R-StatProgamming with functions in R

Timothée Bonnet, BDSI

April 2, 2020

On Friday 03/04/2020 I will be presenting this tutorial live on Zoom. (ask by email/Slack if you do not know how to use Zoom.)

If you have any trouble going through this tutorial then or at a different time you can chat about it on Slack (rsb-r-stats-biology.slack.com , if you are not a member but would like to be, drop me an email) or email me at timotheebonnetc@gmail.com.

If you do not attend the Zoom meeting but would like to receive credit through the COS Career Development Framework program I need you to complete three exercises of your choice. Send me your answers via Slack or email. It does not have to be correct on the first try and you are welcome to get in touch if you are completely stuck. I will provide feedback to help you complete exercises you want to do.

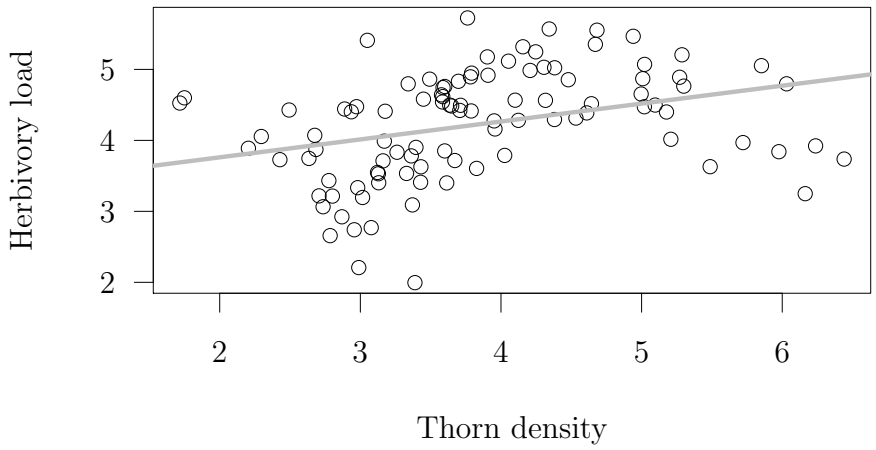
In this tutorial you will learn:

- How unexplained variation in a response variable can compromise statistical inference when that variation is structured.
- How fixed and random effects can correct for structured variation in the response variable.
- How to fit mixed-effects models in R.
- How to choose between fixed and random effects.
- Extract and interpret mixed models output.
- Test for the statistical significance of a random effect.

Contents

1 The dangers of unexplained variation

Imagine you have measured how much a plant species is attacked by big herbivores as a function of how many thorns the plant grows on stems. You have collected data in 5 locations and visualise the relationship between herbivory and quantity of thorns (in arbitrary units):



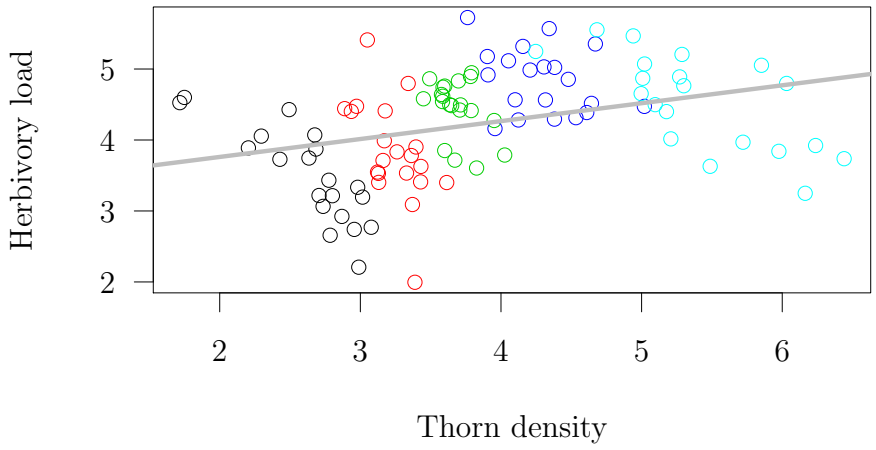
It looks like the more a plant has thorns, the more it is eaten. Let's check that by fitting a linear regression.

```
thorns <- read.csv(file = "Data/thorndata.csv", header=TRUE)
str(thorns)

## 'data.frame': 100 obs. of  3 variables:
##  $ herbivory   : num  4.53 4.6 3.89 4.05 3.73 ...
##  $ thorndensity: num  1.72 1.75 2.2 2.3 2.43 ...
##  $ site        : Factor w/ 5 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...

model_0 <- lm(herbivory ~ thorndensity, data = thorns)
summary(model_0)

##
## Call:
## lm(formula = herbivory ~ thorndensity, data = thorns)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -2.1162 -0.5183  0.0821  0.5435  1.5193
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.2564     0.2854   11.41  < 2e-16 ***
## thorndensity   0.2524     0.0717    3.52  0.00066 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.725 on 98 degrees of freedom
## Multiple R-squared:  0.112,Adjusted R-squared:  0.103
## F-statistic: 12.4 on 1 and 98 DF,  p-value: 0.000657
```

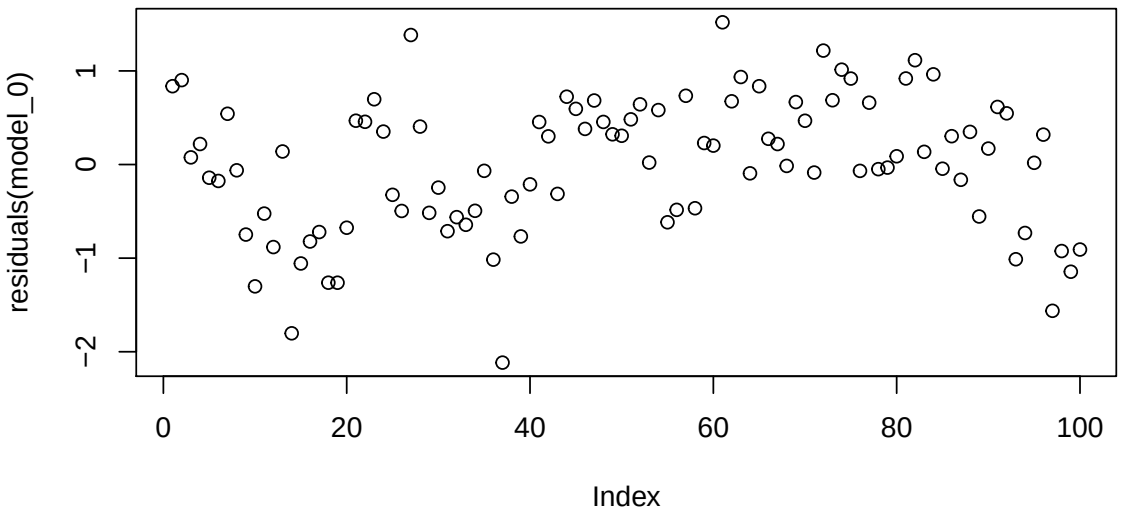
This model confirms a positive relationship between how much a plant has thorns and how much it is consumed by herbivores; which is counter-intuitive.

The twist is that data were collected in 5 different sites, that our two variables have different mean values in different sites, and that our model is not aware of that. We can see the structure in the data by coloring the points by site:



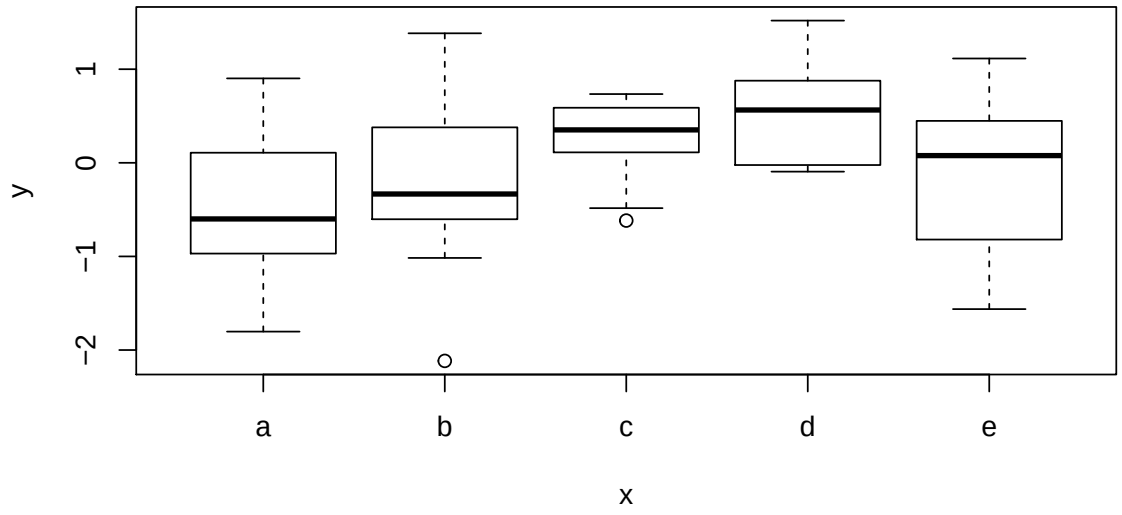
Maybe you remember that in a linear model (such as a linear regression) we assume that the residuals follow a normal distribution without any structure (homoscedasticity, independence...). It is not easy to see patterns by just ploting the residuals:

```
plot(residuals(model_0))
```



However, in our case, we have data about the missing variable that structure residuals, so we can see clearly that residuals are distributed with structure:

```
plot(thorns$site, residuals(model_0))
```

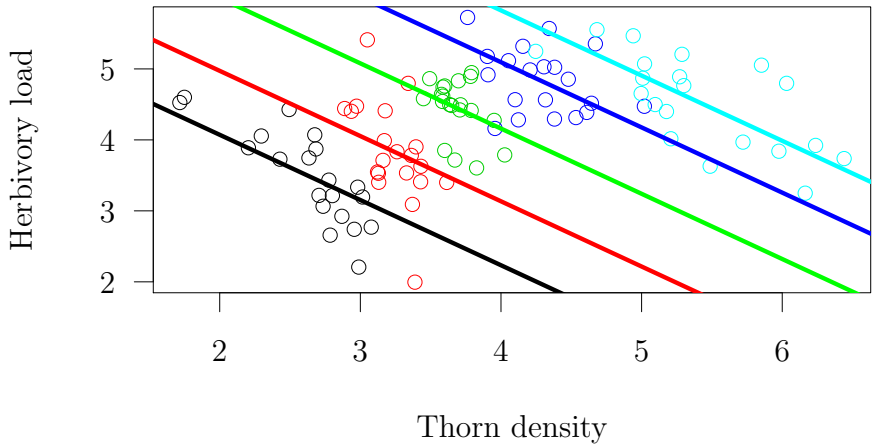


We can correct the problem by fititng site in the mode:

```r
model_1 <- lm(herbivory ~ thorndensity + site, data = thorns)
summary(model_1)
```

```
##
## Call:
## lm(formula = herbivory ~ thorndensity + site, data = thorns)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6877 -0.2889  0.0498  0.2492  1.3968
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)     6.047      0.387   15.62  < 2e-16 ***
## thorndensity   -0.975      0.141   -6.90  6.0e-10 ***
## siteb           0.940      0.176    5.33  6.6e-07 ***
## sitec           1.996      0.215    9.30  5.8e-15 ***
## sited           2.971      0.281   10.56  < 2e-16 ***
## sitee           3.767      0.422    8.93  3.5e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.488 on 94 degrees of freedom
## Multiple R-squared:  0.615,Adjusted R-squared:  0.595
## F-statistic:   30 on 5 and 94 DF,  p-value: <2e-16
```

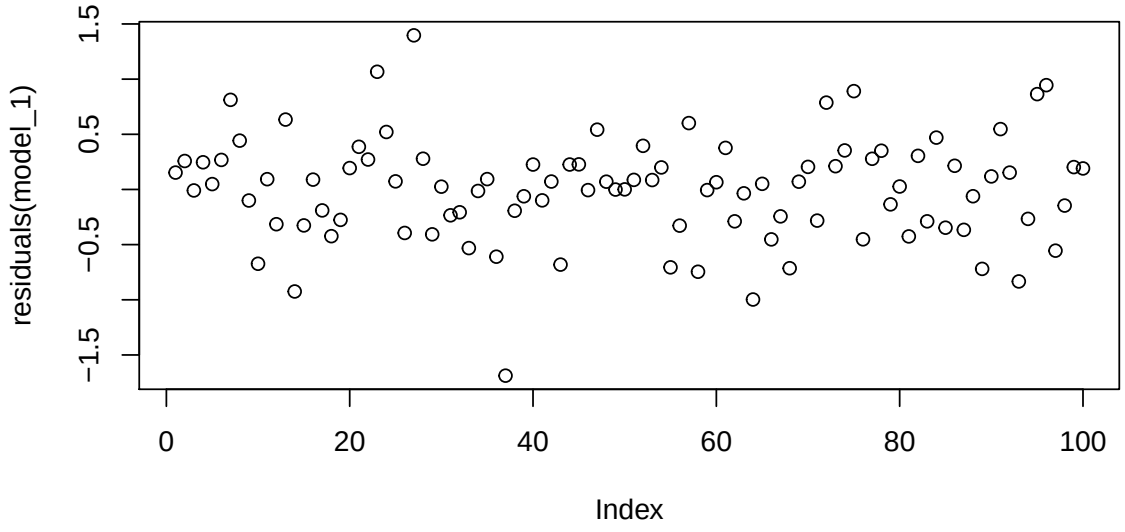
Now we see a negative relationship between thorn density and herbivory. That model corresponds to the plot below:



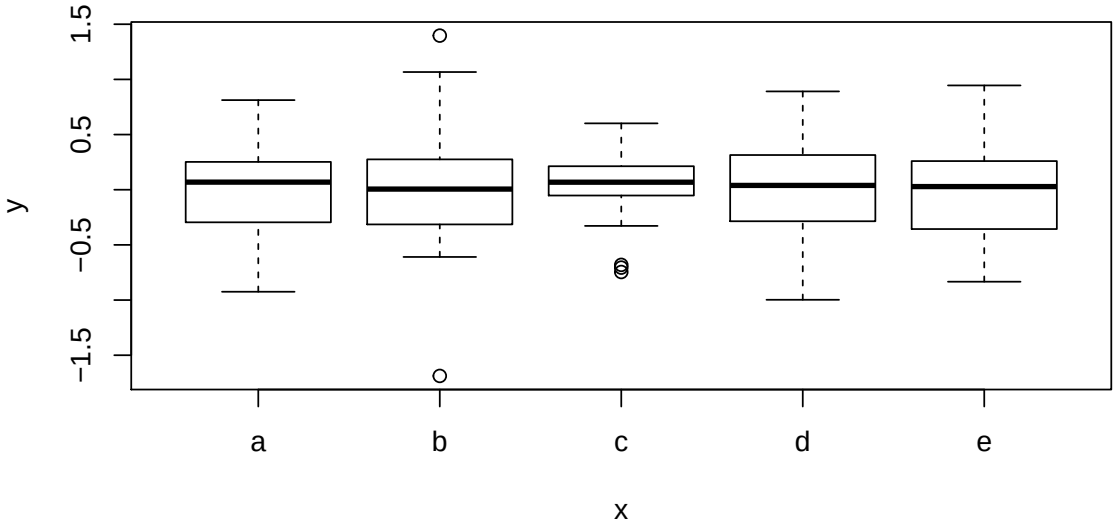
We have 5 parallel regression lines, one for each site. Differences in the mean values within each sites are accounted for but we estimate the common slope across all sites.

The residuals of that model are more clearly without pattern:

```
plot(residuals(model_1))
```



```
plot(thorns$site, residuals(model_1))
```



Note that in the summary of model_1 we have 4 parameter estimates we did not really care about: the deviations of four site relative to the fifth site. It does not hurt to have them there, but we are not really interested in them because 1) we did not have questions about how sites vary in herbivory, 2) site is just a grouping factor and differences between sites are probably due to numerous processes we have not measured. Given our ignorance on what makes sites different, the values of site effects are essentially random; we fit site only to get a correct inference but do not learn much from the estimated effects of sites.

2 Fitting random effects

Now imagine a larger research setting where you would have collected data in 50 sites for a response variable and a predictor.

Check what happens if we approach the problem as in the thorn example:

```
biggerdata <- read.csv("Data/biggerdata.csv")
str(biggerdata)

## 'data.frame': 10000 obs. of  3 variables:
##  $ predictor: num  1.72 1.88 2.78 2.04 2.06 ...
##  $ response : num  3.252 1.476 0.143 2.248 0.489 ...
##  $ site     : Factor w/ 50 levels "s1","s10","s11",..: 25 3 17 46 47 14 4 47 10 27

bdmodel <- lm(response ~ predictor + site, data = biggerdata)
```

```
summary(bdmodel)
```

You will find it is a bit of a mess. We have 49 parameter estimates for sites, but we do not really care about those effects, they are essentially random to us. It would be more efficient to fit site as a random effect!

To do so, load (and install if needed) the package `lme4`.

```
library(lme4)

## Loading required package:  Matrix
```

Then you can copy paste the model we called bdmodel, and change two things: 1) change `lm` to `lmer`; 2) change `site` to `(1|site)`:

```
bdmmixedmodel <- lmer(response ~ predictor + (1|site), data = biggerdata)
```

That's it, you have fitted a mixed effect model: that is, a model with both fixed and random effect. The syntax (1—something) means that the variable "something" is fitted as a random effect, while the variable "predictor" remains a fixed effect. Check the summary of that model:

```
summary(bdmmixedmodel)

## Linear mixed model fit by REML ['lmerMod']
## Formula: response ~ predictor + (1 | site)
##    Data: biggerdata
##
## REML criterion at convergence: 14852
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -3.712 -0.679 -0.011  0.667  3.854
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  site     (Intercept) 5.930    2.435
##  Residual             0.248    0.498
## Number of obs: 10000, groups:  site, 50
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.96051    0.34499    5.68
## predictor    0.17579    0.00999   17.60
```

```
##
## Correlation of Fixed Effects:
##           (Intr)
## predictor -0.058
```

Now we see a single fixed effect (apart from the intercetp), that of the variable "predictor". The 49 site effects are not there anymore. Where did they go? In the Random effects section, labeled as "site (Intercept)", and instead of seeing 49 effects, we have 50 effects summarized in a single number, namely the variance in those effects: 5.9297. There is a second number, Std.Dev. which in this case is 2.4351, but that is just the standard deviation, that is, the square-root of the variance.

So Variance and Std.Dev. carry exactly the same information: the amount of difference among sites. Below site, there is a row for Residual, again giving both a Variance and a Std.Dev. This quantifies the variance left unexplained within each site, after removing the variance explained by fixed effects and random effects.

Time for some practice on your own:

* Exercise 1 — A mixed model for the thorn data set

1. Load the dataset "thorndata.csv" to fit a linear model of "herbivory" as a function of "thorndensity". What is the estimate for the slope?

2. Add a random effect for site. How does the estimate for the slope changes?

3. How much variation is explained by differences among sites? How much variation remains unexplained within sites?

Remember how we went from 49 fixed effect parameters to 1 random effect parameter in the previous example? Doesn't it sound a bit weird? How can we summarize 49 (or 50) numbers in 1. Well, by making one assumption: that these 50 numbers follow a normal distribution, which we define such that its mean is zero. A normal distribution is defined by 2 parameters, the mean and the variance, we fix the mean to zero, so we can describe the distribution with the variance only. Let's visualize that distribution: The model was:

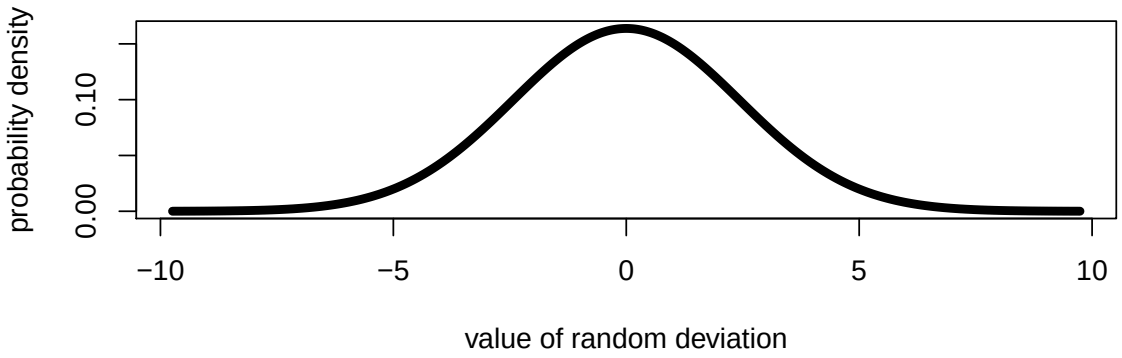
```
mm0 <- lmer(response ~ predictor + (1|site), data=biggerdata)
```

We can extract the random variance for site as:

```
rvar <- as.numeric(VarCorr(mm0)$site)
```

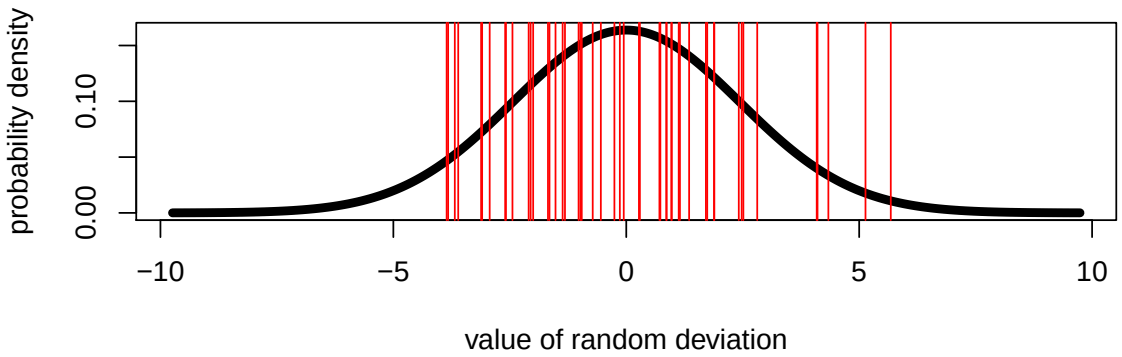
Knowing that most of a normal distribution is contained within +/- 4 times its standard deviation, we can visualize the density distribution of the random effect:

```
x <- seq(-4*sqrt(rvar),4*sqrt(rvar),sqrt(rvar)/100)
plot(x=x, y=dnorm(x = x, mean =0 , sd = sqrt(rvar)), type="l",
     xlab= "value of random deviation", ylab="probability density", lwd=5)
```



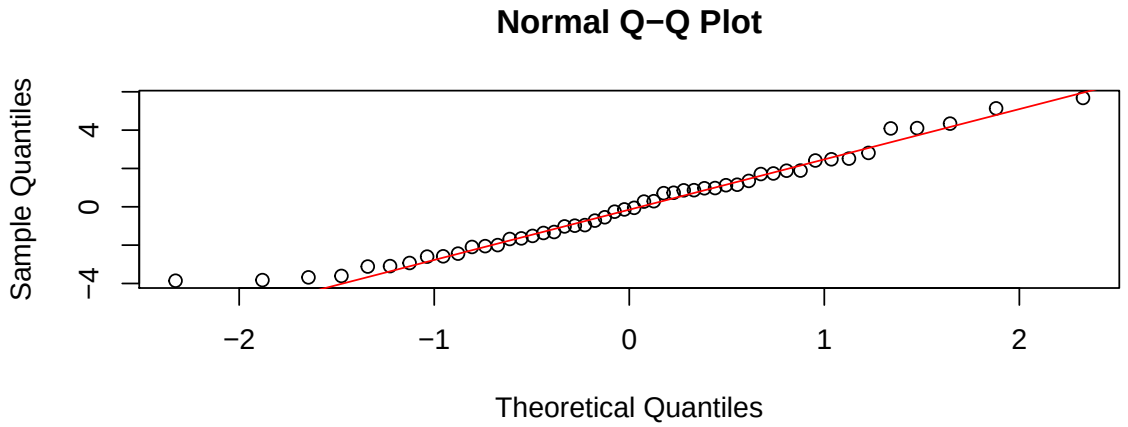
The actual random deviation for each site are "kind of" estimated and can be extracted with the function `ranef()`. In our case `ranef()` will return 50 values. Let's map them onto the previous plot:

```
x <- seq(-4*sqrt(rvar),4*sqrt(rvar),sqrt(rvar)/100)
plot(x=x, y=dnorm(x = x, mean =0 , sd = sqrt(rvar)), type="l",
     xlab= "value of random deviation", ylab="probability density", lwd=5)
abline(v=unlist(ranef(mm0)), col= "red")
```



OK, it makes some sense, but also does not look perfect. Let's see how well the distribution of random levels fits a normal distribution using a qqplot:

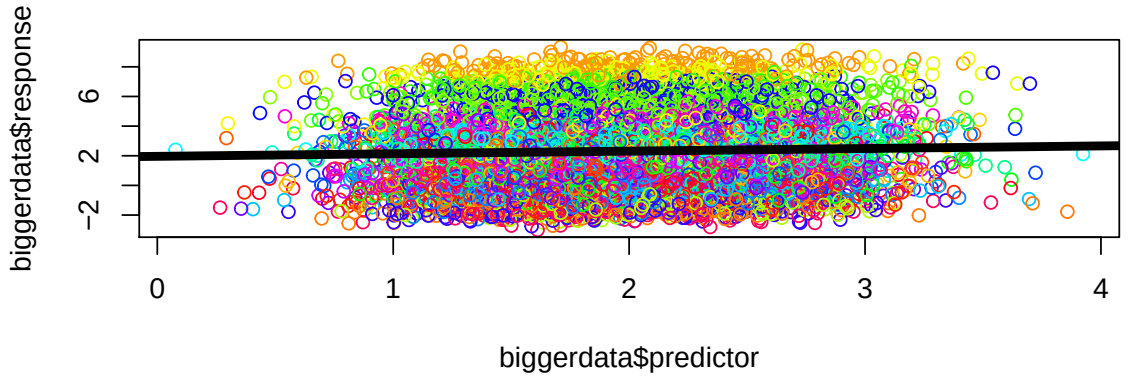
```
qqnorm(unlist(ranef(mm0)))
qqline(unlist(ranef(mm0)), col = 2)
```



Not too bad, but with only 50 samples it is not surprising we do not get a perfect fit.

Now, let's visualize the "median" regression line, that is, the line predicted when the random deviaiton is zero, or in other words using only fixed effects and ignoring the random effect. Let's also colour points by site:

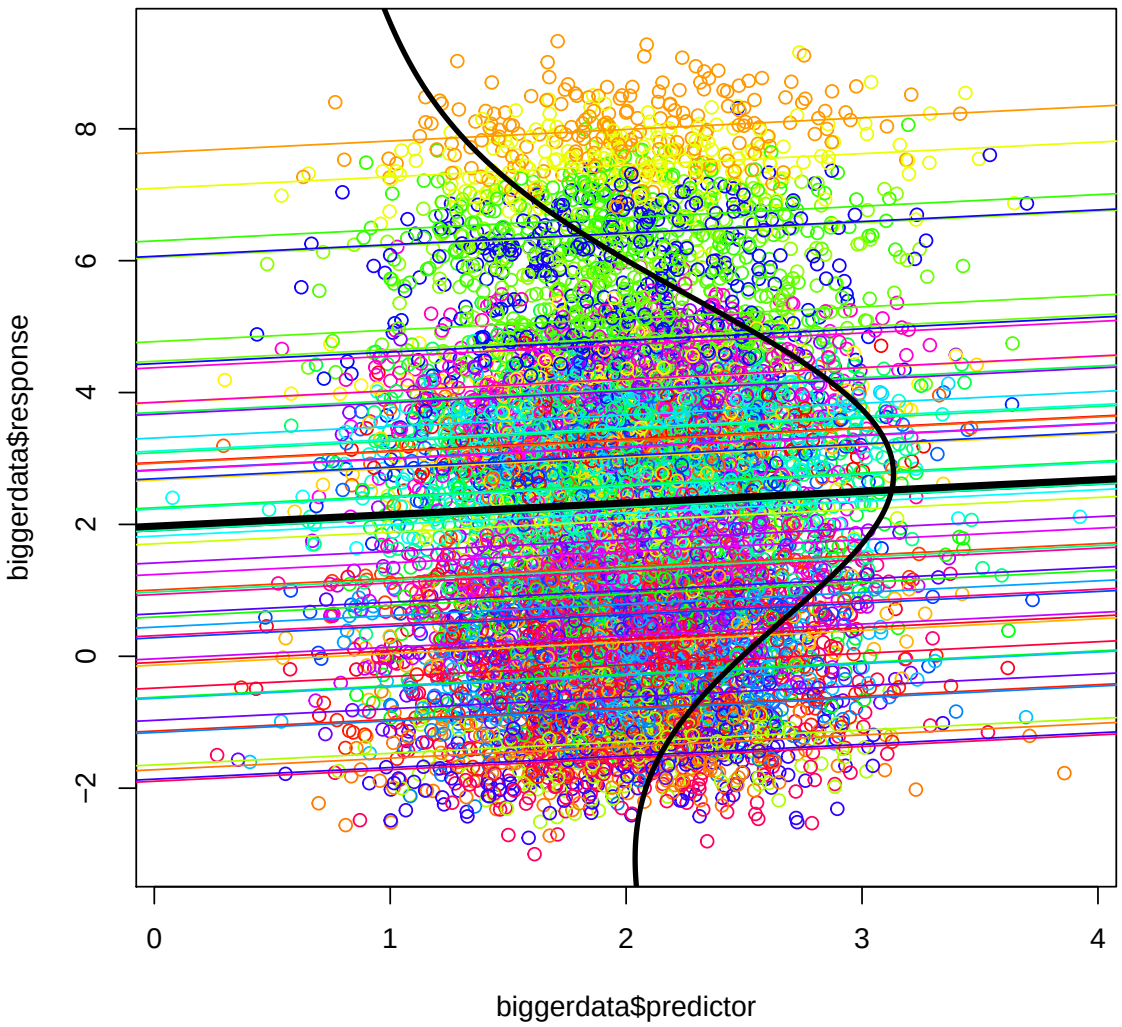
```
plot(biggerdata$predictor, biggerdata$response, col=rainbow(50)[biggerdata$site])
abline(fixef(mm0)[1], fixef(mm0)[2], lwd=5)
```



Now, let's add one thin regression line for each site, using the output of `ranef()` to get a sort of estimate of the site deviation:

```
plot(biggerdata$predictor, biggerdata$response, col=rainbow(50)[biggerdata$site])
abline(fixef(mm0)[1], fixef(mm0)[2], lwd=5)
```

```r
for (i in 1:50)
{
  abline(a = fixef(mm0)[1]+unlist(ranef(mm0))[i],
         b = fixef(mm0)[2], lwd=1, col=rainbow(50)[i])
}

x <- seq(-4*sqrt(rvar),4*sqrt(rvar),sqrt(rvar)/100)

dens <- as.matrix(cbind(x,10*dnorm(x = x, mean =0 , sd = sqrt(rvar))))
dens[,1] <- dens[,1] - mean(dens[,1])
dens[,2] <- dens[,2] - mean(dens[,2])

# library(spdep)
# plot(Rotation(dens, -0.99*pi/2))
#
# plot(dens)
# theta <- -pi/2
# k <- 1
# newcoor <- k * dens %*% matrix(c(cos(theta),-sin(theta),sin(theta),cos(theta)), 2
# plot(0.5*newcoor[,1], 10*newcoor[,2]+fixef(mm0)[1], lwd=3)
#
theta <- -0.95*pi/2
k <- 1
newcoor <- k * dens %*% matrix(c(cos(theta),-sin(theta),sin(theta),cos(theta)), 2, 2)
lines(newcoor[,1]+2, 1+newcoor[,2]+fixef(mm0)[1], lwd=3)
```


biggerdata$response
8
6
4
2
0
−2
0
1
2
3
4
biggerdata$predictor

Fixed or random effect?

When you want to include a categorical variable as a predictor in a model you can generally

- First, if the predictor is a numerical variable (like size, temperature, number of species...) you will generally want a fixed effect, because the order and distance between values is likely meaningful. A random effect consider values as random "names" for grouping levels.
- In general it does not change inference much. Random effects are slightly more efficient because the differences between grouping levels are assumed to come from a normal distribution, instead of being estimated independently from each other.
- A slightly lame but practical reason to use random effects: models output are cleaner with random effects because you get a single parameter estimate for a grouping variable modeled as a random effect, instead of a parameter estimate for each level of the grouping variable when modeled with a fixed effect.
- Using a random effect instead of a fixed effect shifts the focus from the effect of each grouping level to variation among grouping levels. What is more interesting to you?
- Often we use random effect to "correct" for some structure in the data that is not of interest. But random variance parameters can be of interest in themselves. For instance they can quantify genetic variation, individual repeatability, niche specialisation... if a variance parameter is of interest then estimate it using a random effect.
- On the other hand, if you are very interested in how different a particular grouping level is from other levels, use fixed effects.
- Be careful with random effects if the number of grouping levels is small. You definitely need at least 3, probably at least 5, maybe at least 10... With few grouping levels the fitting algorithm could have difficulties estimating the random variance, leading to unstable results (you may get different results from different algorithms) or the random variance being estimated to exactly zero.

3 Confidence intervals and Tests

Sometimes random effects are part of the experimental design and are in the models only to control for confounding effects. But sometimes we care about the value of variance

components or we want to report their statistical significance. Let's try a few approaches to evaluate the significance of a random effect.

* Exercise 2 — CI for variance components in lme4

Use the function `confint` to estimate confidence intervals for the variance component for "site". Hints: 1) by default confint calculate 95% confidence intervals for all fixed and random effects as well as for the residual standard deviation which is labelled ".sigma"; 2) confint returns standard deviations for random effects, instead of variances. Take the square of a standard deviation to obtain a variance.

* Exercise 3 — Testing variance components in lme4

Use the function anova to test the statistical significance of the random effect "site" in the thorn dataset. Hint: 1) you need to fit a simple linear model (lm()) without site to serve as a null model against which to compare your mixed model. The function anova() need to compare two models in this case. 2) the order in which you give your models to anova() matters.

If you have managed to solve this exercise, congrats! What you have done is called a Likelihood Ratio Test (LRT). This is a useful test, but be careful, the models you compare need to be nested, that is, one model is a special case of the other one. In our case the lm() is nested within the lmer() model because in the special case when the variance in the random effect (1—site) is exactly zero the two models are identical. Also, be aware that for random effects Likelihood Ratio Tests using anova are generally conservative.

How does random effect formula work again?

Random effect formula in lme4 and many other R-packages look like (1|group). On the right-hand side of the pipe (|) is the variable that groups data points; for instance location, time, species...

On the left-hand side of the pipe is what varies according to grouping. In today's example we use only 1, which stands for **intercept**. In other words we use random-intercept models. You can write other things on the left-hand side to define more complex models ("random interactions", "random regressions",...) but we will see that next time.

*** Exercise 4 — Do kangaroos have personalities?

We want to know whether the distance at which kangaroos run away when we approach is different among individuals. Load the dataset "roo.csv" and fit linear mixed models of "EscapeDistance" to find out. We can quantify how much individuals differ from each other using repeatability. Repeatability can be defined as the ratio of variance among individuals over the sum variance among individuals plus variance within individuals.

We can estimate the variance among individuals with a random effect of `id` (individual identifier) and the variance within individuals can be estimated as the residual variance.

What other variables to include? Does individual repeatability explain more variance than expected by randomly grouping observations? What is the repeatability of behaviour?

4 More resources

Ben Bolker's GLMM FAQ is a gold mine if you need some technical details about mixed models (generalized or not).

If you are going to do lots of mixed models consider subscribing to the mailing-list https://stat.ethz.ch/mailman/listinfo/r-sig-mixed-models, ask questions or search the archive (anyway, Google is likely to directly send you to the archive).