Hierarchical designs

author: Maria Paniw date: 11.02.2016 width: 1620 height: 1080

The independence assumptions of lm and glm

The analyses we did so far using `lm` or `glm` functions had two critical assumptions about the way we got the data to be analyzed:

- independence of sampling units: this assumption is violated when we have time-series data
- random allocation of treatments or sampling in space or time: this may be violated when we simply do not have the money or time to randomly sample each replicate of our treatment combinations.

Particularly the latter point is a reality in many field studies where you have to allocate resources efficiently and control for unwanted environmental factors. One way to deal with this is to create hierarchical designs.

Random effects

Before going into specific hierarchical experimental designs, let's look at how modern statistical packages analyze them. Typically, this is done via **mixed effect models** in which factors at the highest levels of hierarchy are treated as radom effects, whereas the treatments you are actually interested in are treated as fixed effects.

Fixed vs. Random Effects

So far, we included the categorical variables into our models as **fixed** effects. That means that we assume the following:

> The different treatment levels that we use are the only ones of interest to us and are the ones we actually manipulated. The inference we make is therefore restricted to these levels.

The inference is different for random effects. Here, we assume:

> The treatment levels are just a **random sample** taken from a ditribution of all possible levels. This distribution is typically assumed to be normal with the mean and variance given by the mean and variance of your data (random effects).

In a mathematical formulation, the only difference between a fixed and a random effect β_{ji} for treatment level j and observation i is:

Fixed: $Y_i = \alpha + \beta_{ji} + \delta * X_i + \epsilon_i$

Random: $Y_i = \alpha + \beta_{ji} + \delta * X_i + \epsilon_i$; $\beta_j \sim Normal(0, \sigma_\beta)$

In other words, in a random effects model, β is drawn from a normal distribution with a mean of 0 and standard deviation σ

Example:

When we practiced ANCOVA, we simulated the following data:

```r
n.groups <- 3
n.sample <- 10
n <- n.groups * n.sample    # Total number of data points
x <- rep(1:n.groups, rep(n.sample, n.groups)) # Indicator for population
pop <- factor(x, labels = c("Pyrenees", "Massif Central", "Jura"))
length <- runif(n, 45, 70)      # Obs. body length (cm) is rarely less than 45
```

```r
Xmat <- model.matrix(~ pop*length)
print(Xmat, dig = 2)
```

```
   (Intercept) popMassif Central popJura length popMassif Central:length
1            1                 0       0     56                        0
2            1                 0       0     54                        0
3            1                 0       0     68                        0
4            1                 0       0     63                        0
5            1                 0       0     65                        0
6            1                 0       0     53                        0
7            1                 0       0     62                        0
8            1                 0       0     60                        0
9            1                 0       0     59                        0
10           1                 0       0     45                        0
11           1                 1       0     60                       60
12           1                 1       0     51                       51
13           1                 1       0     58                       58
14           1                 1       0     62                       62
15           1                 1       0     65                       65
16           1                 1       0     47                       47
17           1                 1       0     52                       52
18           1                 1       0     70                       70
19           1                 1       0     54                       54
20           1                 1       0     63                       63
21           1                 0       1     57                        0
22           1                 0       1     53                        0
23           1                 0       1     53                        0
24           1                 0       1     56                        0
25           1                 0       1     50                        0
26           1                 0       1     58                        0
27           1                 0       1     57                        0
28           1                 0       1     69                        0
29           1                 0       1     63                        0
30           1                 0       1     70                        0
   popJura:length
1               0
2               0
3               0
4               0
5               0
6               0
7               0
```

```
8               0
9               0
10              0
11              0
12              0
13              0
14              0
15              0
16              0
17              0
18              0
19              0
20              0
21             57
22             53
23             53
24             56
25             50
26             58
27             57
28             69
29             63
30             70
attr(,"assign")
[1] 0 1 1 2 3 3
attr(,"contrasts")
attr(,"contrasts")$pop
[1] "contr.treatment"
```

```
beta.vec <- c(-250, 150, 200, 6, -3, -4)
```

```
lin.pred <- Xmat[,] %*% beta.vec    # Value of lin.predictor
eps <- rnorm(n = n, mean = 0, sd = 10)  # residuals
mass <- lin.pred + eps          # response = lin.pred + residual
```

Fixed effect model:

```
#We can fit a fixed-effect model to the data:

mod.fx=lm(mass ~ pop + length)
```

```
# Or a random effect model using lmer in the package lme4

library(lme4)

mod.rf=lmer(mass~length+(1|pop))
```

```
summary(mod.rf)
```

```
Linear mixed model fit by REML ['lmerMod']
```

```
Formula: mass ~ length + (1 | pop)

REML criterion at convergence: 244.4

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.71027 -0.63523 -0.08594  0.55956  2.11655

Random effects:
 Groups   Name        Variance Std.Dev.
 pop      (Intercept) 322.2    17.95
 Residual             203.2    14.26
Number of obs: 30, groups:  pop, 3

Fixed effects:
             Estimate Std. Error t value
(Intercept) -117.7289    25.9744  -4.533
length         3.3911     0.4051   8.372

Correlation of Fixed Effects:
       (Intr)
length -0.911
```

In order to see how much residual variance is accounted for by pop:

```r
# exract random variance components of the model
r.var=as.data.frame(VarCorr(mod.rf))
# define proportion of variance explained by parameter uncertainty:
r.var$vcov[1]/sum(r.var$vcov)
```

```
[1] 0.6132391
```

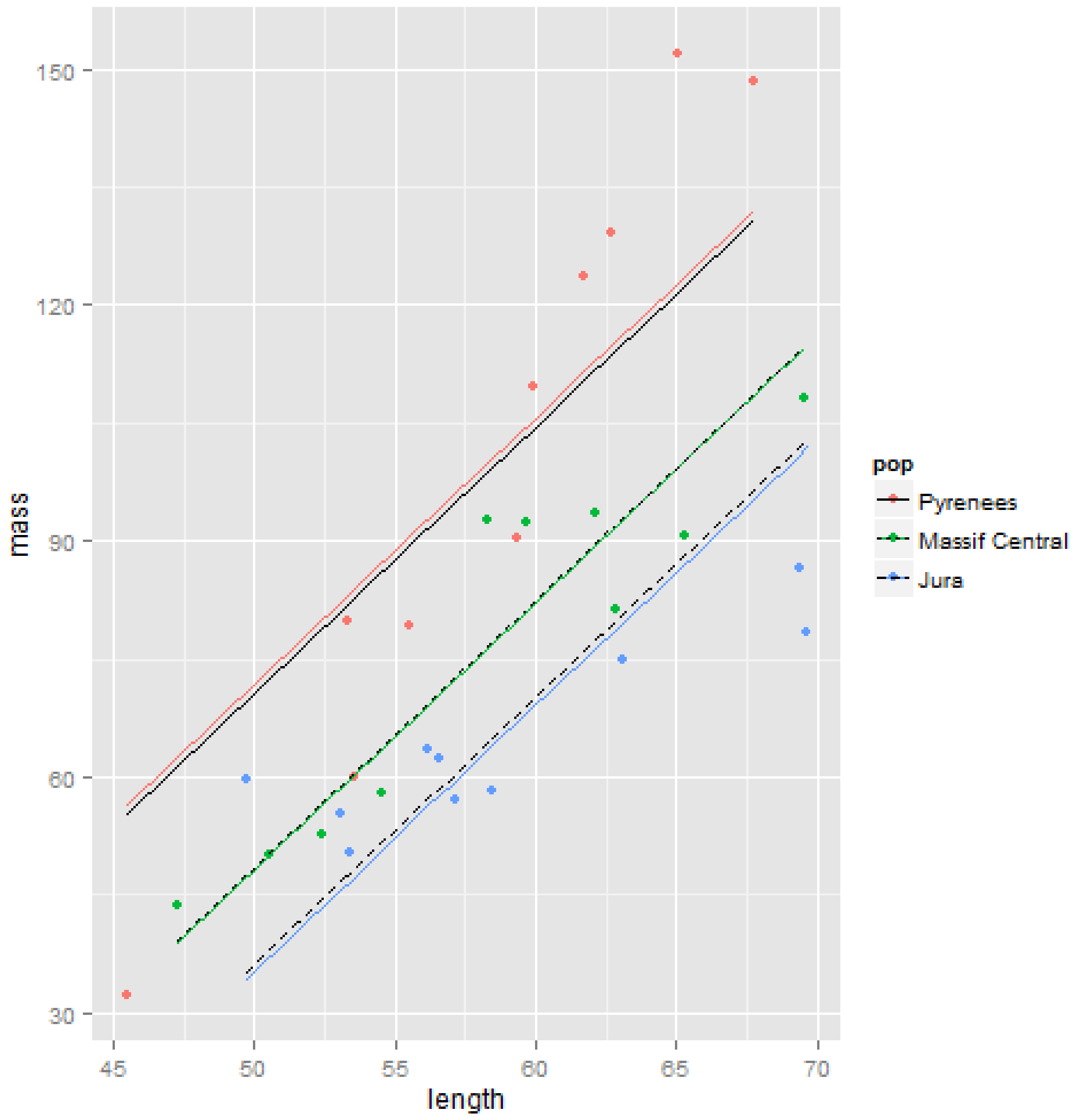
Visualize the difference between random and fixed effect models

```r
# data frame with predictions:
new.data=data.frame(mass=mass,length=length,pop=pop,
                    pred.fx=predict(mod.fx),
                    pred.rf=predict(mod.rf))
```

Plot

```r
library(ggplot2)

ggplot(data=new.data,aes(x=length,y=mass))+geom_point(aes(col=pop))+
  geom_line(aes(x=length,y=pred.fx,col=pop))+
  geom_line(aes(x=length,y=pred.rf,linetype=pop))
```

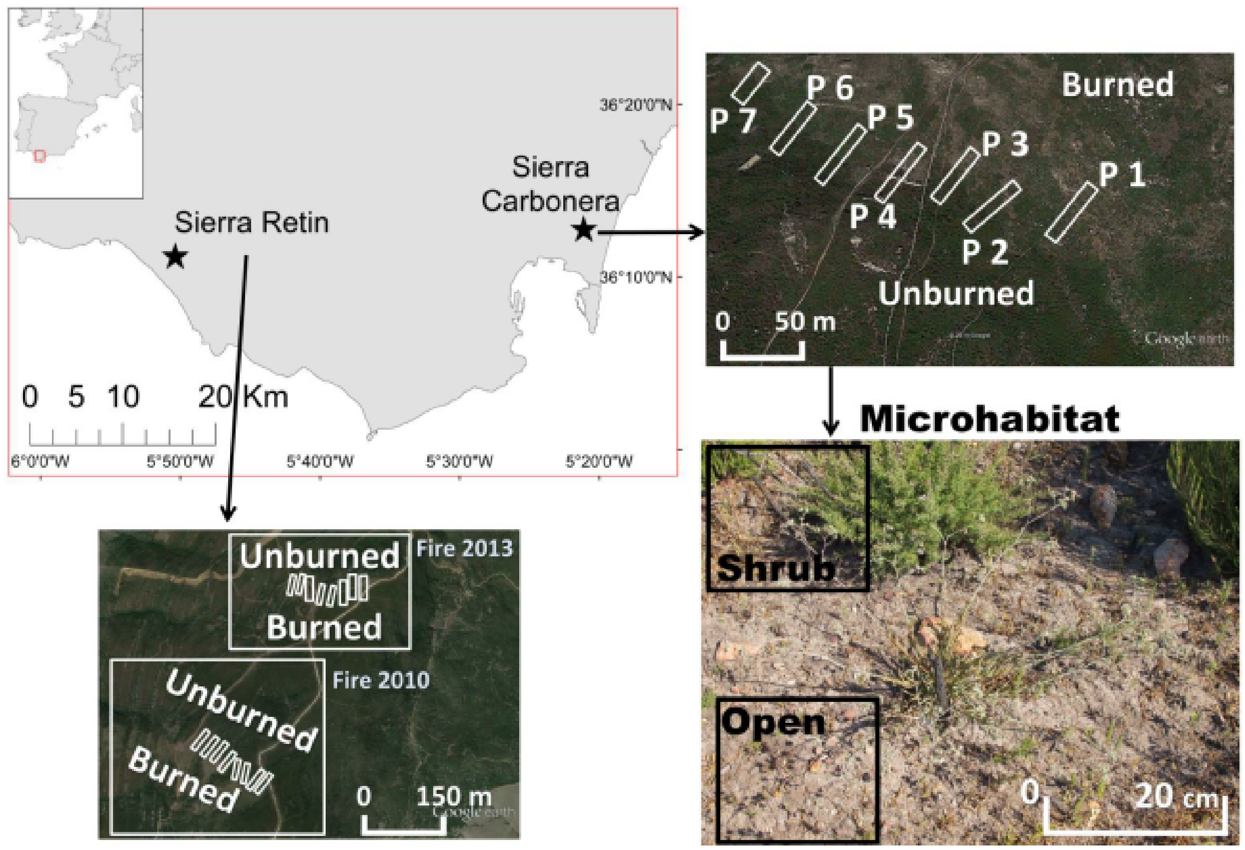


Random effects can be formulated in various ways, as we will see.

WIKI on GLMER

Back to designs: randomized blocks

You can randomly organize your treatment levels into blocks to control for environmental heterogeneity.



In block designs, the effect of blocks is typically not of interest to us, so we treat it as a random factor in lmer. Let's look at an exercise.

Load the data `drosoSB.txt`, where you have data of a seed-bank survival experiment. The response variable, `surv` depicts survival of seeds in the seed bank out of a total of 20 that were sown there. Surival is hypothesized to differ between burned and unburned habitat patches. So, what you have is a logistic regression. However, the treatment levels *burned* and *unburned* were randomly arranged into 7 blocks.

Fit a `glmer` model to the data, using block as a random effect on the mean.

Hierarchical designs: split plot

The simple form of split-plot dessigns applies one treatment to the entire plot while, the other treatment is replicated within and between plots.

For example, a classical experiment was conducted in Iowa in 1944 to see how different varieties of alfalfa responded to the **last cutting day** of the previous year (Snedecor and Cochran 1967). We know that in the fall alfalfa can either continue to grow, or stop growing and store resources belowground in roots for growth during the following year. Thus, we might expect that later cutting dates inhibits growth for the following year. On the other hand, if plants are cut after they have gone into senescence, there should be little effect on productivity during the following year. There are **two factors**: 1) **variety of alfalfa** (three varieties were planted in each of three randomly chosen **whole plots**), and 2) the **date of last cutting** (A=none, B=Sept 1, C=Sept. 20, or D=Oct. 7). The dates were randomly chosen split plots within the whole plots. Replication was accomplished using six blocks of fields.

Here what the experiment looks like (Source):

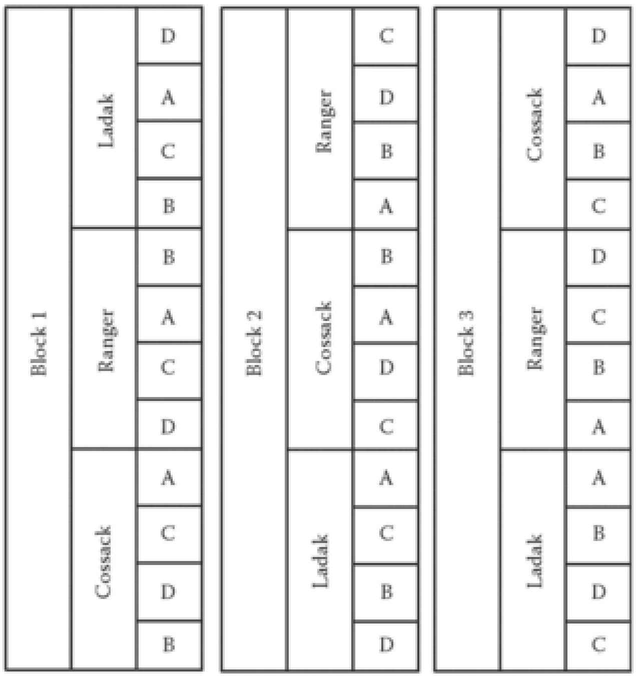


FIGURE 10.11
First three blocks of the alfalfa split-plot experiment showing the random assignment of whole- and split-plot treatments. (Adapted from Snedecor, G. W., and Cochran, W. G. 1989. *Statistical Methods*, 8th ed. Iowa State College Press, Ames, IA.)

The data can be found in the package `asbio` - so install the package!

```
#intall if needed
library(asbio)
data(alfalfa.split.plot)

data=alfalfa.split.plot

head(data)
```

```
  yield variety cut.time block
1  2.17       L     None     1
2  1.58       L       S1     1
3  2.29       L      S20     1
4  2.23       L      O7      1
5  2.33       C     None     1
6  1.38       C       S1     1
```

==== There are several ways we can analyze the data. First, having a classical ANOVA design, we can use the aov function ignoring the hierarchy in the data.

```
summary(aov(yield~variety*cut.time,data))
```

```
                 Df Sum Sq Mean Sq F value  Pr(>F)
variety           2  0.178  0.0890   0.789 0.45905
```

```
cut.time          3  1.962  0.6542   5.797 0.00151 **
variety:cut.time  6  0.211  0.0351   0.311 0.92884
Residuals        60  6.771  0.1128
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

But your residuals are wrong!

Look at the correct result

```
summary(aov(yield~variety*cut.time +Error(block/variety),data))
```

```
Error: block
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  5   4.15    0.83

Error: block:variety
          Df Sum Sq Mean Sq F value Pr(>F)
variety    2  0.178 0.08901   0.653  0.541
Residuals 10  1.362 0.13623

Error: Within
                 Df Sum Sq Mean Sq F value   Pr(>F)
cut.time          3 1.9625  0.6542  23.390 2.83e-09 ***
variety:cut.time  6 0.2106  0.0351   1.255    0.297
Residuals        45 1.2585  0.0280

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

====== Because the `aov` formulation is designrd for classical ANOVA only and does not deal well with non-normal responses or unbalanced designs, we should use mixed-effect models!

1. Formulate different random effects

```
# This model assumes that the grand mean of yield varies between blocks
mod1=lmer(yield~variety+cut.time+(1|block),data)

#This model assumes that yield is not constant for each level
#of variety among different blocks, but instead varies randomly
#between blocks
mod2=lmer(yield~variety+cut.time+(variety|block),data)
```

```
anova(mod1,mod2)# The second model has a higher likelihood
```

```
Data: data
Models:
mod1: yield ~ variety + cut.time + (1 | block)
```

```
mod2: yield ~ variety + cut.time + (variety | block)
     Df     AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
mod1  8 10.2899 28.503  2.8551  -5.7101
mod2 13  4.0078 33.604 10.9961 -21.9922 16.282      5   0.006083 **

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1. Formulate different fixed effects

```
mod2a=lmer(yield~1+(variety|block),data)
mod2b=lmer(yield~variety+(variety|block),data)
mod2c=lmer(yield~variety+cut.time+(variety|block),data)
mod2d=lmer(yield~variety*cut.time+(variety|block),data)
```

```
anova(mod2a,mod2b,mod2c,mod2d)# additive effect is best
```

```
Data: data
Models:
mod2a: yield ~ 1 + (variety | block)
mod2b: yield ~ variety + (variety | block)
mod2c: yield ~ variety + cut.time + (variety | block)
mod2d: yield ~ variety * cut.time + (variety | block)
      Df    AIC    BIC  logLik deviance   Chisq Chi Df Pr(>Chisq)
mod2a  8 41.184 59.397 -12.592   25.184
mod2b 10 44.196 66.963 -12.098   24.197  0.9873      2     0.6104
mod2c 13  4.008 33.604  10.996  -21.992 46.1887      3  5.171e-10 ***
mod2d 19  7.654 50.911  15.173  -30.346  8.3536      6     0.2133

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Final model

```
anova(mod2c)# additive effect is best
```

```
Analysis of Variance Table
         Df Sum Sq Mean Sq F value
variety   2 0.0260 0.01300  0.4512
cut.time  3 1.9625 0.65416 22.7091
```

How to test significance of fixed effect

This question not trivial because it is difficult to estimate the degrees of freedom in you model. But `lm` objects use the t distribution to test significance of coefficients. For `lmer`, there are several opitons: see blog

```r
require(lmerTest)
# extract coefficients from original model
coefs <- data.frame(coef(summary(mod2c)))
# re-fit model
m.semTest <- lmer(yield~variety+cut.time+(variety|block),data,REML=F)
```

```r
# get Satterthwaite-approximated degrees of freedom
coefs$df.Satt <- coef(summary(m.semTest))[, 3]
# get approximate p-values
coefs$p.Satt <- coef(summary(m.semTest))[, 5]
coefs
```

```
                Estimate Std..Error    t.value   df.Satt       p.Satt
(Intercept)   1.75597222 0.13691125 12.8256240  6.961828 2.368651e-06
varietyL      0.09458333 0.10500876  0.9007185  6.000002 3.618984e-01
varietyR     -0.01916667 0.08139324 -0.2354823  6.000003 8.050593e-01
cut.timeO7   -0.09000000 0.05657440 -1.5908255 54.000046 1.074572e-01
cut.timeS1   -0.44055556 0.05657440 -7.7871892 54.000046 9.401213e-11
cut.timeS20  -0.20666667 0.05657440 -3.6530068 54.000046 4.207895e-04
```

Plots!

Plotting the results of mixed models may be a bit tricky with `predict` because you typically only want to represent the predictions of fixed effects, but `predict` includes random effect predictions as well.

Plotting is best accomplished like this:

```r
#Predictions (create new data)

new.data=expand.grid(variety=levels(data$variety),
                     cut.time=levels(data$cut.time),
                     yield=0)


mm <- model.matrix(terms(mod2c),new.data) # create a model matrix

#multiply it by fixed effect parameters -> this is you prediction
new.data$yield <- mm %*% fixef(mod2c)
```

Your turn!

Load the data `DrosoSurvSite.txt`. These data consist of measurements of plant size and survival in plots that differed based on time since fire (TSF). This design was replicated at 6 sites (but not perfectly). I would like you to fit a **generalized mixed model** where `surv~size +/* TSF` and **site** is a random factor.