## Adding predictors to a mixed effects model

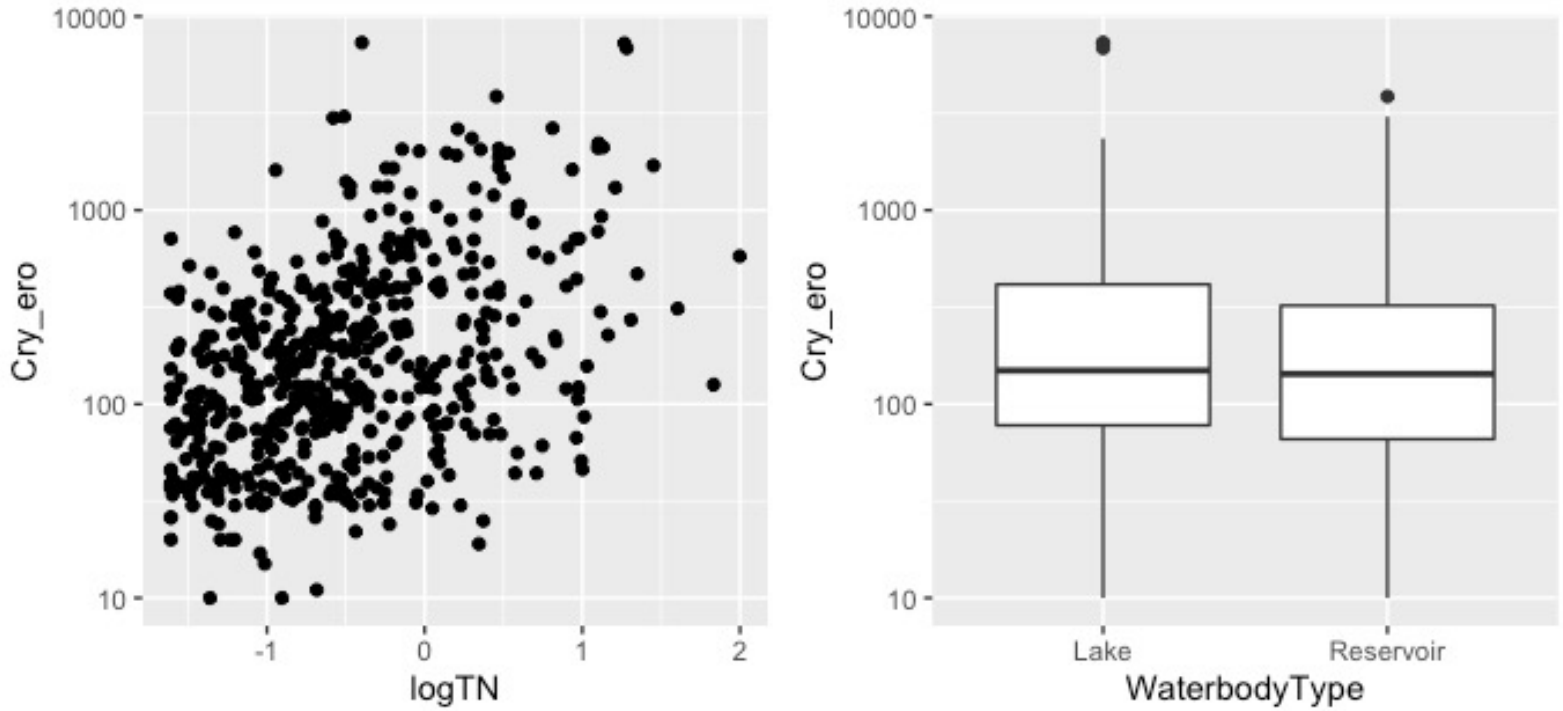
Last time we fit the simplest possible random effects model

```
library(lme4)
```

```
rand.mod = lmer(log(Cry_ero) ~ 1 + (1|WaterbodyID), data = crysub1)
```

Now we'll add in some predictors

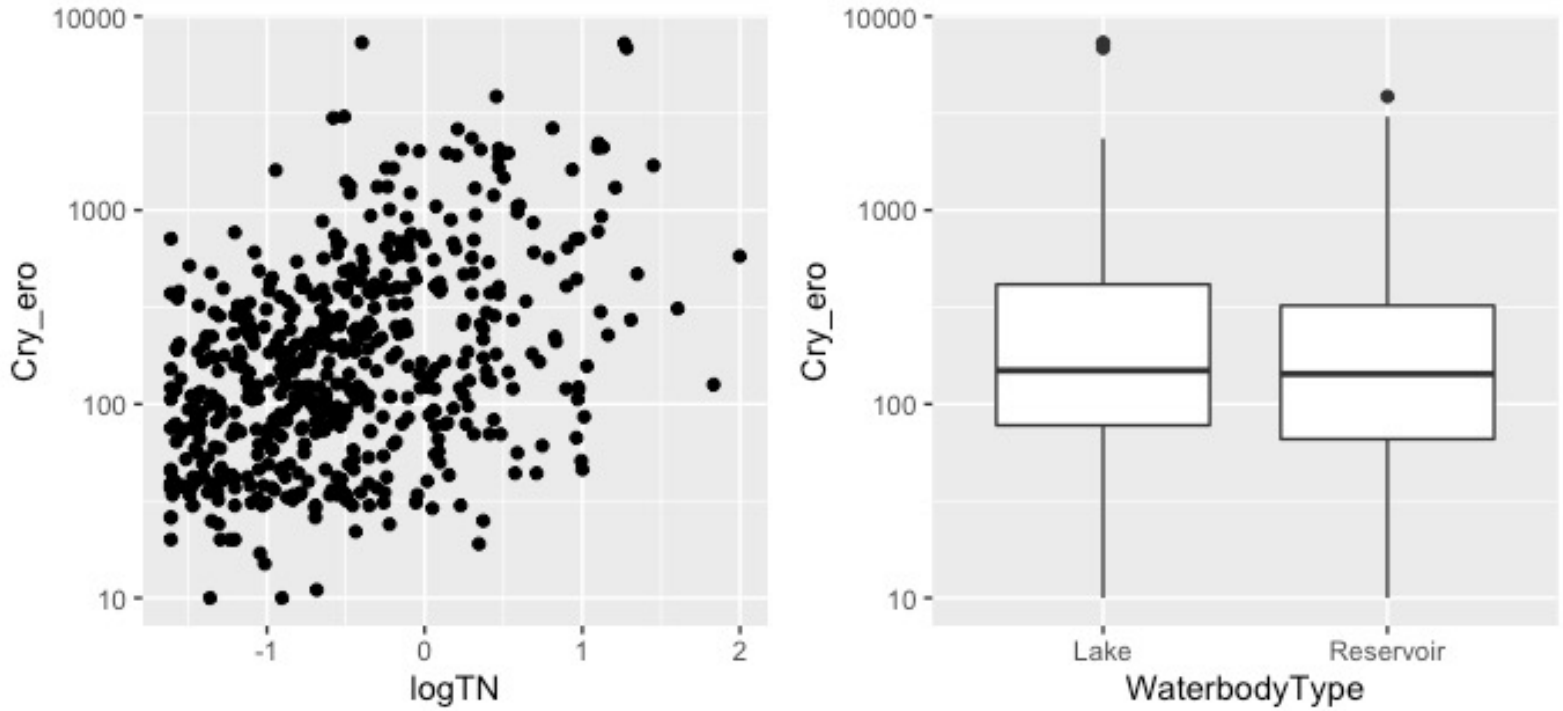
Nitrogen is an important limiting nutrient; waterbody type may have effects on phytoplankton that are not well quantified by other predictors



- Now that we have a model with multiple levels of variation, it is important to think about the **scale(s) at which predictors vary**

- TN is different for every sample: it varies **within and between waterbodies**

- WaterbodyType only varies **among waterbodies**



- How to write it mathematically? One way is to think of a **multi-level model**

**Within each waterbody**

$$\mu_i = \alpha_{j[i]} + \beta_1 * X_i$$
$$Y_i \sim \text{Normal}(\mu_i, \sigma_Y)$$

*X* is log total nitrogen

**Across waterbodies – the intercepts have their own upper-level linear model**

$$\mu_{\alpha j} = \gamma_0 + \gamma_1 * Z_j$$
$$\alpha_j \sim \text{Normal}(\mu_{\alpha j}, \sigma_\alpha)$$

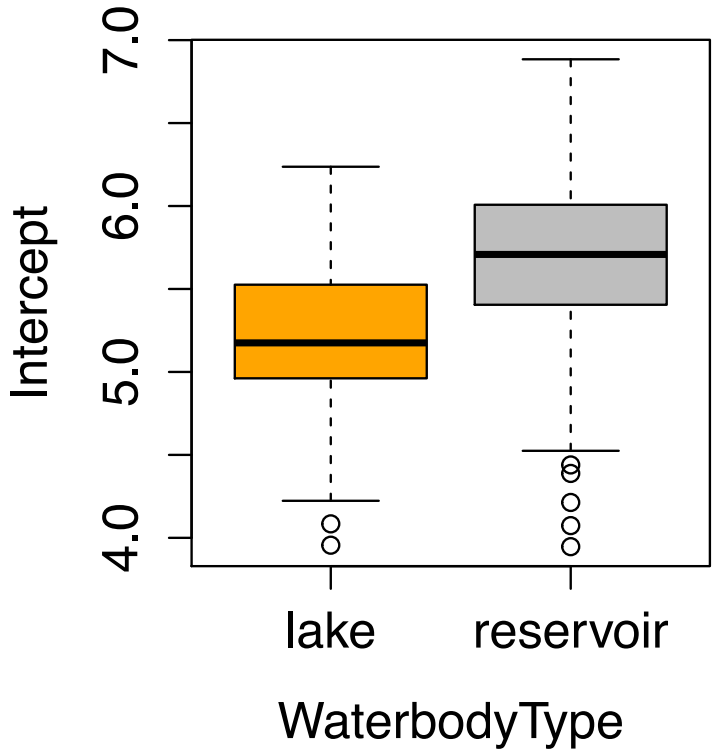
*Z* is indicator for Waterbody type (0 = lake, 1 = reservoir)

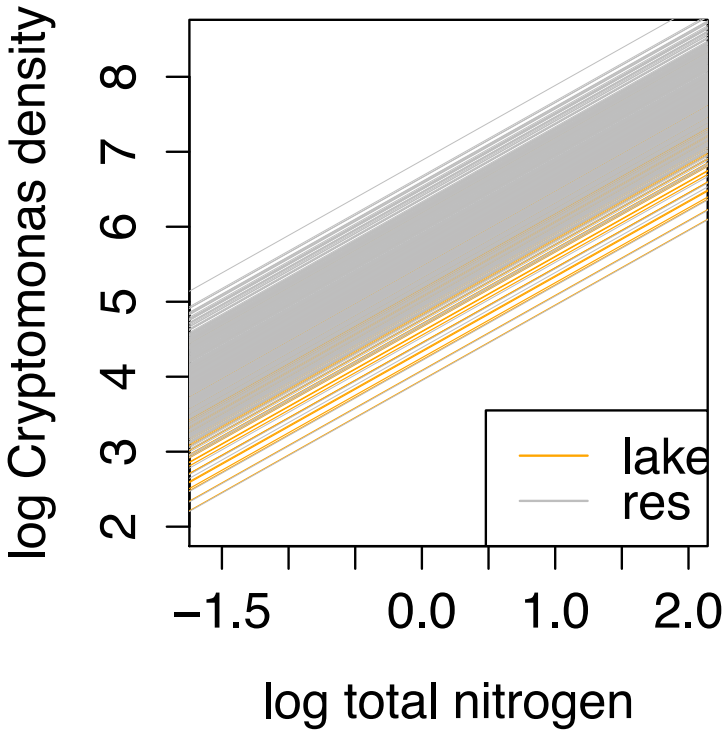
So $\gamma_0$ is the mean for all lakes; $\gamma_0 + \gamma_1$ is the mean for all reservoirs

$\mu_{aj}$ is the expected value for waterbody *j*

$\alpha_j$ is the randomly drawn mean for waterbody *j*

- Some simulated values that illustrate the multi-level model




Different intercepts for different waterbodies,
Same slope

The alpha are random but have a different mean depending on waterbody type

- Although predictors can act at different levels, they are specified the same way

```
epamod = lmer(log(Cry_ero) ~ logTN + WaterbodyType + (1|WaterbodyID),
data = crysub)
```

- Note that every row needs to have the right WaterbodyType

- This makes sense because the expected value of an observation just looks like a multiple regression:

$$\mu_i = \alpha_{j[i]} + \beta_1 * X_i$$

the expected value of $\alpha_{j[i]}$ is $\mu_{\alpha j}$

$$\mu_{\alpha j} = \gamma_0 + \gamma_1 * Z_j$$

$$\mu_i = \gamma_0 + \gamma_1 * Z_i + \beta_1 * X_i$$

- So we have three fixed effects to estimate

- Plus the variance for the intercepts and the residual variance

- Although predictors can act at different levels, they are specified the same way

```
epamod = lmer(log(Cry_ero) ~ logTN + WaterbodyType + (1|WaterbodyID),
data = crysub)
```

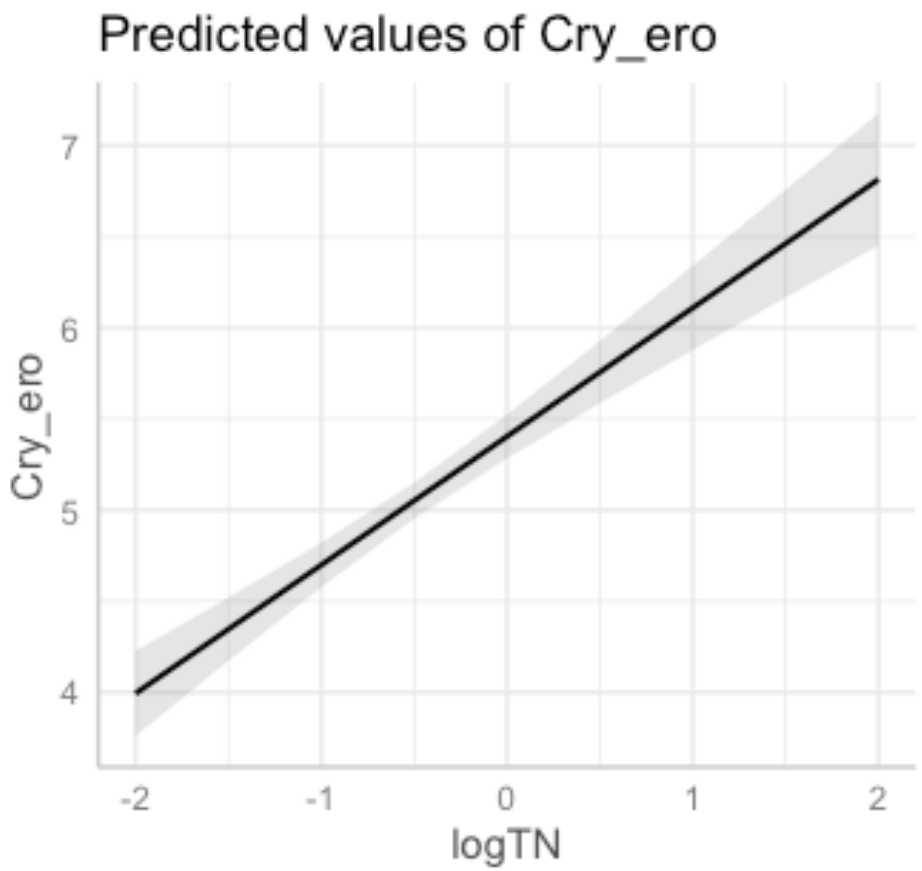
- What if we just did lm() with no random effect for WaterbodyID? Why would this be a problem?

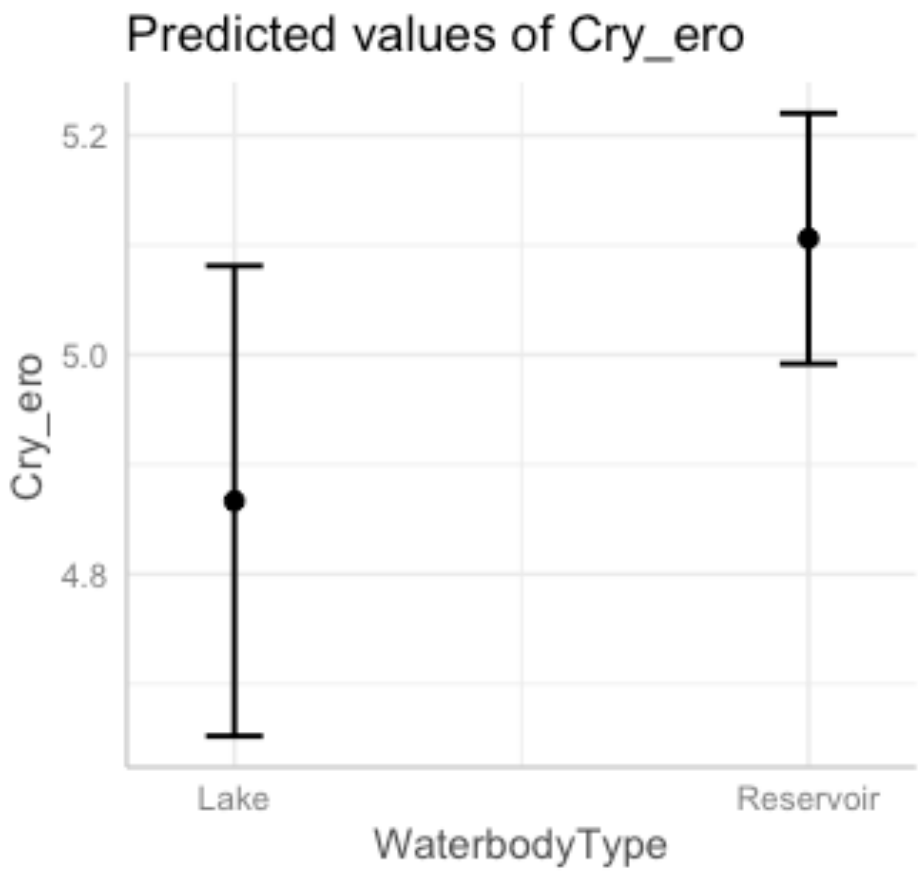
- A general rule: if a predictor only varies across groups of data, need to have the group as a random effect

```r
summary(epamod)
```

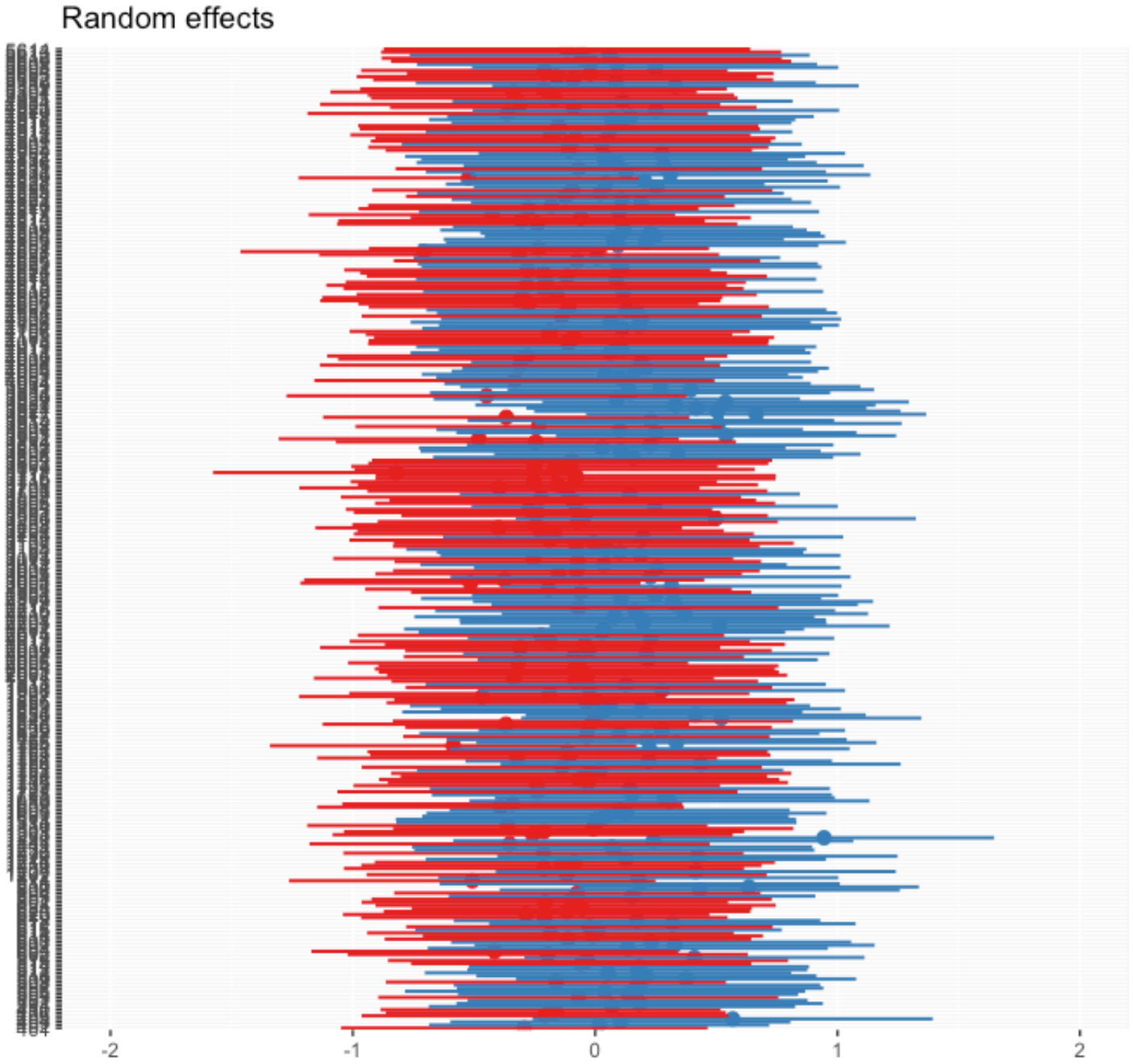
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log(Cry_ero) ~ logTN + WaterbodyType + (1 | WaterbodyID)
##    Data: crysub
##
## REML criterion at convergence: 1582
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.2579 -0.6383 -0.0091  0.5893  3.1371
##
## Random effects:
##  Groups      Name        Variance Std.Dev.
##  WaterbodyID (Intercept) 0.220    0.469
##  Residual                0.924    0.961
## Number of obs: 533, groups:  WaterbodyID, 320
##
## Fixed effects:
##                          Estimate Std. Error t value
## (Intercept)                5.2186     0.1033    50.5
## logTN                      0.7052     0.0713     9.9
## WaterbodyTypeReservoir     0.2391     0.1262     1.9
##
## Correlation of Fixed Effects:
##             (Intr) logTN
## logTN        0.005
## WtrbdyTypRs -0.817  0.351
```

```
grid.arrange(grobs = plot(ggeffect(epamod)), nrow = 1)
```



```
plot_model(epamod, type = 're')
```



These plots let us see the fixed effects and random effects individually

But I also want to combine them to think about the multi-level structure

The predicted intercept for each waterbody includes a fixed effect part and a random effect part

```
#fixed effects estimates
fixef(epamod)

##             (Intercept)                    logTN WaterbodyTypeReservoir
##                  5.2186                   0.7052                   0.2391
```

**For lakes:** (Intercept) + random deviate from ranef()

**For reservoirs:** (Intercept) + WaterbodyTypeReservoir + random deviate from ranef()

```r
#extract the random effects for waterbody
random.intercepts = ranef(epamod)$WaterbodyID[[1]]

#make a new factor that codes WaterbodyType for each waterbody
waterbody.type = factor(tapply(crysub$WaterbodyType, crysub$WaterbodyID,
function(x) as.character(x[1])))

#what is the predicted intercept for each waterbody?
fixed.effects.waterbody = fixef(epamod)["(Intercept)"] +
fixef(epamod)["WaterbodyTypeReservoir"]*(waterbody.type == "Reservoir")

#the estimate for each waterbody is the random effect plus the effect of the
group-level predictor
waterbody.effects = random.intercepts + fixed.effects.waterbody
```

Plot waterbody-level estimates and predictors

```
#plot the waterbody estimate by waterbody type
plot(waterbody.effects ~ waterbody.type, col = c('orange', 'grey'), xlab =
'waterbody type', ylab = 'model-fitted intercept')
```

Plot predicted relationship in each waterbody

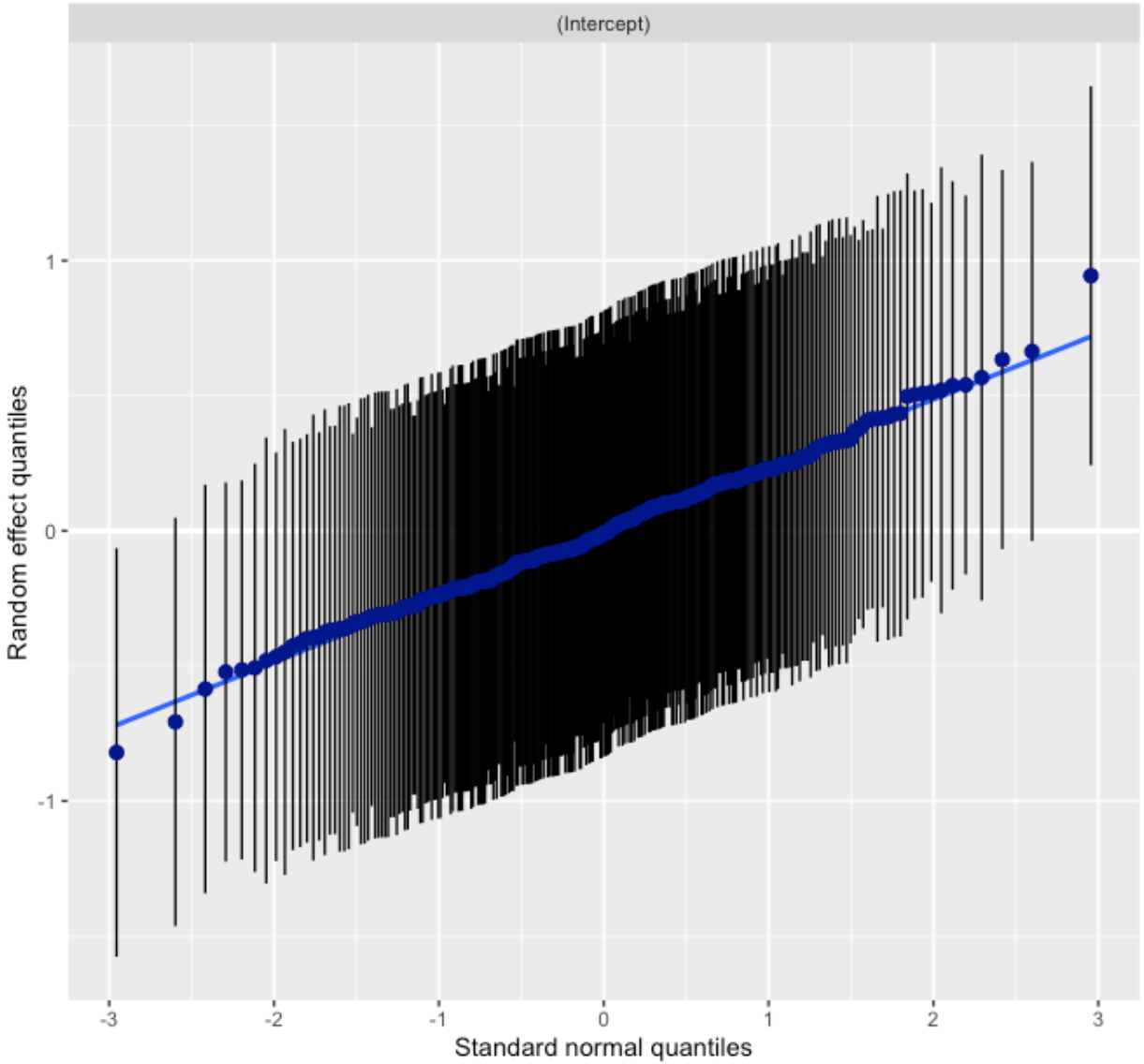
```
#plot the model-predicted regression within each waterbody. just drawing a lin
e with curve(), where the intercept varies by waterbody but each waterbody has
 the same slope.
curve(waterbody.effects[1] + fixef(epamod)["logTN"]*x, from = min(crysub$logT
N), to = max(crysub$logTN), xlab = 'logTN', ylab = 'log density', ylim = c(3.5
 7.5))

for (i in 2:length(waterbody.effects)) {
  curve(waterbody.effects[i] + fixef(epamod)["logTN"]*x, from = min(crysub$log
TN), to = max(crysub$logTN), col = c('orange', 'grey')[waterbody.type[i]], add
 = T)
}
```

model−fitted intercept
5.0
5.5
6.0
Lake
waterbody type


log density
4
5
6
7
−1
0
1
2
logTN

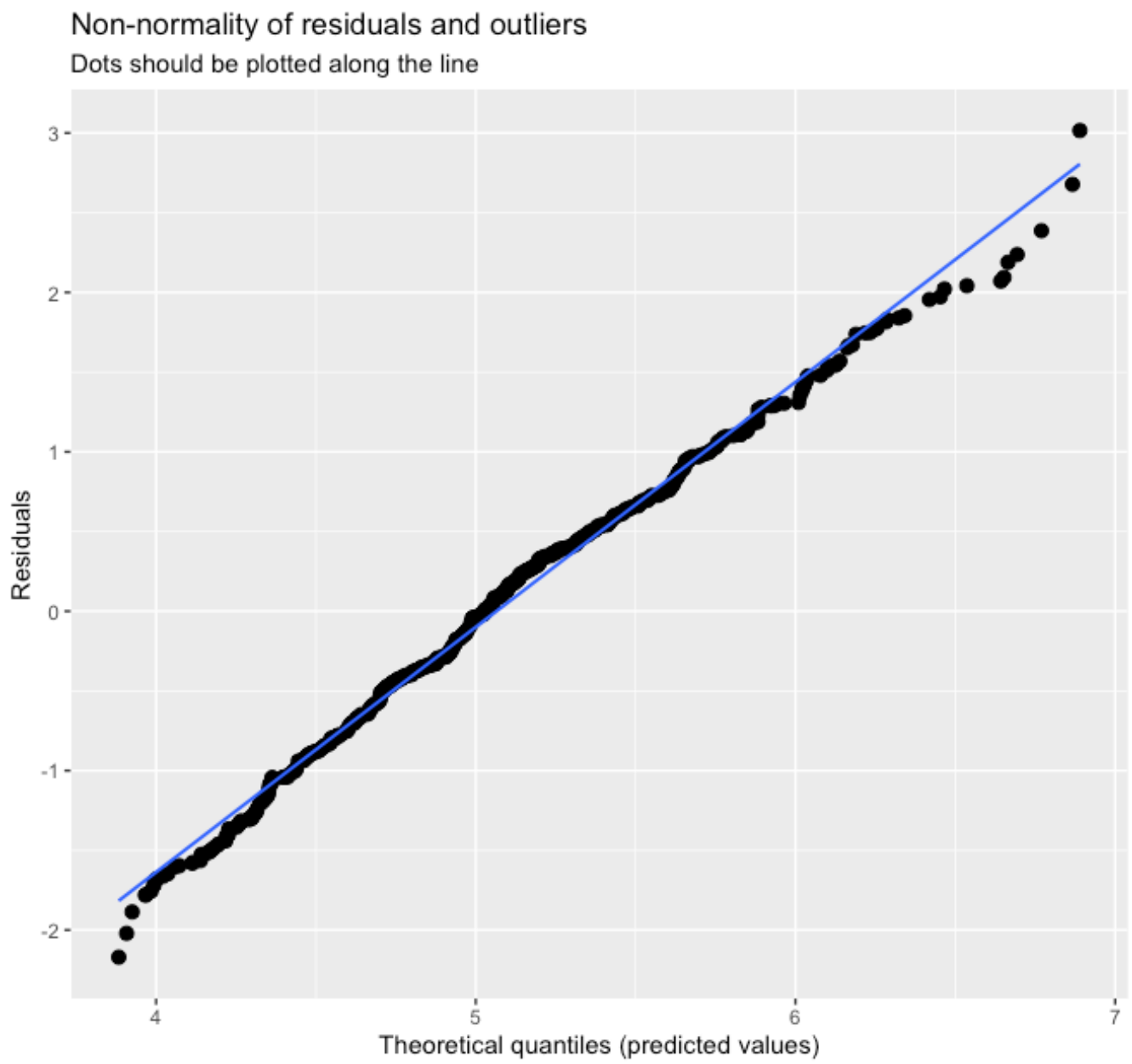
## Assessing model fit

```
plot_model(epamod, type = 'diag')[[2]]
```



## Assessing model fit

```
plot_model(epamod, type = 'diag')[[1]]
```



## How much random variation at different levels?

```
VarCorr(epamod)

##  Groups      Name        Std.Dev.
##  WaterbodyID (Intercept) 0.469
##  Residual                0.961
```

## $R^2$ for mixed models

We would like to say 'how much variation do these predictors explain?'

But now we have **variation at multiple levels**

**One approach:** see how much variation goes up when we remove a predictor.

```
epamod.noType = lmer(log(Cry_ero) ~ logTN + (1|WaterbodyID), data = crysub)
VarCorr(epamod.noType)
```

```
##  Groups      Name        Std.Dev.
##  WaterbodyID (Intercept) 0.473
##  Residual                0.963
```

```
epamod.noTN = lmer(log(Cry_ero) ~ WaterbodyType + (1|WaterbodyID), data = crys
ub)
VarCorr(epamod.noTN)
```

```
##  Groups      Name        Std.Dev.
##  WaterbodyID (Intercept) 0.694
##  Residual                0.954
```

## $R^2$ for mixed models

We would like to say 'how much variation do these predictors explain?'

But now we have variation at multiple levels

**More quantitative:** compare variance estimates for **full vs. null models**

$$1 - \frac{\sigma_F^2}{\sigma_0^2}$$

```
epamod.nopred = lmer(log(Cry_ero) ~ 1 + (1|WaterbodyID), data = crysub)


1 - VarCorr(epamod)$WaterbodyID[1]/VarCorr(epamod.nopred)$WaterbodyID
[1]

## [1] 0.5501
```

```
1 - (sigma(epamod)^2)/(sigma(epamod.nopred)^2)

## [1] -0.01763
```

## $R^2$ for mixed models

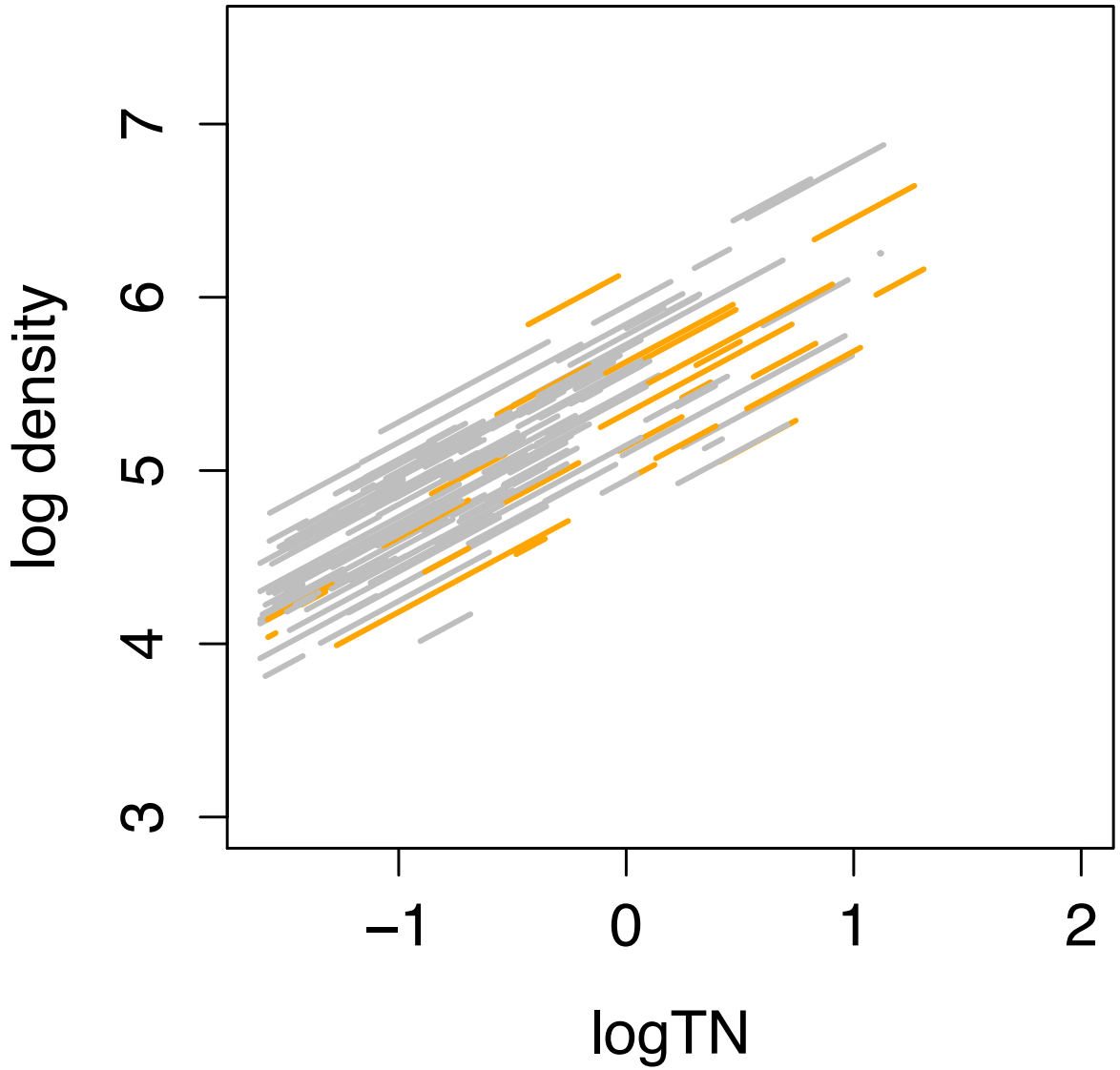
Total nitrogen varies both within and between waterbodies

**TN is a good predictor at the waterbody scale, but not within waterbodies over time**

Why is that?

```
mod.TN = lmer(logTN ~ 1 + (1|WaterbodyID), data = crysub)
VarCorr(mod.TN)

##  Groups      Name        Std.Dev.
##  WaterbodyID (Intercept) 0.712
##  Residual                0.273
```

**Same plot as before, but plot each line using the range of logTN from that waterbody**



**Is there a way to say 'How much of the total variation is explained?'**

Yes…but don't take it too seriously

$$R^2_{LMM(m)} = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_\alpha^2 + \sigma_Y^2}$$

$$R^2_{LMM(c)} = \frac{\sigma_f^2 + \sigma_\alpha^2}{\sigma_f^2 + \sigma_\alpha^2 + \sigma_Y^2}$$

**Marginal $R^2$** – compares the variation predicted by the fixed effects to the total variation, including random effects and residual

**Conditional $R^2$** – compare the variation predicted by the fixed+random effects to the total that adds in the residual

**Is there a way to say 'How much of the total variation is explained?'**

Yes…but don't take it too seriously

$$R^2_{LMM(m)} = \frac{\sigma_f^2}{\sigma_f^2 + \sigma_\alpha^2 + \sigma_Y^2}$$

$$R^2_{LMM(c)} = \frac{\sigma_f^2 + \sigma_\alpha^2}{\sigma_f^2 + \sigma_\alpha^2 + \sigma_Y^2}$$

```
r.squaredGLMM(epamod)
```

```
##     R2m     R2c
## 0.1726 0.3315
```

```
r.squaredGLMM(epamod.noTN)
```

```
##       R2m       R2c
## 0.004345 0.348819
```

**Inference with mixed models: null hypothesis tests and AIC**

Inference with mixed models is hard, we need to pay attention to some details

**Effective sample size can be small / LRT is asymptotic**

- We've been using Chi-square LRT for GLMs
- This is an approximation, derived as sample size -> Infinity
- Usually the approximation is fine; but for small sample sizes it is **anti-conservative**
- Mixed models can have a small effective sample size; depends on the predictor
- E.g. 100 samples from 5 islands. If Island Area is a predictor, we have 5 observations.

**Defining an F-statistic is hard**

- The F-test (and t-test) are designed to account for sample size
- But we need to get an F-statistic

$$F = \frac{\text{explained variance}}{\text{unexplained variance}}$$

- We also need to **count parameters / degrees of freedom**
- **How many parameters** does a mixed model have? Does # groups matter?

**An approximate F-test**

You can get a Chi-squared LRT with drop1() or Anova(), but they can be anti-conservative

Approximate F-test is better

```
library(lmerTest)

anova(epamod, ddf = "Kenward-Roger")

## Analysis of Variance Table of type 3  with  Kenward-Roger
## approximation for degrees of freedom
##                Sum Sq Mean Sq NumDF DenDF F.value Pr(>F)
## logTN            89.6    89.6     1   394    97.3 <2e-16 ***
## WaterbodyType     3.3     3.3     1   342     3.6  0.059 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

**The parametric boostrap**

Instead of making assumptions for a test, we can use simulation

1. We **fit our model**, and calculate the likelihood for the fitted model.

2. We **fit a restricted model** that removes a predictor of interest. We calculate the likelihood for that model.

3. We want to see if the likelihood ratio of the two models is 'significantly' large. But we don't want to use the Chi-square approximation.

4. Instead, **we simulate from the restricted model many times**, which is equivalent to simulating data under the null hypothesis.

5. In each iteration we **fit the full model and the restricted model to the simulated data**, and calculate the likelihood ratio. This lets us ask 'If the null hypothesis were true, what would the distribution of the likelihood ratio look like? And is the real likelihood ratio significantly large, compared to the null distribution?'

**Bootstrap** = using simulated datasets for inference
**Parametric** = assuming the model is accurate

```r
epamod.noType = lmer(log(Cry_ero) ~ logTN + (1|WaterbodyID), data = crysub, RE
ML = FALSE)

sim.null = simulate(epamod.noType)


null.full = lmer(sim.null[[1]] ~ logTN + WaterbodyType + (1|WaterbodyID), data
 = crysub, REML = FALSE)
null.restricted = lmer(sim.null[[1]] ~ logTN + (1|WaterbodyID), data = crysub,
 REML = FALSE)

logLik(null.full)

## 'log Lik.' -773 (df=5)

logLik(null.restricted)

## 'log Lik.' -773 (df=4)

-2*(logLik(null.restricted)[1] - logLik(null.full)[1])

## [1] 0.08931
```

```
epamod = lmer(log(Cry_ero) ~ logTN + WaterbodyType + (1|WaterbodyID), data = c
rysub, REML = FALSE)
epamod.noType = lmer(log(Cry_ero) ~ logTN + (1|WaterbodyID), data = crysub, RE
ML = FALSE)


library(pbkrtest)
PBmodcomp(epamod, epamod.noType)

## Parametric bootstrap test; time: 28.81 sec; samples: 1000 extremes: 55;
## large : log(Cry_ero) ~ logTN + WaterbodyType + (1 | WaterbodyID)
## small : log(Cry_ero) ~ logTN + (1 | WaterbodyID)
##        stat df p.value
## LRT     3.6  1   0.058 .
## PBtest  3.6      0.056 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Requires fewer assumptions

Can be slow

Results not identical

**REML vs ML**

- REML is a bias-corrected maximum likelihood estimator
- But **to compare different models, you have to fit using ML**
- E.g. LRT or AIC

## AIC

- Can be used, but a bit sketchy
- How many parameters?
- K = # fixed effects parameters + # variances estimated
- Tends to be anti-conservative like the Chi-square LRT
- There are other information criteria on the horizon (e.g. DIC)

```
AICc(epamod)
```

```
## [1] 1582
```

```
AICc(epamod.noTN)
```

```
## [1] 1664
```

```
AICc(epamod.noType)
```

```
## [1] 1583
```

**Testing random effects**

- Often you don’t need to, or shouldn’t
- Is the random effect there to account for non-independence in the data?
- Or does it represent a hypothesis?
- If you need to test, LRT and AIC tend to be **conservative**
- Because the null hypothesis is that the variance = 0 (but it can’t go any lower)
- So you can use these but know they’re conservative

## Testing random effects

- You can also use parametric bootstrap
- Necessary for a single random effect
- **Can't compare likelihoods of lmer() and lm() models**

```
library(RLRsim)

epamod = lmer(log(Cry_ero) ~ logTN + WaterbodyType + (1|WaterbodyID), d
ata = crysub, REML = TRUE)

exactRLRT(epamod)

##
##  simulated finite sample distribution of RLRT.
##
##  (p-value based on 10000 simulated values)
##
## data:
## RLRT = 8.937, p-value = 0.0013
```