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

Recall the amylase activity data from example 8.10

```
[1] "aTemp"   "gTemp"   "variety" "amylase"
> amylase.data <- within(emp08.10,aTempF <- factor(aTemp); gTempF <- factor(gTemp))
```

Make factors.

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

We decided to use log data.

```
> anova(lamyl.out)
```

Variety and growth temperature interact, but only weak evidence for other interactions.

```
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
```

```
> qlamyl.out <- lm(log(amylase)~variety*gTemp*(aTemp+I(aTemp^2)+I(aTemp^3)+
   I(aTemp^4)+I(aTemp^5))+variety:gTempF:aTempF,data=amylase.data)
```

Now fit with quantitative factors.

Note what I did here. I wasn't too interested in fitting a very high polynomial term in analysis temperature, so I only went up to power 5 (which is really pretty high already). As the last term in the model, I added the three-way interaction of the three factors. This is will absorb all of the potential model df that were not fit by the earlier terms in the model. What this does is make sure that our residual MS is estimating pure error. It also gives me a check on the terms left out of the model (sixth and seventh powers).

```
> anova(qlamyl.out)
```

In this case, there is no evidence that we missed anything by leaving out the sixth and seventh powers. In fact, it doesn't look like we need fourth or fifth powers either.

```
Analysis of Variance Table

Response: log(amylase)
                       Df  Sum Sq Mean Sq  F value    Pr(>F)
variety                 1 0.58957 0.58957 107.9085 2.305e-15 ***
gTemp                   1 0.00438 0.00438   0.8016 0.3739757
aTemp                   1 0.87537 0.87537 160.2184 < 2.2e-16 ***
I(aTemp^2)              1 2.08972 2.08972 382.4793 < 2.2e-16 ***
I(aTemp^3)              1 0.04199 0.04199   7.6860 0.0072804 **
I(aTemp^4)              1 0.00284 0.00284   0.5196 0.4736411
I(aTemp^5)              1 0.00000 0.00000   0.0002 0.9875662
variety:gTemp           1 0.08599 0.08599  15.7392 0.0001863 ***
variety:aTemp           1 0.00110 0.00110   0.2011 0.6553824
variety:I(aTemp^2)      1 0.01877 0.01877   3.4354 0.0684225 .
```

```
variety:I(aTemp^3)        1 0.00218 0.00218   0.3988 0.5299623
variety:I(aTemp^4)        1 0.00087 0.00087   0.1587 0.6917206
variety:I(aTemp^5)        1 0.00373 0.00373   0.6833 0.4115309
gTemp:aTemp               1 0.03543 0.03543   6.4846 0.0132978 *
gTemp:I(aTemp^2)          1 0.00009 0.00009   0.0163 0.8988203
gTemp:I(aTemp^3)          1 0.02911 0.02911   5.3283 0.0242237 *
gTemp:I(aTemp^4)          1 0.00621 0.00621   1.1368 0.2903255
gTemp:I(aTemp^5)          1 0.00689 0.00689   1.2604 0.2657748
variety:gTemp:aTemp       1 0.00000 0.00000   0.0003 0.9872496
variety:gTemp:I(aTemp^2)  1 0.00028 0.00028   0.0520 0.8203957
variety:gTemp:I(aTemp^3)  1 0.04069 0.04069   7.4470 0.0081950 **
variety:gTemp:I(aTemp^4)  1 0.00003 0.00003   0.0059 0.9392471
variety:gTemp:I(aTemp^5)  1 0.00581 0.00581   1.0629 0.3064448
variety:gTempF:aTempF     8 0.01130 0.00141   0.2586 0.9767623
Residuals                64 0.34967 0.00546
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> qlamyl.out2 <- lm(log(amylase)~gTemp*variety*(aTemp+I(aTemp^2)+I(aTemp^3))+
      variety:gTempF:aTempF,data=amylase.data)
```

Now we just fit up to cubic.

```
> anova(qlamyl.out2)
Analysis of Variance Table

Response: log(amylase)
                        Df  Sum Sq Mean Sq  F value    Pr(>F)
gTemp                    1 0.00438 0.00438   0.8016 0.3739757
variety                  1 0.58957 0.58957 107.9085 2.305e-15 ***
aTemp                    1 0.87537 0.87537 160.2184 < 2.2e-16 ***
I(aTemp^2)               1 2.08972 2.08972 382.4793 < 2.2e-16 ***
I(aTemp^3)               1 0.04199 0.04199   7.6860 0.0072804 **
gTemp:variety            1 0.08599 0.08599  15.7392 0.0001863 ***
gTemp:aTemp              1 0.03543 0.03543   6.4846 0.0132978 *
gTemp:I(aTemp^2)         1 0.00009 0.00009   0.0163 0.8988203
gTemp:I(aTemp^3)         1 0.02911 0.02911   5.3283 0.0242237 *
variety:aTemp            1 0.00110 0.00110   0.2011 0.6553824
variety:I(aTemp^2)       1 0.01877 0.01877   3.4354 0.0684225 .
variety:I(aTemp^3)       1 0.00218 0.00218   0.3988 0.5299623
gTemp:variety:aTemp      1 0.00000 0.00000   0.0003 0.9872496
gTemp:variety:I(aTemp^2) 1 0.00028 0.00028   0.0520 0.8203957
gTemp:variety:I(aTemp^3) 1 0.04069 0.04069   7.4470 0.0081950 **
variety:gTempF:aTempF   16 0.03768 0.00235   0.4310 0.9682507
Residuals               64 0.34967 0.00546
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> (.0011+.01877+.00218+.00028+.04069)/6
```

If we want to keep that gTemp by variety by aTemp cubed, then it implies keeping five other terms, none of which is particularly significant. Let's test whether we need all six. Begin by getting a mean square for all six.

```
[1] 0.01050333
> .0105/.00546
```

Now compute the F.

```
[1] 1.923077
> pf(1.92,6,64,lower=FALSE)
```

As a group, these six are not very significant, so I would probably take them all out of the model rather than keep them all in to accommodate that one term with a small p-value. Remember, you will get some small p-values just by chance.

```
[1] 0.09100609

> qlamyl.out3 <- lm(log(amylase)~gTemp*variety+gTemp*(aTemp+I(aTemp^2)+I(aTemp^3))+
   variety:gTempF:aTempF,data=amylase.data)
```

Take those terms out.

```
> anova(qlamyl.out3)
```

Similarly, we need to decide whether we should keep a quadratic and an cubic in order to get to an apparently significant cubic.

```
Analysis of Variance Table

Response: log(amylase)
                      Df  Sum Sq Mean Sq  F value    Pr(>F)
gTemp                  1 0.00438 0.00438   0.8016 0.3739757
variety                1 0.58957 0.58957 107.9085 2.305e-15 ***
aTemp                  1 0.87537 0.87537 160.2184 < 2.2e-16 ***
I(aTemp^2)             1 2.08972 2.08972 382.4793 < 2.2e-16 ***
I(aTemp^3)             1 0.04199 0.04199   7.6860 0.0072804 **
gTemp:variety          1 0.08599 0.08599  15.7392 0.0001863 ***
gTemp:aTemp            1 0.03543 0.03543   6.4846 0.0132978 *
gTemp:I(aTemp^2)       1 0.00009 0.00009   0.0163 0.8988203
gTemp:I(aTemp^3)       1 0.02911 0.02911   5.3283 0.0242237 *
variety:gTempF:aTempF 22 0.10070 0.00458   0.8378 0.6695299
Residuals             64 0.34967 0.00546
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

> (.02911+.00009)/2
```

Mean square for gTemp by quadratic and cubic in aTemp.

```
[1] 0.0146

> pf(.0146/.00546,2,64,lower=FALSE)
```

Again, not very significant.

```
[1] 0.07667892

> qlamyl.out4 <- lm(log(amylase)~gTemp*variety+(aTemp+I(aTemp^2)+I(aTemp^3)+
   gTemp:aTemp)+variety:gTempF:aTempF,data=amylase.data)

> anova(qlamyl.out4)
```

It looks like we have reduced the model as far as is reasonable.

```
Analysis of Variance Table

Response: log(amylase)
                      Df  Sum Sq Mean Sq  F value    Pr(>F)
gTemp                  1 0.00438 0.00438   0.8016 0.3739757
variety                1 0.58957 0.58957 107.9085 2.305e-15 ***
```

```
aTemp                   1 0.87537 0.87537 160.2184 < 2.2e-16 ***
I(aTemp^2)              1 2.08972 2.08972 382.4793 < 2.2e-16 ***
I(aTemp^3)              1 0.04199 0.04199   7.6860 0.0072804 **
gTemp:variety           1 0.08599 0.08599  15.7392 0.0001863 ***
gTemp:aTemp             1 0.03543 0.03543   6.4846 0.0132978 *
variety:gTempF:aTempF  24 0.12990 0.00541   0.9907 0.4904529
Residuals              64 0.34967 0.00546
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> qlamyl.out5 <- lm(log(amylase)~gTemp*variety+aTemp+I(aTemp^2)+I(aTemp^3)+
    gTemp:aTemp,data=amylase.data)
```

Refit with just significant terms so that we can look at coefficients.

```
> summary(qlamyl.out5)

Call:
lm.default(formula = log(amylase) ~ gTemp * variety + aTemp +
    I(aTemp^2) + I(aTemp^3) + gTemp:aTemp, data = amylase.data)

Residuals:
      Min        1Q    Median        3Q       Max
-0.164478 -0.049143  0.002068  0.051949  0.155316

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     5.160e+00  1.532e-01  33.674  < 2e-16 ***
gTemp          -6.279e-03  3.164e-03  -1.985 0.050310 .
variety1       -1.641e-02  2.502e-02  -0.656 0.513646
aTemp           3.780e-02  2.028e-02   1.863 0.065735 .
I(aTemp^2)      5.904e-04  8.649e-04   0.683 0.496653
I(aTemp^3)     -3.173e-05  1.143e-05  -2.776 0.006724 **
gTemp:variety1  4.988e-03  1.256e-03   3.972 0.000145 ***
gTemp:aTemp     3.151e-04  1.236e-04   2.550 0.012511 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.07382 on 88 degrees of freedom
Multiple R-squared:  0.8859,Adjusted R-squared:  0.8768
F-statistic: 97.58 on 7 and 88 DF,  p-value: < 2.2e-16
```

```
> 5.160-.01641
```

"Intercept" for variety 1.

```
[1] 5.14359
> 5.160+.01641
```

"Intercept" for variety 2.

```
[1] 5.17641
> -.006279+.004988
```

Slope of gTemp for variety 1.

```
[1] -0.001291
> -.006279-.004988
```

Slope of gTemp for variety 2.

```
[1] -0.011267
```

```
> qlamyl.out6 <- lm(log(amylase)~variety-1 + gTemp:variety+aTemp+I(aTemp^2)+
    I(aTemp^3)+gTemp:aTemp,data=amylase.data)
```

We can make R fit the combined intercepts and slopes by "subtracting out 1" to remove the overall constant, and by first entering gTemp:variety without having a gTemp main effect.

```
> summary(qlamyl.out6)
```

Here we see the combined coefficients directly.

```
Call:
lm.default(formula = log(amylase) ~ variety - 1 + gTemp:variety +
    aTemp + I(aTemp^2) + I(aTemp^3) + gTemp:aTemp, data = amylase.data)

Residuals:
      Min        1Q    Median        3Q       Max
-0.164478 -0.049143  0.002068  0.051949  0.155316

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
varietyB73         5.144e+00  1.553e-01  33.128  < 2e-16 ***
varietyOh73        5.177e+00  1.553e-01  33.340  < 2e-16 ***
aTemp              3.780e-02  2.028e-02   1.863  0.06573 .
I(aTemp^2)         5.904e-04  8.649e-04   0.683  0.49665
I(aTemp^3)        -3.173e-05  1.143e-05  -2.776  0.00672 **
varietyB73:gTemp  -1.291e-03  3.404e-03  -0.379  0.70542
varietyOh73:gTemp -1.127e-02  3.404e-03  -3.310  0.00135 **
gTemp:aTemp        3.151e-04  1.236e-04   2.550  0.01251 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.07382 on 88 degrees of freedom
Multiple R-squared:  0.9999,Adjusted R-squared:  0.9998
F-statistic: 7.431e+04 on 8 and 88 DF,  p-value: < 2.2e-16
```

```
> catch<-c(19.1,50.1,123,23.4,166.1,407.4,
29.5,223.9,398.1,23.4,58.9,229.1,16.6,64.6,251.2)
```

These data are average numbers of insects trapped for 3 kinds of traps used in 5 periods. Data from Snedecor and Cochran.

```
> trap<-factor(rep(1:3,5))
```

Factors to indicate traps and periods. Replication is just n=1.

```
> period<-factor(rep(1:5,each=3))
```

```
> fit1 <- lm(catch~period+trap); anova(fit1)
```

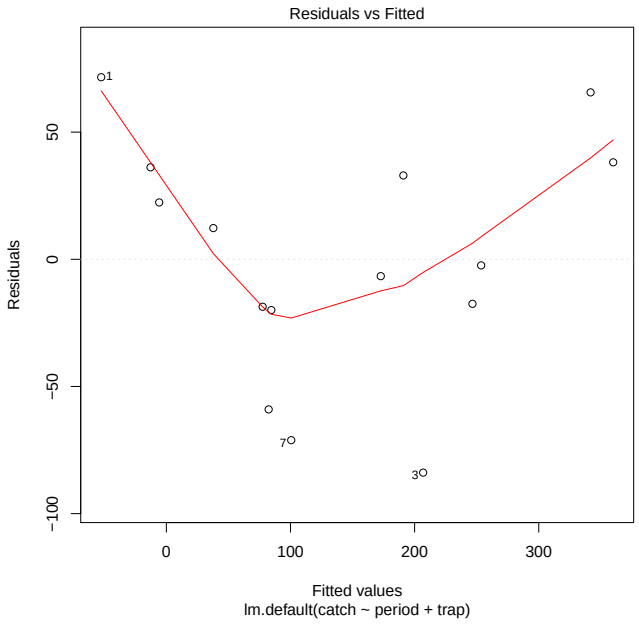
Here is the standard ANOVA. Because we have only a single replication, we have no estimate of pure error. Our error here is really interaction, so we might want to check to see if some of this could be reduced via transformation.

```
Analysis of Variance Table

Response: catch
          Df Sum Sq Mean Sq F value    Pr(>F)
period     4  52066   13016  3.4022 0.0661095 .
trap       2 173333   86667 22.6528 0.0005073 ***
Residuals  8  30607    3826
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> plot(fit1,which=1)
```

OK, here is trouble. If the interaction can truly be ignored, this plot should just look like random noise. It doesn't. This curved shape suggests that a transformation could help.




```
> preds<-predict(fit1)
```

To do Tukey One DF, start by getting the predicted values.

```
> coef(fit1)
(Intercept)     period1     period2     period3     period4        trap1
  138.96000   -74.89333    60.00667    78.20667   -35.16000  -116.56000
      trap2
  -26.24000
```

```
> rspv<-preds^2/2/coef(fit1)[1]
```

Get the squares of the predicted values and rescale them by dividing by twice the mean. (The first coefficient is $\hat{\mu}$.)

```
> fit2 <- lm(catch~period+trap+rspv);anova(fit2)
```

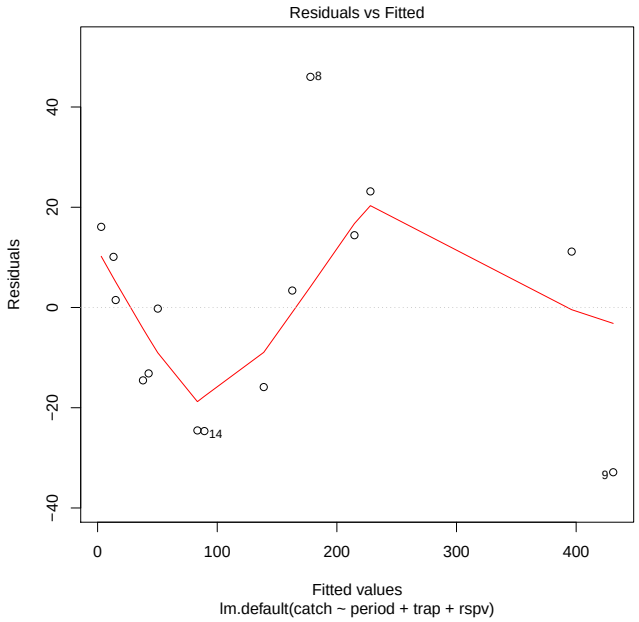
Get the Tukey 1df test by doing an ANOVA with the squared predicted values as a third term after rows and columns. This is the simplest way to get a test for Tukey nonadditivity. In this case, preds2 is highly significant, so the Tukey test is indicating that interaction can be reduced through a transformation.

```
Analysis of Variance Table

Response: catch
          Df Sum Sq Mean Sq F value    Pr(>F)
period     4  52066   13016  14.491  0.001693 **
trap       2 173333   86667  96.482 8.026e-06 ***
rspv       1  24319   24319  27.073  0.001249 **
Residuals  7   6288     898
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> plot(fit2,which=1)
```

The residual plot is improved, but still not great.



> **summary(fit2)**

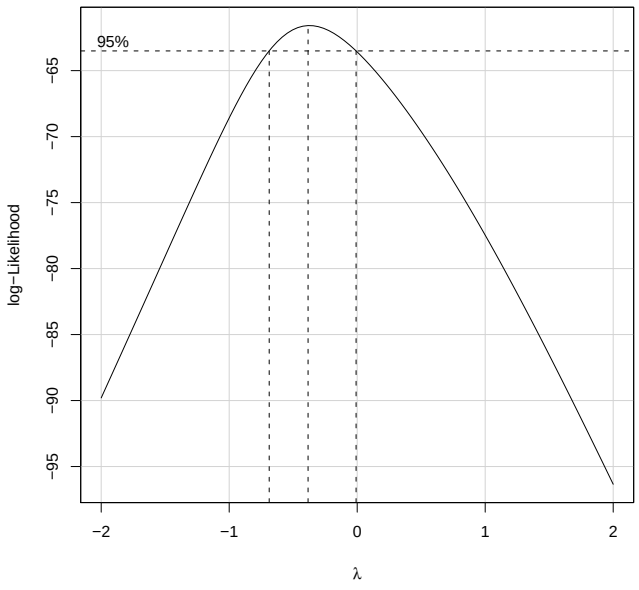
The coefficient of preds2 is 0.88, with a standard error of .17. Thus, a reasonable range for the coefficient is .54 to 1.22. So a reasonable range for 1 minus the coefficient is -.22 up to .46. So log is in the range, as is a fourth root, (almost) a square root, and various others. The fact of the matter is that this is not always the best way to find the transformation power. A better way is to try a bunch of different powers and then take the one that minimizes the Tukey F-test. Here, the power that minimizes things is about -.5 (not shown).

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  29.8090    22.3595   1.333  0.22423
period1     -15.5236    19.2284  -0.807  0.44604
period2       6.5802    18.5734   0.354  0.73356
period3       0.7047    21.4802   0.033  0.97475
period4       3.0070    17.1273   0.176  0.86560
trap1       -20.0381    21.5381  -0.930  0.38315
trap2        31.4869    15.5839   2.020  0.08307 .
rspv          0.8835     0.1698   5.203  0.00125 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 29.97 on 7 degrees of freedom
Multiple R-squared: 0.9754,Adjusted R-squared: 0.9509
F-statistic: 39.71 on 7 and 7 DF,  p-value: 4.122e-05
```

> **boxCox(fit1)**

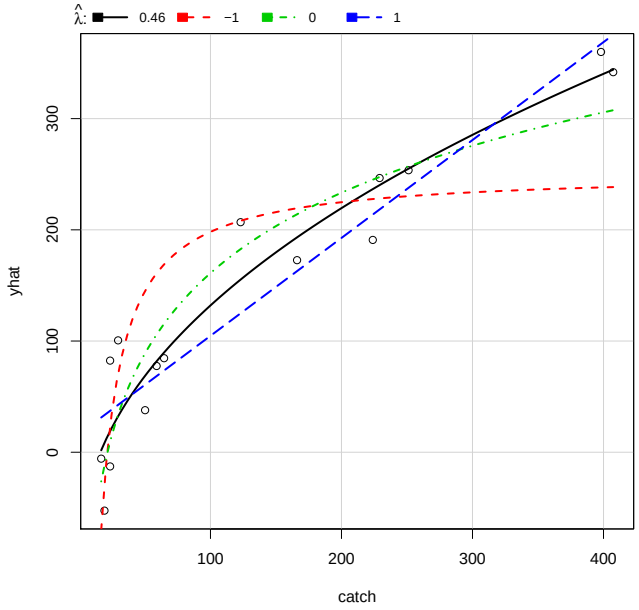
Box Cox also suggests a reciprocal square root.



```
> inverseResponsePlot(fit1)
```

The inverse response plot is also supposed to transform to a better fit. It likes the regular square root.

```
      lambda      RSS
1  0.4649587 20109.07
2 -1.0000000 65903.09
3  0.0000000 26153.72
4  1.0000000 26947.73
```




```
> #
```

So we now have three suggested transformations ranging from square root to reciprocal square root. Which one do we use? The proof of the pudding is in the tasting, and the proof of the transformation is in doing the transformation and then looking at the residuals.

```
> fit3 <- lm(log(catch)~period+trap);anova(fit3)
```

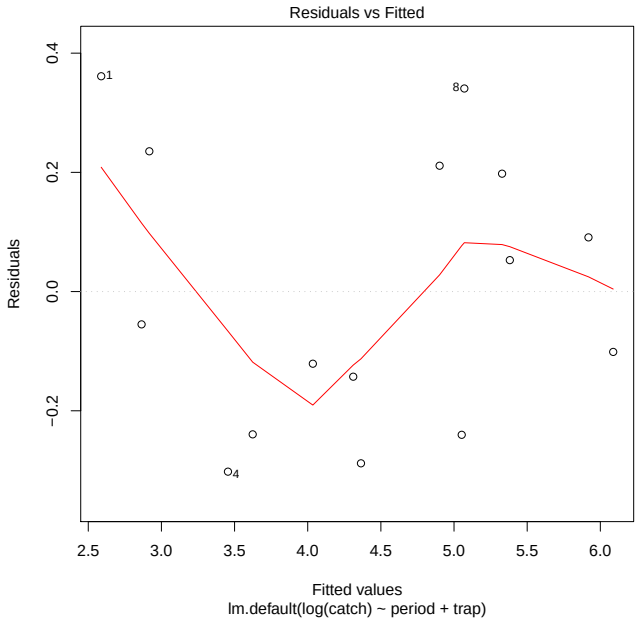
Here we look at the log. Note that the "error," which is also the interaction, is about 4% of total variation (it was about 12% on the original scale).

```
Analysis of Variance Table

Response: log(catch)
          Df  Sum Sq Mean Sq F value    Pr(>F)
period     4  2.2512  0.5628  6.1305   0.01470 *
trap       2 15.3320  7.6660 83.5041 4.366e-06 ***
Residuals  8  0.7344  0.0918
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

> **`plot(fit3,which=1)`**

This is better than the original scale, but in no way could it be called good. Compare this to the residuals when we include rspv.



> **`fit4 <- lm(catch^-.5~period+trap);anova(fit4)`**

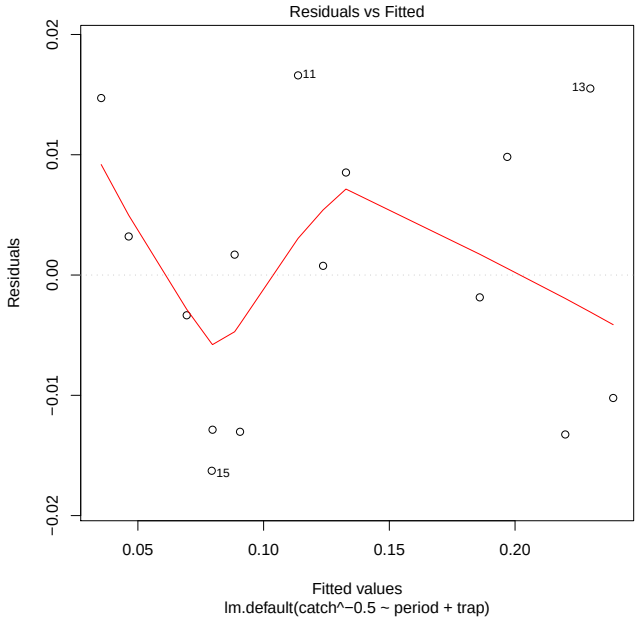
Repeat with reciprocal square root. Now the "error" is about 0.3% of total variation, so it's the best one yet.

```
Analysis of Variance Table

Response: catch^-0.5
          Df   Sum Sq   Mean Sq  F value    Pr(>F)
period     4 0.005981 0.0014953   6.6082   0.01186 *
trap       2 0.059878 0.0299388 132.3083 7.416e-07 ***
Residuals  8 0.001810 0.0002263
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> plot(fit4,which=1)
```

This is the best we've seen, but still not very good.




```
> fit5 <- lm(sqrt(catch)~period+trap);anova(fit5)
```

Repeat with square root. Now the "error" is about 7% of total variation, so not so good.

```
Analysis of Variance Table

Response: sqrt(catch)
          Df Sum Sq Mean Sq F value    Pr(>F)
period     4  75.59  18.898  4.5549   0.03276 *
trap       2 344.79 172.394 41.5514 5.946e-05 ***
Residuals  8  33.19   4.149
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> #
```

Let's look at what a Tukey style interaction looks like in residuals. We work from the original model. What these steps will do is to put the residuals in a matrix, and then order the rows and columns so that the row and column effects are in increasing order.

```
> m <- matrix(residuals(fit1),3);m
```

Here is the matrix of residuals. Note that this worked because the data were entered in a systematic order. If things were randomized we would need to work harder to associate treatments with their residuals in a matrix form.

```
          [,1]       [,2]      [,3]   [,4]   [,5]
[1,]  71.59333 -59.006667 -71.10667  36.16  22.36
[2,]  12.27333  -6.626667  32.97333 -18.66 -19.96
[3,] -83.86667  65.633333  38.13333 -17.50  -2.40
```

```
> p <- model.effects(fit1,"period");p
```

These are the period effects.

```
        1         2         3         4         5
-74.89333  60.00667  78.20667 -35.16000 -28.16000
```

```
> op <- order(p);op
```

This is the order in which we will need to sort the columns to put column effects in increasing order.

```
[1] 1 4 5 2 3
```

```
> t<-model.effects(fit1,"trap");t
```

Trap effects.

```
      1       2       3
-116.56  -26.24  142.80
```

```
> ot <- order(t);ot
```

Order to put trap effects into increasing order. Well, they were already in increasing order, so this doesn't change anything.

```
[1] 1 2 3
```

```
> m[ot,op]
```

Here are the residuals reordered into increasing row and column effect orders. Note the pattern of negatives and positives in opposite corners. This is the Tukey 1 df interaction pattern in residuals. It looks a lot like a linear by linear interaction (once the rows and columns have been properly ordered).

```
          [,1]   [,2]   [,3]       [,4]      [,5]
[1,]  71.59333  36.16  22.36 -59.006667 -71.10667
[2,]  12.27333 -18.66 -19.96  -6.626667  32.97333
[3,] -83.86667 -17.50  -2.40  65.633333  38.13333
```

```
> #
```

What we're going to do now is to redo the Tukey using comparison values (in preparation for the Mandel models).

```
> p <- model.effects(fit1,"period");allp <- p[period]
```

As above, but now repeat these values as appropriate for levels of period.

```
> t <- model.effects(fit,"trap");allt <- t[trap]
```

Similarly for trap effects.

```
> compvals <- allp*allt/coef(fit1)[1]
```

These are the comparison values.

```
> fit2b <- lm(catch~period+trap+compvals)
```

Refit the Tukey model with comparison values.

```
> summary(fit2b)
```

Two points. First, the estimated coefficient using compvals is the same as using rspv. However, when you use compvals the coefficients for period and trap are the same as they were in fit1 (this is not true when using rspv).

```
Call:
lm.default(formula = catch ~ period + trap + compvals)

Residuals:
    Min      1Q  Median      3Q     Max
-32.869 -15.205   1.492  12.788  46.020

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  138.9600     7.7385  17.957 4.10e-07 ***
period1      -74.8933    15.4770  -4.839  0.00188 **
period2       60.0067    15.4770   3.877  0.00608 **
period3       78.2067    15.4770   5.053  0.00147 **
period4      -35.1600    15.4770  -2.272  0.05733 .
trap1       -116.5600    10.9439 -10.651 1.41e-05 ***
trap2        -26.2400    10.9439  -2.398  0.04763 *
compvals       0.8835     0.1698   5.203  0.00125 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 29.97 on 7 degrees of freedom
Multiple R-squared:  0.9754,Adjusted R-squared:  0.9509
F-statistic: 39.71 on 7 and 7 DF,  p-value: 4.122e-05
```

```
> fit.rowmodel <- lm(catch~trap+period+period*allt)
```

The distinction of rows versus columns is arbitrary, but here we we fit a different slope to trap effects separately for each period.

```
> fit.rowmodel
```

The ally will always be non-estimable, and the trap and period effects are the same as in the base model.

```
Call:
lm.default(formula = catch ~ trap + period + period * allt)

Coefficients:
 (Intercept)         trap1         trap2       period1       period2
    138.9600     -116.5600      -26.2400      -74.8933       60.0067
     period3       period4          allt  period1:allt  period2:allt
     78.2067      -35.1600            NA       -0.5955        0.4738
period3:allt  period4:allt
      0.3712       -0.1795
```

```
> model.effects(fit.rowmodel,"period:allt")
```

The slope adjustments sum to 0.

```
         allt
1 -0.59547589
2  0.47377405
3  0.37120481
4 -0.17954369
5 -0.06995928
```

```
> anova(fit.rowmodel)
```

The row model is borderline significantly better than the additive model.

```
Analysis of Variance Table

Response: catch
            Df Sum Sq Mean Sq F value    Pr(>F)
trap         2 173333   86667 77.5687 0.0006318 ***
period       4  52066   13016 11.6501 0.0177592 *
period:allt  4  26138    6534  5.8485 0.0577367 .
Residuals    4   4469    1117
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> anova(fit1,fit2b,fit.rowmodel)
```

Tukey is significantly better than additive, but the row model does not significantly improve on Tukey.

```
Analysis of Variance Table

Model 1: catch ~ trap + period
Model 2: catch ~ period + trap + compvals
Model 3: catch ~ trap + period + period * allt
  Res.Df     RSS Df Sum of Sq       F   Pr(>F)
1      8 30607.0
2      7  6287.9  1   24319.1 21.7662 0.009551 **
3      4  4469.1  3    1818.7  0.5426 0.678594
```

```
> fit.colmodel <- lm(catch~trap+period+trap*allp)
```

Here we fit different period "slopes" for each trap.

```
> anova(fit.colmodel)
```

Column model is significantly better than additive.

```
Analysis of Variance Table

Response: catch
          Df Sum Sq Mean Sq F value   Pr(>F)
trap       2 173333   86667 140.016 9.23e-06 ***
period     4  52066   13016  21.029 0.001122 **
trap:allp  2  26893   13447  21.724 0.001787 **
Residuals  6   3714     619
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> anova(fit1,fit2b,fit.colmodel)
```

But the column model is not really significantly better than the Tukey model.

```
Analysis of Variance Table

Model 1: catch ~ trap + period
Model 2: catch ~ period + trap + compvals
Model 3: catch ~ trap + period + trap * allp
  Res.Df     RSS Df Sum of Sq       F    Pr(>F)
1      8 30607.0
2      7  6287.9  1     24319 39.2893 0.0007659 ***
3      6  3713.8  1      2574  4.1585 0.0875351 .
```

```
> fit.slpmodel <- lm(catch~trap+period+trap*allp+period*allt)
```

We can fit the model with both the slopes.

```
> anova(fit.slpmodel)
```

. . . but it is not significantly better than period slopes model.

```
Analysis of Variance Table

Response: catch
            Df Sum Sq Mean Sq  F value   Pr(>F)
trap         2 173333   86667 137.1945 0.001125 **
period       4  52066   13016  20.6053 0.016107 *
trap:allp    2  26893   13447  21.2861 0.016890 *
period:allt  3   1819     606   0.9597 0.513094
Residuals    3   1895     632
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> #
```

Note that we get the same predicted values but different individual terms estimated in the model depending on which order we enter the rows and column slope effects.