```
> #
```

We now want to explore how we might use restricted model assumptions. Recall that restricted model assumptions arise when we have an interaction between a fixed factor and a random factor and we assume that the random coefficients in the interaction term will add to zero when we add across the subscript for the fixed factor.

Our example will have four types of cheese and 10 randomly selected raters. The random terms in the model will be a rater effect and the cheese by rater interaction. Consider two ways that these random terms might come about. In the first approach, every potential rater has his/her own implicit likings for different kinds of cheese. If you choose Joe, he always rates all cheeses high and rates cheese 1 particularly high. If you choose Sally, she gives intermediate ratings, but always rates cheese 3 very low. In this kind of situation, choosing a rater really chooses a vector of four cheese ratings, and you would get the same four ratings again if you choose that particularly rater again. In this case the restricted model assumptions are appropriate.

On the other hand, suppose that the individual cheeses were highly variable internally, and some raters received unusually good bits of cheese and other raters received unusually bad bits of cheese. If you Joe, you could get different orderings for the cheeses depending on which particular bits of cheese Joe happens to receive. Here the interactions are really independent of Joe, and the unrestricted assumptions are appropriate.

```
> cheese.data <- read.table("cheese.txt",header=TRUE)
```

These are simulated data. There are four types of cheese, 10 randomly chosen raters (called tasters) and each rater tastes each cheese twice (all in random order).

```
> attach(cheese.data);cheese <- factor(cheese);taster <- factor(taster)
> t2 <- taster
```

t2 is just a copy of taster. We'll need it for fooling lme into thinking we have nested another variable.

```
> cheese.data
```

For your reference

```
   cheese taster score
1       1      1    27
2       2      1    20
3       3      1    49
4       4      1    39
5       1      1    29
6       2      1    20
7       3      1    51
8       4      1    36
9       1      2    29
10      2      2    37
11      3      2    41
12      4      2    49
13      1      2    28
14      2      2    37
15      3      2    41
16      4      2    50
17      1      3    21
18      2      3    19
19      3      3    16
20      4      3    21
21      1      3    22
22      2      3    20
23      3      3    15
24      4      3    20
25      1      4    72
26      2      4    65
27      3      4    85
28      4      4    83
```

```
29      1      4    72
30      2      4    66
31      3      4    86
32      4      4    85
33      1      5    57
34      2      5    59
35      3      5    37
36      4      5    50
37      1      5    59
38      2      5    58
39      3      5    38
40      4      5    52
41      1      6    68
42      2      6    63
43      3      6    75
44      4      6    68
45      1      6    69
46      2      6    65
47      3      6    74
48      4      6    67
49      1      7    26
50      2      7    25
51      3      7    28
52      4      7    32
53      1      7    28
54      2      7    25
55      3      7    30
56      4      7    31
57      1      8    25
58      2      8     1
59      3      8    26
60      4      8    19
61      1      8    26
62      2      8     5
63      3      8    25
64      4      8    20
65      1      9    71
66      2      9    59
67      3      9    66
68      4      9    57
69      1      9    70
70      2      9    56
71      3      9    66
72      4      9    58
73      1     10    42
74      2     10    40
75      3     10    51
76      4     10    57
77      1     10    41
78      2     10    38
79      3     10    52
80      4     10    60
> library(Stat5303)
> library(nlme)
```

By default, both lme() and lmer() do unrestricted model assumptions. For some models, we can force lme() to use restricted assumptions, but I don't know how to make lmer() do restricted assumptions. lme() *always* nests random effects, so we will need a way to curb its nesting enthusiasm. We do this by making copies of some random factors. For example, t2 is just a copy of taster above. When we nest t2 in taster, we just get taster again. In this way we can make a random factor appear more than once. This is nothing deep, it's just a trick to get lme() to do what we want. More generally, we can do restricted assumptions in models that have a single nested chain of random effects, because we have to use lme(), and lme() assumptions we have a chain of nested random effects with no crossing.

```
> cheese.lme <- lme(score ~ cheese,random=list(taster=~1,
    t2=pdIdent(~Restrict(cheese)-1)))
```

This command fits the mixed effects model using restricted assumptions for the cheese by taster interaction. lme() lets you specify the random portion of the model in several different ways, and we have to use every trick in the book to get restricted assumptions.

Here we specify the random portion as a list of named objects: list(name1=something, name2=somethingelse). name1 is a random factor, and we will get an independent "something" for every level of name1. In our first term, we get a different additive adjustment for every level of taster. This is just a taster main effect that we could specify by (1|taster) in lmer. As we know, lme always nests random terms, so the second term will actually be name2 nested in name1. In our command, t2 is just a copy of taster, so t2 nested in taster is just taster again. We use a new name to keep labeling working well later. We get an independent set of somethingelse effects for every level of name2 nested in name1. The hideous "pdIdent(~Restrict(cheese)-1)" is a set of effects with equal variance that add to zero across the levels of cheese, and that is what we need for restricted assumptions.

```
> summary(cheese.lme)
```

The only unusual thing here is that the Restrict(cheese) variance component is listed with three identical standard deviations. They will always be identical; just look at one of them.

```
Linear mixed-effects model fit by REML
 Data: NULL
       AIC      BIC    logLik
  455.1465 471.4617 -220.5733

Random effects:
 Formula: ~1 | taster
        (Intercept)
StdDev:    20.39859

 Formula: ~Restrict(cheese) - 1 | t2 %in% taster
 Structure: Multiple of an Identity
        Restrict(cheese)1 Restrict(cheese)2 Restrict(cheese)3 Residual
StdDev:          7.637665          7.637665          7.637665 1.118037

Fixed effects: score ~ cheese
             Value Std.Error DF   t-value p-value
(Intercept) 44.575  6.451813 67  6.908911  0.0000
cheese1     -0.475  2.102836 67 -0.225885  0.8220
cheese2     -5.675  2.102836 67 -2.698736  0.0088
cheese3      3.025  2.102836 67  1.438533  0.1549
 Correlation:
        (Intr) chees1 chees2
cheese1  0.000
cheese2  0.000 -0.333
cheese3  0.000 -0.333 -0.333

Standardized Within-Group Residuals:
         Min           Q1          Med           Q3          Max
-1.889616773 -0.476087452 -0.002249045  0.478544977  1.688083163

Number of Observations: 80
Number of Groups:
        taster t2 %in% taster
            10             10
```

```
> cheese.lme.notaster <- lme(score ~ cheese,random=list(t2=pdIdent(~Restrict(cheese)-1)))
```

Using restricted assumptions will have its greatest effect on testing of the taster effect. Here we fit the model without the taster effect.

```
> anova(cheese.lme,cheese.lme.notaster)
```

We can use anova() to compare the two models. Remember this likelihood ratio value of 251.3.

```
                  Model df      AIC      BIC    logLik   Test  L.Ratio p-value
cheese.lme            1  7 455.1465 471.4617 -220.5733
cheese.lme.notaster   2  6 704.4922 718.4766 -346.2461 1 vs 2 251.3456  <.0001
```

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

Now switch to lme4 so we can use lmer.

```
> cheese.lmer <- lmer(score ~ cheese + (1|taster) + (1|cheese:taster))
```

Use lmer to fit the model with unrestricted model assumptions.

```
> summary(cheese.lmer)
Linear mixed model fit by REML
Formula: score ~ cheese + (1 | taster) + (1 | cheese:taster)
   AIC   BIC logLik deviance REMLdev
 455.1 471.8 -220.6    456.1   441.1
Random effects:
 Groups        Name        Variance Std.Dev.
 cheese:taster (Intercept)  58.334   7.6377
 taster        (Intercept) 401.521  20.0380
 Residual                    1.250   1.1180
Number of obs: 80, groups: cheese:taster, 40; taster, 10

Fixed effects:
            Estimate Std. Error t value
(Intercept)   44.575      6.452   6.909
cheese1       -0.475      2.103  -0.226
cheese2       -5.675      2.103  -2.699
cheese3        3.025      2.103   1.439

Correlation of Fixed Effects:
        (Intr) chees1 chees2
cheese1  0.000
cheese2  0.000 -0.333
cheese3  0.000 -0.333 -0.333
```

```
> #
```

Nearly everything is the same for restricted and unrestricted except that our estimate of the taster variance component changes. In this example, the change is only a little, but it is real. In other examples, the change could be substantial.

> **cheese.lmer.notaster <- lmer(score~cheese+(1|taster:cheese))**

To do a likelihood ratio test for taster, we need to fit the model without taster and then compare.

> **anova(cheese.lmer,cheese.lmer.notaster)**

Note that the likelihood ratio test using the unrestricted model assumptions is much smaller than that using the restricted model assumptions. In general, unrestricted model assumptions produce more conservative tests.

```
Data:
Models:
cheese.lmer.notaster: score ~ cheese + (1 | taster:cheese)
cheese.lmer: score ~ cheese + (1 | taster) + (1 | cheese:taster)
                     Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
cheese.lmer.notaster  6 516.87 531.16 -252.44
cheese.lmer           7 470.06 486.73 -228.03 48.809      1  2.821e-12 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```