```
> #
```

Make sure that you have the latest version of Stat5303 (0.7-1).

```
> library(Stat5303);library(lme4)
```

We're going to need lme4, so we may as well load it now. *Warning: if you Google for lme4 you may find a link to an "lme4 book." It appears that some of the cooler functions illustrated and used in that book are not in our version of lme4. The book uses an lme4a, but it is not readily available.*

```
> bulls <- read.table("kuehl1.dat.txt",header=TRUE)
```

These are the data from exercise 5-1 of Kuehl (1994, Duxbury). There are 5 bulls selected at random, and we observe the birth weights of male calves. Sire is considered random, and we make it a factor.

```
> bulls <- within(bulls,sire <- as.factor(sire))

> bulls
   sire wts
1     1  61
2     1 100
3     1  56
4     1 113
5     1  99
6     1 103
7     1  75
8     1  62
9     2  75
10    2 102
11    2  95
12    2 103
13    2  98
14    2 115
15    2  98
16    2  94
17    3  58
18    3  60
19    3  60
20    3  57
21    3  57
22    3  59
23    3  54
24    3 100
25    4  57
26    4  56
27    4  67
28    4  59
29    4  58
30    4 121
31    4 101
32    4 101
33    5  59
34    5  46
35    5 120
36    5 115
37    5 115
```

```
38    5  93
39    5 105
40    5  75
```

```
> #
```

OK, before jumping into REML, we will take just a little taste of the “old school” method for random effects. This example is one of the situations where old school is just dead easy. The old school basically takes the fixed effects approach, ANOVA, and tries to fix it up for random effects. It works reasonably well in simple situations, but it doesn’t extend well.

```
> calves.lm <- lm(wts~sire,data=bulls)
```

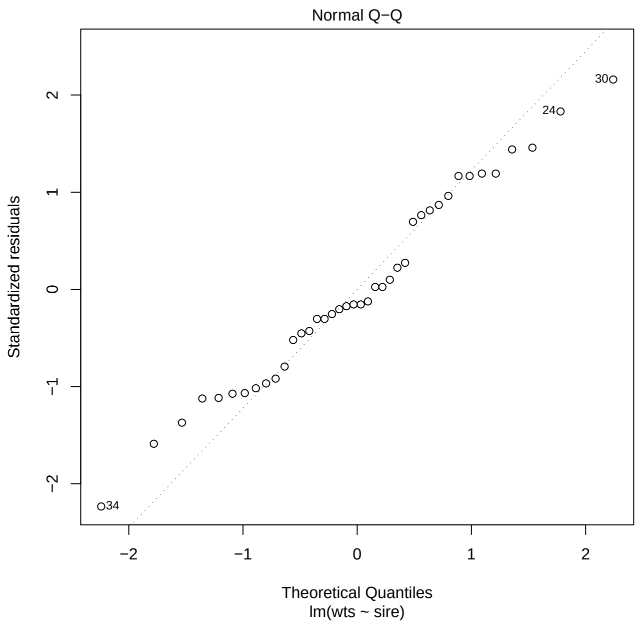
```
> anova(calves.lm)
```

This is the ordinary ANOVA. It doesn’t know anything about fixed or random effects. The DF, SS, and MS are correct. Under the null hypothesis of no sire effect, the random effects model and the fixed effects model are the same. Thus this F-test and p-value are valid. Put another way, this is the old school way of analyzing the data.

```
Analysis of Variance Table

Response: wts
          Df  Sum Sq Mean Sq F value  Pr(>F)
sire       4  5591.1  1397.8  3.0138 0.03087 *
Residuals 35 16232.8   463.8
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

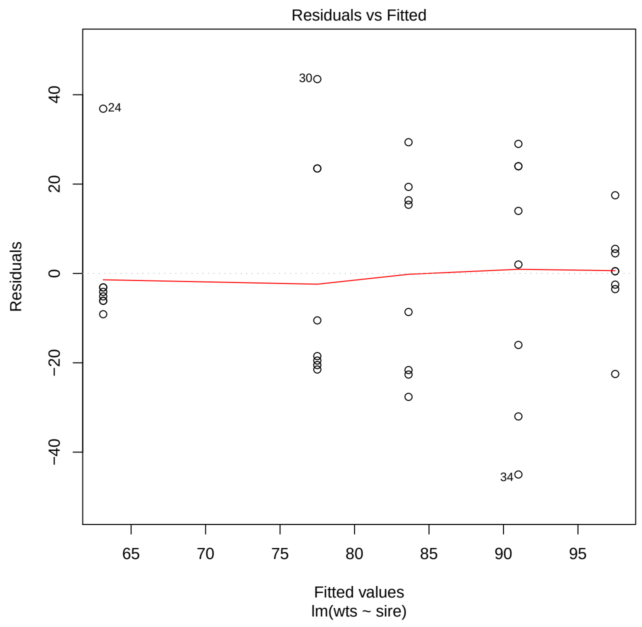
```
> plot(calves.lm,which=2)
```

Normality is pretty good.



```
> plot(calves.lm,which=1)
```

Constant variance is a little bit doubtful. If you do BoxCox, the optimum is near the log, but leaving the data alone (power 1) is well within the confidence interval. (It takes a pretty strong power family transformation to do much since the ratio of largest to smallest response is only about 2.



```
> (1397.79-463.79)/8
```

In the old school approach, we set the expected values of MS equal to their observed values and then solve for the variance components. The EMS for MSE is $\sigma^2$, and in this problem the EMS for MSA is $\sigma^2 + 8\sigma^2_\alpha$. Here we solve for an estimate of $\sigma^2_\alpha$.

```
[1] 116.75
```

```
> #
```

There are two main functions for doing REML. The first is `lme` from the nlme package, and the second is `lmer` from the lme4 package. We will mostly use lmer, but we will dabble with lme from time to time.

```
> calves.lmer <- lmer(wts ~ 1 + (1|sire),data=bulls)
```

In lmer, the fixed effects terms are entered as usual. In our case, the only fixed effect term is the overall mean. The random effects terms are entered inside parentheses. In this case, the 1 in the `1|sire` means give us an additive effect (which R will call an intercept), and the `|sire` part means give us a separate, independent effect for each level of sire.

```
> anova(calves.lmer)
```

The anova function applied to an lmer model gives test statistics for the fixed effects. However, it doesn't give anything for the intercept, which is the only fixed effect in this model. Thus, the anova output for this lmer model is a little unfulfilling.

```
Analysis of Variance Table
     Df Sum Sq Mean Sq F value
```

```
> summary(calves.lmer)
```

The summary has three main parts. The first part is some statistics that give a measure of how good the fit is. REMLdev is the REML deviance, which is computed as -2 times the REML log likelihood. Differences in REMLdev are treated, kind of, sort of, like chisquare statistics. AIC and BIC penalize the REML deviance by adding a multiple of the number of parameters. For AIC, it is 2 times the number of parameters, for BIC it is log(n) times the number of parameters. logLik is the REML log likelihood, and deviance is the ordinary (likelihood, not REML) deviance. Models for a given set of data are preferred if they have smaller AIC or BIC. AIC is fairly easily convinced that you need another term in the model; BIC takes more convincing.

The next section is estimates of the random effects. We have two random effects here, one for sire and one for error. The output gives the estimated variance, along with the standard deviation (just the square root of the variance, not the variability of our estimate of variance). Here the values of 116.75 for the sire variance and 463.79 for error variance are exactly the same as what we obtained from the old school method. This is generally the case in simple, balanced problems.

The final section is the fixed effects, their estimates and their standard errors. In this model the only fixed effect is the intercept (grand mean). NOTE: standard errors for fixed effects are usually a little too small. The reason is that they are computed assuming that we know the random effects, but we don't really know the random effects. Not knowing the random effects introduces more variability.

```
Linear mixed model fit by REML
Formula: wts ~ 1 + (1 | sire)
   AIC   BIC logLik deviance REMLdev
 364.2 369.3 -179.1    363.6   358.2
Random effects:
 Groups   Name        Variance Std.Dev.
 sire     (Intercept) 116.75   10.805
 Residual             463.79   21.536
Number of obs: 40, groups: sire, 5

Fixed effects:
            Estimate Std. Error t value
(Intercept)   82.550      5.911   13.97
```

```
> 116.75+463.79/8
```

This is our estimate of the variance of $\bar{y}_{i\bullet}$. We have one $\alpha_i$ with variance $\sigma_\alpha^2$ and we average the 8 $\epsilon_{ij}$s within the treatment.
Note that this variance is simply the MS for sire from the fixed effects ANOVA divided by 8.

```
[1] 174.7237
```

```
> sqrt(174.7237/5)
```

We have 5 sires, so this should be the SE of the average of 5 treatment means. Evidently, this matches what REML is producing.
However, when we do this simple case by hand, we note that we should be using t with 4 df when forming confidence intervals, because this is really coming from the MS for sire with 4 df. Thus we should not think of this as estimate plus or minus two times standard error, because we need a t multiplier. Sadly, REML does not let us know about the “degrees of freedom.”

```
[1] 5.911408
```

```
> logLik(calves.lmer,REML=TRUE)
```

The standard measure of overall model fit is the log likelihood. We saw the log likelihood in the model summary above.

```
'log Lik.' -179.156 (df=3)
```

```
> logLik(lm(wts~1,data=bulls),REML=TRUE)
```

When we want to compare two models that differ by a random effect, we take twice the difference of the log likelihoods (this is the difference of the deviances) as a test statistic and compare it to a chi-square distribution with degrees of freedom equal to the difference in parameters. For a single random effect, this is just one df.
When we test random effects, we can regular or REML likelihoods. When we test fixed effects, we must use regular likelihoods (i.e., REML=FALSE).

```
'log Lik.' -180.5634 (df=2)
```

```
> 2*(-179.156- -180.5634)
[1] 2.8148
```

```
> pchisq(2.8148,1,lower.tail=FALSE)
```

The *apparent* p-value is about .09.

```
[1] 0.09339854
```

```
> #
```

Everyone should be a little queasy about the “apparent” in the previous annotation. Well, there’s good news and bad news here. The good news is we can do a chi square test, but the bad news is that the distribution of the chi square test when testing variance components isn’t really chi square. Ouch.
The problem with the likelihood ratio tests of variance components is that they tend to be conservative. That is, the p-values that you compute using the chi square distribution tend to be bigger than they should be. I’ve seen recommendations to divide the nominal p-value by 2, but I don’t know that anything is a sure bet.
Dividing by 2, our corrected p-value is about .045.

```
> exactRLRT(calves.lmer)
```

If you have a model with a single variance component (other than error), then you can test that component simply with exactRLRT() from the RLRsim package. As we can see, the p-value is about .03, which is what the old school anova gave us.

```
simulated finite sample distribution of RLRT.  (p-value based on 10000 simulated values)

data:
RLRT = 2.9099, p-value = 0.031
```

```
> #
```

The exactRLRT function is cool and gives us good p-values. However, what's all this about simulation?
What a function like exactRLRT does is simulate the real distribution. That is, it uses the (restricted) likelihood ratio as a test statistic, but it simulates the real distribution to get a p-value. It doesn't compare it to a chi square distribution.

```
> lmer(wts ~ 1 +(1|sire),REML=FALSE)
```

I mentioned that sometimes it is better to do ordinary likelihood instead of REML. You get that by setting REML to FALSE in the lmer command. You can see that the estimated variances tend to be smaller. In fact, the likelihood estimates of variance tend to be biased downwards, which is why people like REML.

```
Linear mixed model fit by maximum likelihood
Formula: wts ~ 1 + (1 | sire)
   AIC   BIC logLik deviance REMLdev
 369.5 374.6 -181.7    363.5   358.3
Random effects:
 Groups   Name        Variance Std.Dev.
 sire     (Intercept)  81.804   9.0446
 Residual             463.793  21.5359
Number of obs: 40, groups: sire, 5

Fixed effects:
            Estimate Std. Error t value
(Intercept)   82.550      5.287   15.61
```

```
> calves.mcmc <- lmer.mcmc(calves.lmer,10000)
```

This function allows us to do some Bayesian procedures and produces samples (tries to anyway) from the posterior distribution of the parameters. Bayes methods are becoming more popular, but they are still a little outre in applications.
The default prior distribution is assumed to be flat, but you can optionally add a very slightly informative prior that just slightly prefers small variances.
Think of the posterior distribution as telling us where we think the parameters should be, or how we should spread out our belief of where the parameters are. Posterior intervals are not confidence intervals, but they are similar in usage and a heck of a lot easier to compute in this case.
Here we compute 10,000 samples from the posterior for the fixed effects and random effects variances. By default, the function saves every 10th value; you can change that.
I'm sorry, but this function is slow.
*Note: the lme4 package includes an MCMC function of its own, but it does not appear to be working quite right (and some of the package documentation agrees).* So I wrote my own; it's dumb and it's slow, but I think it is giving the right answers.

```
> par(mfrow=c(3,2))
```

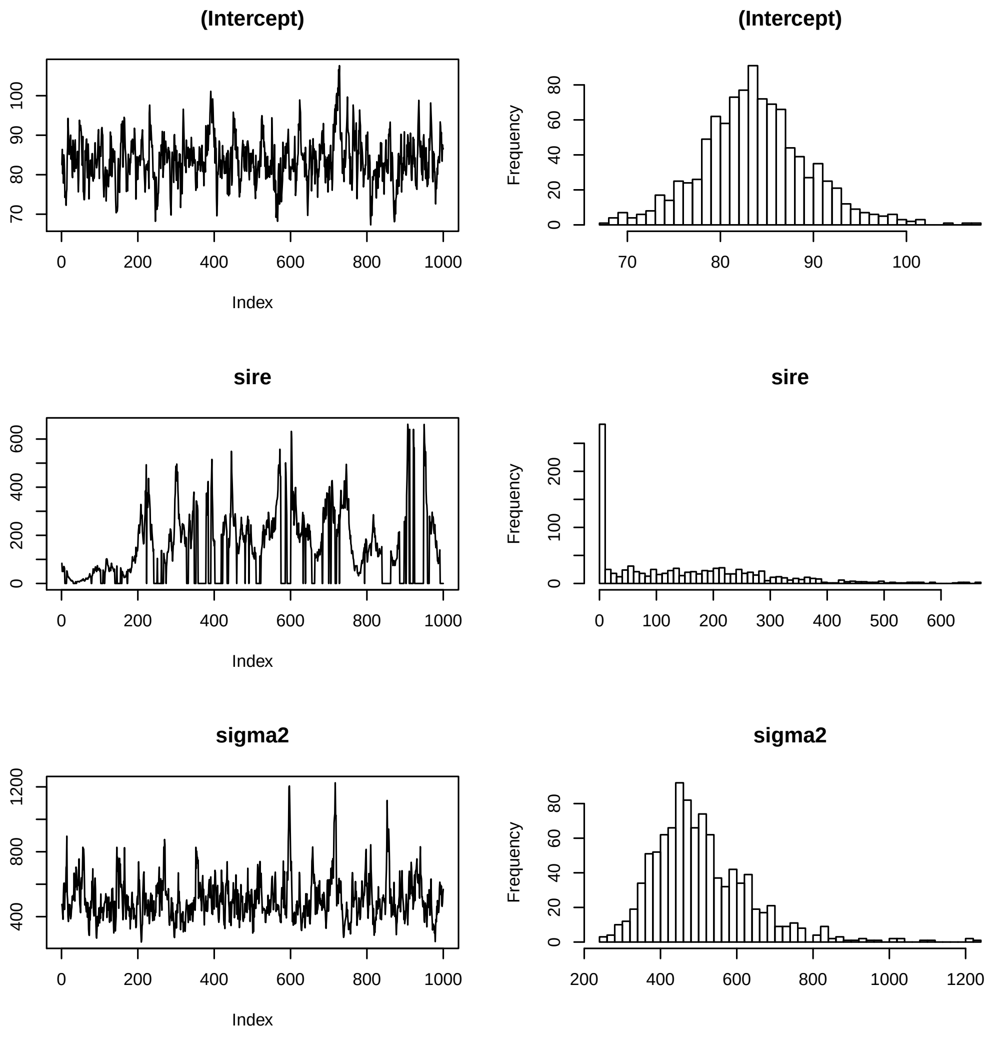
We're going to draw six plots here in a moment, and this command lets us arrange them in a 3x2 matrix and fill by rows.

```
> lmer.mcmc.plots(calves.mcmc)
```

Before we go do some inference, we want to make sure that the simulation has settled down. We do this by plotting the values against run number. The trace should look like a lot of blah nothing. If it doesn't look like blah nothing, then we need to run the simulation for more iterations. (A strong argument could be made that we may need to modify the way that our MCMC simulation is done, but my function only does things one way, and our only available "fix" is to do more runs.)

The trace plots are on the left. The traces for the intercept and for $\sigma^2$ look fairly stable, but we can't say that for the sire variance. Note that this simulation does explore some exact zero values for the sire variance.

Let's try more runs.

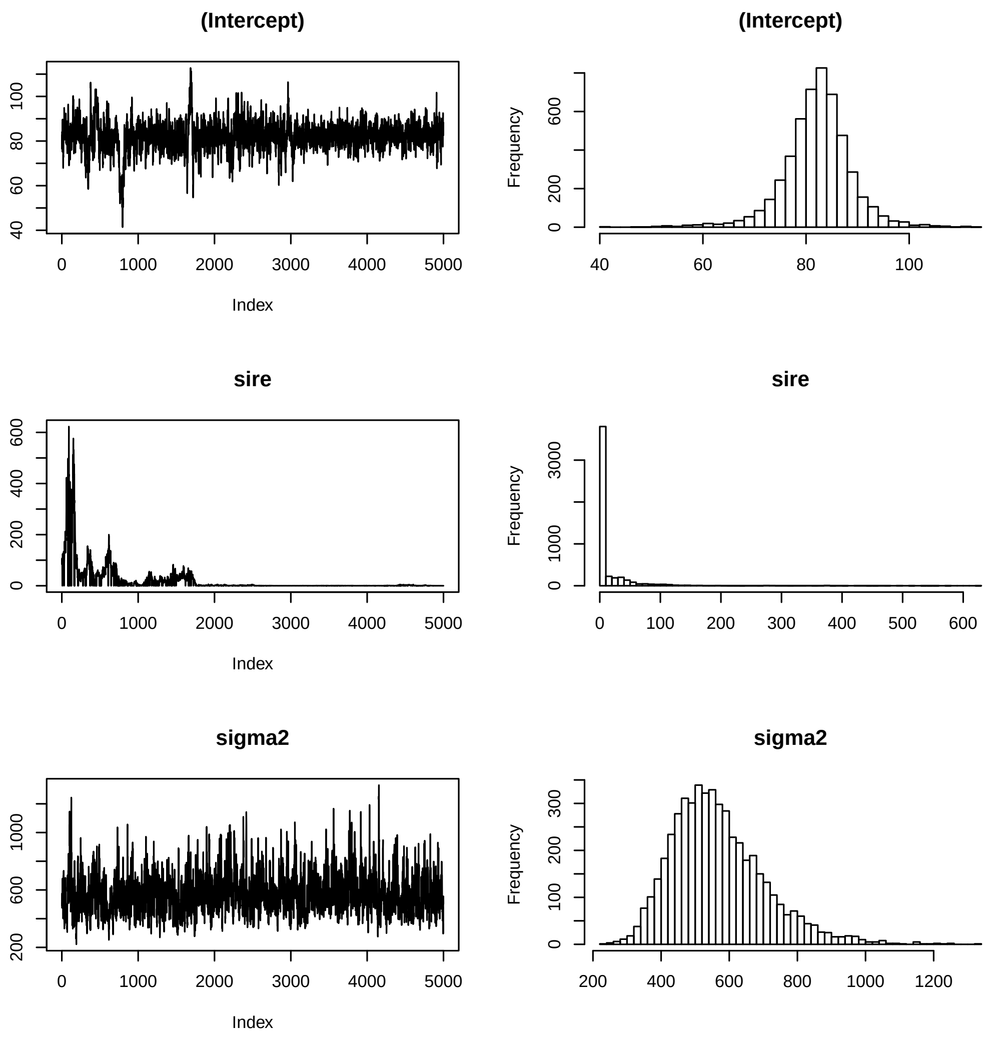


```
> calves.mcmc2 <- lmer.mcmc(calves.lmer,50000)
```

Well, let's up the iterations and see if it looks any better. Again, I have not had time to try to make this function fast. This run took about 85 seconds on my laptop.

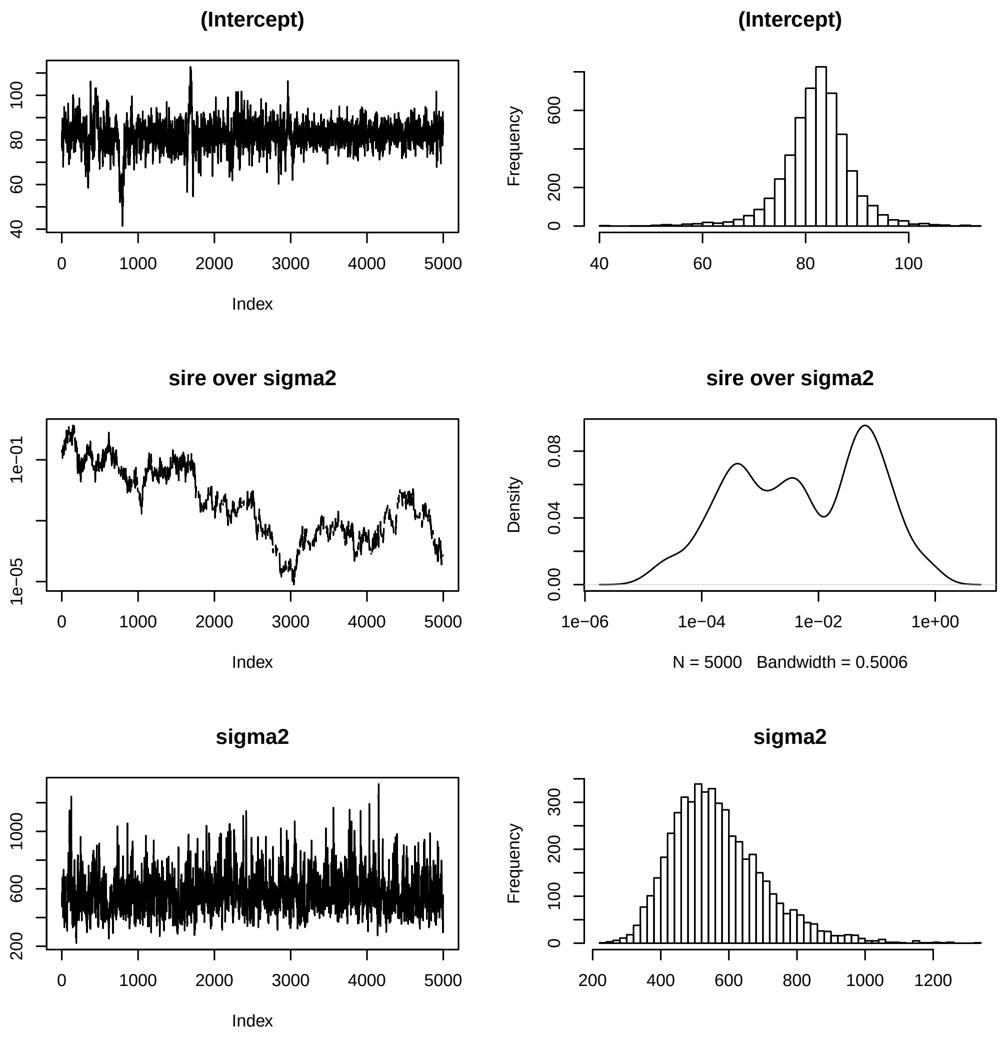
```
> lmer.mcmc.plots(calves.mcmc2)
```

I still would not go so far as to say that the sire component has really settled down, but it's better, at least. The histograms on the right are approximations to the posterior distribution. Note that the simulation has really settled in on a small value for the sire variance.



```
> lmer.mcmc.plots(calves.mcmc2,log=TRUE)
```

We can also plot variance components as the log of the ratio of the variance component to the error variance. Here we can see that the sire effect is just keeps getting smaller and smaller and smaller. That's a good hint that it might really be 0. Also, the little gaps in the sire trace are where the sire variance was simulated at 0 (and it can't plot the log of 0).



```
> lmer.mcmc.intervals(calves.mcmc2)
```

So what can we do with this? We can get posterior intervals, which are analogous to confidence intervals. By default, it's 95%, but you can reset that if you like.

A few things to notice here. First, the SE we have for the intercept is larger than that we saw from the summary of calves.lmer (5.911). That is typical, but not universal for fixed effects.

Second, the median values for the random effects variances are not necessarily matched closely with the REML estimates. In this problem, the REML estimates are not right in the middle of the intervals.

Third, we don't have a lot of information about sire to sire variability, and that shows up as an interval that is a mile wide. That is often typical for variance components based on only a few levels of the random effect (5 here); that's just life in the big city.

We also need to realize that the intervals for variance components are rather sensitive to non-normality. Well, actually they are very sensitive to non-normality. So exercise some caution in using them.

```
                lower    median     upper         SE
(Intercept)  67.66784  82.47333  94.91444   6.632444
sire          0.00000   0.12923 136.99136  52.701548
sigma2      352.74659 548.25954 891.54205 137.033587
```

```
> lmer.mcmc.anova(calves.mcmc2)
```

We can get an "anova" for the fixed effects. The degrees of freedom given are the df for the hypothesis, that is, the number of parameters being tested. The p-value is computed from the variability present in the MCMC samples. This one is a little boring, but others will be more interesting.

```
               chisq Df MC p-value
(Intercept) 154.9127  1          0
```

```
> lmer.mcmc.vcov(calves.mcmc2)
```

We can obtain the posterior variance/covariance matrix for the fixed effects. Note that this variance is just the square of the SE reported in the intervals output.

```
[1] 43.98931
```

```
> residuals(calves.lm)
```

Here are the usual residuals from the fixed effects model.

```
      1       2       3       4       5       6       7       8       9
-22.625  16.375 -27.625  29.375  15.375  19.375  -8.625 -21.625 -22.500
...
```

```
> residuals(calves.lmer)
```

Here are the residuals for the random effects model. *They are not the same!* Similarly, the estimated random level for the first sire is not the same as the estimated fixed effect for the first sire. In general, the random effects are shrunk in a little bit toward zero.

```
 [1] -22.268347  16.731653 -27.268347  29.731653  15.731653  19.731653
...
```

```
> par(mfrow=c(2,2))
```

Some more plots coming, let's set them up 2 by 2.

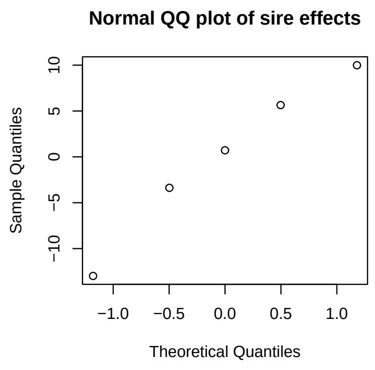
```
> lmer.plot(calves.lmer)
```

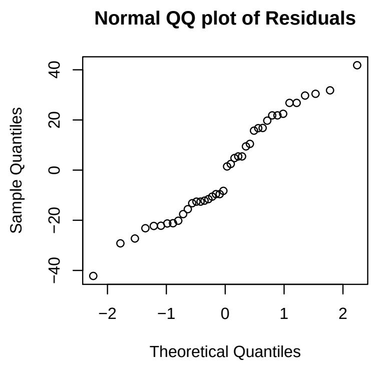
We need to try to check up on our model. We are assuming that all random effects are normal, constant variance (at least within an effect group), and independent. We can make the usual plots for residuals, subject to the consideration that they may be not quite what we are expecting, but there are usually so few individual random effects that the best we might hope for is a check of normality; constant variance seems beyond us.

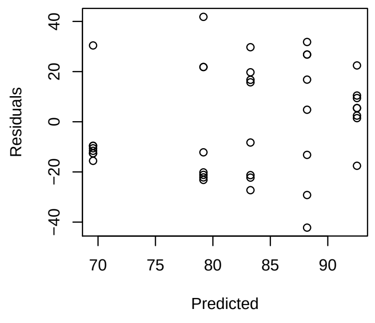
Now the fact we just saw that residuals sort of mean something different in random effects models will imply that our hard-won intuition about residuals from fixed effects models is not always going to be correct in random effects models.

What the lmer.plot() function (from the Stat5303 package) does is a normal probability plot for each random effect, and a probability plot and residuals versus predicted plot for residuals.

Things look pretty good in this data set.







```
> detach(library:lme4);library(nlme)
```

We will briefly peek at the lme() function in the nlme library. lme4 and nlme seem to step on each other's toes occasionally, so I detach one before using the other.

```
> calves.lme <- lme(wts ~ 1, random= ~1|sire,data=bulls)
```

In lme(), you specify the random effects in a separate argument from the fixed effects. For this example, there's not much to choose between them. In general, lmer() can do crossed random effects while that is very difficult in lme(). However, lme() can do some very complicated special covariance structures for random effects that cannot be done in lmer(). In addition, the non-crossed nature of the random effects in lme() makes estimating "denominator" degrees of freedom simpler, so lme() is happier about doing an anova for fixed effects.

```
> summary(calves.lme)
```

The presentation is different, but the information agrees.

```
Linear mixed-effects model fit by REML
 Data: NULL
      AIC      BIC    logLik
  364.217 369.2077 -179.1085

Random effects:
 Formula: ~1 | sire
        (Intercept) Residual
StdDev:    10.80530 21.53582

Fixed effects: wts ~ 1
            Value Std.Error DF  t-value p-value
(Intercept) 82.55  5.911487 35 13.96434       0

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.9593563 -0.7458505 -0.1580621  0.8142560  1.9420875

Number of Observations: 40
Number of Groups: 5
```

```
> detach(package:nlme);library(lme4)
```

Back to lmer.

```
> resistors <- read.table("hicksturner.dat.txt",header=TRUE);resistors
```

These are data from problem 6.18 of Hicks and Turner (1999 Oxford). Ten resistors are chosen at random, and three operators are chosen at random. Each operator measures the resistance of each resistor twice, with the 20 measurements made in random order. Response is in milliohms.

```
   part oper mohms
1     1    1   417
2     1    2   394
3     1    3   404
4     1    1   419
5     1    2   398
6     1    3   410
7     2    1   417
8     2    2   387
9     2    3   398
10    2    1   417
11    2    2   399
12    2    3   402
13    3    1   423
14    3    2   389
15    3    3   407
16    3    1   418
17    3    2   407
18    3    3   402
19    4    1   412
20    4    2   389
21    4    3   407
22    4    1   410
23    4    2   405
24    4    3   411
25    5    1   407
26    5    2   386
27    5    3   400
28    5    1   409
29    5    2   405
30    5    3   410
31    6    1   408
32    6    2   382
33    6    3   405
34    6    1   413
35    6    2   400
36    6    3   410
37    7    1   409
38    7    2   385
39    7    3   407
40    7    1   408
41    7    2   400
42    7    3   400
43    8    1   408
44    8    2   384
45    8    3   402
46    8    1   411
47    8    2   401
```

```
48    8    3   405
49    9    1   412
50    9    2   387
51    9    3   412
52    9    1   408
53    9    2   401
54    9    3   405
55   10    1   410
56   10    2   386
57   10    3   418
58   10    1   404
59   10    2   407
60   10    3   404
```

```
> resistors <- within(resistors, {oper <- as.factor(oper);part <- as.factor(part)}))
```

Make factors.

```
> mohms.lmer <- lmer(mohms ~ 1 + (1|part) + (1|oper) + (1|part:oper),data=resistors)
```

This model has the constant as the only fixed effect. We have an independent random level for each part, for each operator, and for each part by operator combination.

```
> summary(mohms.lmer)
```

An interesting feature here is that two of the random effects (part and part by operator) are estimated to have zero variance.

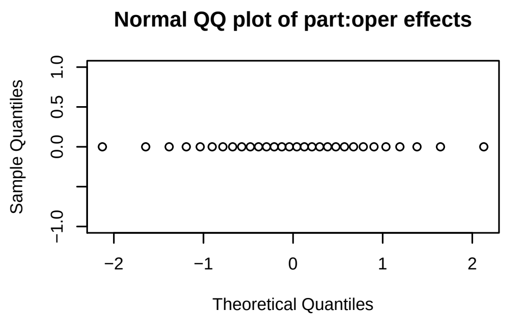
```
Linear mixed model fit by REML
Formula: mohms ~ 1 + (1 | part) + (1 | oper) + (1 | part:oper)
   AIC   BIC logLik deviance REMLdev
 406.9 417.4 -198.5      402   396.9
Random effects:
 Groups    Name        Variance Std.Dev.
 part:oper (Intercept)  0.000   0.0000
 part      (Intercept)  0.000   0.0000
 oper      (Intercept) 76.026   8.7193
 Residual              40.311   6.3491
Number of obs: 60, groups: part:oper, 30; part, 10; oper, 3

Fixed effects:
            Estimate Std. Error t value
(Intercept)    404.2        5.1   79.25
```

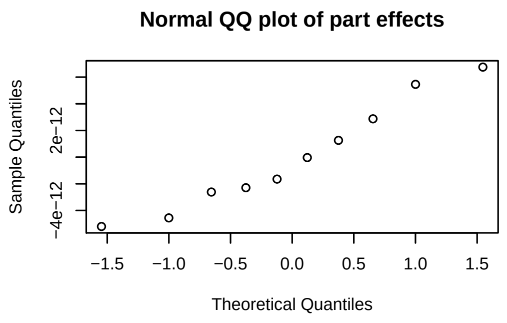
```
> par(mfrow=c(3,2))
```

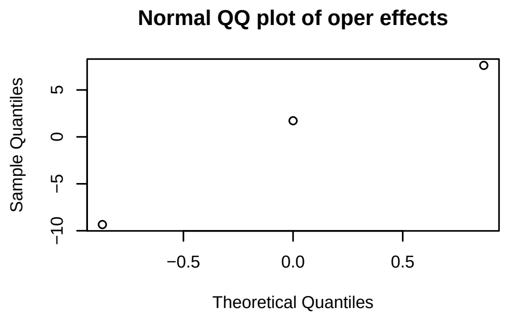
Before going further, let's look at the effects and residuals. We'll have five plots (three random effects and two residuals plots).

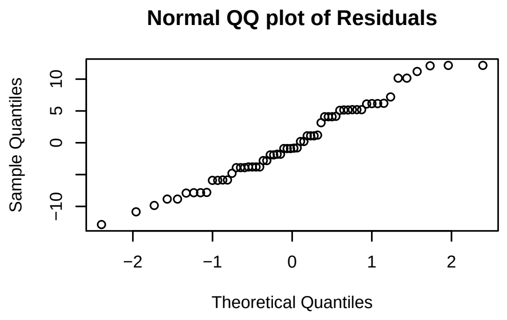
```
> lmer.plot(mohms.lmer)
```

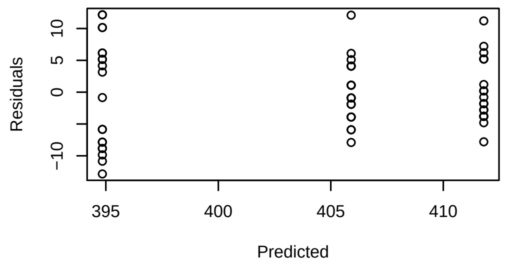
Residuals look OK, although one of the operators may be a bit more variable than the others. We only have three effects for operator, so we can't really say much about their distribution. The other two random effects are zero (or nearly so).











```
> #
```

The exactRLRT() function is a little more complex when there is more than one random effect. To test one random effect, call it A, we are going to need three fitted lmer models. The first is a model with A as the only random effect; the second is the full alternative model (with all random effects including A); the third is the null model, with all the random effects except A. So let's set this up.

```
> mohms.partonly.lmer <- lmer(mohms ~ 1 + (1|part),data=resistors)
> mohms.operonly.lmer <- lmer(mohms ~ 1 + (1|oper),data=resistors)
> mohms.intronly.lmer <- lmer(mohms ~ 1 + (1|part:oper),data=resistors)
> mohms.nopart.lmer <- lmer(mohms ~ 1 + (1 | oper) + (1 | part:oper),data=resistors)
> mohms.nooper.lmer <- lmer(mohms ~ 1 + (1 | part) + (1 | part:oper),data=resistors)
> mohms.nointr.lmer <- lmer(mohms ~ 1 + (1 | part) + (1 | oper),data=resistors)
> exactRLRT(mohms.partonly.lmer,mohms.lmer,mohms.nopart.lmer)
```

To test part we need three models, the one with only part, the full model, and the one without part. We're not surprised to find a p-value of 1 for a random effect estimated to have zero variance.

```
simulated finite sample distribution of RLRT.  (p-value based on 10000
simulated values)

data:
RLRT = 0, p-value = 1
```

```
> exactRLRT(mohms.operonly.lmer,mohms.lmer,mohms.nooper.lmer)
```

For operator, we have a significant p-value.

```
simulated finite sample distribution of RLRT.  (p-value based on 10000
simulated values)

data:
RLRT = 35.5323, p-value < 2.2e-16
```

```
> exactRLRT(mohms.intronly.lmer,mohms.lmer,mohms.nointr.lmer)
```

The interaction is not significant (it was also estimated at zero).

```
simulated finite sample distribution of RLRT.  (p-value based on 10000
simulated values)

data:
RLRT = 0, p-value = 1
```

```
> BIC(mohms.lmer,mohms.partonly.lmer,mohms.operonly.lmer,mohms.intronly.lmer,
+ mohms.nopart.lmer,mohms.nooper.lmer,mohms.nointr.lmer)
```

BIC selects the operator-only model.

```
                    df      BIC
mohms.lmer           5 417.4146
mohms.partonly.lmer  3 450.4996
mohms.operonly.lmer  3 409.2260
mohms.intronly.lmer  3 444.7582
mohms.nopart.lmer    4 413.3203
mohms.nooper.lmer    4 448.8526
mohms.nointr.lmer    4 413.3203
```

```
> AIC(mohms.lmer,mohms.partonly.lmer,mohms.operonly.lmer,mohms.intronly.lmer,
+ mohms.nopart.lmer,mohms.nooper.lmer,mohms.nointr.lmer)
```

AIC selects the same model.

```
                    df      AIC
mohms.lmer           5 406.9429
mohms.partonly.lmer  3 444.2165
mohms.operonly.lmer  3 402.9429
mohms.intronly.lmer  3 438.4752
mohms.nopart.lmer    4 404.9429
mohms.nooper.lmer    4 440.4752
mohms.nointr.lmer    4 404.9429
```

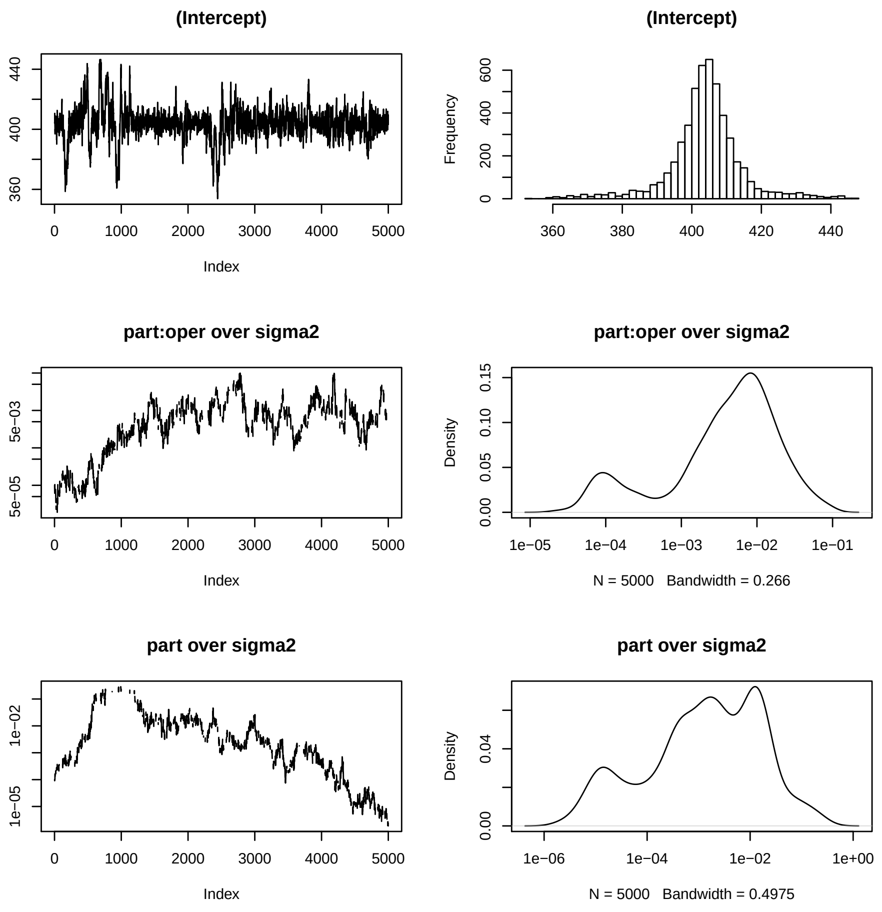
```
> mohms.mcmc <- lmer.mcmc(mohms.lmer,50000)
```

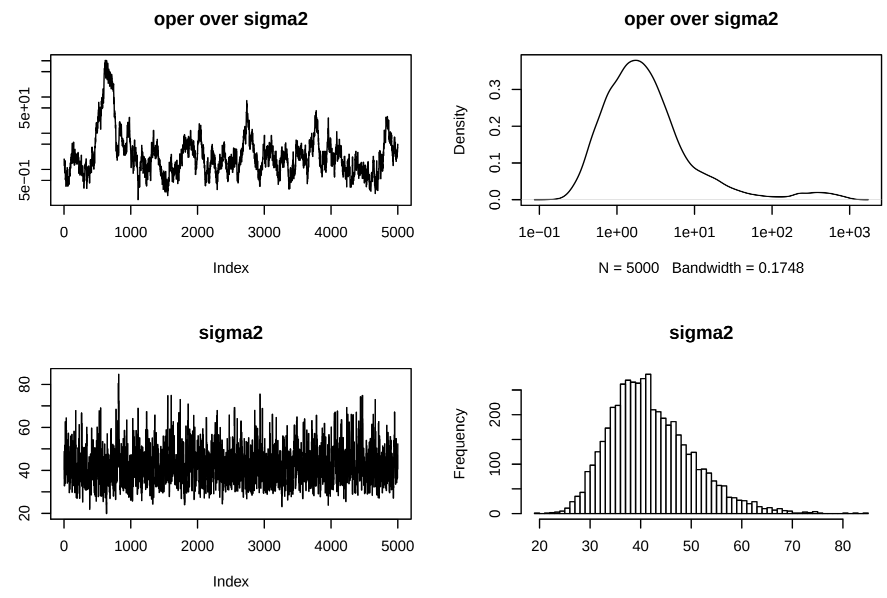
Let's see what we get with the posterior samples. I'm just going straight to 50,000 samples, although that might be overkill (162 seconds)

```
> par(mfrow=c(3,2))
```

```
> lmer.mcmc.plots(mohms.mcmc,log=TRUE)
```

We see a couple of unusual excursions for the intercept, the part and part:oper effects seem to have settled down or are heading off to zero; operator and the error variance look OK.





```
> lmer.mcmc.intervals(mohms.mcmc)
```

These are about what you might expect except for operator. The part and part:oper variances are down near zero, and the error variance is between about 30 and 60. Operator, on the other hand, is somewhere between 16.5 and 9,202, with 81.5 in the middle.

As a point of comparison, the "old style" estimates for the operator variance component aren't any tighter, with a 95% confidence interval running from 16.2 up to 6203.

```
                lower    median        upper           SE
(Intercept) 377.18066 404.18177   428.369991   10.5261268
part:oper     0.00000   0.00000     1.263120    0.3782880
part          0.00000   0.00000     1.314201    0.7630323
oper         16.54980  81.49098  9202.304403 2996.6920157
sigma2       28.96265  40.91059    60.939407    8.1788918
```

```
> #
```

So what do we conclude from all of this? There is very little variability between parts or between parts separately by operator. On the other hand, there is some substantial variability between operators, possibly much larger in size than error variability. However, we do not have enough levels of operator to get a tight estimate of the operator variance.