DOE

Cody Frisby

March 25, 2016

Here, I do the in class example using R rather than minitab.

```r
y <- c(1.68,1.98,2.07,2.44,4.98,5.70,7.77,9.43,3.24,3.44,4.09,4.53,
       9.97,9.07,11.75,16.30)
A <- rep(c(-1, 1), 2)
B <- rep(c(rep(-1,4),rep(1,4)), 2)
C <- c(rep(-1,8), rep(1,8))
D <- rep(c(rep(-1,2), rep(1,2)), 2)
df <- data.frame(y,A,B,C,D)
fit.all <- aov(y ~ .^4, df)
fit.two <- aov(y ~ .*., df) # all two way interactions.
summary(fit.all)
```

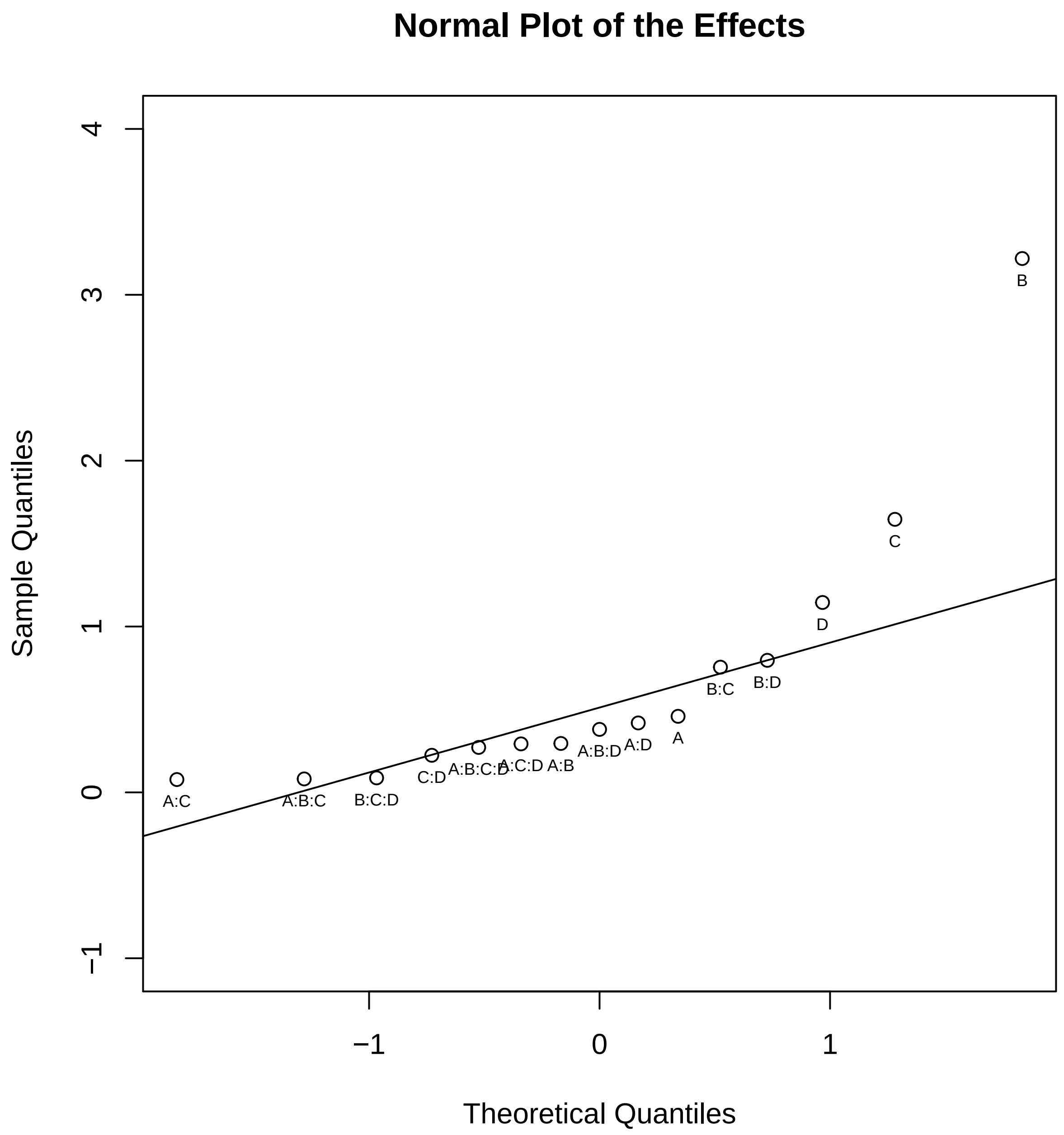
```
##             Df Sum Sq Mean Sq
## A            1   3.37    3.37
## B            1 165.77  165.77
## C            1  43.36   43.36
## D            1  20.98   20.98
## A:B          1   1.39    1.39
## A:C          1   0.10    0.10
## A:D          1   2.81    2.81
## B:C          1   9.12    9.12
## B:D          1  10.14   10.14
## C:D          1   0.80    0.80
## A:B:C        1   0.11    0.11
## A:B:D        1   2.31    2.31
## A:C:D        1   1.37    1.37
## B:C:D        1   0.12    0.12
## A:B:C:D      1   1.18    1.18
```

Oh, no. We can't use this model. There are no remaining *df* so we can not estimate the errors.

If we have a 2^6 factorial design. How many two-way interactions? 6 choose 2, or using R, 15. How many greater than 2 interactions? 6 choose 3 + 6 choose 4 + 6 choose 5 + 6 choose 6. Or $2^6 - (6 + 15) - 1$.

In the spirit of replicating the minitab normal plot of the effects, here I attempt to do it in R.

```r
# let's try to create a "good" normal prob plot of the effects
# another way
tmp <- qqnorm(coef(fit.all)[-1], ylim = c(-1, 4),
              main = "Normal Plot of the Effects")
qqline(coef(fit.all)[-1])
text(tmp$x, tmp$y, names(coef(fit.all)[-1]), pos=1, cex=0.6)
```



So, from this plot, we could fit a new model keeping B, C, D, A, AD, ABD, AB, ACD, and AC. We can also simply look at the main effects by using R's function effects(model).

```
effects(fit.all)
```

```
## (Intercept)           A           B           C           D         A:B
##     -24.610      -1.835      12.875       6.585       4.580      -1.180
##         A:C         A:D         B:C         B:D         C:D       A:B:C
##      -0.310      -1.675       3.020       3.185      -0.895       0.325
##       A:B:D       A:C:D       B:C:D     A:B:C:D
##      -1.520      -1.170      -0.350       1.085
## attr(,"assign")
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
## attr(,"class")
## [1] "coef"
```

We can look at the values of these and reduce our model using the leading canidates. Here it appears B, C, D, BC, BD are the leaders.

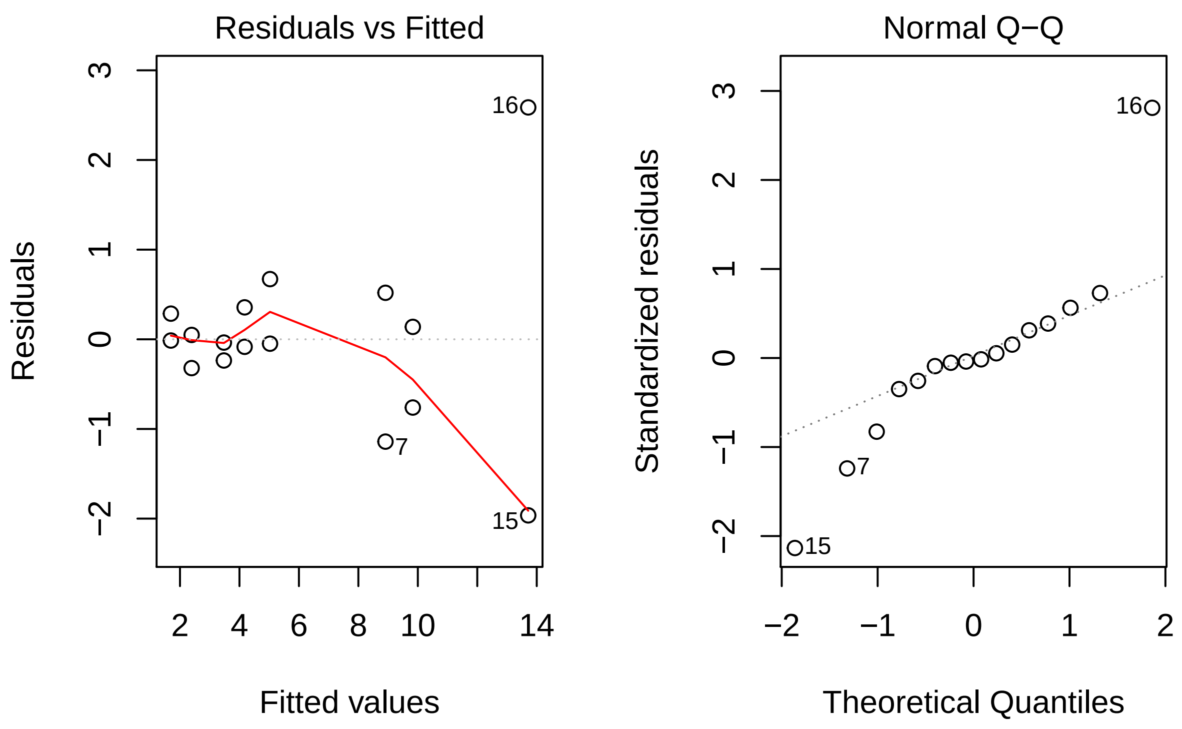
Fit the reduced model.

```
fit.red <- aov(y ~ B+C+D+B*C+B*D, data = df)
anova(fit.red)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq  F value    Pr(>F)
## B          1 165.766 165.766 122.3631 6.258e-07 ***
## C          1  43.362  43.362  32.0087 0.0002102 ***
## D          1  20.976  20.976  15.4841 0.0027975 **
## B:C        1   9.120   9.120   6.7324 0.0267341 *
## B:D        1  10.144  10.144   7.4882 0.0209601 *
## Residuals 10  13.547   1.355
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Diagnostics of the reduced model.

```
par(mfrow=c(1,2))
plot(fit.red, which = c(1,2))
```



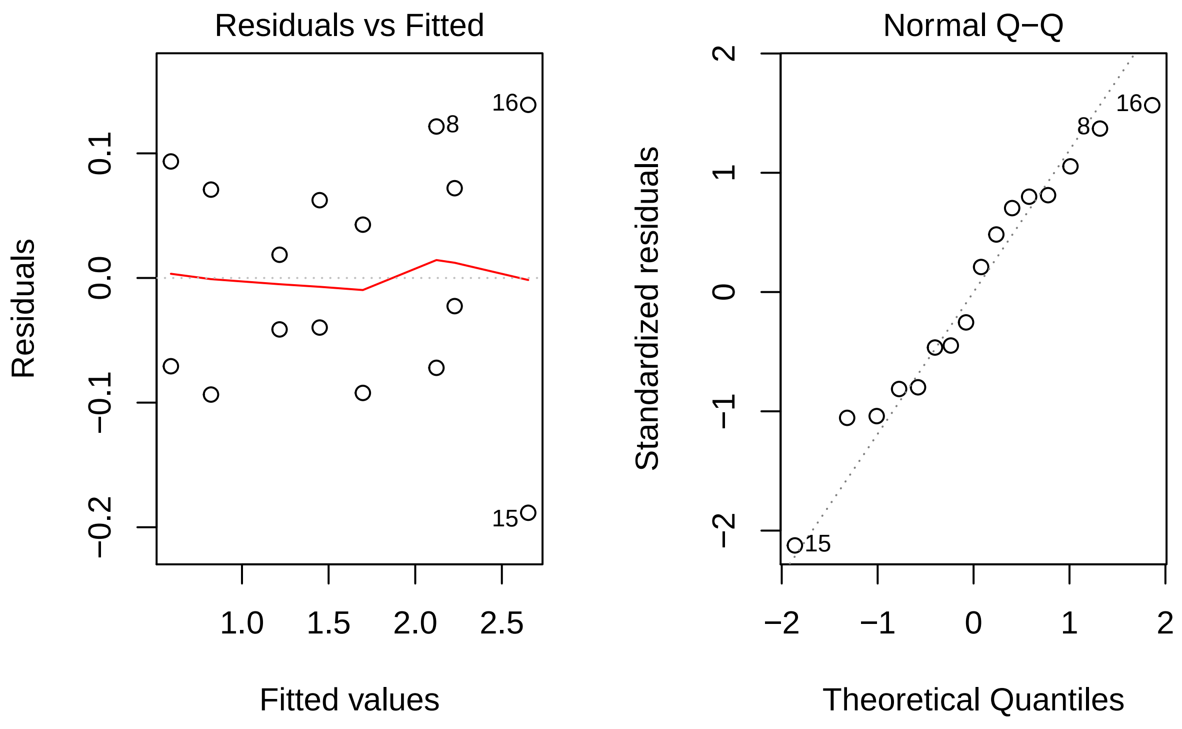
These don't look great. We will fit a new model with a transformation on the response.

```r
fit.trans <- aov(log(y) ~ B+C+D+B*C+B*D, data = df)
anova(fit.trans)
```

```
## Analysis of Variance Table
##
## Response: log(y)
##           Df Sum Sq Mean Sq  F value    Pr(>F)
## B          1 5.3452  5.3452 424.7138 1.601e-09 ***
## C          1 1.3389  1.3389 106.3825 1.196e-06 ***
## D          1 0.4305  0.4305  34.2092 0.0001619 ***
## B:C        1 0.0095  0.0095   0.7529 0.4059060
## B:D        1 0.0373  0.0373   2.9663 0.1157454
## Residuals 10 0.1259  0.0126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now let's take a look at the residuals from this model.

```r
par(mfrow=c(1,2))
plot(fit.trans, which = c(1,2))
```



After transformation, this is looking much better.

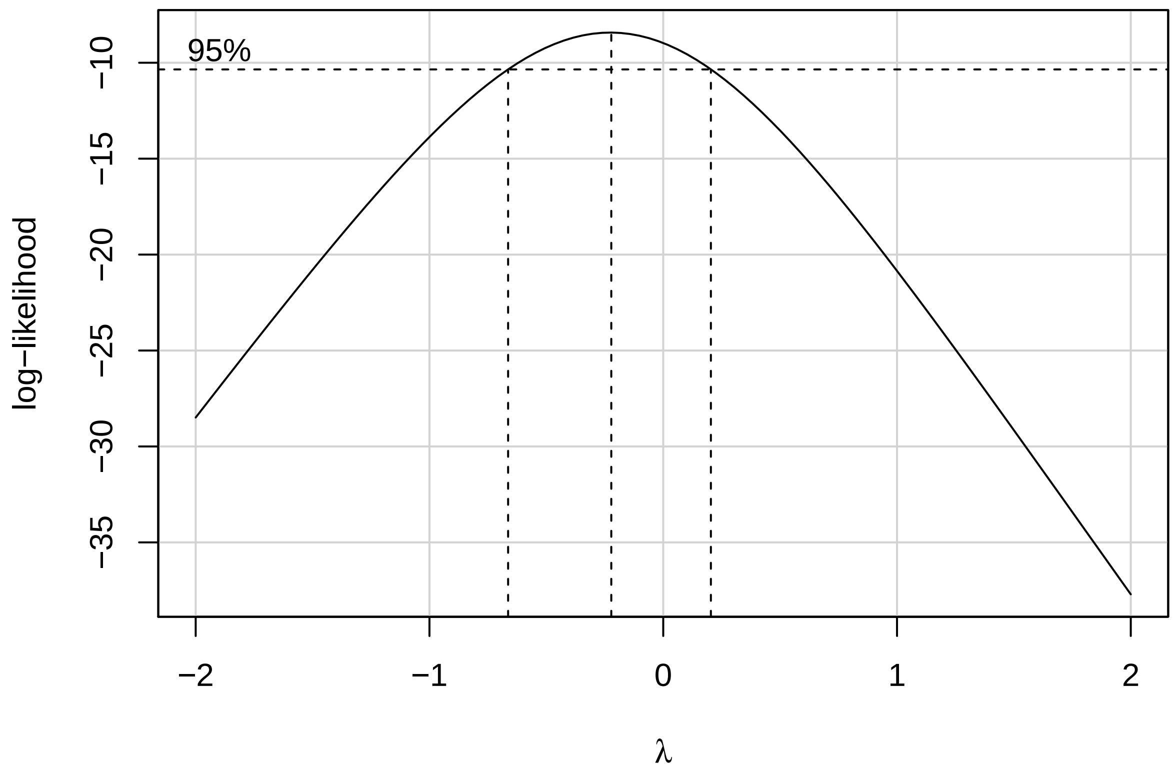
We can also use some built in functions in R to find the best transformation to use. PowerTransform() and boxcox() from the car package are very easy to use and very cool.

```r
car::powerTransform(fit.red)
```

```
## Estimated transformation parameters
##         Y1
## -0.2259652
```

We can also visually plot this:

```
library(car)
boxCox(fit.red)
```



So, this power of lambda, -0.2259, may be even better than log for our transformation. Although, zero is inside the dotted lines on the boxCox plot, meaning log could be an appropriate transformation.

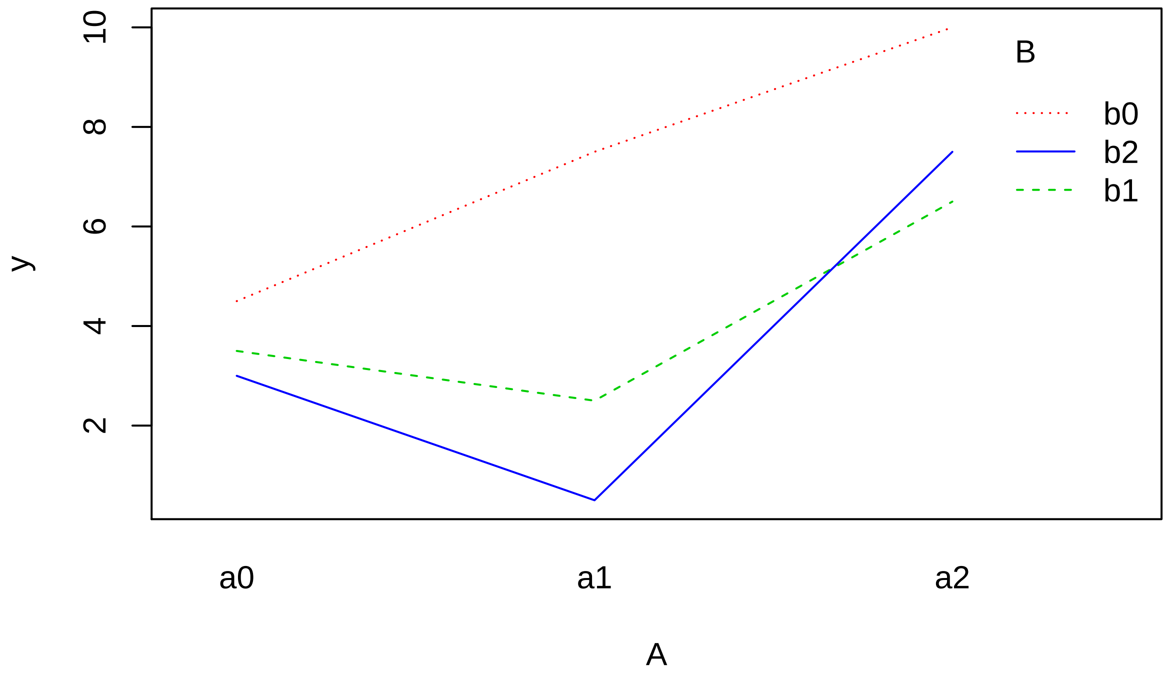
In class 4-1-16, Chapter 9 3^2 design

```
y <- c(4,5,2,5,3,3,6,9,2,3,0,1,7,13,5,8,7,8)
A <- c(rep("a0",6), rep("a1",6), rep("a2",6))
B <- rep(c(rep("b0",2), rep("b1",2), rep("b2",2)),3)
df <- data.frame(y,A,B)
fit <- aov(y ~ A*B, df)
summary(fit)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## A            2  78.11   39.06  10.493 0.00445 **
## B            2  47.44   23.72   6.373 0.01887 *
## A:B          4  19.89    4.97   1.336 0.32864
## Residuals    9  33.50    3.72
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

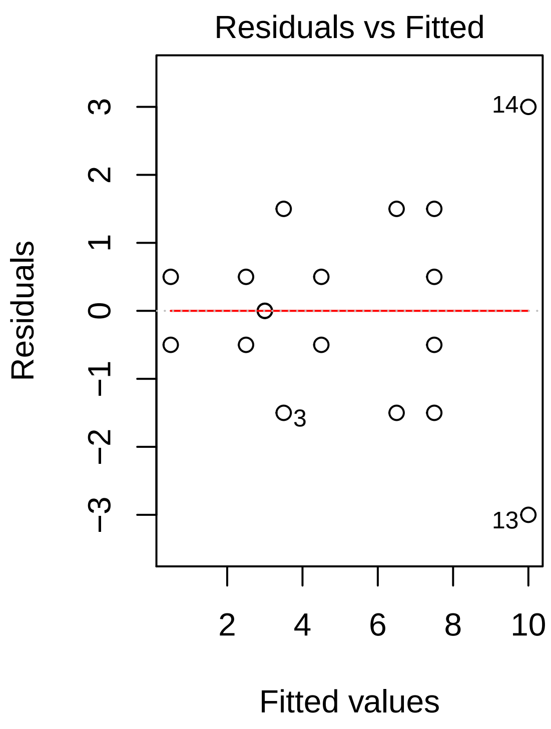
Here, we look at the interaction plot:

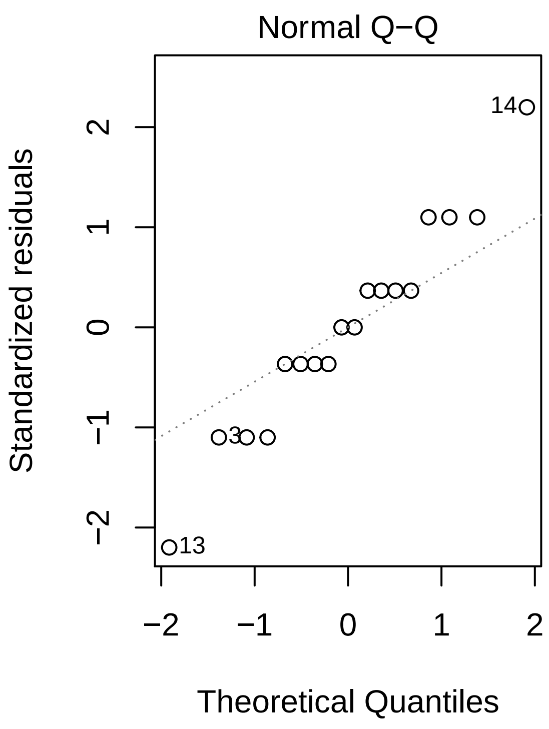
```
with(df, interaction.plot(A, B, y, col = 2:4, xlab = "A", ylab = "y",
                          trace.label = "B"))
```



And some residual plots:

```
par(mfrow=c(1,2))
plot(fit, which = c(1,2))
```





Residuals don't look great here.