Multilevel Modeling Application in R

```
library(car)
library(lme4)
library(effects)
library(ggplot2)
library(dplyr)
set.seed(1234)
radon <- read.table("radon.txt", header = TRUE)
```

The Radon Study

We will use data on levels of radon gas in houses in Minnesota. Radon is a carcinogen estimated to cause several thousand lung cancer deaths per year in the U.S.

The distribution of radon in American houses varies greatly. Some houses have dangerously high concentrations.

The EPA did a study of 80,000 houses throughout the country, in order to better understand the distribution of radon.

Higher levels of uranium are expected to lead to higher radon levels, in general. And, in general, more radon will be measured in the basement than on the first floor.

Houses are situated within 85 counties. Uranium soil levels are measured at the county level. The level-1 data is at the house level, and the level-2 data is at the county level. The houses are nested in the counties.

Variables in the data:

`radon`: outcome variable, log level of radon

`uranium`: county level predictor variable, log level of soil uranium

`floor`: house level predictor variable, basement=0 and first floor=1

`county`: indicates the county by numbering them 1 - 85

```
summary(radon)
```

```
##      radon             floor           uranium             county
##  Min.   :-2.3026   Min.   :0.0000   Min.   :-0.88183   Min.   : 1.00
##  1st Qu.: 0.6419   1st Qu.:0.0000   1st Qu.:-0.47467   1st Qu.:21.00
##  Median : 1.2809   Median :0.0000   Median :-0.09652   Median :44.00
##  Mean   : 1.2246   Mean   :0.1665   Mean   :-0.13171   Mean   :43.52
##  3rd Qu.: 1.7918   3rd Qu.:0.0000   3rd Qu.: 0.18324   3rd Qu.:70.00
##  Max.   : 3.8754   Max.   :1.0000   Max.   : 0.52802   Max.   :85.00
```

```
head(radon)
```

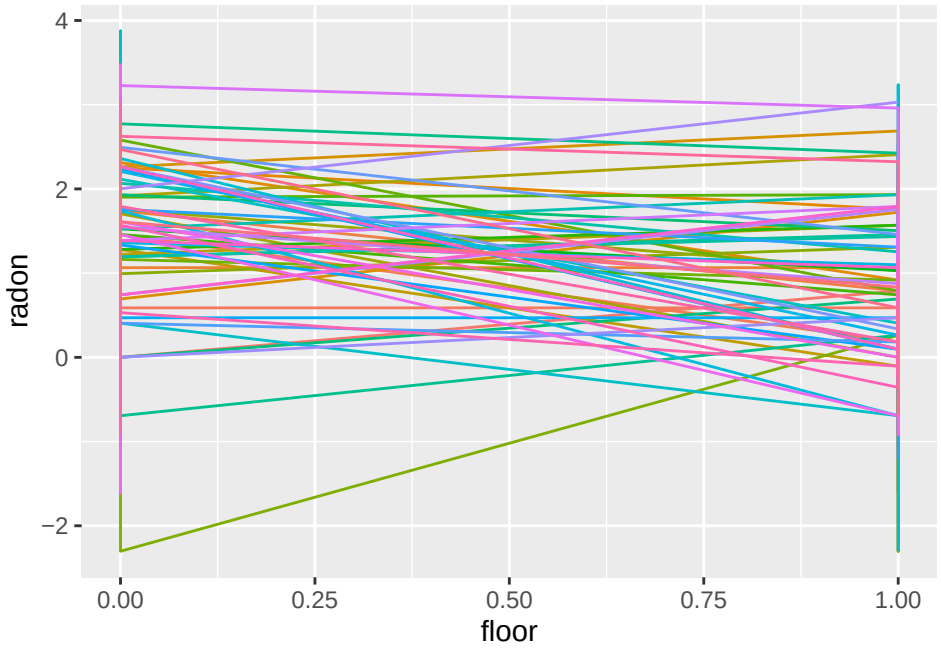
```
##       radon floor    uranium county
## 1 0.7884574     1 -0.6890476      1
## 2 0.7884574     0 -0.6890476      1
## 3 1.0647107     0 -0.6890476      1
## 4 0.0000000     0 -0.6890476      1
## 5 1.1314021     0 -0.8473129      2
```

```
## 6 0.9162907     0 -0.8473129      2
```

```
tail(radon)
```

```
##        radon floor     uranium county
## 914 2.251292     0 -0.09002427     84
## 915 1.856298     0 -0.09002427     84
## 916 1.504077     0 -0.09002427     84
## 917 1.609438     0 -0.09002427     84
## 918 1.308333     0  0.35528698     85
## 919 1.064711     0  0.35528698     85
```

```
ggplot(data = radon, aes(x = floor, y = radon,
                         group = county, color=as.factor(county))) +
    geom_line() +
    theme(legend.position="none")
```



For the sake of comparison, let's fit the OLS regression model:

```
ols <- lm(radon ~ floor + uranium, data=radon)
summary(ols)
```

```
##
## Call:
## lm(formula = radon ~ floor + uranium, data = radon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9676 -0.4809  0.0267  0.4993  2.4297
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.43489    0.02953  48.593   <2e-16 ***
## floor       -0.64589    0.06843  -9.438   <2e-16 ***
## uranium      0.78004    0.06982  11.172   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7721 on 916 degrees of freedom
## Multiple R-squared:  0.1831, Adjusted R-squared:  0.1813
## F-statistic: 102.7 on 2 and 916 DF,  p-value: < 2.2e-16
```

This model does not take into account variation predicted by county (other than the county uranium levels).

Note that we cannot include `uranium` in the next model because it is a county level predictor and is redundant with county because it is constant within county.

Fixed effects model

```
fixed <- lm(radon~ floor + factor(county) - 1, data=radon)
# factor(county) tells R to create dummy varianbles for every county
# The -1 removes the usual intercept so that we instead obtain an intercept for each county
summary(fixed)
```

```
##
## Call:
## lm(formula = radon ~ floor + factor(county) - 1, data = radon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.14595 -0.45405  0.00065  0.45376  2.65987
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## floor             -0.72054    0.07352  -9.800  < 2e-16 ***
## factor(county)1    0.84054    0.37866   2.220 0.026701 *
## factor(county)2    0.87482    0.10498   8.333 3.23e-16 ***
## factor(county)3    1.52870    0.43946   3.479 0.000530 ***
## factor(county)4    1.55272    0.28897   5.373 1.00e-07 ***
## factor(county)5    1.43257    0.37866   3.783 0.000166 ***
## factor(county)6    1.51301    0.43672   3.464 0.000558 ***
## factor(county)7    2.01216    0.20243   9.940  < 2e-16 ***
## factor(county)8    1.98958    0.37999   5.236 2.08e-07 ***
## factor(county)9    1.00304    0.23931   4.191 3.07e-05 ***
## factor(county)10   1.56391    0.31099   5.029 6.04e-07 ***
## factor(county)11   1.40113    0.33828   4.142 3.80e-05 ***
## factor(county)12   1.73025    0.37821   4.575 5.49e-06 ***
## factor(county)13   1.03872    0.30881   3.364 0.000804 ***
## factor(county)14   1.98838    0.20325   9.783  < 2e-16 ***
## factor(county)15   1.33797    0.37999   3.521 0.000453 ***
## factor(county)16   0.66486    0.53487   1.243 0.214204
## factor(county)17   1.27480    0.38221   3.335 0.000890 ***
## factor(county)18   1.12155    0.21913   5.118 3.83e-07 ***
## factor(county)19   1.33831    0.09541  14.026  < 2e-16 ***
## factor(county)20   1.80032    0.43672   4.122 4.13e-05 ***
## factor(county)21   1.73399    0.25227   6.873 1.23e-11 ***
## factor(county)22   0.63679    0.30905   2.060 0.039663 *
## factor(county)23   1.39999    0.53613   2.611 0.009183 **
## factor(county)24   2.10162    0.25267   8.318 3.64e-16 ***
## factor(county)25   1.95072    0.20243   9.636  < 2e-16 ***
## factor(county)26   1.36058    0.07422  18.332  < 2e-16 ***
```

```
## factor(county)27  1.77336    0.30978   5.725 1.45e-08 ***
## factor(county)28  1.24159    0.34115   3.639 0.000290 ***
## factor(county)29  1.05600    0.43672   2.418 0.015818 *
## factor(county)30  0.92576    0.22807   4.059 5.39e-05 ***
## factor(county)31  2.02057    0.33828   5.973 3.45e-09 ***
## factor(county)32  1.23629    0.37821   3.269 0.001124 **
## factor(county)33  2.06187    0.37821   5.452 6.58e-08 ***
## factor(county)34  1.59044    0.43946   3.619 0.000314 ***
## factor(county)35  0.81920    0.28897   2.835 0.004695 **
## factor(county)36  2.95897    0.53613   5.519 4.55e-08 ***
## factor(county)37  0.40209    0.25227   1.594 0.111345
## factor(county)38  1.86772    0.37999   4.915 1.07e-06 ***
## factor(county)39  1.74807    0.33860   5.163 3.05e-07 ***
## factor(county)40  2.31580    0.37866   6.116 1.48e-09 ***
## factor(county)41  1.96715    0.26759   7.351 4.69e-13 ***
## factor(county)42  1.36098    0.75642   1.799 0.072343 .
## factor(county)43  1.60224    0.25543   6.273 5.69e-10 ***
## factor(county)44  1.04099    0.28609   3.639 0.000291 ***
## factor(county)45  1.29541    0.21101   6.139 1.28e-09 ***
## factor(county)46  1.21461    0.33828   3.591 0.000349 ***
## factor(county)47  0.88393    0.53613   1.649 0.099583 .
## factor(county)48  1.14812    0.25227   4.551 6.13e-06 ***
## factor(county)49  1.70211    0.21010   8.102 1.93e-15 ***
## factor(county)50  2.49321    0.75642   3.296 0.001022 **
## factor(county)51  2.16504    0.37821   5.724 1.45e-08 ***
## factor(county)52  1.92769    0.43672   4.414 1.15e-05 ***
## factor(county)53  1.25080    0.43741   2.860 0.004348 **
## factor(county)54  1.30676    0.15802   8.270 5.28e-16 ***
## factor(county)55  1.61799    0.26885   6.018 2.64e-09 ***
## factor(county)56  1.10110    0.43946   2.506 0.012415 *
## factor(county)57  0.76218    0.30905   2.466 0.013855 *
## factor(county)58  1.86092    0.37866   4.915 1.07e-06 ***
## factor(county)59  1.72178    0.37999   4.531 6.73e-06 ***
## factor(county)60  1.27939    0.53487   2.392 0.016979 *
## factor(county)61  1.15873    0.13389   8.654  < 2e-16 ***
## factor(county)62  1.98301    0.33860   5.856 6.80e-09 ***
## factor(county)63  1.67070    0.43741   3.820 0.000144 ***
## factor(county)64  1.84784    0.22817   8.099 1.97e-15 ***
## factor(county)65  1.29912    0.53487   2.429 0.015357 *
## factor(county)66  1.66574    0.20648   8.067 2.50e-15 ***
## factor(county)67  1.80312    0.21101   8.545  < 2e-16 ***
## factor(county)68  1.09002    0.26743   4.076 5.02e-05 ***
## factor(county)69  1.24245    0.37821   3.285 0.001062 **
## factor(county)70  0.86763    0.07096  12.227  < 2e-16 ***
## factor(county)71  1.49184    0.15174   9.832  < 2e-16 ***
## factor(county)72  1.57990    0.23920   6.605 7.08e-11 ***
## factor(county)73  1.79176    0.53487   3.350 0.000845 ***
## factor(county)74  0.98704    0.37821   2.610 0.009223 **
## factor(county)75  1.72372    0.43741   3.941 8.80e-05 ***
## factor(county)76  2.00844    0.37866   5.304 1.45e-07 ***
## factor(county)77  1.82168    0.28609   6.367 3.17e-10 ***
## factor(county)78  1.28569    0.33956   3.786 0.000164 ***
## factor(county)79  0.61488    0.37866   1.624 0.104785
## factor(county)80  1.32952    0.11181  11.890  < 2e-16 ***
```

```
## factor(county)81  2.70953    0.43946   6.166 1.09e-09 ***
## factor(county)82  2.23001    0.75642   2.948 0.003286 **
## factor(county)83  1.62292    0.21048   7.711 3.57e-14 ***
## factor(county)84  1.64535    0.20987   7.840 1.38e-14 ***
## factor(county)85  1.18652    0.53487   2.218 0.026801 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7564 on 833 degrees of freedom
## Multiple R-squared:  0.7671, Adjusted R-squared:  0.7431
## F-statistic: 31.91 on 86 and 833 DF,  p-value: < 2.2e-16
```

As mentioned previously, that is a lot of parameters... in addition, some counties do not have measurements from very many houses (e.g., only 2 houses) and counties with data on fewer houses can end up with more extreme and less precise estimates of α_j.

Random intercept model, no predictors

Let's fit a random intercept model with no predictors so that we can obtain the intraclass correlation coefficient.

```
forICC <- lmer(radon ~ 1 + (1|county), data=radon)
summary(forICC)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: radon ~ 1 + (1 | county)
##    Data: radon
##
## REML criterion at convergence: 2259.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.4661 -0.5734  0.0441  0.6432  3.3516
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  county   (Intercept) 0.09581  0.3095
##  Residual             0.63662  0.7979
## Number of obs: 919, groups:  county, 85
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.31258    0.04891   26.84
```

The ICC is:

```
.0958/(.0958 + .6366)
```

```
## [1] 0.1308028
```

13% of the variance can be explained by counties. If the ICC were 0, that would indicate that the counties convey no information about log radon levels.

Random intercept model with level 1 predictor

We can combine random and fixed effects in the same model, which is why they are called mixed-effects models. The random intercept model with predictors is given as:

$$Y_{ij} = \alpha_j + \beta_1 x_{1ij} + \beta_2 x_{2ij} + \ldots + \epsilon_{ij}$$

$\beta_1, \beta_2, \ldots, \beta_p$ are the fixed effects coefficients, which are identical for all groups and $x_{1ij}, \ldots, x_{pij}$ are the p fixed-effect regressors for observation i in group j. As before, $\alpha_j \sim N(\mu_\alpha, \sigma^2_\alpha)$ and $\epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$. So σ^2_α is the variance among the average log radon levels of the different counties.

Let's fit the random intercept model with the level 1 predictor, floor, to our radon data:

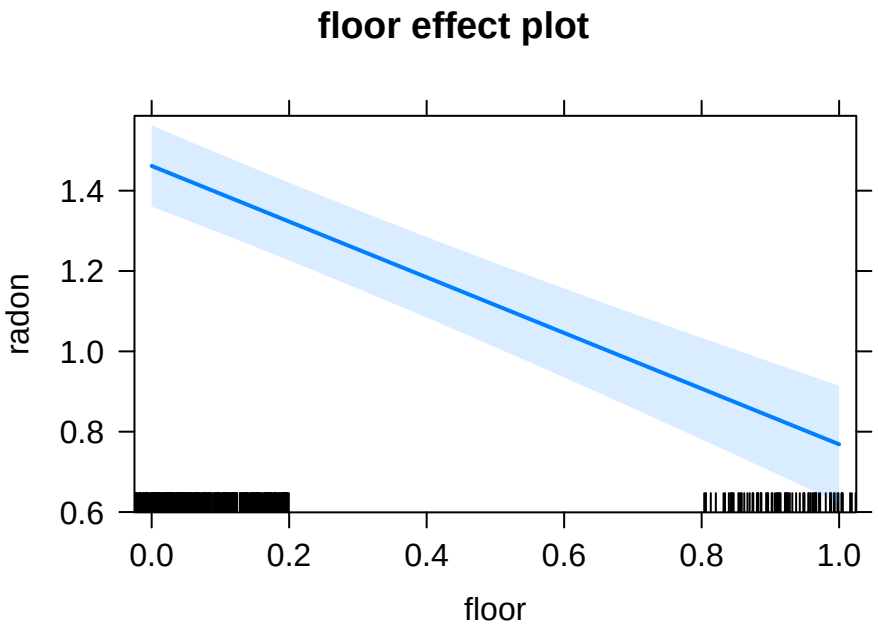
```
randomInt <- lmer(radon ~ floor + (1|county), data=radon)
summary(randomInt)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: radon ~ floor + (1 | county)
##    Data: radon
##
## REML criterion at convergence: 2171.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.3989 -0.6155  0.0029  0.6405  3.4281
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  county   (Intercept) 0.1077   0.3282
##  Residual             0.5709   0.7556
## Number of obs: 919, groups:  county, 85
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.46160    0.05158  28.339
## floor       -0.69299    0.07043  -9.839
##
## Correlation of Fixed Effects:
##       (Intr)
## floor -0.288
```

The average intercept across all the counties, that is the average log level of radon in the basement, is 1.46 but the intercepts across counties have a standard deviation of .328. The effect of measurement on the first floor, averaged across all the counties, is to decrease the level of radon by -0.693 compared to measurement in the basement.

The following code plots the fixed, that is, average effects over all counties.

```
plot(allEffects(randomInt))
```




```
fix <- fixef(randomInt)   #fixef obtains the fixed effects
rand <- ranef(randomInt)  #ranef obtains the random effects
params <- coef(randomInt) #coefficients in each county
```

The `fixef` function obtains the fixed, that is, average effects over all the counties.

```
fixef(randomInt)
```

```
## (Intercept)       floor
##   1.4615979  -0.6929937
```

To see the estimated model in each county, use the `coef` function.

```
coef(randomInt)
```

```
## $county
##    (Intercept)      floor
## 1    1.1915003 -0.6929937
## 2    0.9276468 -0.6929937
## 3    1.4792143 -0.6929937
## 4    1.5045012 -0.6929937
## 5    1.4461503 -0.6929937
## 6    1.4801817 -0.6929937
## 7    1.8581255 -0.6929937
## 8    1.6827736 -0.6929937
## 9    1.1600746 -0.6929937
## 10   1.5086099 -0.6929937
## 11   1.4322449 -0.6929937
## 12   1.5771520 -0.6929937
## 13   1.2370518 -0.6929937
## 14   1.8380232 -0.6929937
## 15   1.4024982 -0.6929937
## 16   1.2432992 -0.6929937
## 17   1.3723633 -0.6929937
## 18   1.2209415 -0.6929937
## 19   1.3462611 -0.6929937
```

```
## 20   1.5840333 -0.6929937
## 21   1.6311136 -0.6929937
## 22   1.0211902 -0.6929937
## 23   1.4409443 -0.6929937
## 24   1.8605721 -0.6929937
## 25   1.8135585 -0.6929937
## 26   1.3626875 -0.6929937
## 27   1.6222663 -0.6929937
## 28   1.3467692 -0.6929937
## 29   1.3149878 -0.6929937
## 30   1.0999775 -0.6929937
## 31   1.7329563 -0.6929937
## 32   1.3646863 -0.6929937
## 33   1.7197951 -0.6929937
## 34   1.5015319 -0.6929937
## 35   1.0870316 -0.6929937
## 36   1.8680900 -0.6929937
## 37   0.7928241 -0.6929937
## 38   1.6303574 -0.6929937
## 39   1.5979923 -0.6929937
## 40   1.8260565 -0.6929937
## 41   1.7636308 -0.6929937
## 42   1.4456250 -0.6929937
## 43   1.5404841 -0.6929937
## 44   1.2199767 -0.6929937
## 45   1.3375197 -0.6929937
## 46   1.3416955 -0.6929937
## 47   1.2995480 -0.6929937
## 48   1.2623707 -0.6929937
## 49   1.6294468 -0.6929937
## 50   1.6253581 -0.6929937
## 51   1.7641694 -0.6929937
## 52   1.6300755 -0.6929937
## 53   1.3820836 -0.6929937
## 54   1.3328317 -0.6929937
## 55   1.5494611 -0.6929937
## 56   1.3246513 -0.6929937
## 57   1.0877738 -0.6929937
## 58   1.6303984 -0.6929937
## 59   1.5675867 -0.6929937
## 60   1.4116740 -0.6929937
## 61   1.1995431 -0.6929937
## 62   1.7120493 -0.6929937
## 63   1.5338618 -0.6929937
## 64   1.7205672 -0.6929937
## 65   1.4170797 -0.6929937
## 66   1.5982671 -0.6929937
## 67   1.6981946 -0.6929937
## 68   1.2380854 -0.6929937
## 69   1.3673371 -0.6929937
## 70   0.8899487 -0.6929937
## 71   1.4829168 -0.6929937
## 72   1.5389227 -0.6929937
## 73   1.5520593 -0.6929937
```

```
## 74   1.2574762 -0.6929937
## 75   1.5530274 -0.6929937
## 76   1.6938491 -0.6929937
## 77   1.6642923 -0.6929937
## 78   1.3708520 -0.6929937
## 79   1.0944377 -0.6929937
## 80   1.3404792 -0.6929937
## 81   1.9060483 -0.6929937
## 82   1.5835784 -0.6929937
## 83   1.5716875 -0.6929937
## 84   1.5906331 -0.6929937
## 85   1.3862294 -0.6929937
##
## attr(,"class")
## [1] "coef.mer"
```

All of the slopes are identical because the model constrains them to be. The `ranef` function gives the county level errors and tell us how much the intercept is shifted up or down in a particular county.

```
ranef(randomInt)
```

```
## $county
##    (Intercept)
## 1  -0.27009754
## 2  -0.53395109
## 3   0.01761646
## 4   0.04290332
## 5  -0.01544759
## 6   0.01858386
## 7   0.39652763
## 8   0.22117574
## 9  -0.30152324
## 10  0.04701207
## 11 -0.02935303
## 12  0.11555414
## 13 -0.22454609
## 14  0.37642531
## 15 -0.05909971
## 16 -0.21829873
## 17 -0.08923457
## 18 -0.24065636
## 19 -0.11533677
## 20  0.12243538
## 21  0.16951573
## 22 -0.44040766
## 23 -0.02065353
## 24  0.39897423
## 25  0.35196061
## 26 -0.09891043
## 27  0.16066847
## 28 -0.11482868
## 29 -0.14661004
## 30 -0.36162041
## 31  0.27135840
## 32 -0.09691161
```

```
## 33  0.25819719
## 34  0.03993398
## 35 -0.37456633
## 36  0.40649210
## 37 -0.66877374
## 38  0.16875955
## 39  0.13639437
## 40  0.36445858
## 41  0.30203295
## 42 -0.01597291
## 43  0.07888624
## 44 -0.24162116
## 45 -0.12407820
## 46 -0.11990237
## 47 -0.16204985
## 48 -0.19922714
## 49  0.16784894
## 50  0.16376021
## 51  0.30257150
## 52  0.16847766
## 53 -0.07951425
## 54 -0.12876623
## 55  0.08786325
## 56 -0.13694658
## 57 -0.37382404
## 58  0.16880053
## 59  0.10598885
## 60 -0.04992386
## 61 -0.26205481
## 62  0.25045139
## 63  0.07226396
## 64  0.25896936
## 65 -0.04451817
## 66  0.13666919
## 67  0.23659675
## 68 -0.22351246
## 69 -0.09426078
## 70 -0.57164916
## 71  0.02131895
## 72  0.07732481
## 73  0.09046142
## 74 -0.20412165
## 75  0.09142954
## 76  0.23225117
## 77  0.20269438
## 78 -0.09074584
## 79 -0.36716018
## 80 -0.12111868
## 81  0.44445038
## 82  0.12198054
## 83  0.11008962
## 84  0.12903525
## 85 -0.07536847
##
```

```
## with conditional variances for "county"
```

For example, for county 1:

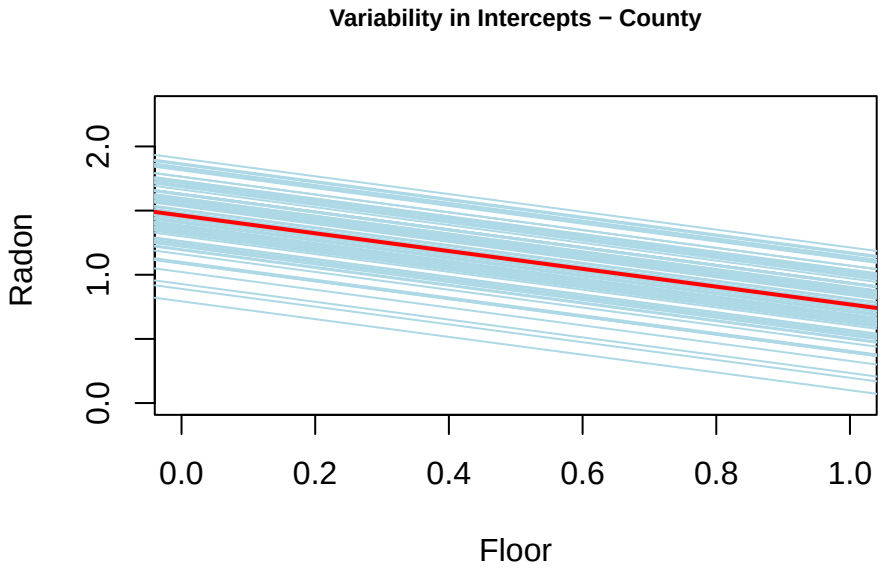
$$\widehat{y} = (1.46 - 0.27) - 0.69 * floor = 1.19 - 0.69 * floor$$

The following plots the regession line for each county.

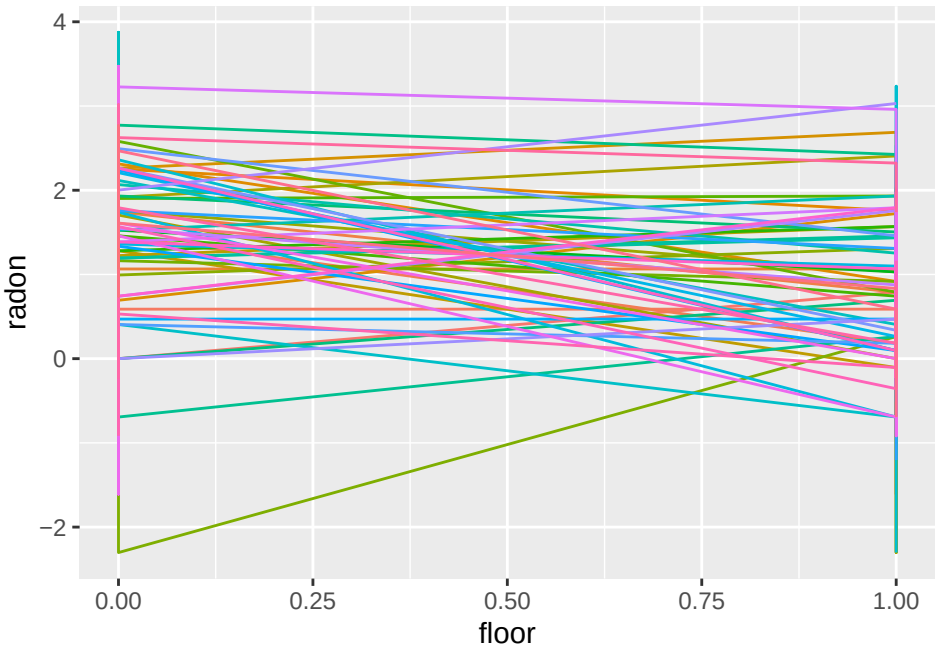
```
plot(data = radon, radon ~ floor, type = 'n',
     ylim = c(0, 2.3),
     xlim = c(0, 1),
     cex.main = .75,
     xlab = 'Floor', ylab = "Radon ",
     main = "Variability in Intercepts - County")

for(i in 1:length(unique(radon$county))){
  abline(a = params$county[i,1], b = params$county[i,2], col = 'lightblue')
  par <- par(new=F)
}

abline(a = fix[1], b = fix[2], lwd= 2,col = 'red')
```




```
ggplot(data = radon, aes(x = floor, y = radon,
                         group = county, color=as.factor(county))) +
    geom_line() +
    theme(legend.position="none")
```



An examination of our actual data though shows that not all counties have the same slope. That is, clearly the lines in the above plot of our data are not parallel. Thus, we will fit a model that allows the slopes to also vary over counties.

Random intercept, random slope model

We could also allow the coefficient of any level 1 predictor, in this case floor, to vary across counties. This is called a random intercept and random slope model. This might make sense if we expected the effect of floor to vary across county.

The model is given as:

$$Y_{ij} = \alpha_j + \beta_{1j} x_{1ij} + \epsilon_{ij}$$

where $\alpha_j \sim N(\mu_\alpha, \sigma^2_\alpha)$ and $\beta_{1j} \sim N(\mu_\beta, \sigma^2_\beta)$. β_{1j} is the random effects coefficient, which is allowed to vary across counties, hence the j subscript. As before, $\epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$. α_j and β_{1j} are allowed to be correlated but both are independent (i.e., uncorrelated) with ϵ_{ij}.

```
randomIntSlope <- lmer(radon ~ floor + (floor|county), data=radon)
summary(randomIntSlope)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: radon ~ floor + (floor | county)
##    Data: radon
##
## REML criterion at convergence: 2168.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.4044 -0.6224  0.0138  0.6123  3.5682
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  county   (Intercept) 0.1216   0.3487
```

```
##           floor       0.1181   0.3436   -0.34
##  Residual             0.5567   0.7462
## Number of obs: 919, groups:  county, 85
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.46277    0.05387  27.155
## floor       -0.68110    0.08758  -7.777
##
## Correlation of Fixed Effects:
##       (Intr)
## floor -0.381
```

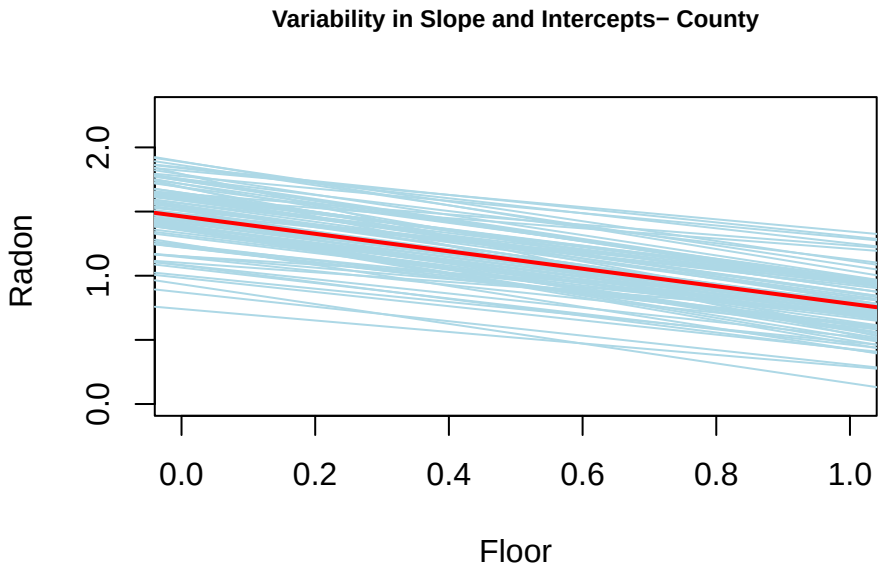
The average intercept over all the counties, that is the average log level of radon in the basement, is 1.46, but the intercepts across counties have a standard deviation of .349, which is similar to the previous model. The difference between this and the previous model is that now we have that the average effect (across all counties) of measurement on the first floor is to decrease the log level of radon by -0.661 compared to measurement in the basement, but this effect varies across the counties and has a standard deviation of 0.344. In addition, the correlation between the random intercept and random slope is -0.34 which means that a larger intercept is associated with having a smaller slope (and a smaller intercept is associated with a larger slope). It is common for the random intercept and random slope to be negatively correlated.

```
fix <- fixef(randomIntSlope)
rand <- ranef(randomIntSlope)
params <- coef(randomIntSlope)
```

```
plot(data = radon, radon ~ floor, type = 'n',
     ylim = c(0, 2.3),
     xlim = c(0, 1),
     cex.main = .75,
     xlab = 'Floor', ylab = "Radon ",
     main = "Variability in Slope and Intercepts- County")

for(i in 1:length(unique(radon$county))){
  abline(a = params$county[i,1], b = params$county[i,2], col = 'lightblue')
  par <- par(new=F)
}

abline(a = fix[1], b = fix[2], lwd= 2,col = 'red')
```



Symbols used in specifying random effects in `lme4`:

- `(1 | g)` - random intercept where `g` is the level 2 group variable.
- `x + (x | g)` - random intercept and random slope for level 1 variable `x`. You can also write this as `x + (1 + x | g)`
- `x + (-1 + x | g)` - fixed intercept but random slope for level 1 variable `x`.

In general the code looks like this:

```
random.effect.model <- lmer(outcome ~ predictor1 + predictor2 +
(predictor1 | level 2 group variable), data=yourdata)
```

Note that random effects for level 2 predictors are not possible.

Random slope only

We can also have a random slope only model. For one level 1 predictor, this model is given as:

$$Y_{ij} = \alpha + \beta_{1j} x_{1ij} + \epsilon_{ij}$$

where $\beta_{1j} \sim N(\mu_\beta, \sigma^2_\beta)$.

Note that the intercept no longer has a j subscript because it is no longer allowed to vary across counties. β_{1j} is independent of ϵ_{ij}.

```
randomSlope <- lmer(radon ~ floor + (- 1 + floor |county), data=radon)
summary(randomSlope)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: radon ~ floor + (-1 + floor | county)
##    Data: radon
##
## REML criterion at convergence: 2250.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.4721 -0.6633  0.0422  0.6899  3.2364
##
```

```
## Random effects:
##  Groups   Name  Variance Std.Dev.
##  county   floor 0.1154   0.3396
##  Residual       0.6586   0.8115
## Number of obs: 919, groups:  county, 85
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.32674    0.02932  45.247
## floor       -0.55462    0.08920  -6.218
##
## Correlation of Fixed Effects:
##       (Intr)
## floor -0.329
```

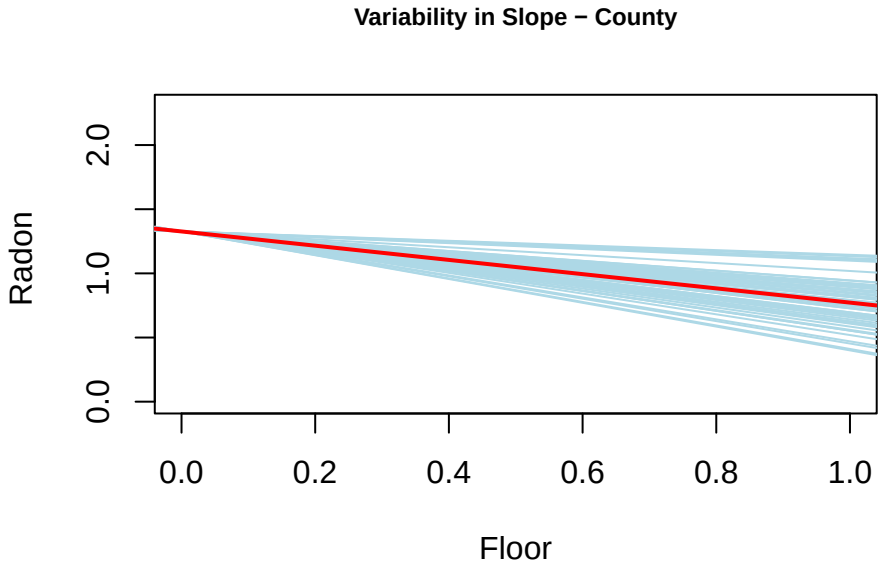
This model estimates the intercept, that is, the log level of radon in the basement, to be 1.327 across all the houses in all counties. The effect of measurement on the first floor is to decrease the log level of radon by -0.555 compared to measurement in the basement, and this effect varies across the counties with a standard deviation of 0.34.

```
fix <- fixef(randomSlope)
rand <- ranef(randomSlope)
params <- coef(randomSlope)
```

```
plot(data = radon, radon ~ floor, type = 'n',
     ylim = c(0, 2.3),
     xlim = c(0, 1),
     cex.main = .75,
     xlab = 'Floor', ylab = "Radon ",
     main = "Variability in Slope - County")

for(i in 1:length(unique(radon$county))){
  abline(a = params$county[i,1], b = params$county[i,2], col = 'lightblue')
  par <- par(new=F)
}

abline(a = fix[1], b = fix[2], lwd= 2,col = 'red')
```



In general, a random slope only model is a little strange. One case in which it may be plausible is in a weight loss study because then everyone begins with 0 kg of weight loss. Here, we know from the plot of our observed data, that this is not really a plausible model.

Adding Level 2 predictors

We can add the level 2 predictor, log soil uranium levels, to the random intercept model, which allows us to examine *contextual effects*, that is, characteristics of the county which may influence the level 1 response variable, log radon levels.

This model can be written as:

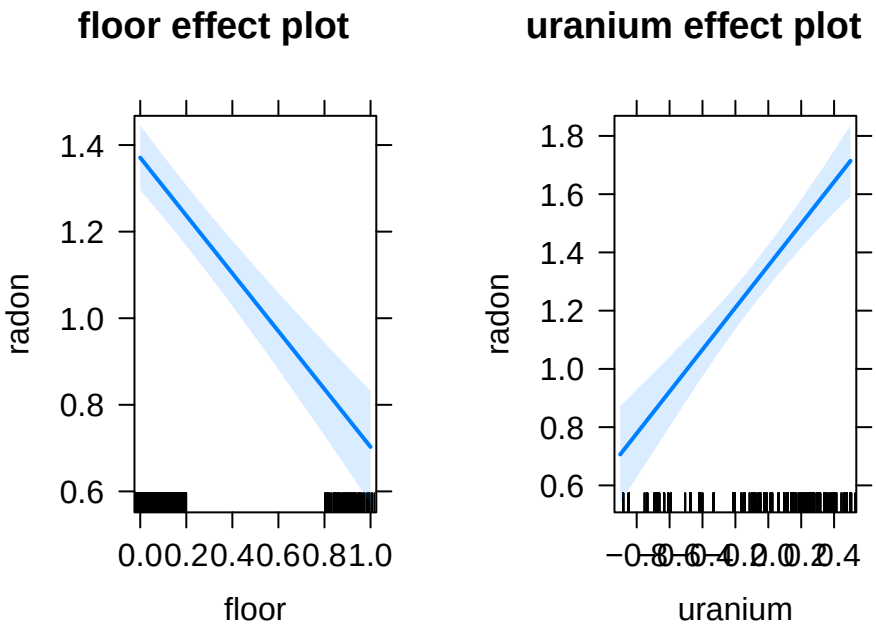
$$Y_{ij} = \alpha_j + \beta_1 floor_{ij} + \epsilon_{ij}$$

where, as usual, $\epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$ and $\alpha_j \sim N(\gamma_0 + \gamma_1 * uranium_j, \sigma^2_\alpha)$. σ^2_α represents variation among the counties that is not explained by either the house or county-level predictors (in this case, floor and log soil uranium level). Thus, the idea of adding the county level predictor is to predict the variability among the counties in the intercepts.

```
level2pred <- lmer(radon ~ floor + uranium + (1|county), data=radon)
summary(level2pred)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: radon ~ floor + uranium + (1 | county)
##    Data: radon
##
## REML criterion at convergence: 2134.2
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -4.9673 -0.6117  0.0274  0.6555  3.3848
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  county   (Intercept) 0.02446  0.1564
##  Residual             0.57523  0.7584
## Number of obs: 919, groups:  county, 85
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.46576    0.03794  38.633
## floor       -0.66824    0.06880  -9.713
## uranium      0.72027    0.09176   7.849
##
## Correlation of Fixed Effects:
##         (Intr) floor
## floor   -0.357
## uranium  0.145 -0.009
```

```
plot(allEffects(level2pred))
```



The model estimates the intercept to be 1.466. This would be the expected log radon level in the basement when log uranium soil level is zero. The house level residual variance is 0.575 and the county level residual variance is .024. That is, the residual standard deviations are 0.76 at the individual level and 0.156 at the county level. In comparison, these residual standard deviations were 0.76 and 0.33 without the uranium predictor. This predictor has left the within county variation unchanged - which makes sense, since it is a county-level predictor and cannot explain variation within any county - but has drastically reduced the unexplained variation between counties.

The effect of a measurement on the first floor (versus the basement) is to decrease the log radon level by -.668, holding constant log uranium level. For a one unit increase in log uranium level, log radon level increases by .72, holding constant the floor on which the measurement takes place.

Finally, we can have a level 2 predictor, random intercepts, and random slopes.

```
level2predRandomIntSlp <- lmer(radon ~ floor + uranium + (floor|county), data=radon)
summary(level2predRandomIntSlp)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: radon ~ floor + uranium + (floor | county)
##    Data: radon
##
## REML criterion at convergence: 2128.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.1161 -0.6127  0.0288  0.6379  3.5101
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  county   (Intercept) 0.01668  0.1291
##           floor       0.12753  0.3571   0.21
##  Residual             0.56006  0.7484
## Number of obs: 919, groups:  county, 85
##
## Fixed effects:
##             Estimate Std. Error t value
```

```
## (Intercept)  1.46265    0.03565  41.026
## floor       -0.64239    0.08737  -7.353
## uranium      0.76801    0.08888   8.641
##
## Correlation of Fixed Effects:
##         (Intr) floor
## floor   -0.260
## uranium  0.181 -0.045
```

The model estimates the intercept to be 1.463, which would be the expected radon level in the basement when uranium level is zero. The county level residual variance is 0.017 and the house level residual variance is 0.128. The effect of a measurement on the first floor (versus the basement) is to decrease the log radon level by -.642, holding constant log uranium level, and this effect varies across the counties with a standard deviation of 0.357. For a one unit increase in log uranium level, log radon level increases by .768, holding constant the floor on which the measurement takes place.

Inference

The output has not contained statistical significance for the random effects or p-values for the fixed effects.

There are two options for obtaining p-values for the fixed effects. First the Satterthwaite approximation implemented in the package `lmerTest`. When you load `lmerTest` is takes over the `lmer` function so that all you have to do is refit the model and the p-values will appear.

```
library(lmerTest)
```

```
##
## Attaching package: 'lmerTest'
```

```
## The following object is masked from 'package:lme4':
##
##     lmer
```

```
## The following object is masked from 'package:stats':
##
##     step
```

```
randomSlope2 <- lmer(radon ~ floor + uranium + (floor|county), data=radon)
summary(randomSlope2)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: radon ~ floor + uranium + (floor | county)
##    Data: radon
##
## REML criterion at convergence: 2128.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -5.1161 -0.6127  0.0288  0.6379  3.5101
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  county   (Intercept) 0.01668  0.1291
##           floor       0.12753  0.3571   0.21
##  Residual             0.56006  0.7484
## Number of obs: 919, groups:  county, 85
##
```

```
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  1.46265    0.03565 41.35183  41.026  < 2e-16 ***
## floor       -0.64239    0.08737 40.49436  -7.353 5.65e-09 ***
## uranium      0.76801    0.08888 43.89177   8.641 5.04e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##         (Intr) floor
## floor   -0.260
## uranium  0.181 -0.045
```

The second option is the Kenward-Roger approximation.

```
anova(randomSlope2, ddf = "Kenward-Roger")
```

```
## Type III Analysis of Variance Table with Kenward-Roger's method
##         Sum Sq Mean Sq NumDF  DenDF F value    Pr(>F)
## floor   28.634  28.634     1 46.427  51.127 5.163e-09 ***
## uranium 38.042  38.042     1 50.666  67.924 6.400e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To test the variance components, we can use the `anova` function. Remember, when testing variance components, it is ok to use models estimated by REML as long as the fixed effects remain the same.

The following tests whether the random slope is necessary.

```
anova(randomInt, randomIntSlope, refit=FALSE)
```

```
## Data: radon
## Models:
## randomInt: radon ~ floor + (1 | county)
## randomIntSlope: radon ~ floor + (floor | county)
##                npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## randomInt         4 2179.3 2198.6 -1085.7   2171.3
## randomIntSlope    6 2180.3 2209.3 -1084.2   2168.3 2.9807  2     0.2253
```

From this, we can conclude that the random slope does not significantly improve the model and thus, we can use the random intercept only model.

Longitudinal Data

Consider the following example, drawn from work by Blackmore, Davis, and Fox on the exercise histories of 138 teenaged girls who were hospitalized for eating disorders and of 93 "control" subjects.

There are several observations for each subject, but because the girls were hospitalized at different ages, the number of observations and the age at the last observation vary.

The variables in the data set are:

`subject`: an identification number, necessary to keep track of which observations belong to each subject.

`age`: the subject's age, in years, at the time of observation. All but the last observation for each subject were collected retrospectively at intervals of two years, starting at age eight. The age at the last observation is recorded to the nearest day.

`exercise`: the amount of exercise in which the subject engaged, expressed as hours per week.

`patient`: a factor indicating whether the subject is a patient or a control.

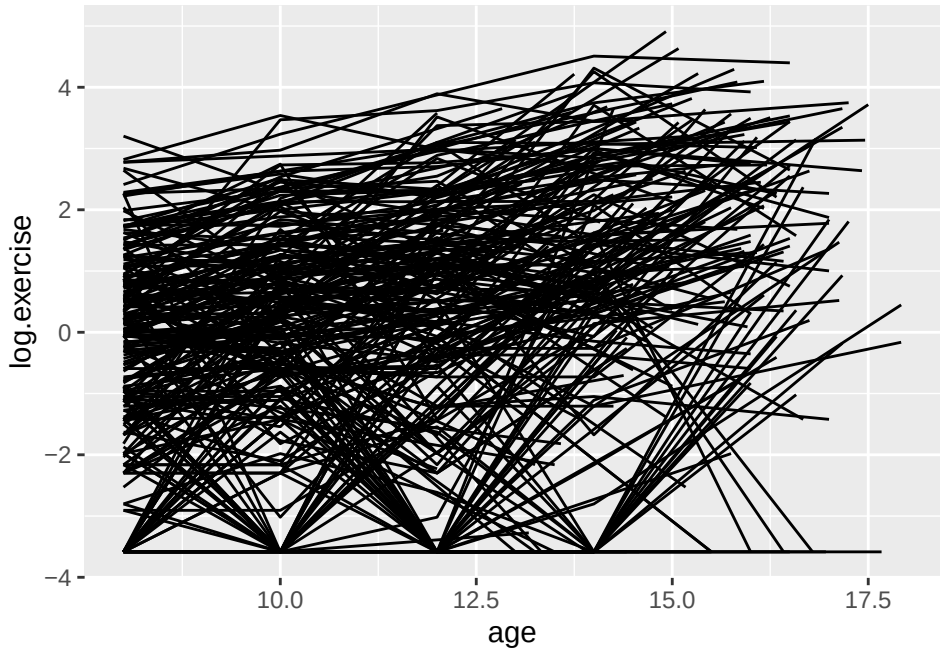
It is of interest here to determine the typical trajectory of exercise over time, and to establish whether this trajectory differs between eating-disordered and control subjects.

Preliminary examination of the data suggests a log transformation of exercise. Because about 12 percent of the data values are 0, it is necessary to add a small constant to the data before taking logs. An alternative would be to fit a model (such as an appropriate generalized linear model) that takes explicit account of the nonnegative character of the response variable.

```
Blackmore$log.exercise <- log(Blackmore$exercise + 5/60, 2)
Blackmore$patient <- Blackmore$group
```

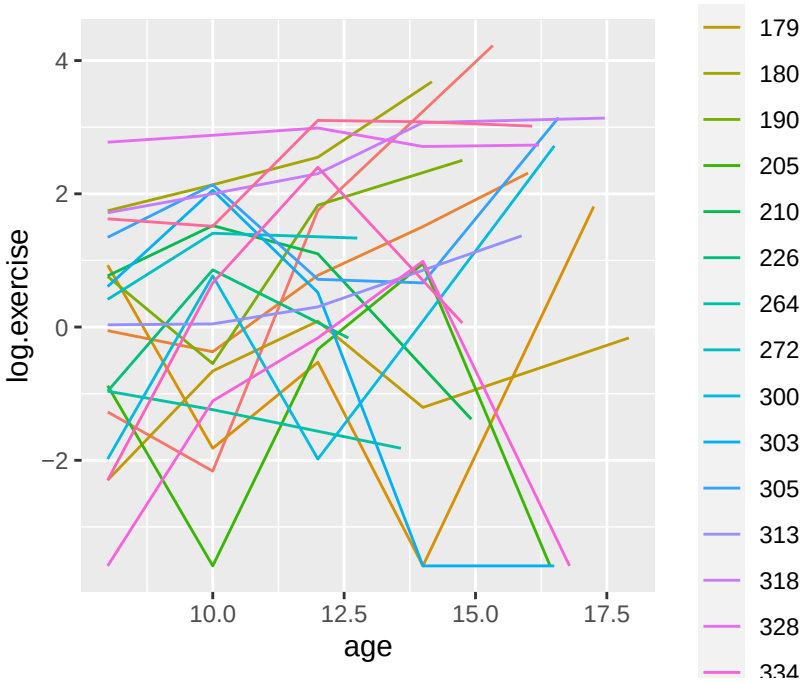
First, we need to plot the data to get an idea of the trajectories over time. Note that for longitudinal data, level-1 is the repeated observations, and level-2 is the individual. Thus, the level 2 grouping variable is the subject identification number, `subject`. *This is the variable that is used in* ***`group =`***

```
ggplot(data = Blackmore, aes(x = age, y = log.exercise, group = subject)) +
    geom_line()
```



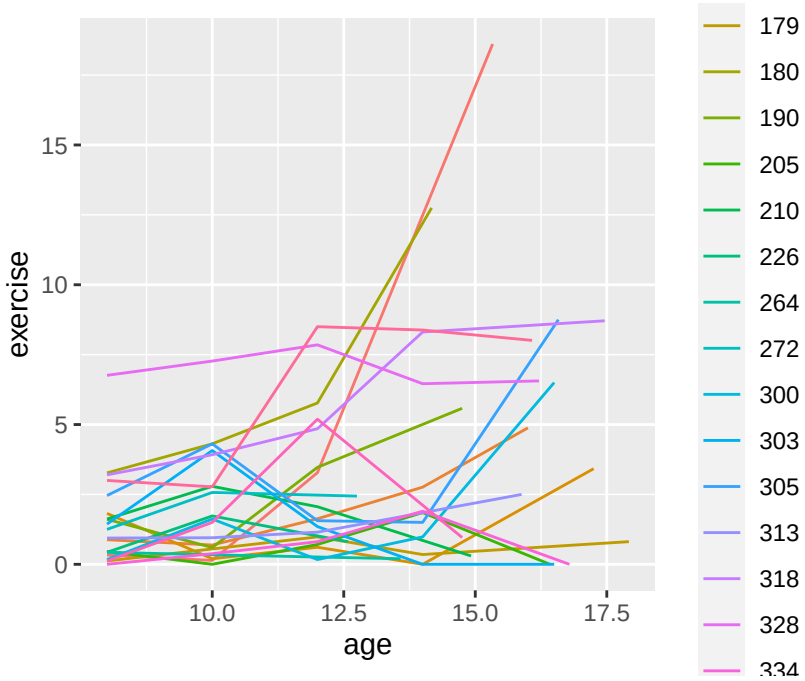
It is difficult to see any sort of pattern in the data because there are 231 individuals, so we can randomly select 20 individuals from the sample and re-create the plot.

```
set.seed(1234)
# randomly sampling 20 individuals
ids <- sample(unique(Blackmore$subject), 20)
# creating dataframe of the 20 individuals
samp <- Blackmore[Blackmore$subject %in% ids, ]
ggplot(data = samp, aes(x = age, y = log.exercise,
                        group = subject, color=as.factor(subject))) +
    geom_line()
```



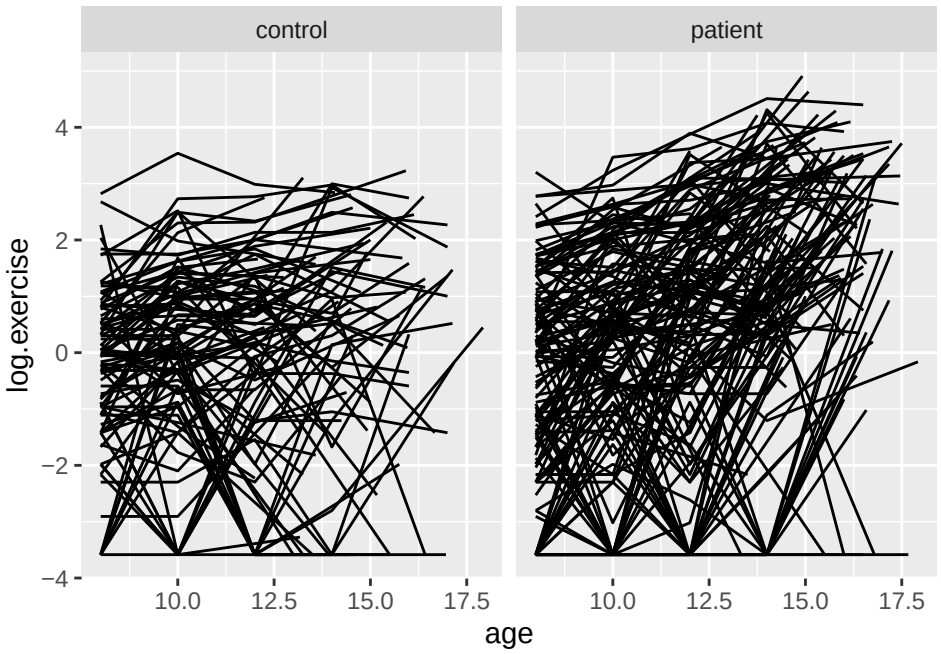
And on the original scale of min. per week:

```
ggplot(data = samp, aes(x = age, y = exercise,
                        group = subject, color=as.factor(subject))) +
    geom_line()
```



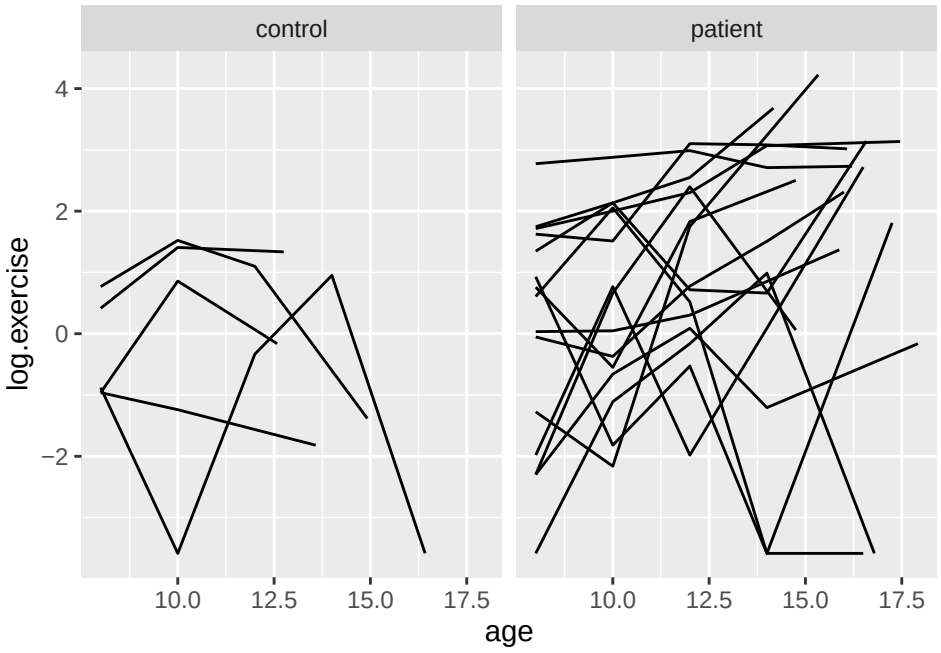
We can facet on whether or not the subject is a patient vs. control.

```
ggplot(data = Blackmore, aes(x = age, y = log.exercise, group = subject)) +
    geom_line() +
    facet_wrap(~ patient)
```



But again, we cannot really see any trends. Let's use the random sample of 20 subjects.

```
ggplot(data = samp, aes(x = age, y = log.exercise, group = subject)) +
    geom_line() +
    facet_wrap(~ patient)
```



Symbols used in specifying random effects for longitudinal data in `lme4`:

- `(1 | id)` - random intercept where `id` is the person identifier variable.
- `x + (x | id)` - random intercept and random slope for level 1 variable `x`. You can also write this as `x + (1 + x | id)`
- `x + (-1 + x | id)` - fixed intercept but random slope for level 1 variable `x`.

In general the code looks like this:

```
random.effect.model <- lmer(outcome ~ predictor1 + predictor2 +
(predictor1 | individual ID variable), data=yourdata)
```

Next, I create a time variable based on age. I subtracted 8 from the age variable. Recall that it was coded 8, 10, 12, 14, and 16. If we subtract 8, then age is coded as 0, 2, 4, 6, and 8. This means that intercept will be interpretable as the mean log exercise at the beginning of the study at age 8.

```
Blackmore$time <- Blackmore$age - 8
```

We can fit the model with a random intercept and a random slope for time and fixed effects for patient, time, and a patient by time interaction:

```
bm.lme.1 <- lmer(log.exercise ~ time*patient + (time|subject),
                 data=Blackmore)
summary(bm.lme.1)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log.exercise ~ time * patient + (time | subject)
##    Data: Blackmore
##
## REML criterion at convergence: 3614.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7349 -0.4245  0.1228  0.5280  2.6362
##
## Random effects:
##  Groups   Name        Variance Std.Dev. Corr
##  subject  (Intercept) 2.08385  1.4436
##           time        0.02716  0.1648   -0.28
##  Residual             1.54777  1.2441
## Number of obs: 945, groups:  subject, 231
##
## Fixed effects:
##                      Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)          -0.27602    0.18237 236.27190  -1.514   0.1315
## time                  0.06402    0.03136 237.53070   2.041   0.0423 *
## patientpatient       -0.35399    0.23529 233.73381  -1.504   0.1338
## time:patientpatient   0.23986    0.03941 221.21925   6.087 5.03e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) time   ptntpt
## time        -0.489
## patientptnt -0.775  0.379
## tm:ptntptnt  0.389 -0.796 -0.489
```

Thus, the mean log hours of exercise per week for the control group at the beginning of the study is -.276, and this is not statistically different from 0, $t(236.27) = -1.514, p = .132$ using the Satterwaithe approximation. The standard deviation of the log mean hours per week of exercise in the control group at baseline is 1.444.

For a 1 unit increase in time, the mean log hours per week of exercise increases by .064 in the control group and this effect is statistically significant, $t(237.53) = 2.041, p = .042$. Using the Kenward-Roger approximation

gives a similar conclusion although the p-value is much smaller, $F(1, 218.53) = 86.894, p < .001$. The standard deviation of this effect across individuals is 0.165. In addition, the random slopes are negatively correlated with the random intercepts, -.28.

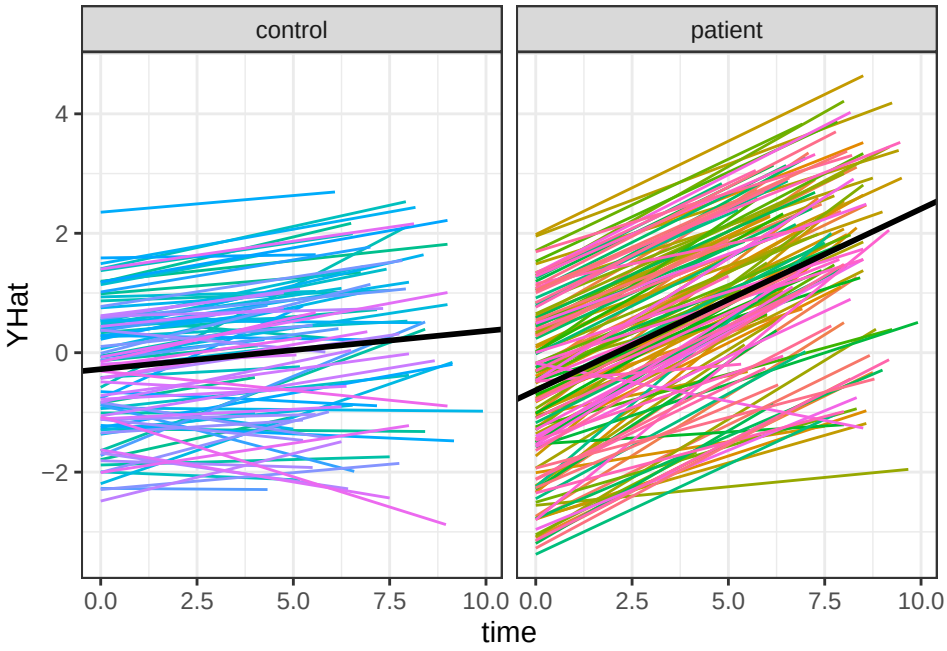
The difference in the log mean hours per week of exercise between the control and patient group is not statistically significantly different from 0, $t(233.73) = -1.504, p = .134$ using the Satterwaithe approximation. The Kenward-Roger approximation gives a similar result, $F(1, 230.11) = 2.263, p = .134$.

Finally, for a 1 unit increase in time, the mean log hours per week of exercise increases by 0.304 in the patient group, and this effect is statistically significant, $t(221.22) = 6.087, p < .001$ using the Satterwaithe approximation. The Kenward-Roger approximation gives a similar result, $F(1, 218.53) = 36.935, p < .001$.

```
anova(bm.lme.1, ddf="Kenward-Roger")
```

```
## Type III Analysis of Variance Table with Kenward-Roger's method
##                Sum Sq Mean Sq NumDF  DenDF F value    Pr(>F)
## time         134.492 134.492     1 218.53 86.8936 < 2.2e-16 ***
## patient        3.502   3.502     1 230.11  2.2629    0.1339
## time:patient  57.166  57.166     1 218.53 36.9345 5.375e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Let's examine whether the random slopes for age are necessary.

```
bm.lme.2 <- lmer(log.exercise ~ time*patient + (1 | subject),
                 data=Blackmore)
summary(bm.lme.2)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
```

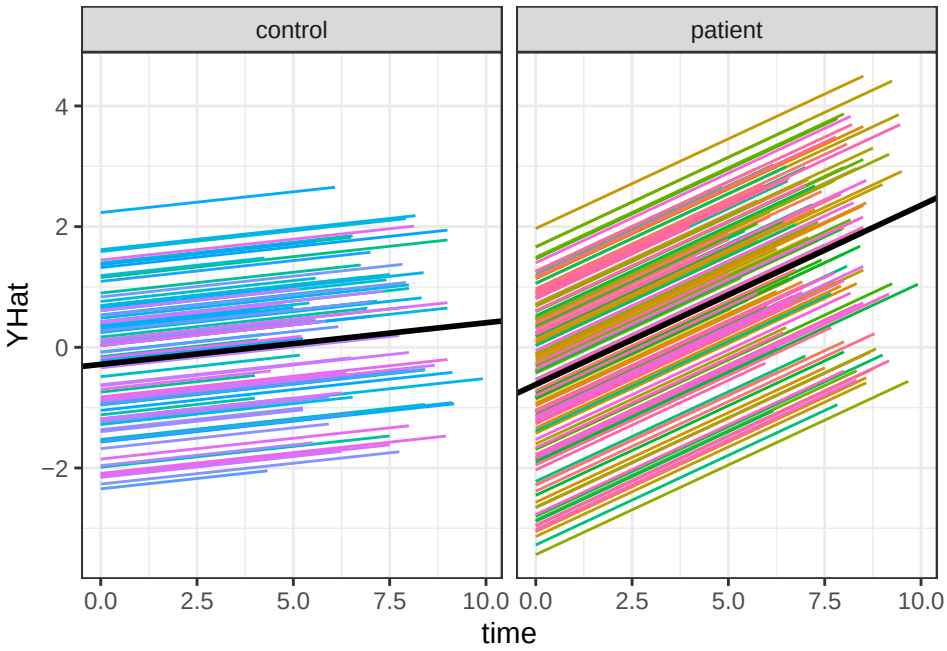
```
## Formula: log.exercise ~ time * patient + (1 | subject)
##    Data: Blackmore
##
## REML criterion at convergence: 3632.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2142 -0.4735  0.1269  0.5643  2.7214
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  subject  (Intercept) 1.875    1.369
##  Residual             1.807    1.344
## Number of obs: 945, groups:  subject, 231
##
## Fixed effects:
##                      Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)          -0.28322    0.18066 381.62865  -1.568   0.1178
## time                  0.06901    0.02717 738.45474   2.540   0.0113 *
## patientpatient       -0.33354    0.23306 377.86380  -1.431   0.1532
## time:patientpatient   0.22816    0.03387 735.37738   6.737 3.26e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) time   ptntpt
## time        -0.471
## patientptnt -0.775  0.365
## tm:ptntptnt  0.378 -0.802 -0.473
```

```
anova(bm.lme.1, bm.lme.2, refit=FALSE) # test for random slopes
```

```
## Data: Blackmore
## Models:
## bm.lme.2: log.exercise ~ time * patient + (1 | subject)
## bm.lme.1: log.exercise ~ time * patient + (time | subject)
##          npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## bm.lme.2    6 3644.3 3673.4 -1816.1   3632.3
## bm.lme.1    8 3630.1 3668.9 -1807.1   3614.1 18.122  2  0.0001161 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is significant variability ($\chi^2(2) = 18.122, p < .001$) in the slopes and therefore, we will retain the random slope model. Note that the difference in the degrees of freedom is because there is a random slope variance but also a random slope- random intercept correlation. Thus, the random intercept only model has two fewer parameters to estimate.

Let's examine what the predicted trajectories would look like without the random slope.



We could also fit a random slope only model and compare it to a random intercept and random slope model.

```
bm.lme.3 <- lmer(log.exercise ~ time*patient + (time - 1 | subject),
                 data=Blackmore)
summary(bm.lme.3)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: log.exercise ~ time * patient + (time - 1 | subject)
##    Data: Blackmore
##
## REML criterion at convergence: 3822.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.3685 -0.4842  0.1386  0.6036  2.3796
##
## Random effects:
##  Groups   Name Variance Std.Dev.
##  subject  time 0.05843  0.2417
##  Residual      2.60670  1.6145
## Number of obs: 945, groups:  subject, 231
##
## Fixed effects:
##                       Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)          -0.27775    0.13538 746.25591  -2.052   0.0405 *
## time                  0.06408    0.04265 493.49983   1.502   0.1336
## patientpatient       -0.36205    0.17362 742.47961  -2.085   0.0374 *
## time:patientpatient   0.24146    0.05366 470.75332   4.500 8.59e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) time   ptntpt
```

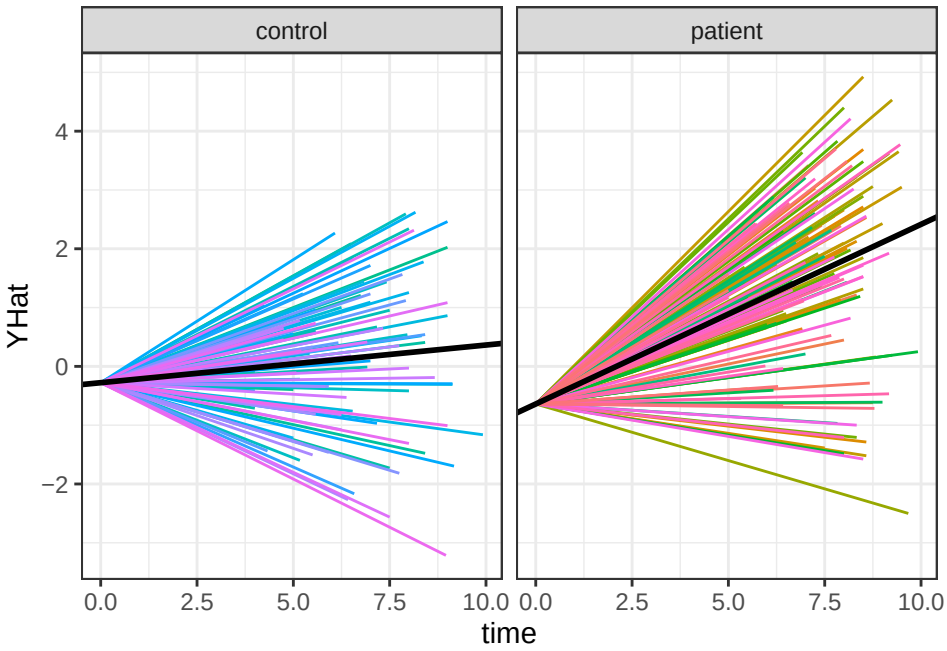
```
## time        -0.614
## patientptnt -0.780  0.479
## tm:ptntptnt  0.488 -0.795 -0.609
```

```
anova(bm.lme.1, bm.lme.3, refit=FALSE)  # test for random intercepts
```

```
## Data: Blackmore
## Models:
## bm.lme.3: log.exercise ~ time * patient + (time - 1 | subject)
## bm.lme.1: log.exercise ~ time * patient + (time | subject)
##          npar    AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## bm.lme.3    6 3834.1 3863.2 -1911.0   3822.1
## bm.lme.1    8 3630.1 3668.9 -1807.1   3614.1 207.95  2  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is significant variability in the intercepts, $\chi^2(2) = 207.6, p < .001$ so we will retain the random intercept - random slope model.

Again, we will plot the model predicted trajectories.



Note that profile confidence intervals can be obtained using the `confint` function but they take a while to run.

```
CIs <- confint(bm.lme.1)
```

```
## Computing profile confidence intervals ...
```

```
CIs
```

```
##                            2.5 %      97.5 %
## .sig01                1.246360558 1.640285253
## .sig02               -0.484511670 0.007556466
## .sig03                0.111925140 0.209333347
## .sigma                1.170951594 1.324668138
## (Intercept)          -0.633450042 0.081264029
## time                  0.002485022 0.125434237
```

```
## patientpatient      -0.814766779 0.107492854
## time:patientpatient  0.162603338 0.317345929
```

The random effect estimates can be obtained using the `VarCorr()` function. The log-likelihood can be obtained using the `logLik()` function.

```
VarCorr(bm.lme.1) #gives the random effects by themselves
```

```
##  Groups   Name        Std.Dev. Corr
##  subject  (Intercept) 1.4436
##           time        0.1648   -0.281
##  Residual             1.2441
```

```
logLik(bm.lme.1)  #gives the log-likelihood for the model
```

```
## 'log Lik.' -1807.068 (df=8)
```

```
AIC(bm.lme.1)     #gives the AIC for the model
```

```
## [1] 3630.136
```