```
> #
```

We will first go through some of the techniques we have for detecting problems with assumptions. Then we will go back and consider various data analytic techniques for accommodating violations of assumptions.

```
> library(Stat5303libs);library(cfcdae)
> library(oehlert);emp03.2
```

These are the log resin lifetime data again.

```
> emp03.2 <- within(emp03.2,{Time<-10^logTime);ftemp<-factor(temp)})
```

Within the data frame we will add an un-logged time and a factor for temperature.

```
> outTime <- lm(Time~ftemp,data=emp03.2)
```

OK, now let's fit the model.

```
> anova(outTime)
```

And here's the anova. It's still significant on the original scale, at least it looks that way.

```
Analysis of Variance Table

Response: Time
          Df  Sum Sq Mean Sq F value    Pr(>F)
ftemp      4 28027.9  7007.0  101.81 < 2.2e-16 ***
Residuals 32  2202.4    68.8
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> residuals(outTime)
```

There are several functions to examine residuals from a fitted model. Most simply, you can get residuals.

```
          1           2           3           4           5
 23.2236332  -5.1411348  13.5758135  -3.2478094 -15.6296080
          6           7           8           9          10
  4.7768975 -10.5664290  -6.9913630   2.1489895   7.7263089
         11          12          13          14          15
-17.2571495  13.9841643   2.1489895  -2.8218017  -8.0784905
         16          17          18          19          20
  2.1489895   9.3647619  10.1540313  -0.5313245  -4.1022743
         21          22          23          24          25
 -2.1324423  -5.8987824  -6.3226451  -0.5313245  -1.5932700
         26          27          28          29          30
  0.8772236  -0.9275616  -1.2642477   0.4994555   3.3359617
         31          32          33          34          35
 -0.9275616   6.3243432  -5.1118356   0.1499790  -1.4013799
         36          37
  0.4300224  -0.3911291
```

```
> rstudent(outTime)
```

Or Studentized residuals. The `studres()` function does the same thing. These are the leave-one-out residuals.

```
          1           2           3           4           5 
 3.47102831 -0.65658373  1.81061902 -0.41306135 -2.12136129 
          6           7           8           9          10 
 0.60948736 -1.38076739 -0.89819969  0.27288990  0.99549093 
         11          12          13          14          15 
-2.38042596  1.87113246  0.27288990 -0.35863919 -1.04242228 
         16          17          18          19          20 
 0.27288990  1.21574629  1.32376557 -0.06739441 -0.52259046 
         21          22          23          24          25 
-0.27078367 -0.75500743 -0.81036921 -0.06739441 -0.20431072 
         26          27          28          29          30 
 0.11243679 -0.11889164 -0.16207861  0.06400812  0.42876029 
         31          32          33          34          35 
-0.11889164  0.83104956 -0.66914098  0.01949216 -0.18222802 
         36          37 
 0.05589072 -0.05083526 
```

```
> rstandard(outTime)
```

Or standardized residuals.

```
          1           2           3           4           5 
 2.99265371 -0.66249911  1.74941227 -0.41852059 -2.01406921 
          6           7           8           9          10 
 0.61556260 -1.36161567 -0.90092400  0.27692400  0.99563092 
         11          12          13          14          15 
-2.22379816  1.80203334  0.27692400 -0.36362421 -1.04101389 
         16          17          18          19          20 
 0.27692400  1.20676594  1.30847311 -0.06846777 -0.52862902 
         21          22          23          24          25 
-0.27479169 -0.76013141 -0.81475139 -0.06846777 -0.20744029 
         26          27          28          29          30 
 0.11421261 -0.12076650 -0.16460230  0.06502802  0.43433496 
         31          32          33          34          35 
-0.12076650  0.83509597 -0.67499076  0.01980393 -0.18504478 
         36          37 
 0.05678217 -0.05164653 
```

```
> cooks.distance(outTime)
```

You can also get Cook's distance, but it is generally less useful in standard experimental designs because the leverages are all very close to equal. That is, there are no influential points. You can get the leverages with the `hatvalues(out)` function.

```
           1            2            3            4            5 
2.558850e-01 1.254014e-02 8.744124e-02 5.004557e-03 1.158993e-01 
           6            7            8            9           10 
1.082621e-02 5.297135e-02 2.319040e-02 2.191054e-03 2.832231e-02 
          11           12           13           14           15 
1.412937e-01 9.278069e-02 2.191054e-03 3.777788e-03 3.096314e-02 
          16           17           18           19           20 
2.191054e-03 4.160812e-02 4.891720e-02 1.339382e-04 7.984247e-03 
          21           22           23           24           25 
2.157442e-03 1.650856e-02 1.896628e-02 1.339382e-04 1.434382e-03 
          26           27           28           29           30 
4.348173e-04 4.861516e-04 9.031306e-04 1.409548e-04 6.288229e-03 
          31           32           33           34           35 
4.861516e-04 2.789541e-02 1.822450e-02 1.568783e-05 1.369663e-03 
          36           37 
1.289686e-04 1.066945e-04 
```

```
> plot(outTime)
```

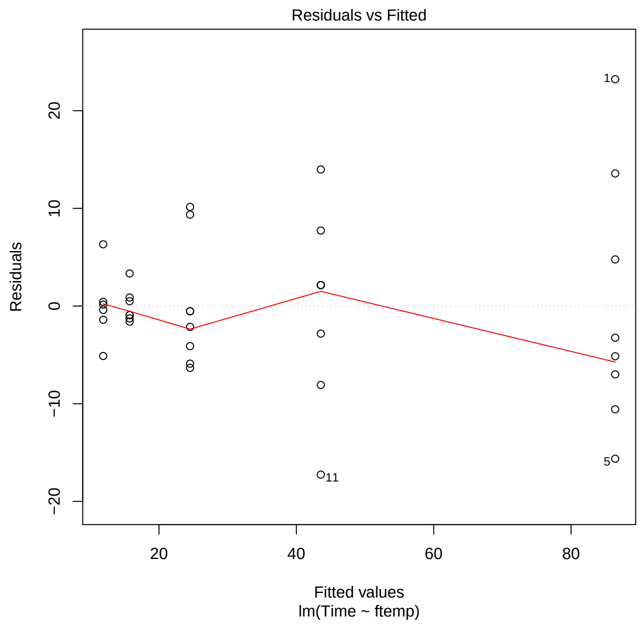
Our default method of working with residuals is to use them in diagnostic plots. When you plot a fitted model, you get four diagnostic plots. Actually, this is what you get by default. There are a couple of other plots that you can ask for as well, and you can also select plots individually.

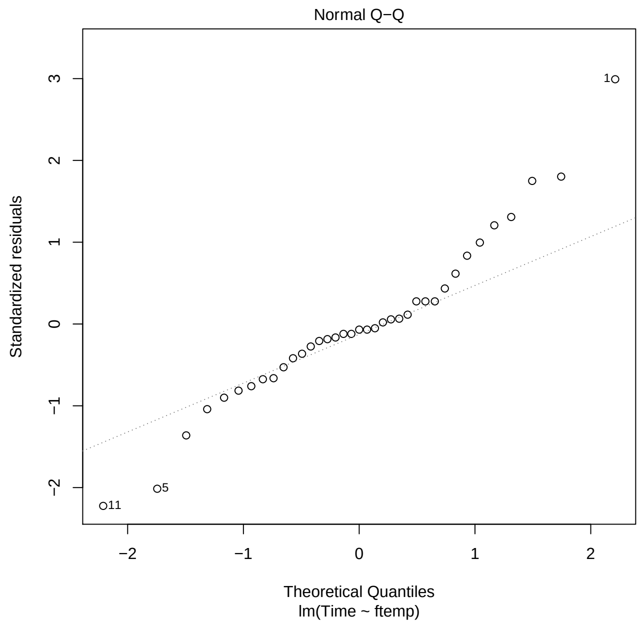
The first plot is residuals against fitted values. This ought to look very blah and even. For this model, the fitted values are the group means. We see the classic right opening megaphone shape — responses with big means have big variances.

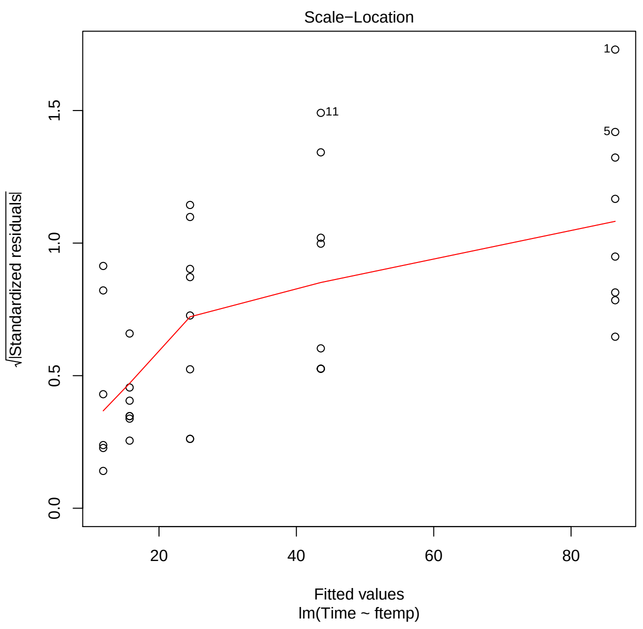
The second plot is a normal probability plot of the residuals, also called a normal qq plot. In the assumption of normality of statistical errors is correct, then this plot should look like a straight line. Here we see long tails.

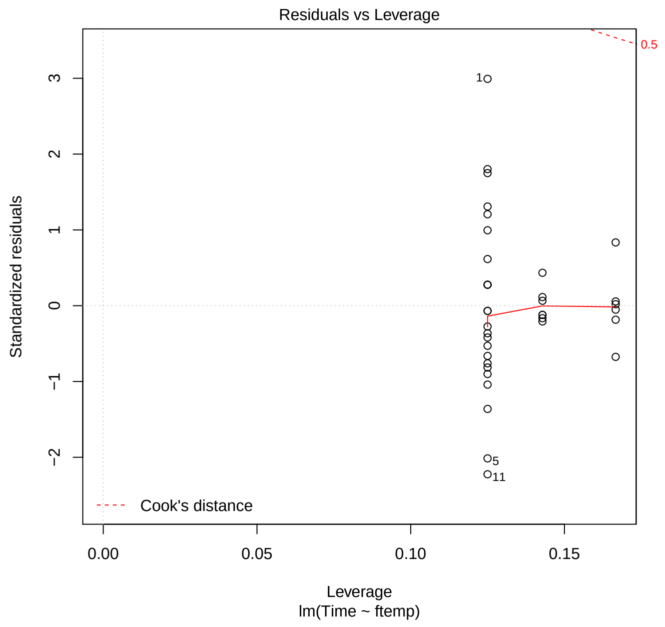
The third plot is the square root of absolute residuals plotted against fitted values. Once again, this should look flat if the residuals have constant variance.

The final standard plot is residuals against leverage. This one is not all that useful for most designed experiments, because the leverages are generally close to equal.


Residuals vs Fitted
Residuals
Fitted values
lm(Time ~ ftemp)
1
5
11
20
10
0
−10
−20
20
40
60
80


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Time ~ ftemp)
1
5
11
3
2
1
0
−1
−2
−2
−1
0
1
2

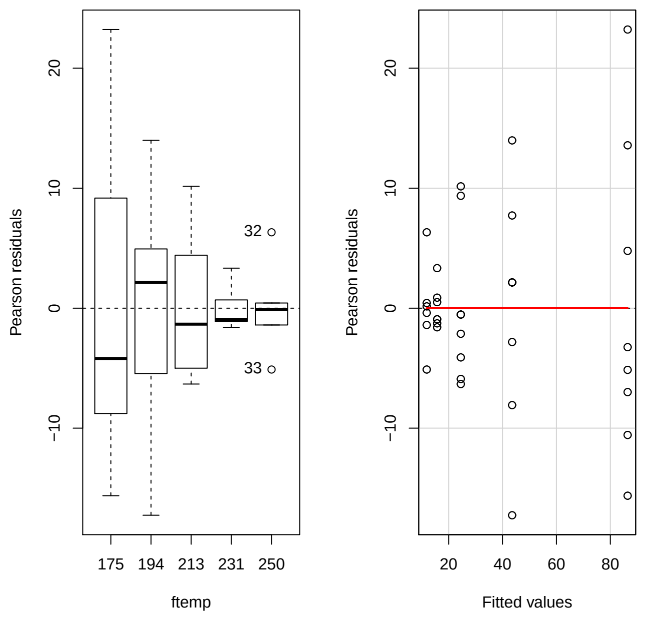

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
20
40
60
80
Fitted values
lm(Time ~ ftemp)
1
11
5


Residuals vs Leverage
Standardized residuals
−2
−1
0
1
2
3
0.00
0.05
0.10
0.15
Leverage
lm(Time ~ ftemp)
Cook's distance
0.5
1
5
11

```
> residualPlots(outTime)
```

The `car` package has a slightly different collection of plots. In some cases it also prints out some things that we haven't talked about yet (like the Tukey one degree of freedom test for non-additivity.)

The plots are residual boxplots against predictors (in this case just the single treatment factor) and residuals versus fitted values.



```
> residualPlots(outTime,layout=c(1,1))
```

For my money, the plots are too skinny. If you use this layout, then you get one plot per frame. (Not shown.)

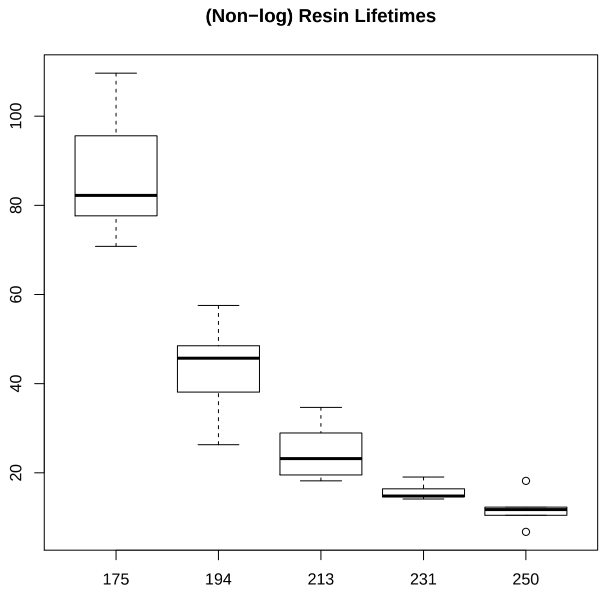
```
> leveneTest(outTime)
```

I don't usually test for equality of variances, because many of the tests are *highly* dependent on normality of the statistical errors. If you absolutely have to test for equality of variances, use Levene's test (from the car library). Levene's test is robust in the sense that when the null hypothesis is true, it only rejects at the 5% level in 5% of cases, even if the data don't follow the normal distribution. The nonrobust tests can give all kinds of nonsense. The price of this robustness is that the test may not be highly sensitive to the alternative hypothesis.

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value  Pr(>F)
group  4  2.3746 0.07286 .
      32
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

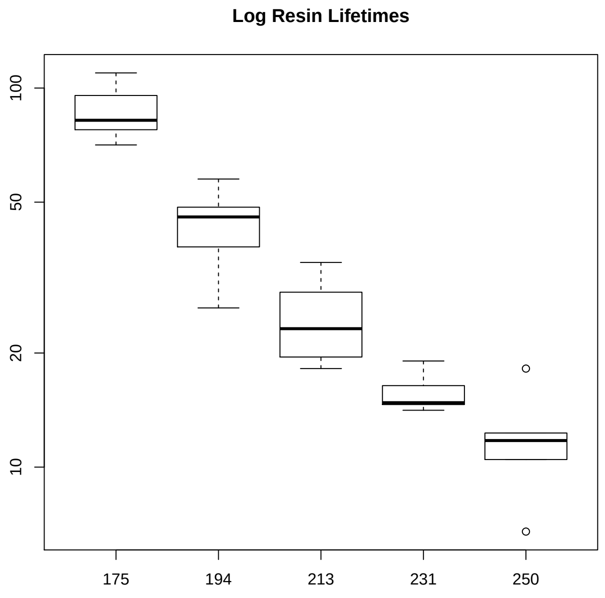
```
> boxplot(Time~temp,data=emp03.2,main="(Non-log) Resin Lifetimes")
```

Here are the data on the original scale. Notice how the larger values are also more spread out.



```
> boxplot(Time~temp,main="Log Resin Lifetimes",log='y',data=emp03.2)
```

Here are the data using a log vertical axis (but not using a specific variable). Note that this is a way to get the picture on the log scale but with labeling reflects the original scale.



```
> clouds <- read.table("http://www.stat.umn.edu/~gary/book/fcdae.data/exmpl6.1",
header=TRUE)
```

Now we're going to use the cloud seeding data of example 6.1. The response is rainfall in acre feet.

```
> clouds <- within(clouds,trt <- factor(trt))
```

Make trt a factor.

```
> outClouds <- lm(y~trt,data=clouds)
```

Fit the model.

```
> anova(outClouds)
```

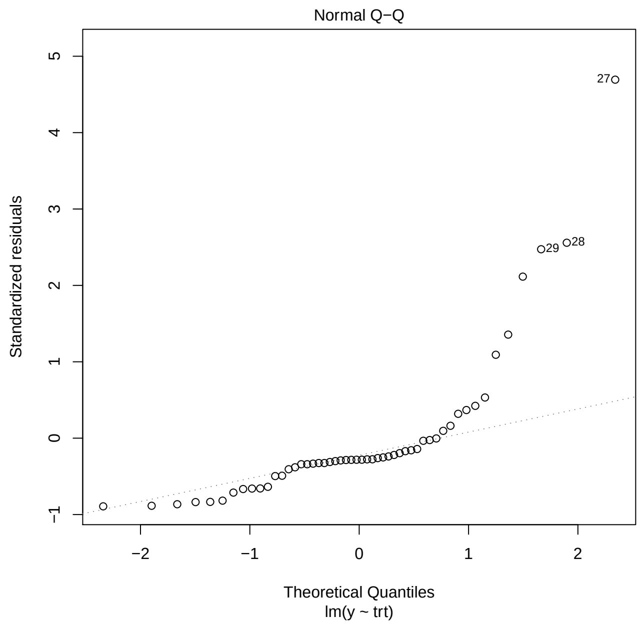
Look at the anova. Sort of marginally significant.

```
Analysis of Variance Table

Response: y
          Df   Sum Sq  Mean Sq F value  Pr(>F)
trt        1  1000332  1000332   3.993 0.05114 .
Residuals 50 12526130   250523
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> plot(outClouds,which=2)
```

Let's look at the residuals. The `which=2` just gives us the qq plot. It looks asymmetric, with possible outliers.



```
> #
```

Looking at the qq plot, we might think that the problem could be asymmetry or maybe outliers. In fact, the problem is asymmetry, but let's suppose that we looked first for outliers and went down that path instead of a transformation. We can do the real fix later.

```
> max(abs(rstudent(outClouds)))
```

Get the maximum Studentized residual (we want maximum of the absolute values). The max is about 6.2, and the plot shows that this is at point 27.

```
[1] 6.212291
```

```
> qt(.05/2/52,50,lower.tail=FALSE)
```

Compute the Bonferroni t cutoff for the studentized residuals. This is for the 5% level (the .05), two-sided (the /2), and 52 tests (the /52, one for each observation). We use 50 degrees of freedom (from df for error in the anova()).

```
[1] 3.509051
```

```
> oven <- read.table("http://www.stat.umn.edu/~gary/book/fcdae.data/exmpl6.3",
+ header=TRUE)
```

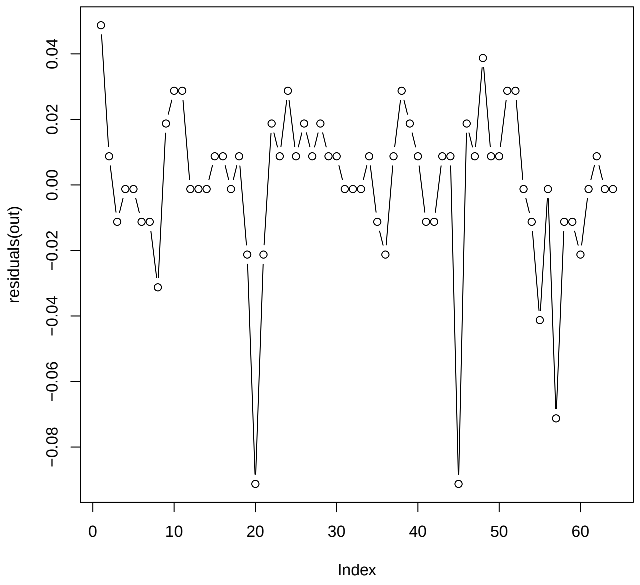
These are the oven temperature data. These data evolve over time and may have a temporal component.

```
> outOven <- lm(tempdiff~1,data=oven)
```

Fit the model.

```
> plot(residuals(outOven),type="b")
```

Plot the residuals in sequence. In this case the data are ordered in time sequence, so just plotting them against their index is good enough. The `type="b"` gives us both the points and lines between them.



```
> durbinWatsonTest(outOven)
```

The car library has a Durbin Watson test. (The stat5303 library loads car automatically.) According to Durbin Watson, serial correlation is just barely statistically significant. However, don't be too ruled by the p-value. A longer time sequence with the same degree of correlation would be significant, and a shorter sequence with stronger correlation could be insignificant.

This function simulates the distribution to get the p-value, so you will get a different value every time you use this function.

```
 lag Autocorrelation D-W Statistic p-value
   1       0.2131219      1.513854    0.044
 Alternative hypothesis: rho != 0
```

```
> runs.test(factor(sign(residuals(outOven))))
```

The tseries library (loaded by Stat5303) has a version of the runs test. This version of the runs test (and there are many versions), looks at an arrangement of a sequence of two labels, call them P and N. It then counts consecutive strings of Ps and Ns. In our case, we get the sign() of the residuals (output either +1 or -1 depending on whether the residual is above or below 0) and then turn this into a factor with two levels.

The runs test is much more convinced that there is serial correlation than is the Durbin Watson test.

Note, runs.test is going to complain if a residual comes up exactly zero. That can happen.

```
Runs Test

data:  factor(sign(residuals(out)))
Standard Normal = -3.2761, p-value = 0.001052
alternative hypothesis: two.sided
```

```
> #
```

Remember: the Durbin-Watson test and other related tests for serial correlation are only appropriate when the data have a relevant time order. You can always do the arithmetic, but the results only make sense if the statistic is based on a real time order.

```
> #
```

We have seen how to recognize problems, let's look at how to fix them (if we can).

```
> anova(outTime)
```

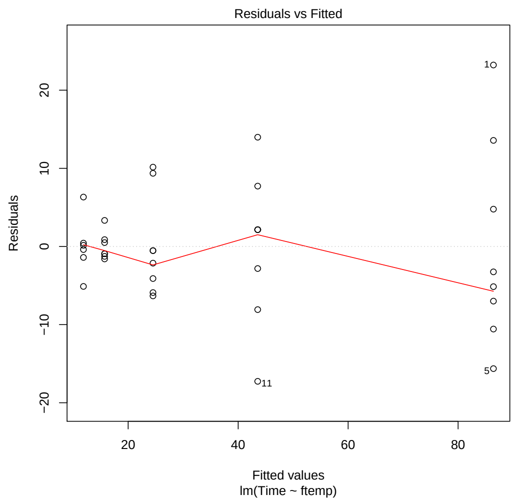
Let's go back to the resin data and the model of Time by the factor form of temperature.

```
Analysis of Variance Table

Response: Time
          Df  Sum Sq Mean Sq F value    Pr(>F)
ftemp      4 28027.9  7007.0  101.81 < 2.2e-16 ***
Residuals 32  2202.4    68.8
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

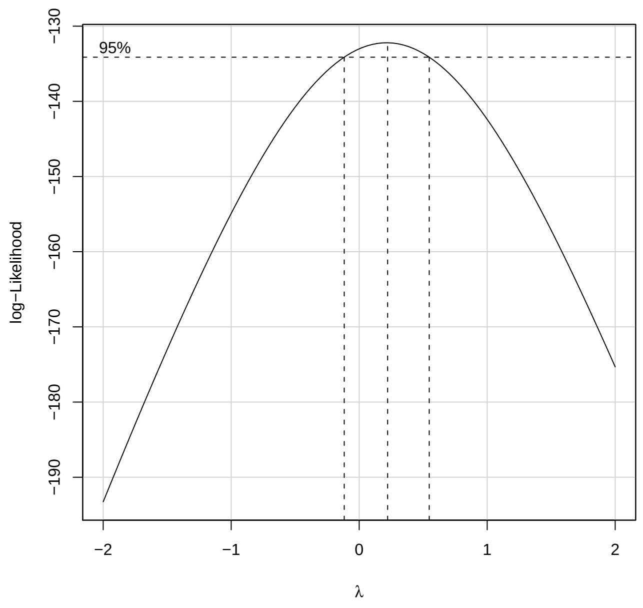
```
> plot(outTime,which=1)
```

And recall the problem of non-constant variance in the usual residuals versus predicted plot.



```
> boxCox(outTime)
```

Our major tool for dealing with nonconstant variance is a Box-Cox transformation. This function tries out a lot of different powers (lambda), and then plots the likelihoods against the power. This looks different than what the book describes, but the rescaled SS in the book is basically just the opposite of the log-likelihoods shown in this plot. Here, we want powers that give us a high log-likelihood. In this case, anything from about -.15 up to .6 or so is in our interval. So the log, .25 power, and .5 power transformations are all consistent with the data.



```
> boxCox(outTime,plotit=FALSE)
```

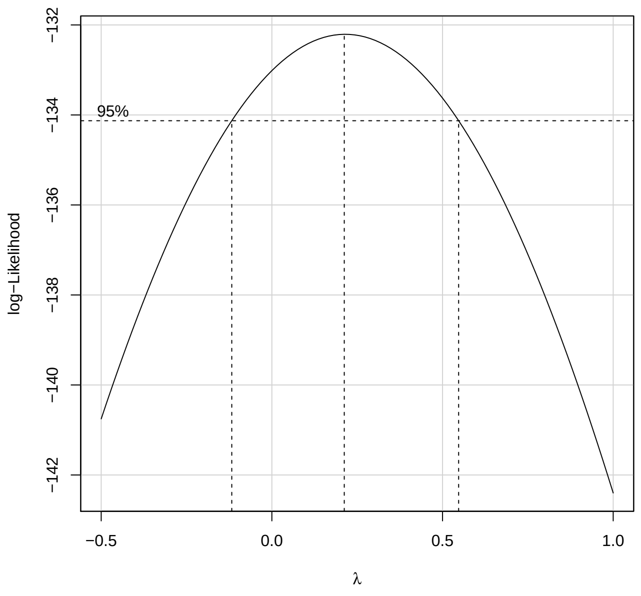
If you tell it not to plot, then you get the powers (the x component) and the log-likelihoods (the y component). So the maximum is somewhere between a power of .2 and .3, but there's no need to get really exact about it.

```
$x
 [1] -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9
[13] -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3
[25]  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4  1.5
[37]  1.6  1.7  1.8  1.9  2.0

$y
 [1] -193.2655 -189.0923 -184.9725 -180.9121 -176.9183 -172.9993
 [7] -169.1647 -165.4252 -161.7933 -158.2828 -154.9094 -151.6904
[13] -148.6446 -145.7922 -143.1546 -140.7534 -138.6105 -136.7469
[19] -135.1822 -133.9336 -133.0157 -132.4394 -132.2117 -132.3353
[25] -132.8086 -133.6258 -134.7768 -136.2479 -138.0221 -140.0799
[31] -142.3998 -144.9594 -147.7359 -150.7069 -153.8506 -157.1468
[37] -160.5767 -164.1232 -167.7712 -171.5072 -175.3192
```

```
> boxCox(outTime,lambda=seq(-.5,1,1/20))
```

You can also change the range of powers considered.



```
> #
```

There are several additional or alternative approaches to working with nonconstant variance besides the Box-Cox approach. We discuss a couple of them here.

```
> brown.forsythe.test(Time~ftemp,data=emp03.2)
```

The Stat5303 library has a function to implement the Brown Forsythe test, which does one way anova when the variances are unequal.

```
One-way analysis of means, Brown-Forsythe (unequal variances)

data:  Time and ftemp
F = 111.745, num df = 4.000, denom df = 18.298, p-value = 1.360e-12
```

```
> oneway.test(Time~ftemp,data=emp03.2)
```

The oneway.test function (built in) does a related procedure called the Welch test. This is a generalization of the unequal variance Welch t-test. It uses a different approach than Brown-Forsythe, but gives similar results in this case.

```
One-way analysis of means (not assuming equal variances)

data:  Time and ftemp
F = 68.9434, num df = 4.000, denom df = 14.373, p-value = 3.273e-09
```

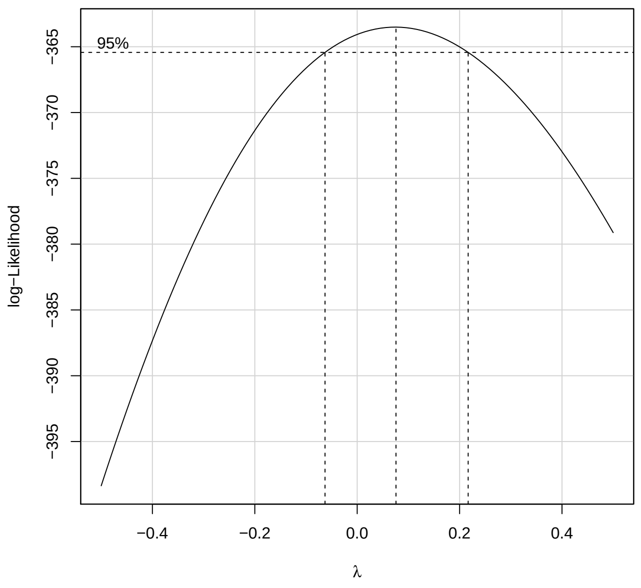
```
> max(abs(rstudent(outClouds)))
```

Now let's go back to the clouds data. It looks like outliers.

```
[1] 6.212291
```

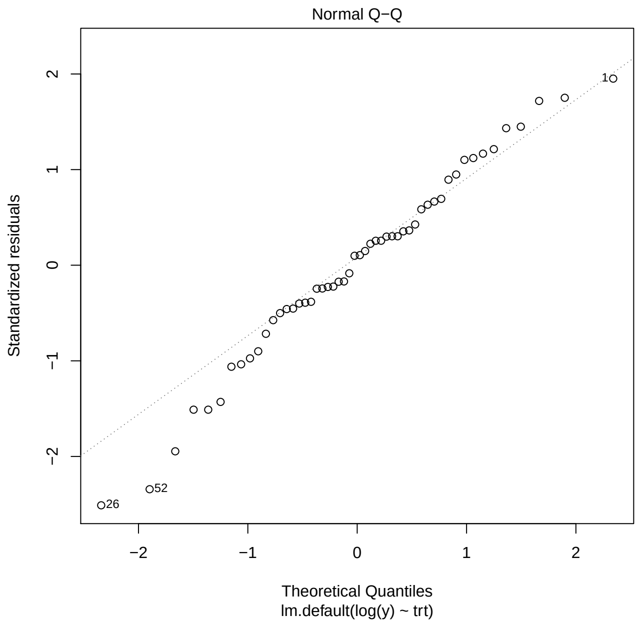
```
> boxCox(outClouds,lambda=seq(-.5,.5,.05))
```

We will look at the "outliers" in a moment, but we said before that the problem was really the scale. Note that Box Cox says these data should be logged.



```
> plot(lm(log(y)~trt,data=clouds),which=2)
```

A log transformation pretty much fixes the non-normality problem. (Constant variances looks good too.)



```
> #
```

Now let's approach this as an outlier problem. The maximum Studentized residual is way beyond our cutoff, so it looks like we have an outlier. What we'll do is make a new response variable, but set the 27th entry to "missing" and repeat the analysis.

```
> clouds <- within(clouds,yb <- y);clouds <- within(clouds,yb[27] <-NA)
```

New response, and set entry 27 to NA, which is the missing value code.

```
> outClouds2 <- lm(yb ~ trt,data=clouds);anova(outClouds2)
```

Fit the model and do the anova; the p-value has increased.

```
Analysis of Variance Table

Response: yb
          Df  Sum Sq Mean Sq F value  Pr(>F)
trt        1  437388  437388  3.0586 0.08657 .
Residuals 49 7007221  143005
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
> rstudent(outClouds2);max(abs(rstudent(outClouds2)))
```

Look at the Studentized residuals again. We have a maximum of 4.2 (at index 28), which is again bigger than the cutoff.

```
          1           2           3           4           5
 3.022770811  1.837744020  0.556455381  0.484047569  0.418774348
. . .
         26          28          29          30          31
-0.437503224  4.214632659  4.038565904  1.729452162  0.953341248
. . .

[1] 4.214633
```

```
> clouds <- within(clouds,yb[28]<-NA);outClouds3<-lm(yb~trt,data=clouds);anova(outClouds3)
```

So delete element 28 and refit the model; p-value increases again.

```
Analysis of Variance Table

Response: y2
          Df  Sum Sq Mean Sq F value Pr(>F)
trt        1  207958  207958  1.9517 0.1688
Residuals 48 5114516  106552
```

```
> max(abs(rstudent(outClouds3)))
```

Recheck residuals. They're getting bigger! Big outliers can hide or mask smaller outliers. In fact, you have to delete seven values before all the Studentized residuals are less than 3.5 in absolute value (we should probably recompute our cutoff at each step too, since we have fewer test and fewer df for error after each removal, but it doesn't change much).
The problem here is not a few outliers in normal data, the data just aren't normal at all. We have already seen that a transformation helps.

```
[1] 5.351822
```

```
>>>>> summary(outOven)
```

I know I said that we wouldn't try to do anything about temporal dependence in this course, but for the truly hard core we can go back to the oven example with temporal correlation for a moment. Here is the summary for the standard model. Note that the standard error of the mean (intercept) is about .003.

```
Call:
lm.default(formula = tempdiff ~ 1, data = oven)

Residuals:
     Min       1Q   Median       3Q      Max
-0.09125 -0.01125  0.00375  0.00875  0.04875

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.141250   0.003138    1001   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 0.0251 on 63 degrees of freedom
```

```
>>>>> oven <- within(oven,one<-0*tempdiff+1)
```

I'm just adding a variable that is a constant 1.

```
>>>>> library(nlme)
```

We will use this library more later in the class, but this is just a sneak peak.

```
>>>>> outOven2 <- lme(tempdiff~1,random=~1|one,correlation=corARMA(p=1,q=0),data=oven)
```

This complicated command says to fit the mean but allow the errors to have a specific kind of temporal correlation structure—ARMA(1,0)

```
>>>>> summary(outOven2)
```

There is a lot to read here, but I just want to direct you to the standard error of the mean (intercept) is about .009. That is three times what it was estimated to be ignoring the correlation.

```
Linear mixed-effects model fit by REML
 Data: oven
        AIC      BIC   logLik
  -276.9285 -268.356 142.4643

Random effects:
 Formula: ~1 | one
        (Intercept)   Residual
StdDev: 0.008417104 0.02525238

Correlation Structure: AR(1)
 Formula: ~1 | one
 Parameter estimate(s):
      Phi
0.2435789
Fixed effects: tempdiff ~ 1
               Value   Std.Error DF  t-value p-value
(Intercept) 3.141487 0.009330874 63 336.6766       0

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-3.6228911 -0.4548726  0.1391309  0.3371320  1.9211413

Number of Observations: 64
Number of Groups: 1
```