degrees of freedom for estimating error
```



> **outmoe <- lm(moe~species*size,data=moedata); anova(outmoe)**

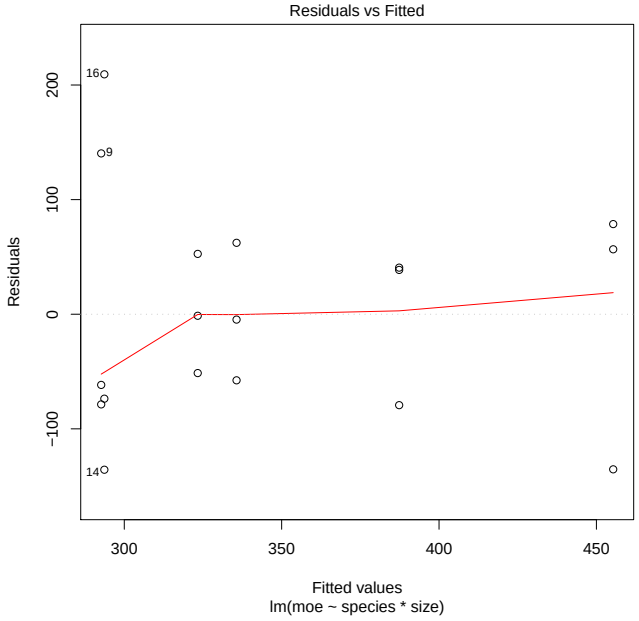
OK, back to the moe data. Here is the ANOVA. *None* of the effects is significant. What we saw in the interaction plot is not distinguishable from noise with this sample size. Let's say it again. You can't fully interpret an interaction plot without some idea of the standard error of the treatment means. Conversely, even a significant F test tells you nothing about what is happening within an interaction.

```
Analysis of Variance Table

Response: moe
             Df Sum Sq Mean Sq F value Pr(>F)
species       2  14511    7256  0.5917 0.5687
size          1   3308    3308  0.2698 0.6129
species:size  2  41707   20854  1.7007 0.2237
Residuals    12 147138   12261
```

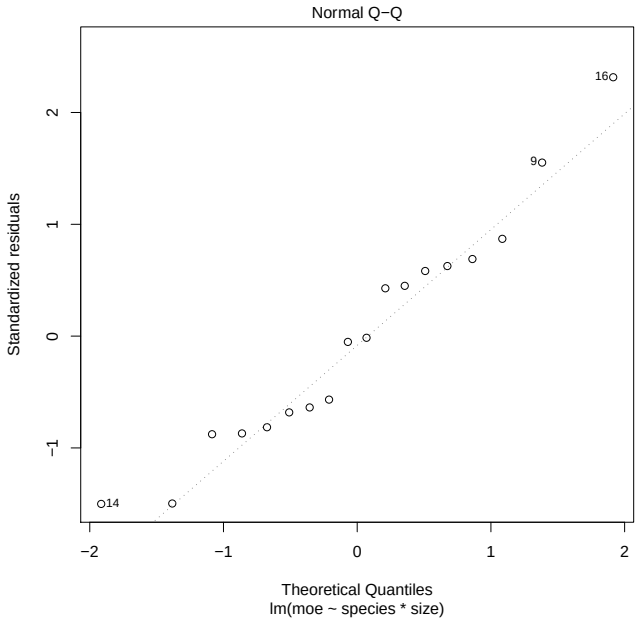
> **plot(outmoe,which=1)**

Just a hint of decreasing variance?



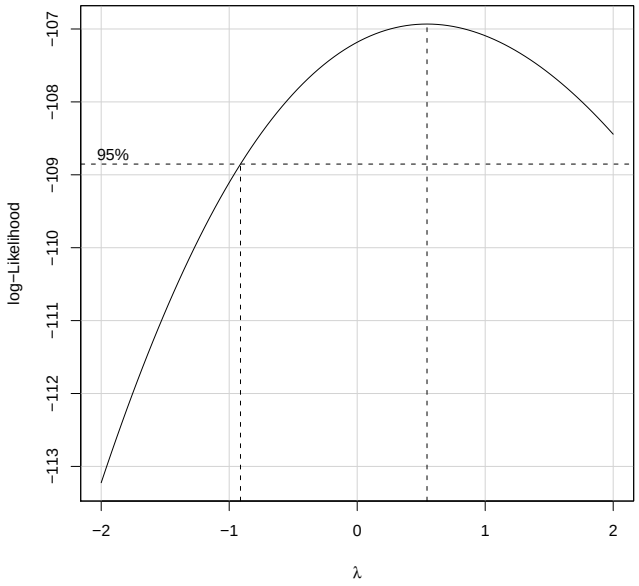
> **plot(outmoe,which=2)**

A little skewed to the right, but not too bad.



> **boxCox(outmoe)**

We can check Box-Cox, but it does not suggest a transformation (the acceptable range is from about -1 to more than 2). This is not surprising for two reasons. First, there was not much in the residual plot to suggest nonconstant variance. Second, there is not much dynamic range in the data (ratio of max to min for moe is under 3.5), so even if there was nonconstant variance, a power transformation would not be of much help.



>>>> **with(moedata, tapply(moe,list(size,species),var))**

Here are the sample variances within each treatment (factor/level combination), each with 2 df.

```
          aspen    birch     maple
0.015 4721.333 14842.33  2705.333
0.025 3616.333 13857.33 33826.333
```

>>>> **33826/2705**

I know I told you not to do F-tests with variances, but I wanted to show you that with few degrees of freedom, even big variance ratios such as this one are not particularly significant, even assuming that all the assumptions are met. This is the greatest ratio.

```
[1] 12.50499
```

>>>> **pf(12.5,2,2,lower.tail=FALSE)**

This is not very significant, and that is before any adjustment for the fact that we searched to find the largest possible variance ratio.

```
[1] 0.07407407
```

```
> model.effects(outmoe,"species")
```

The Stat5303 package has a function to extract the model effects out in a somewhat more comprehensible way. In particular, if the effects are supposed to add to zero, then they add to zero.

Please note that in some cases R just sets things up without using the "add to zero" that we like. You will see this when lower order terms are missing or sometimes when you don't use an intercept. model.effects() tries to account for that, but it's very tricksy and this is a new function I just wrote. Let me know if you think it's doing the wrong thing

```
aspen birch maple
 13.5  26.0 -39.5

> model.effects(outmoe,"size")
    0.015     0.025
-13.55556  13.55556

> model.effects(outmoe,"species:size")
          0.015     0.025
aspen  39.38889 -39.38889
birch -67.77778  67.77778
maple  28.38889 -28.38889

> model.effects(outmoe,"size:species")
```

You need to enter the term exactly as it appears in the model, ie, you have to have the variables in the correct order or it won't find the term.

```
Error in model.effects(out, "size:species") :
  size:species is not an explicit term in the model

> effect("species:size",outmoe)
```

The `effect` function comes from the "effects" package. I don't like the name effect, but I didn't write it so I just live with it. What it's really doing is giving you predictions for various combinations of treatments. In this case, it's just the treatment means.

```
 species*size effect
       size
species    0.015    0.025
  aspen 387.3333 335.6667
  birch 292.6667 455.3333
  maple 323.3333 293.6667

> effect("species:size",lm(moe~species+size,data=moedata))
```

It's also possible to ask for a combination of factors that isn't in the model, provided that all of the factors are in the model. Here we fit a model with just main effects and no interactions but ask for the two way table of predictions. Notice that they are not equal to the previous set of predictions, because they do not take interaction into account. The function warns you if you do this.

```
 species*size effect
       size
species    0.015    0.025
  aspen 347.9444 375.0556
  birch 360.4444 387.5556
  maple 294.9444 322.0556
Warning message:
```

```
In analyze.model(term, mod, xlevels, default.levels) :
  species:size does not appear in the model
```

```
> effect("species",outmoe)
```

You can ask for a prediction just on species, which averages across any other factors in the model (in this case, size). This can be very misleading if there is a large interaction, so the function warns you about this.

```
 species effect
species
aspen birch maple
361.5 374.0 308.5
Warning message:
In analyze.model(term, mod, xlevels, default.levels) :
  species is not a high-order term in the model
```

```
> allEffects(outmoe)
```

This function gives you all of the prediction tables from the model that aren't included in some other interaction. In this case, it's just the two-way interaction, but more complex models could have more than one table. For example, in a three way model, if you just include two of the two factor interactions and omit the three factor, then you will have two two-way tables corresponding to the two interactions.

```
 model: moe ~ species * size

 species*size effect
       size
species    0.015    0.025
  aspen 387.3333 335.6667
  birch 292.6667 455.3333
  maple 323.3333 293.6667
```

```
> allEffects(lm(moe~species+size,data=moedata))
```

Here's an example with allEffects showing multiple prediction tables. In this case, we just fit main effects, and since neither main effect is included in an interaction, it predicts along both main effects.

```
 model: moe ~ species + size

 species effect
species
aspen birch maple
361.5 374.0 308.5

 size effect
size
   0.015    0.025
334.4444 361.5556
```

```
> zinc.data <- read.table("http://www.stat.umn.edu/~gary/book/fcdae.data/exmpl8.5",
+ header=TRUE)
```

This is the Hunt and Larson liver zinc data. This will be our first more-than-two-way example.

```
> names(zinc.data)
[1] "mp" "mz" "dz" "y"
```

> **`zinc.data <- within(zinc.data, {mp <- factor(mp);mz<-factor(mz);dz<-factor(dz)})`**

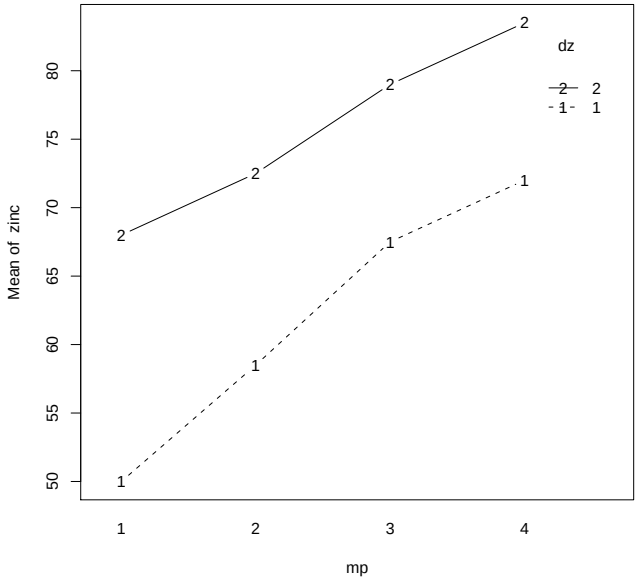
Make factors.

> **`zinc.data <- within(zinc.data, zinc <- y)`**

I'm going to rename this so that it's easier to remember what we are measuring.

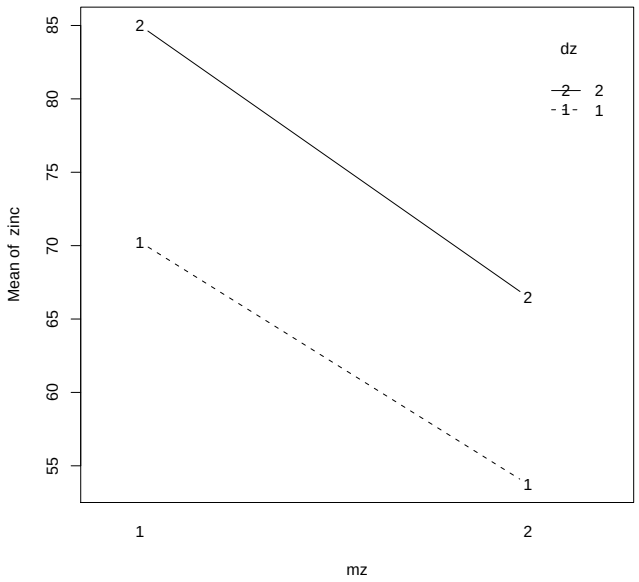
> **`with(zinc.data, interactplot(mp,dz,zinc))`**

Not much interaction between meal protein and diet zinc.



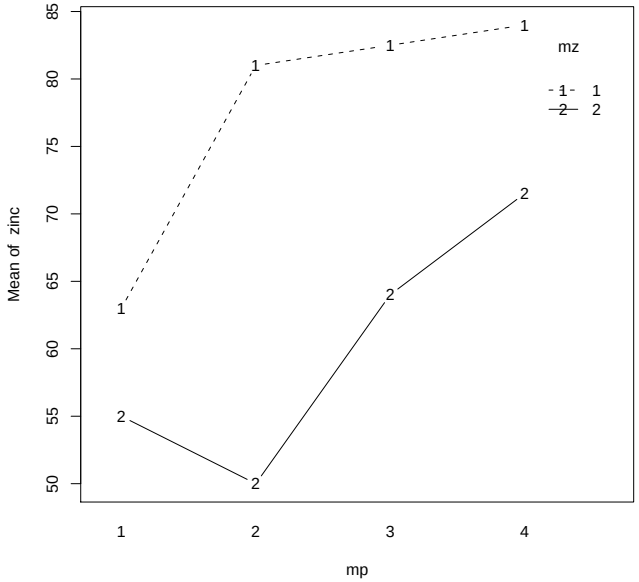
> **`with(zinc.data, interactplot(mz,dz,zinc))`**

Not much interaction between meal zinc and diet zinc either.



>**with(zinc.data, interactplot(mp,mz,zinc))**

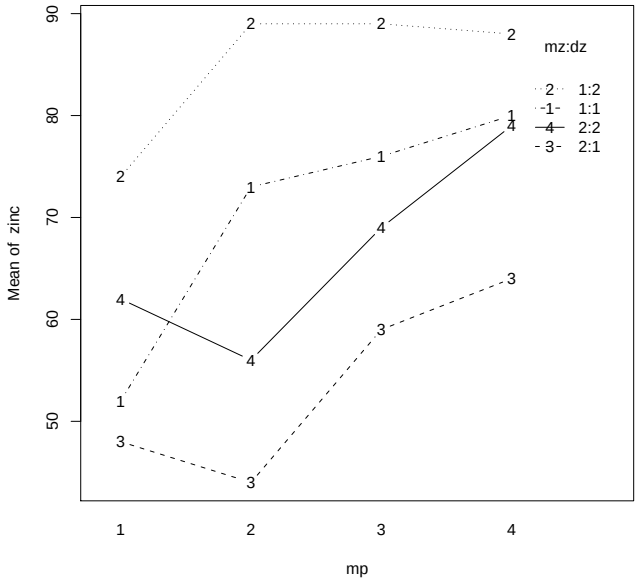
It looks like the effect of meal zinc varies by protein, with perhaps a bigger effect at protein 2 and a smaller effect at protein 1.



> **with(zinc.data, interactplot(mp,mz:dz,zinc))**

Here we look at protein on the horizontal axis, with separate lines for all the meal zinc by diet zinc combinations. The lines are numbered with the meal zinc/diet zinc combinations. This just reinforces the earlier impression that protein and meal zinc interact, but nothing else.

```
Warning message:
In interactplot(mp, mz:dz, zinc) :
  no degrees of freedom for estimating error
```



```
> wire <- read.table("carbonwire.txt",header=TRUE)
```

These are data from a three-factor factorial with four replications. We are interested in whether the treatments affect the resistivity of a carbon wire.

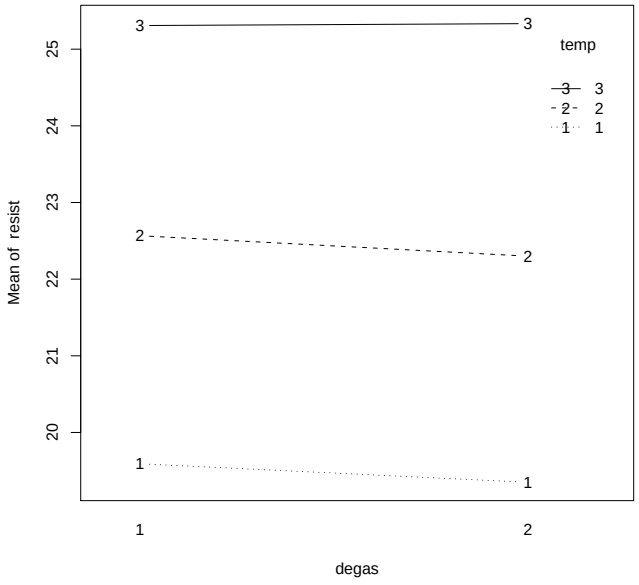
```
> names(wire)
[1] "resist" "degas"  "temp"   "diff"
```

```
> wire <- within(wire, {degas<-factor(degas);temp<-factor(temp);diff<-factor(diff)})
```

Make factors.

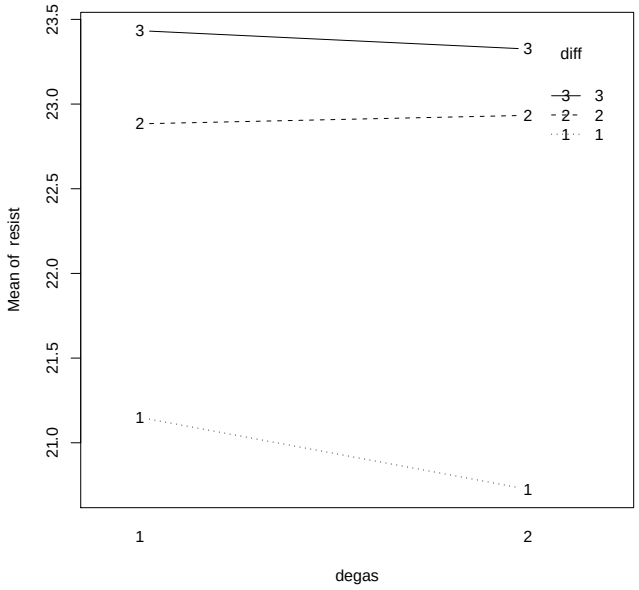
```
> with(wire, interactplot(degas,temp,resist))
```

Here are some interaction plots; not much interaction between degassing and temperature.



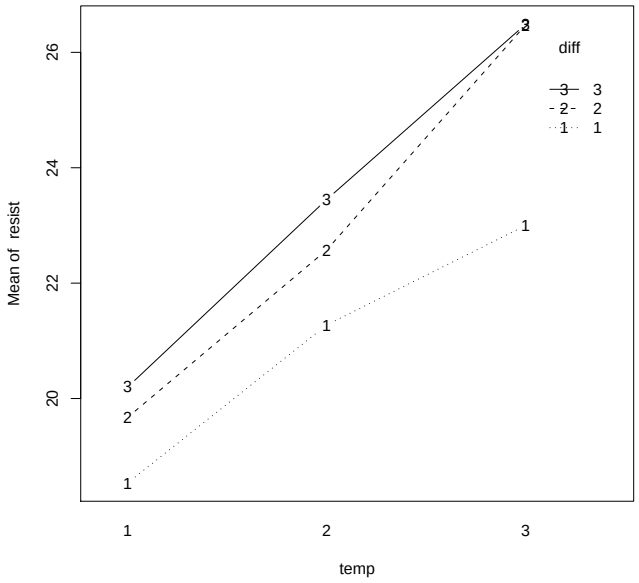
> **`with(wire, interactplot(degas,diff,resist))`**

Not much interaction between degassing and time.



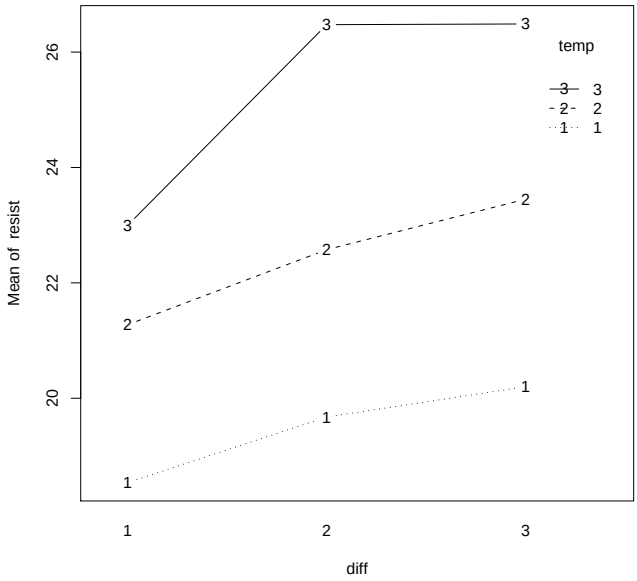
> **`with(wire, interactplot(temp,diff,resist))`**

Looks like there might be some interaction between time and temperature.



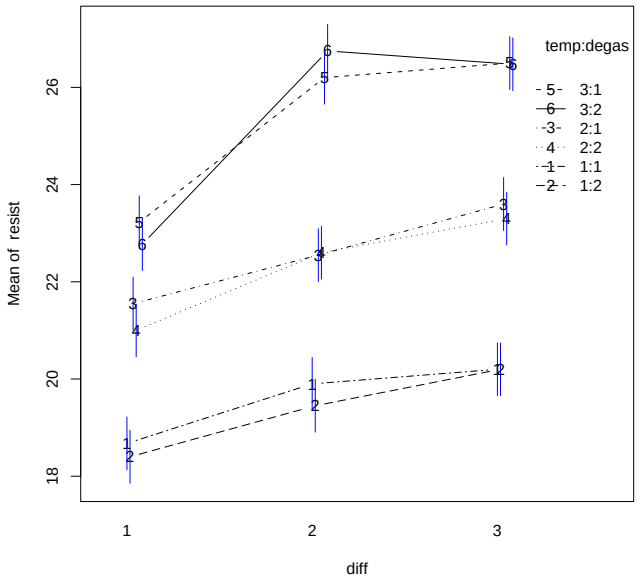
```
> with(wire, interactplot(diff,temp,resist))
```

The time and temperature interaction seems more clear (to me at least) with time on the horizontal.




```
> with(wire, interactplot(diff,temp:degas,resist,confidence=.95))
```

Not much sign of a three factor interaction.



```
> outwire <- lm(resist~degas*temp*diff,data=wire);anova(outwire)
```

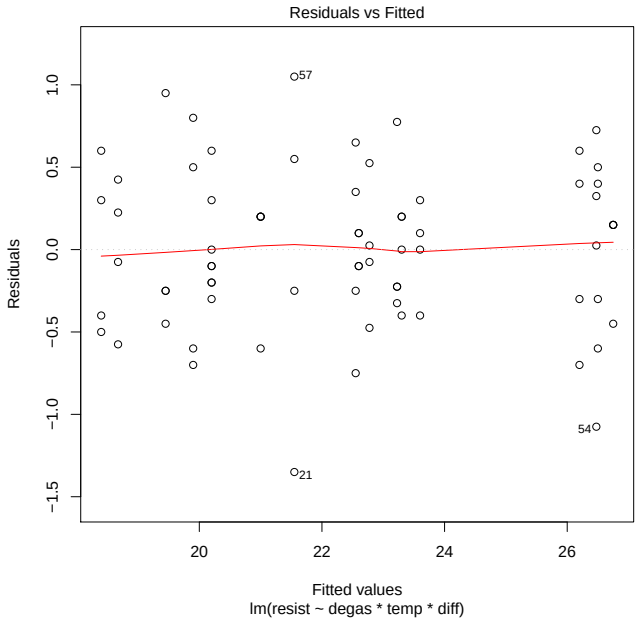
Now we fit the full three-way model. Nothing involving degas is significant, but the others are. But let's look at residuals before we go too far along.

```
Analysis of Variance Table

Response: resist
               Df Sum Sq Mean Sq  F value    Pr(>F)
degas           1   0.47    0.47   1.5918    0.2125
temp            2 410.69  205.35 699.6024 < 2.2e-16 ***
diff            2  80.54   40.27 137.1989 < 2.2e-16 ***
degas:temp      2   0.31    0.16   0.5342    0.5892
degas:diff      2   0.70    0.35   1.1957    0.3104
temp:diff       4  14.81    3.70  12.6177 2.566e-07 ***
degas:temp:diff 4   0.87    0.22   0.7450    0.5656
Residuals      54  15.85    0.29
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

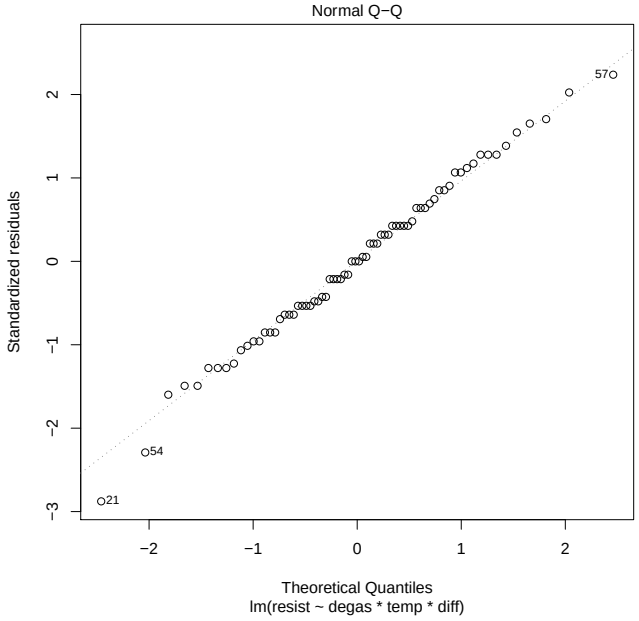
```
> plot(outwire,which=1)
```

Variance is fine.



> **plot(outwire,which=2)**

Normality is incredibly good.



> **anova(lm(resist~(degas+temp+diff)^2,data=wire))**

The exponent form only adds interactions up to the level of the exponent.
This is done here just to demonstrate the exponent form; there is no need to remove the three factor interaction.

```
Analysis of Variance Table

Response: resist
           Df Sum Sq Mean Sq  F value    Pr(>F)
degas       1   0.47    0.47   1.6203    0.2081
temp        2 410.69  205.35 712.1244 < 2.2e-16 ***
diff        2  80.54   40.27 139.6546 < 2.2e-16 ***
degas:temp  2   0.31    0.16   0.5438    0.5835
degas:diff  2   0.70    0.35   1.2171    0.3035
temp:diff   4  14.81    3.70  12.8436 1.499e-07 ***
Residuals  58  16.72    0.29
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> anova(lm(resist~degas*temp*diff-degas:temp:diff,data=wire))
```

You can also "subtract" a term that you previously entered into the model.

```
Analysis of Variance Table

Response: resist
           Df Sum Sq Mean Sq  F value    Pr(>F)
degas       1   0.47    0.47   1.6203    0.2081
temp        2 410.69  205.35 712.1244 < 2.2e-16 ***
diff        2  80.54   40.27 139.6546 < 2.2e-16 ***
degas:temp  2   0.31    0.16   0.5438    0.5835
degas:diff  2   0.70    0.35   1.2171    0.3035
temp:diff   4  14.81    3.70  12.8436 1.499e-07 ***
Residuals  58  16.72    0.29
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> anova(lm(resist~temp+diff+temp:diff+degas,data=wire))
```

Even if you write the terms out in a specific order, R is going to rearrange them to put lower order terms first.

For balanced data, order doesn't matter. But it will matter when we get to unbalanced data.

```
Analysis of Variance Table

Response: resist
          Df Sum Sq Mean Sq  F value    Pr(>F)
temp       2 410.69  205.35 717.6589 < 2.2e-16 ***
diff       2  80.54   40.27 140.7400 < 2.2e-16 ***
degas      1   0.47    0.47   1.6329    0.2061
temp:diff  4  14.81    3.70  12.9434 1.014e-07 ***
Residuals 62  17.74    0.29
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> anova(lm(terms(resist~temp+diff+temp:diff+degas,keep.order=TRUE),data=wire))
```

It's possible to make R keep your order, but you have to work harder.

```
Analysis of Variance Table

Response: resist
          Df Sum Sq Mean Sq  F value    Pr(>F)
temp       2 410.69  205.35 717.6589 < 2.2e-16 ***
diff       2  80.54   40.27 140.7400 < 2.2e-16 ***
temp:diff  4  14.81    3.70  12.9434 1.014e-07 ***
degas      1   0.47    0.47   1.6329    0.2061
Residuals 62  17.74    0.29
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> anova(lm(terms(resist~temp:diff+temp+diff+degas,keep.order=TRUE),data=wire))
```

If you force the order with higher order terms before included lower order terms, the included lower order terms get sucked up into the higher order term and then removed as explicit terms.

```
Analysis of Variance Table

Response: resist
          Df Sum Sq Mean Sq  F value Pr(>F)
temp:diff  8 506.05   63.26 221.0714 <2e-16 ***
degas      1   0.47    0.47   1.6329 0.2061
Residuals 62  17.74    0.29
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> model.effects(outwire,"temp")
```

We can get the model effects. Here are the significant main effects and interactions.

```
      1       2       3
-2.9375  0.0250  2.9125
```

```
> model.effects(outwire,"diff")
         1          2          3
-1.4708333  0.5000000  0.9708333
```

```
> model.effects(outwire,"temp:diff")
        1          2           3
1  0.5375 -0.2958333 -0.24166667
2  0.3125 -0.3583333  0.04583333
3 -0.8500  0.6541667  0.19583333
```

```
> #
```

Some earlier versions of effect() and allEffects() did not work on this model. but they both appear to work this year. It turns out that the name "diff" was the problem. Presumably there are other names that could be a problem as well, so consider renaming if you have trouble in the future.

```
> effect("temp:diff",outwire)
```

This did not work last year.

```
 temp*diff effect
    diff
temp       1      2       3
   1 18.5375 19.675 20.2000
   2 21.2750 22.575 23.4500
   3 23.0000 26.475 26.4875
Warning message:
In analyze.model(term, mod, xlevels, default.levels) :
  temp:diff is not a high-order term in the model
```

```
> allEffects(outwire)
```

Three tables, one for each value of diff.

```
 model: resist ~ degas * temp * diff

 degas*temp*diff effect
, , diff = 1

     temp
degas      1     2      3
    1 18.675 21.55 23.225
    2 18.400 21.00 22.775

, , diff = 2

     temp
degas     1     2     3
    1 19.90 22.55 26.20
    2 19.45 22.60 26.75

, , diff = 3

     temp
degas    1    2      3
    1 20.2 23.6 26.500
    2 20.2 23.3 26.475
```

```
> out3 <- lm(resist~temp*diff);anova(out3)
```

Here we have refit omitting the nonsignificant terms. This ANOVA is not needed, and, in fact, I would consider it inappropriate; I include it only to illustrate that the sums of squares and coefficients in a balanced factorial do not depend on which terms are in the model. The line in the anova for Residuals has changed, because it is now the pooling of the former Residuals line with all of the effects involving degas.

```
Analysis of Variance Table

Response: resist
          Df Sum Sq Mean Sq F value    Pr(>F)
temp       2 410.69  205.35 710.521 < 2.2e-16 ***
diff       2  80.54   40.27 139.340 < 2.2e-16 ***
temp:diff  4  14.81    3.70  12.815 1.085e-07 ***
Residuals 63  18.21    0.29
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> model.effects(out3,"temp")
      1       2       3
-2.9375  0.0250  2.9125
```

```
> anova(lm(zinc~(mp+dz+mz)^2,data=zinc.data))
```

What if we wanted to do ANOVA on the meal zinc data? That data set has just a single replication, so our first step is to remove the three factor interaction from the model and use that as a surrogate error. Here it looks like significant main effects and one marginally significant two factor interaction.

```
Analysis of Variance Table

Response: zinc
          Df  Sum Sq Mean Sq  F value   Pr(>F)
mp         3  827.25  275.75  27.1230 0.011261 *
dz         1  756.25  756.25  74.3852 0.003278 **
mz         1 1225.00 1225.00 120.4918 0.001619 **
mp:dz      3   28.25    9.42   0.9262 0.524375
mp:mz      3  298.50   99.50   9.7869 0.046563 *
dz:mz      1    4.00    4.00   0.3934 0.574988
Residuals  3   30.50   10.17
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> anova(lm(zinc~(mp+dz+mz)+mp:mz,data=zinc.data))
```

Because we only have 3 df for "error," it makes sense to pool more terms into error if we can. Here both mp:dz and dz:mz have very small F statistics (less than 1), so we pool them into error. There MSE does not change much, but the p-values are much smaller, because going from 3 df for error to 7 df for error is huge.

```
Analysis of Variance Table

Response: zinc
          Df  Sum Sq Mean Sq F value    Pr(>F)
mp         3  827.25  275.75  30.761 0.0002106 ***
dz         1  756.25  756.25  84.362 3.736e-05 ***
mz         1 1225.00 1225.00 136.653 7.578e-06 ***
mp:mz      3  298.50   99.50  11.100 0.0047289 **
Residuals  7   62.75    8.96
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> pagefaults <- read.table("http://www.stat.umn.edu/~gary/book/fcdae.data/exmpl8.8",
header=TRUE)
```

This is the page faults data. Numbers of faults depending on how the computer is set up. It has a single replication.

```
> names(pagefaults)
```

What's there?

```
[1] "alg"   "seq"   "psize" "ram"   "y"
> pagefaults <- within(pagefaults, {alg <- factor(alg);sq <- factor(seq);
      psize <- factor(psize); ram <- factor(ram); faults <- y})
```

Make factors (note the name change for seq (which is a function name in R and can cause problems). Rename response.

> **`outfaults4 <- lm(faults~(alg+sq+psize+ram)^3,data=pagefaults)`**

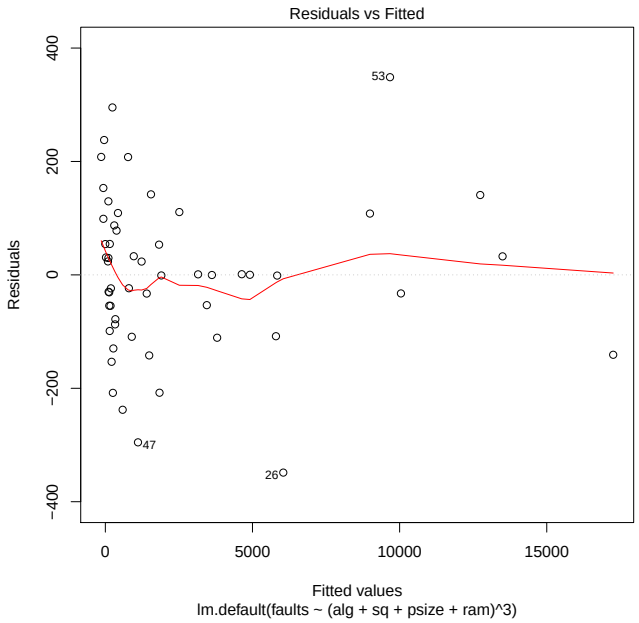
Since we have a single replication, we need a surrogate error. We will look at two approaches. The first just pools the four-way interaction into error.

> **`outfaults34 <- lm(faults~(alg+sq+psize+ram)^2,data=pagefaults)`**

The second approach jumps ahead and pools all three and four way into error.

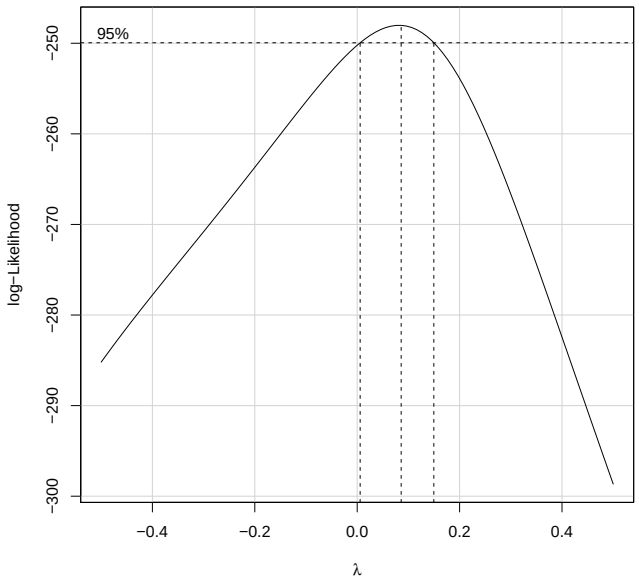
> **`plot(outfaults4,which=1)`**

You can't really see much in the residuals versus fitted values plot.



> **`boxCox(outfaults4,lambda=seq(-.5,.5,.01)`**

Box Cox is really convinced of the need to transform, with the optimum somewhere around power .1. Log is right at the edge of the interval, and 1 is so far away it's funny. We really need to transform.

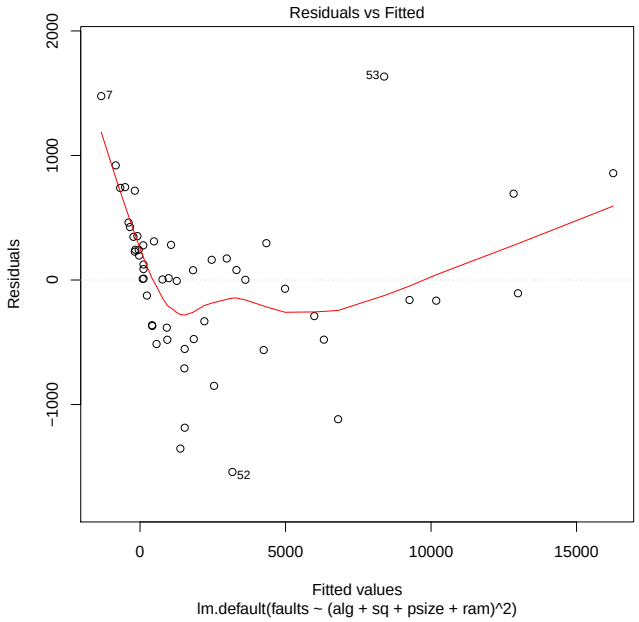


> **`plot(outfaults34,which=1)`**

Holy cow! What happened here?

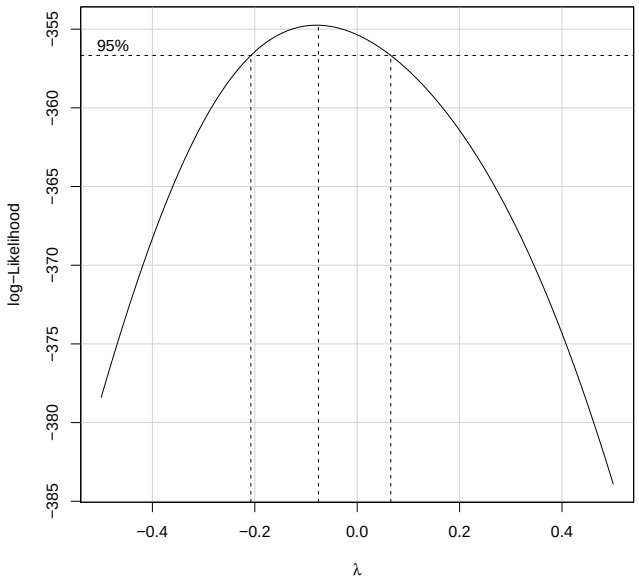
When you do not include all terms in your model, some non-ignorable mean effects can drop down into error giving you systematic patterns in the residuals. This can often be fixed via transformation, but you sometimes need additional predictive terms.

I call this pattern the flopping fish.



> **`boxCox(outfaults34,lambda=seq(-.5,.5,.02)`**

Box Cox is suggesting a power of -1, although log is well within the interval and 1 is a mile away.



```
> summary(pagefaults$faults)
```

You can see that they range over several orders of magnitude. Since these are positive data, the small ones almost surely have smaller variance than the large ones. The log is a good default transformation to start with on this kind of data. It may not turn out to be the optimum, but it's probably better than the natural scale.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   32.0   136.0   782.5  2521.0  3334.0 17120.0
```

```
> #
```

All signs point to the log transformation. I wonder if we should take logs?

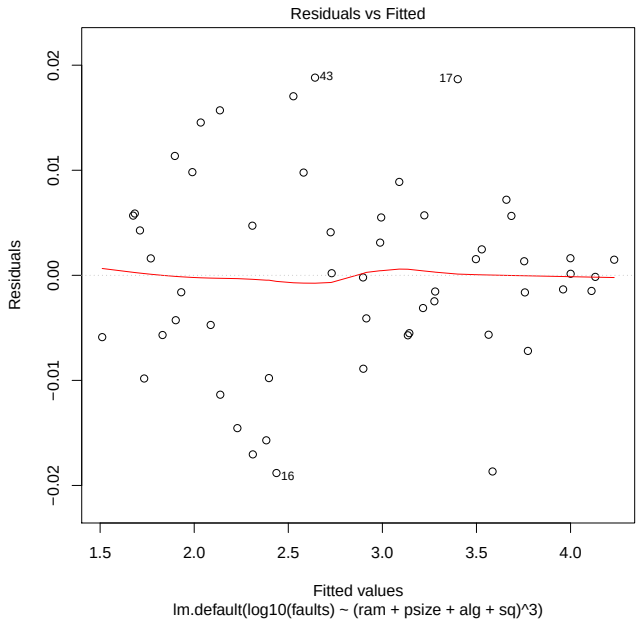
```
> outfaults4l <- lm(log10(faults)~(ram+psize+alg+sq)^3,data=pagefaults)
```

Refit with logs.

```
> outfaults34l <- lm(log10(faults)~(ram+psize+alg+sq)^2,data=pagefaults)
```

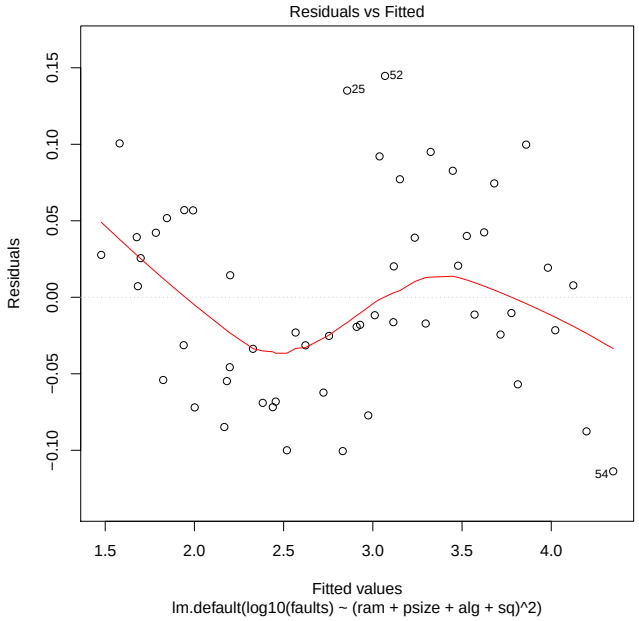
```
> plot(outfaults4l,which=1)
```

This looks much improved.



> **plot(outfaults34l,which=1)**

OK, the log stabilized the variability, but there is still a dominant pattern in the residuals. There is an important three-factor interaction being ignored.



> **anova(outfaults4l)**

Many significant terms and some non-significant terms. Remember that which interaction terms are significant depends on scale.

```
Analysis of Variance Table

Response: log10(faults)
              Df  Sum Sq Mean Sq    F value    Pr(>F)
ram            2 17.4838  8.7419 16265.4282 3.654e-15 ***
psize          2  7.8635  3.9318  7315.5596 8.919e-14 ***
alg            1  0.4719  0.4719   877.9857 1.824e-09 ***
sq             2  4.6473  2.3236  4323.4054 7.300e-13 ***
ram:psize      4  0.0951  0.0238    44.2447 1.689e-05 ***
ram:alg        2  0.0113  0.0057    10.5350  0.005736 **
ram:sq         4  1.7938  0.4484   834.3927 1.632e-10 ***
psize:alg      2  0.0042  0.0021     3.8979  0.065794 .
psize:sq       4  0.1564  0.0391    72.7278 2.511e-06 ***
alg:sq         2  0.0033  0.0017     3.0947  0.101042
ram:psize:alg  4  0.0008  0.0002     0.3511  0.836454
ram:psize:sq   8  0.1984  0.0248    46.1535 6.726e-06 ***
ram:alg:sq     4  0.0049  0.0012     2.2818  0.149073
psize:alg:sq   4  0.0027  0.0007     1.2778  0.354761
Residuals      8  0.0043  0.0005
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> outfaults4lb <- lm(log10(faults)~(ram+psize+alg+sq)^3-ram:psize:alg,data=pagefaults);
anova(outfaults4lb)
```

We only have 8 degrees of freedom for error, so we may wish to pool one or more terms with F-ratios less than 2 into error. We can remove a previously included term by "subtracting" it from the model. Here we take out the three factor interaction with smallest F (less than 1, actually). The mean squares of the other terms don't change, the error df increases by 4, and the error mean square decreases (because the F of the included term was less than 1); all terms become more significant using this surrogate error.

```
Analysis of Variance Table

Response: log10(faults)
             Df  Sum Sq Mean Sq    F value    Pr(>F)
ram           2 17.4838  8.7419 20754.9071 < 2.2e-16 ***
psize         2  7.8635  3.9318  9334.7533 < 2.2e-16 ***
alg           1  0.4719  0.4719  1120.3217 3.210e-13 ***
sq            2  4.6473  2.3236  5516.7239 < 2.2e-16 ***
ram:psize     4  0.0951  0.0238    56.4568 1.105e-07 ***
ram:alg       2  0.0113  0.0057    13.4428 0.0008637 ***
ram:sq        4  1.7938  0.4484  1064.6964 3.436e-15 ***
psize:alg     2  0.0042  0.0021     4.9738 0.0267153 *
psize:sq      4  0.1564  0.0391    92.8017 6.423e-09 ***
alg:sq        2  0.0033  0.0017     3.9489 0.0481125 *
ram:psize:sq  8  0.1984  0.0248    58.8925 1.849e-08 ***
ram:alg:sq    4  0.0049  0.0012     2.9116 0.0675582 .
psize:alg:sq  4  0.0027  0.0007     1.6304 0.2302135
Residuals    12  0.0051  0.0004
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> outfaults4lc <- lm(log10(faults)~(ram+psize+alg+sq)^3-ram:psize:alg-psize:alg:sq,
        data=pagefaults)
```

Pushing things one step further, we might also pool into error the next three factor interaction which also has an F less than 2. I probably wouldn't do this because I already have 12 degrees of freedom for error, but some would pool again. This makes things slightly less significant, because it increases the error mean square. I definitely wouldn't go beyond here.

```
> anova(outfaults4lc)
Analysis of Variance Table

Response: log10(faults)
             Df  Sum Sq Mean Sq    F value    Pr(>F)
ram           2 17.4838  8.7419 17929.1216 < 2.2e-16 ***
psize         2  7.8635  3.9318  8063.8244 < 2.2e-16 ***
alg           1  0.4719  0.4719   967.7896 9.683e-16 ***
sq            2  4.6473  2.3236  4765.6206 < 2.2e-16 ***
ram:psize     4  0.0951  0.0238    48.7702 9.148e-09 ***
ram:alg       2  0.0113  0.0057    11.6126 0.0007664 ***
ram:sq        4  1.7938  0.4484   919.7377 < 2.2e-16 ***
psize:alg     2  0.0042  0.0021     4.2966 0.0320944 *
psize:sq      4  0.1564  0.0391    80.1667 2.243e-10 ***
alg:sq        2  0.0033  0.0017     3.4113 0.0583516 .
ram:psize:sq  8  0.1984  0.0248    50.8743 6.241e-10 ***
```

```
ram:alg:sq    4  0.0049  0.0012     2.5151 0.0825350 .
Residuals    16  0.0078  0.0005
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

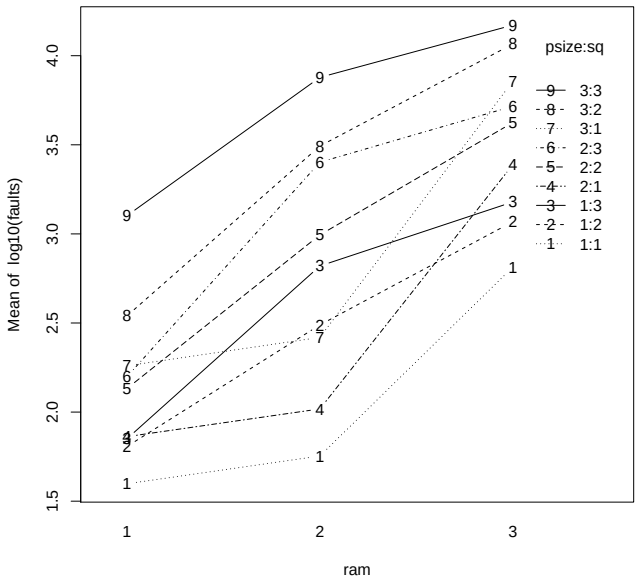
> **outfaults4ld <- update(outfaults4lb, ~.-psize:alg:sq);anova(outfaults4ld)**

There's actually a shortcut for updating a few terms in a model. It's a bit less typing this way.

```
Response: log10(faults)
             Df  Sum Sq Mean Sq    F value    Pr(>F)
ram           2 17.4838  8.7419 17929.1216 < 2.2e-16 ***
psize         2  7.8635  3.9318  8063.8244 < 2.2e-16 ***
alg           1  0.4719  0.4719   967.7896 9.683e-16 ***
sq            2  4.6473  2.3236  4765.6206 < 2.2e-16 ***
ram:psize     4  0.0951  0.0238    48.7702 9.148e-09 ***
ram:alg       2  0.0113  0.0057    11.6126 0.0007664 ***
ram:sq        4  1.7938  0.4484   919.7377 < 2.2e-16 ***
psize:alg     2  0.0042  0.0021     4.2966 0.0320944 *
psize:sq      4  0.1564  0.0391    80.1667 2.243e-10 ***
alg:sq        2  0.0033  0.0017     3.4113 0.0583516 .
ram:psize:sq  8  0.1984  0.0248    50.8743 6.241e-10 ***
ram:alg:sq    4  0.0049  0.0012     2.5151 0.0825350 .
Residuals    16  0.0078  0.0005
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

> **interactplot(ram,psize:sq,logfaults)**

The best way to understand interaction is usually through a plot. Here we see traces 1, 4, and 7 bow down (these are sequence 1), and traces 3, 6, and 9 bow up (these are sequence 3). But the gaps between these change in size, and that's the third factor interaction.



```
> effect("alg",outfaults4lc)
```

A principal question of interest in the experiment was the role of the algorithm. alg, ram:alg, and psize:alg are significant, but the first two have larger effects and so we'll just look at them.

Start with the main effect. The difference is about .187, so algorithm 2 leads to about $10^{.187} = 1.54$ times as many page faults; that's bad.

```
 alg effect
alg
       1        2
2.739949 2.926908
Warning message:
In analyze.model(term, mod, xlevels, default.levels) :
  alg is not a high-order term in the model
```

```
> model.effects(outfaults4lc,"alg")
```

Another way to see the effect.

```
          1           2
-0.09347962  0.09347962
```

```
> effect("ram:alg",outfaults4lc)
```

Here's the largest interaction. Algorithm 2 is always worse than algorithm 1, but for ram allocation 3 it is worse by a smaller amout; that's the interaction.

```
 ram*alg effect
   alg
ram        1        2
  1 2.045567 2.254129
  2 2.704376 2.910671
  3 3.469903 3.615924
Warning message:
In analyze.model(term, mod, xlevels, default.levels) :
  ram:alg is not a high-order term in the model
```

```
> model.effects(outfaults4lc,"ram:alg")
```

Might be a little clearer here.

```
            1           2
1 -0.01080151  0.01080151
2 -0.00966761  0.00966761
3  0.02046912 -0.02046912
```

```
> names(emp08.10)
```

These are the maize sprouting data from example 8.10, an 8x2x2 factorial.

```
[1] "aTemp"   "gTemp"   "variety" "amylase"
```

```
> amylase.data <- within(emp08.10,{aTempF <- factor(aTemp); gTempF <- factor(gTemp)})
```

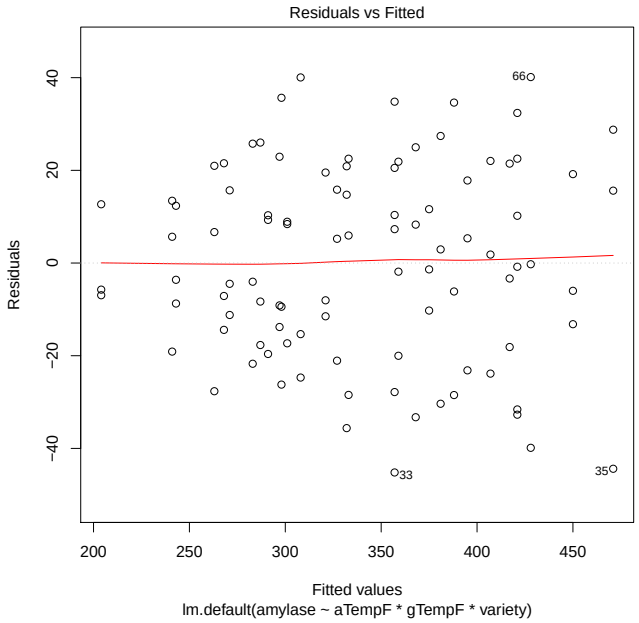
Make factors. In this data set, variety is already a factor.

```
> amyl.out <- lm(amylase ~ aTempF*gTempF*variety,data=amylase.data)
```

Let's fit the full model.

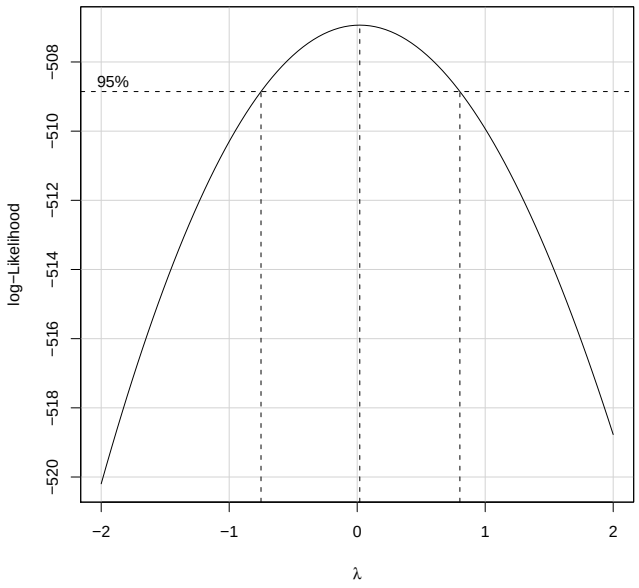
```
> plot(amyl.out,which=1)
```

There could be increasing variance here.




```
> boxCox(amyl.out)
```

BoxCox is suggesting a log, but 1 is not far from the interval.

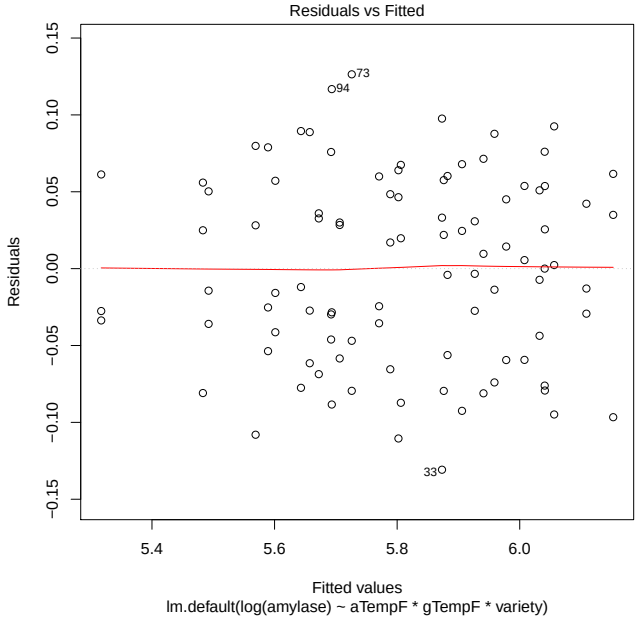



```
> lamyl.out <- lm(log(amylase) ~ aTempF*gTempF*variety,data=amylase.data)
```

Refit with log data.

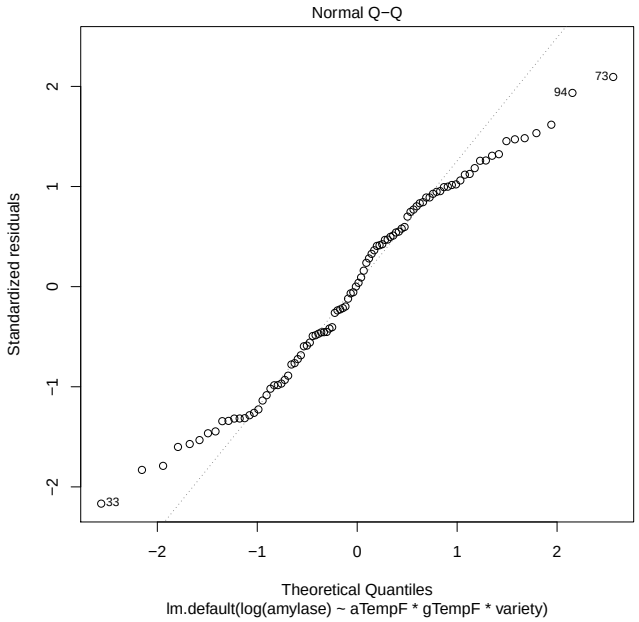
> **`plot(lamyl.out,which=1)`**

Constant variance is improved.



> **`plot(out,which=2)`**

Normality is a little questionable, with slightly short tails here. But we did better on variances, so we will stick with this scale.



> **anova(lamyl.out)**

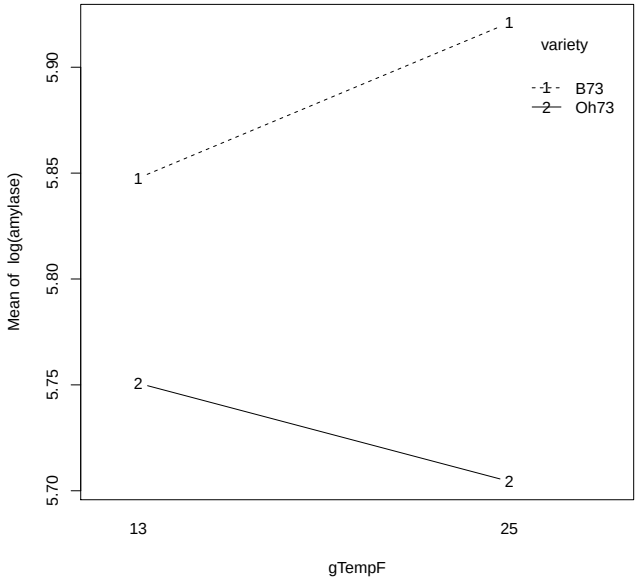
Three highly significant terms, and atemp:gtemp is worth looking at. However, we have a significant gTempF by variety interaction without a significant gTempF main effect.

```
Analysis of Variance Table

Response: log(amylase)
                     Df  Sum Sq Mean Sq  F value    Pr(>F)
aTempF                7 3.01613 0.43088  78.8628 < 2.2e-16 ***
gTempF                1 0.00438 0.00438   0.8016 0.3739757
variety               1 0.58957 0.58957 107.9085 2.305e-15 ***
aTempF:gTempF         7 0.08106 0.01158   2.1195 0.0539203 .
aTempF:variety        7 0.02758 0.00394   0.7212 0.6543993
gTempF:variety        1 0.08599 0.08599  15.7392 0.0001863 ***
aTempF:gTempF:variety 7 0.04764 0.00681   1.2457 0.2916176
Residuals            64 0.34967 0.00546
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

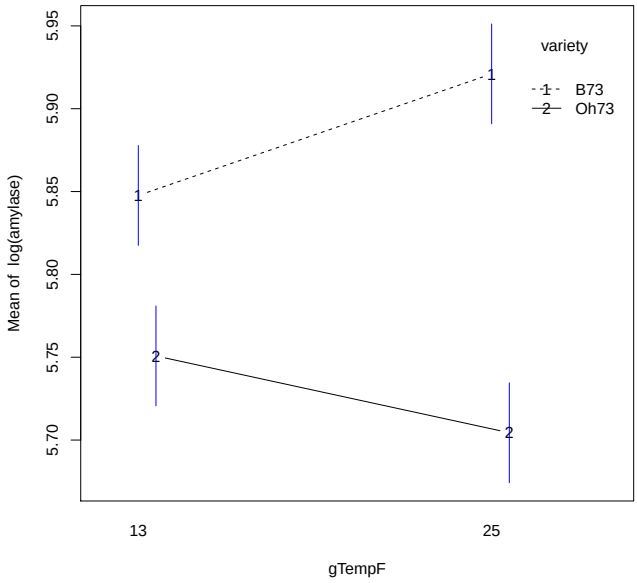
> **with(amylase.data, interactplot(gTempF,variety,log(amylase)))**

OK, now let's look into this mysterious issue of gTempF and variety interacting, but there being no significant gTempF main effect. It's pretty clear from the plot. Variety B73 is higher than variety Oh73, but gTempF has opposite effects of nearly equal magnitude on the two varieties. So no main effect of gTempF, but a significant interaction.



```
> with(amylase.data, interactplot(gTempF,variety,log(amylase),sigma2=0.00546,
      df=64,conf=.95))
```

We can add confidence intervals, using the MSE and its degrees of freedom. We have to do this, because interactplot doesn't know about all the other effects, just these two. So all of that other treatment variability would show up in the error. When you put confidence intervals on an interaction plot but you're ignoring some of the other terms, be sure to use the MSE and df from the anova. Here we see that the size of the effects is quite large compared to error.




```
> with(amylase.data, interactplot(aTempF,gTempF,log(amylase),sigma2=0.00546,
    df=64,conf=.95,jitter=.2))
```

This is a look at the marginally significant aTempF:gTempF interaction. Growth temperature 2 starts out below growth temperature 1, and ends up above. But not very far below or very far above, which is why the interaction is only marginally significant.