# Linear Mixed-Effect Modeling in R

Clay Ford

Spring 2018

# Agenda

- Conceptual overview of mixed-effect models
- How to implement in R
- How to assess significance of parameters
- Diagnostics
- Predictions
- Model selection

# What are Linear Mixed-Effect Models?

- Linear models with fixed and random effects.
- LME (**L**inear **M**ixed-**E**ffect) models are suitable for clustered, longitudinal or repeated-measures data in which the data are grouped by *random levels* and the dependent variable is continuous.
- "Grouped by random levels" means there are several observations that belong to a single subject or group that can be thought of as a random sample from a population.
- These groups are sometimes referred to as "random effects".

# Candidate for linear mixed-effect modeling

- We have a new medical treatment. We carry out an experiment where we randomly prescribe the treatment or placebo to patients and collect response data every week for two months to gauge efficacy.
- Treatment is the *fixed effect*. Other fixed effects may include age, gender, weight, time, etc. If we were to replicate this study, we would use these variables again.
- The data is grouped by patients, a sample from a population. We will likely have *random effects* associated with patients. If we were to replicate this study, we would not use the same patients again.

# Another candidate for linear mixed-effect modeling

We have a new computer-based math curriculum. We carry out an experiment where we randomly prescribe either the new curriculum or current curriculum to teachers in various schools and measure change in student pre- and post-test scores.

- Curriculum is the *fixed effect*. Other fixed effects might include gender, race, etc. If we were to replicate this study, we would use these variables again.
- The data is grouped by school, and teacher within school. We will likely have *random effects* associated with schools and teachers. If we were to replicate this study, we would not necessarily use the same teachers and schools again.

# Example of fixed effect vs random effect

In our medical treatment example, let's say we add age to our model.

We get a coefficient for age that summarizes the relationship between age and the response.

If we think the relationship between age and the response *varies between patients*, we "add a random effect for age". This produces an estimate of variability of how the age coefficient differs between patients.

**Fixed effects have coefficients.**
**Random effects have estimates of variation.**

# Specification of a Linear Model

Let's review the specification for a basic linear model:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \epsilon$$

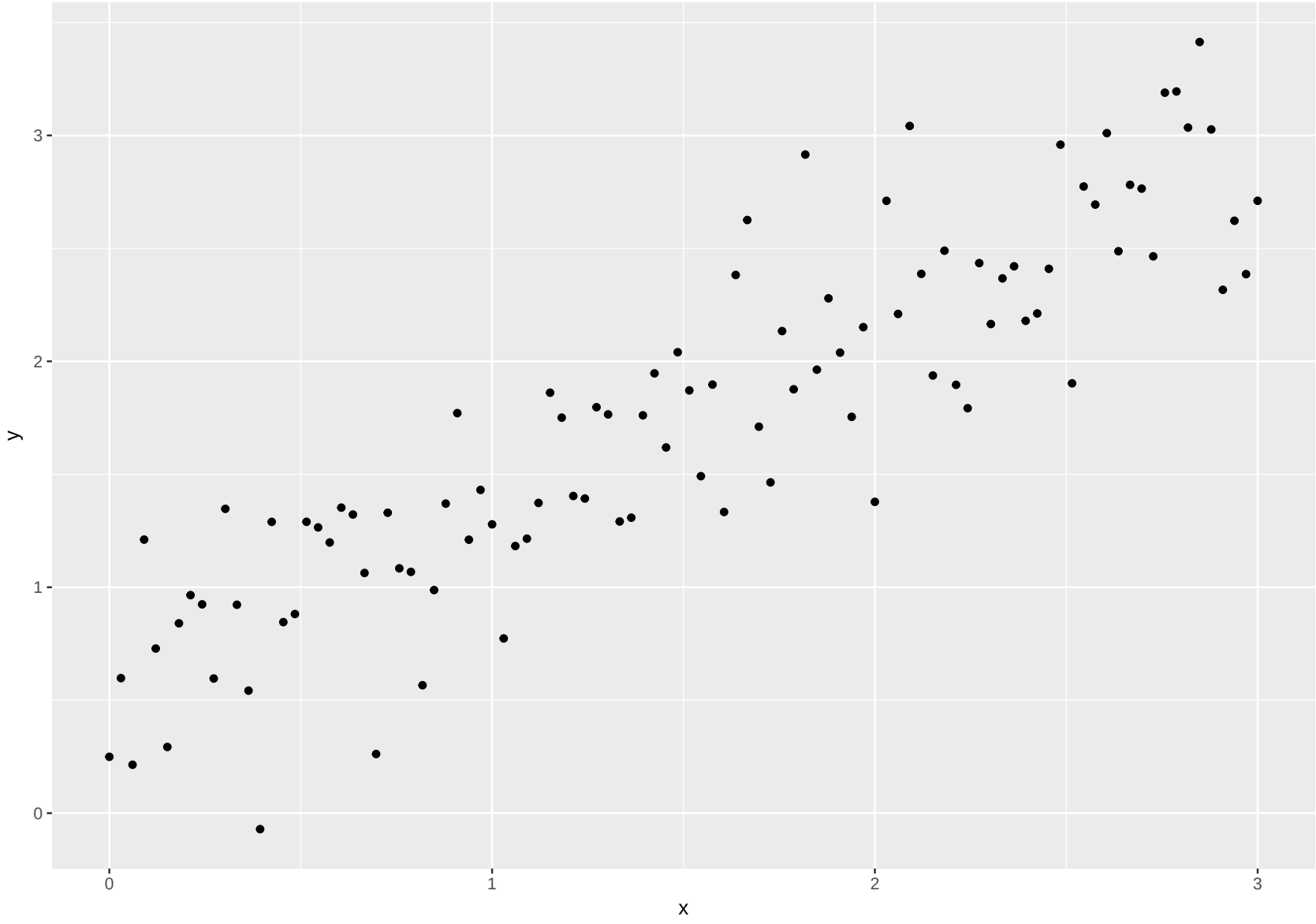
$$\epsilon \sim N(0, \sigma)$$

where $\boldsymbol{X}\boldsymbol{\beta} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$.

This says our dependent variable ($\boldsymbol{Y}$) is equal to our independent variables ($\boldsymbol{X}$) multiplied by coefficients ($\boldsymbol{\beta}$) and added up, plus some random error ($\epsilon$) that comes from a Normal distribution with mean 0 and standard deviation $\sigma$.

A key assumption is the independence of the random errors (ie, independent observations).

# Simple data for a linear model



# Fitting a linear model in R

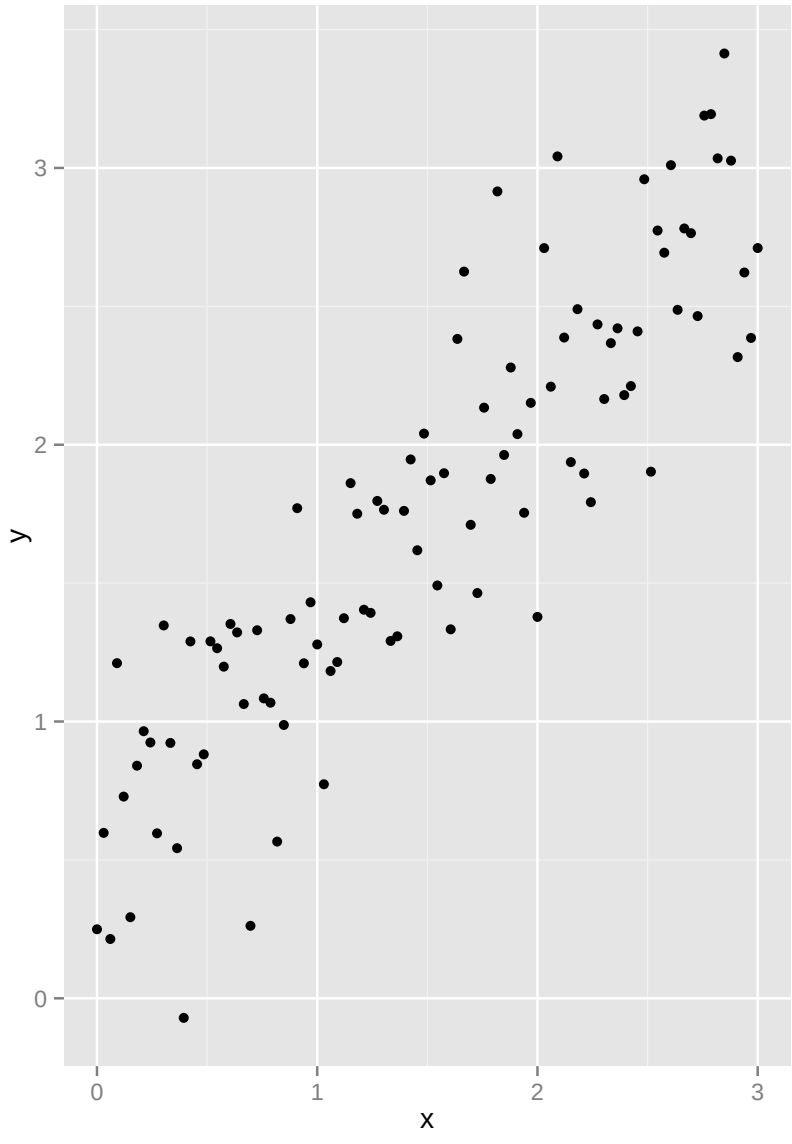
```
m1 <- lm(y ~ x)
coef(m1) # model coefficients
```

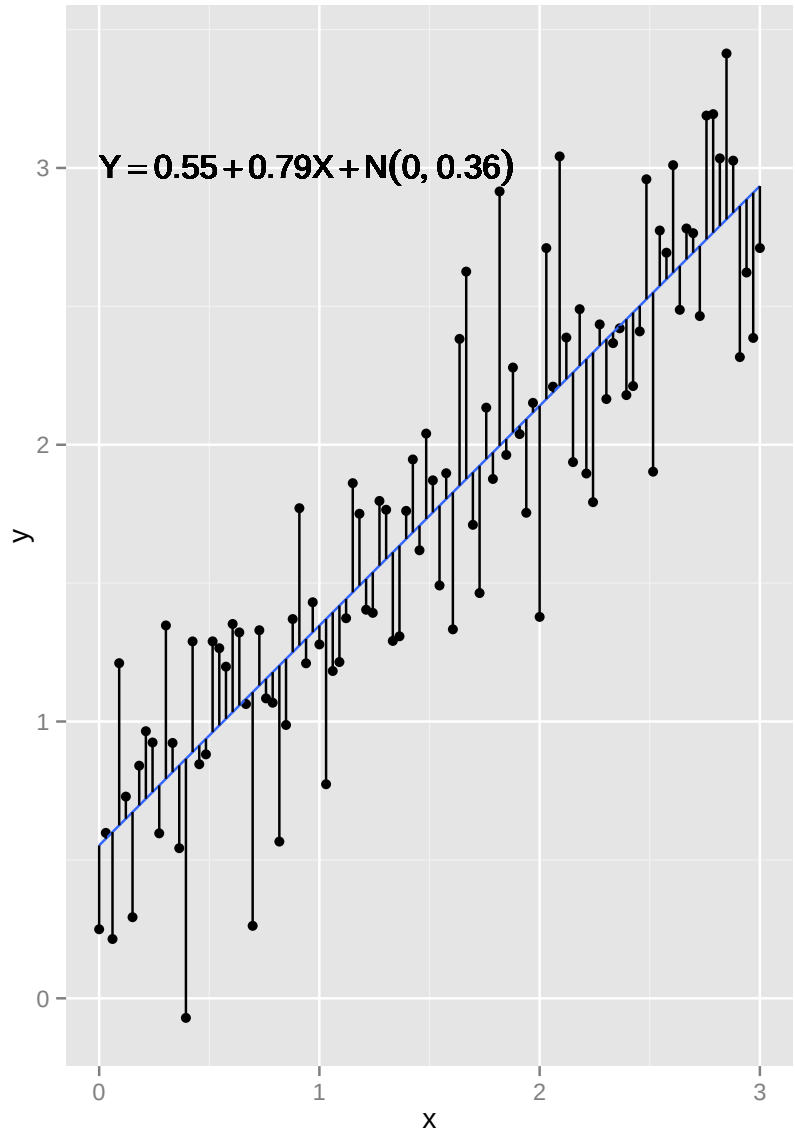
```
## (Intercept)           x
##      0.5525      0.7940
```

```
summary(m1)$sigma # estimate of error standard deviation
```

```
## [1] 0.3611
```

# Visualizing a linear model



# Specification of a Linear Mixed-Effect Model

The linear mixed-effect model is more complex:

$$\boldsymbol{Y_i} = \boldsymbol{X_i}\beta + \boldsymbol{Z_i}\boldsymbol{b_i} + \epsilon_i$$

$$\boldsymbol{b_i} \sim N(\mathbf{0}, \boldsymbol{D})$$

$$\epsilon_i \sim N(\mathbf{0}, \boldsymbol{R_i})$$

Notice the `i` subscript. Linear mixed-effect models are fit to groups of data, say repeated measures made on subjects. $\boldsymbol{X_i}$ are the predictors. $\beta$ are the fixed effect coefficients. $\boldsymbol{Z_i}$ is a subset of the predictors. $\boldsymbol{b_i}$ are the random effect *predictions*. The $\epsilon_i$ are the random errors.

In contrast to a standard linear models, random errors can be correlated.

# Random effects and errors

In a standard linear model we just have to estimate one random error parameter: $\sigma$

In a linear mixed-effect model we have two *matrices* of error structures:
- one for the random effects: $\boldsymbol{D}$
- one for the random errors: $\boldsymbol{R_i}$

$\boldsymbol{D}$ is sometimes referred to as *between-groups* variation while $\boldsymbol{R_i}$ is called *within-groups* variation

These matrices can take on different *structures*.

# What do we mean by "structures"?

The $\boldsymbol{D}$ and $\boldsymbol{R_i}$ matrices are known as *variance-covariance* matrices.

The variance is on the diagonal. The covariance is on the off-diagonal. For example, a $\boldsymbol{D}$ matrix for two random effects might be structured as follows:

$$\begin{pmatrix} \sigma^2_{b1} & \sigma_{b1,b2} \\ \sigma_{b1,b2} & \sigma^2_{b2} \end{pmatrix}$$

In this matrix there are three parameters to estimate: $\sigma^2_{b1}$, $\sigma^2_{b2}$, and $\sigma_{b1,b2}$. This structure assumes the two random effects have covariance, ie, they vary together. *This is the default in R when you specify that two predictors have random effects.*

# Example of another $\boldsymbol{D}$ matrix structure

We could also constrain the covariance to be 0 in a $\boldsymbol{D}$ matrix, like so:

$$\begin{pmatrix} \sigma^2_{b1} & 0 \\ 0 & \sigma^2_{b2} \end{pmatrix}$$

In this matrix there are only two parameters to estimate, $\sigma^2_{b1}$ and $\sigma^2_{b2}$, since we assume covariance is 0.

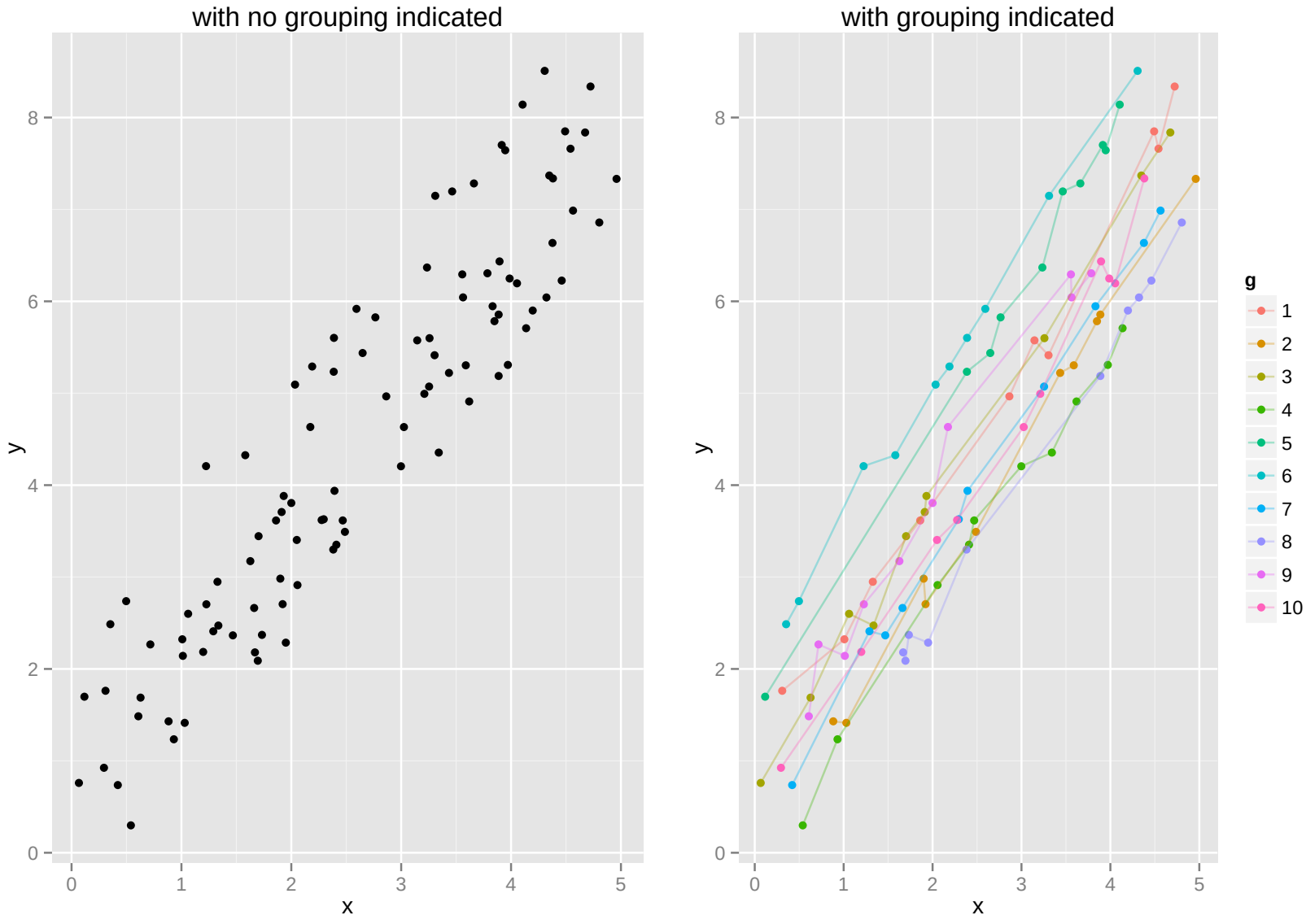
# Example of a $\boldsymbol{R_i}$ matrix structure

The simplest covariance matrix for the random errors is the diagonal structure:

$$\begin{pmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{pmatrix}$$

This says that residuals associated with observations on the same subject have equal variance and are uncorrelated. We only have one parameter to estimate. This is the same as the standard linear model. *This is the default in R and all that we'll cover today.*

# Simple data for a linear mixed-effect model



# What do we notice about the data?

- The different groups seem to have the same trajectory, or *slope*
- The different groups seem to have different starting points, or *intercepts*
- Each group of data appears as if it could be modeled by a straight line model

If we were to imagine what kind of mathematical process gave rise to this data, we might propose a straight line model with a *fixed* slope coefficient but a *random* intercept.

# Fitting a linear mixed-effect model in R

Fit a model with a "random" intercept

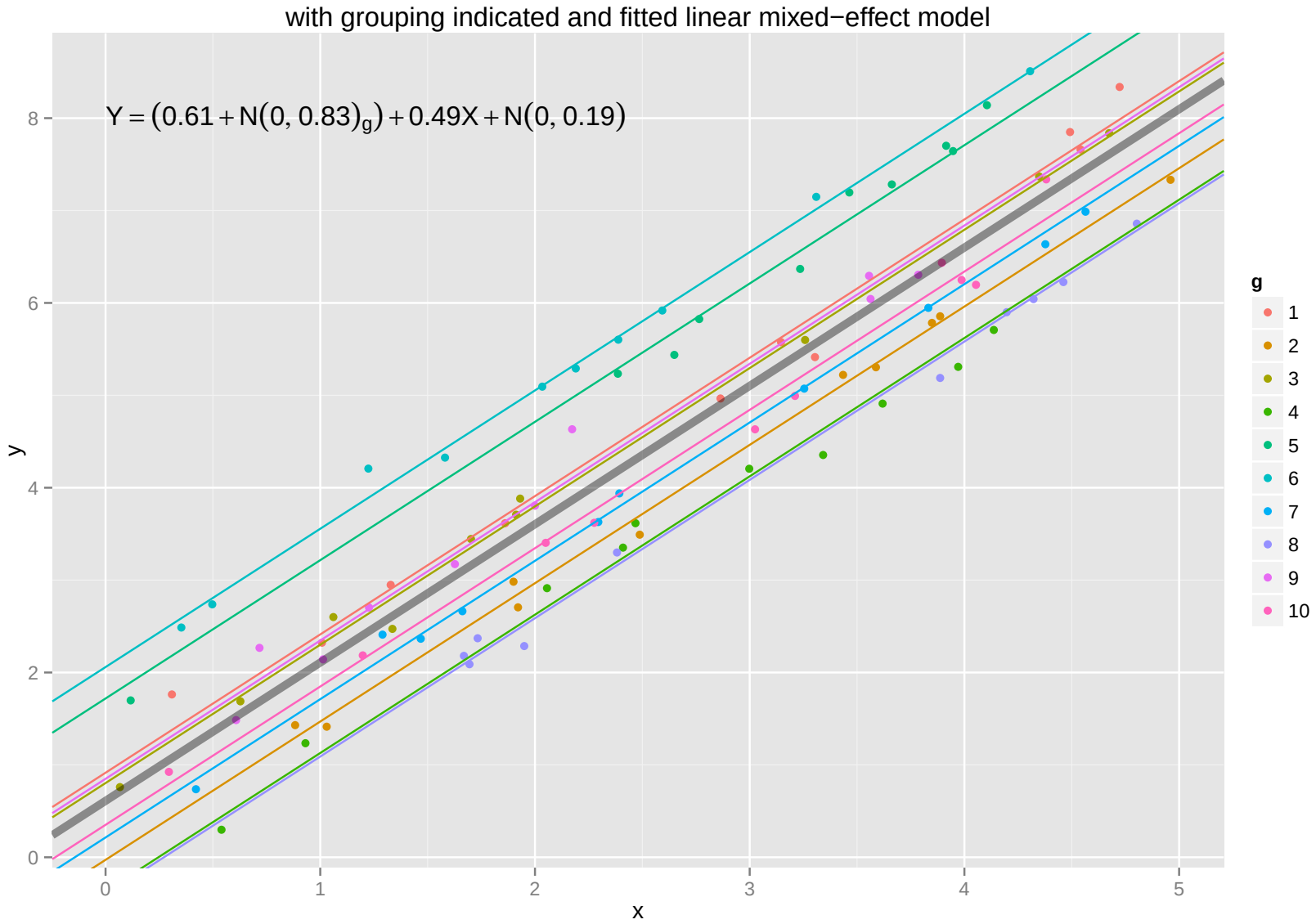
```
library(lme4) # package for fitting lme
lme1 <- lmer(y ~ x + (1 | g), data = dat)
fixef(lme1) # fixed effect estimates
```

```
## (Intercept)           x
##      0.6106      1.4972
```

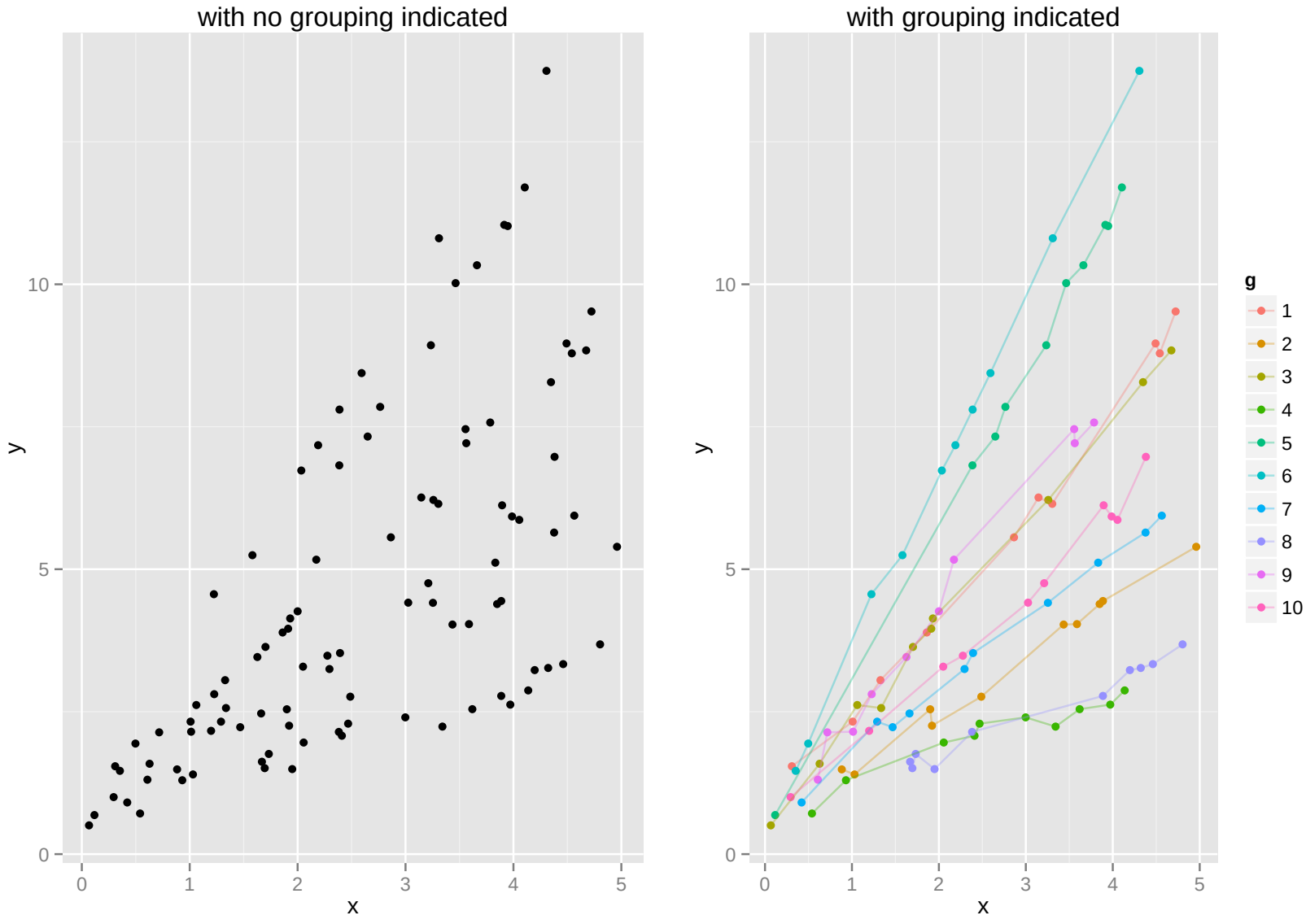
```
VarCorr(lme1) # D and R estimates
```

```
##  Groups   Name        Std.Dev.
##  g        (Intercept) 0.828
##  Residual             0.189
```

# Visualizing a linear mixed-effect model



# Random slopes



# What do we notice about the data?

- The different groups seem to have the same intercept
- The different groups seem to have *different slopes*
- Each group of data appears as if it could be modeled by a straight line model

If we were to imagine what kind of mathematical process gave rise to this data, we might propose a straight line model with a *fixed* intercept coefficient but a *random* slope.

# Fit a LME model for a random slope

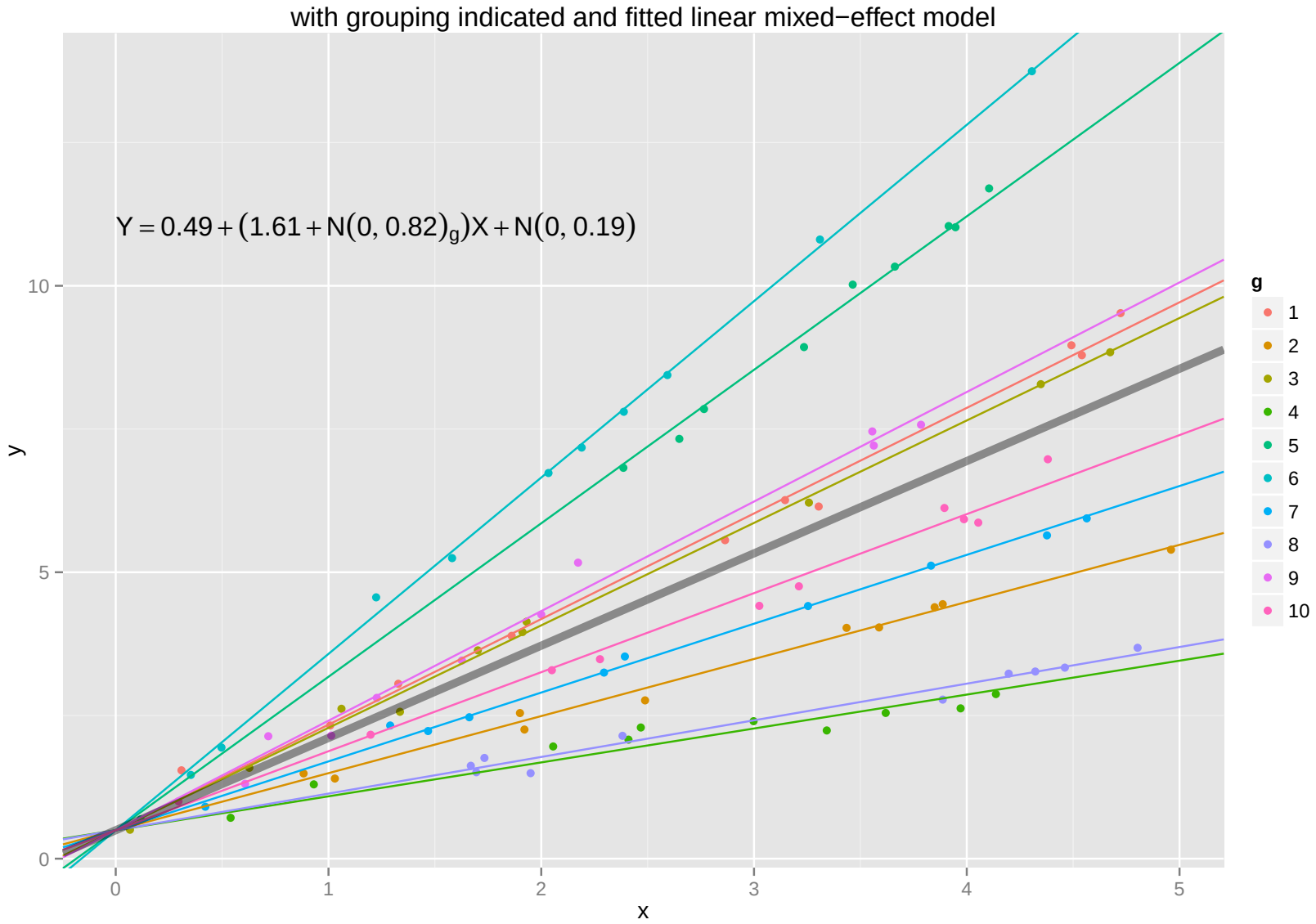
```
lme2 <- lmer(y ~ x + (-1 + x | g), data = dat)
fixef(lme2) # fixed effect estimates
```

```
## (Intercept)           x
##      0.4962      1.6111
```

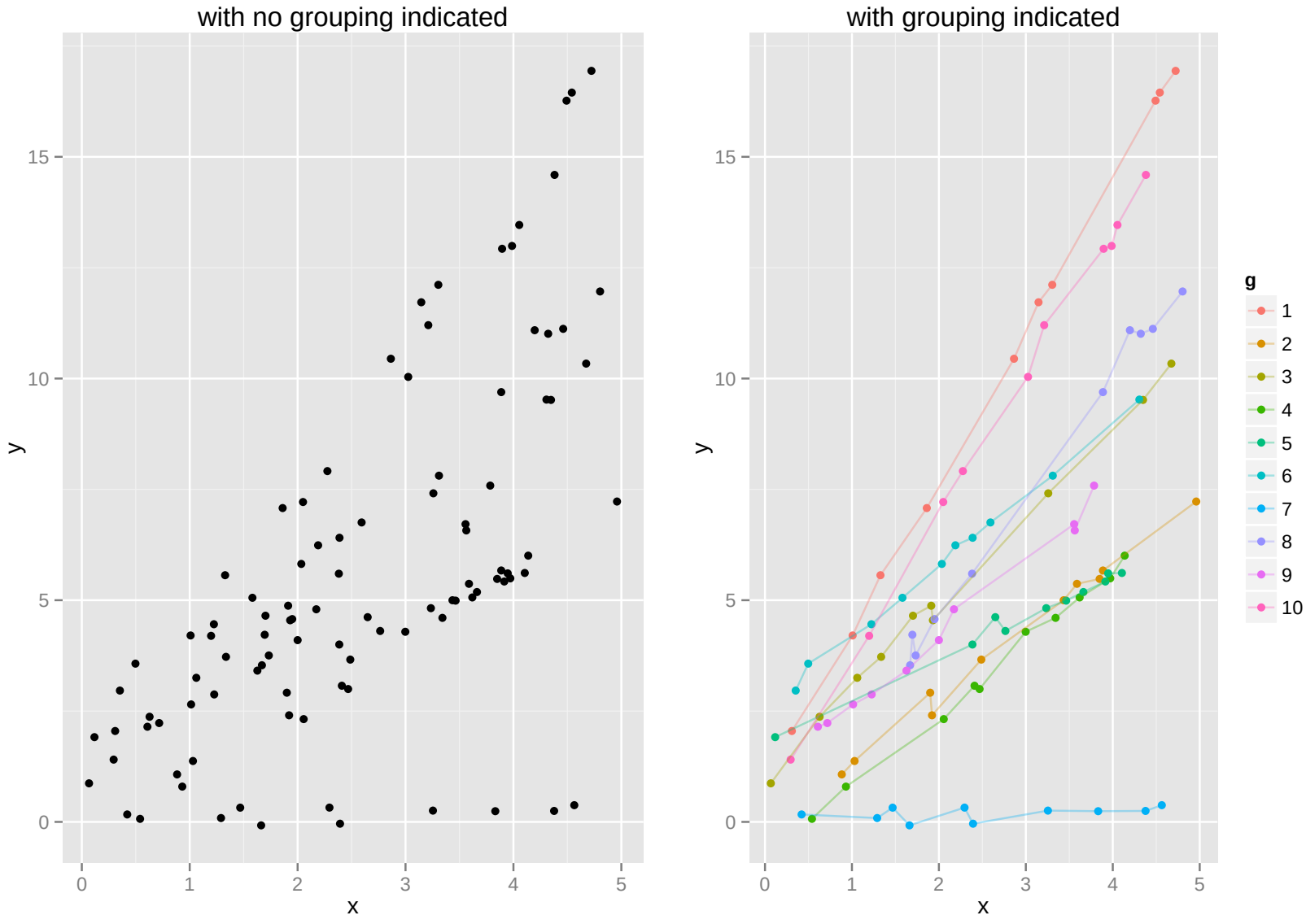
```
VarCorr(lme2) # D and R estimates
```

```
##  Groups   Name Std.Dev.
##  g        x    0.820
##  Residual      0.189
```

# Visualizing the fit



# Random intercepts and slopes



# Fit LME model for random intercept and slope

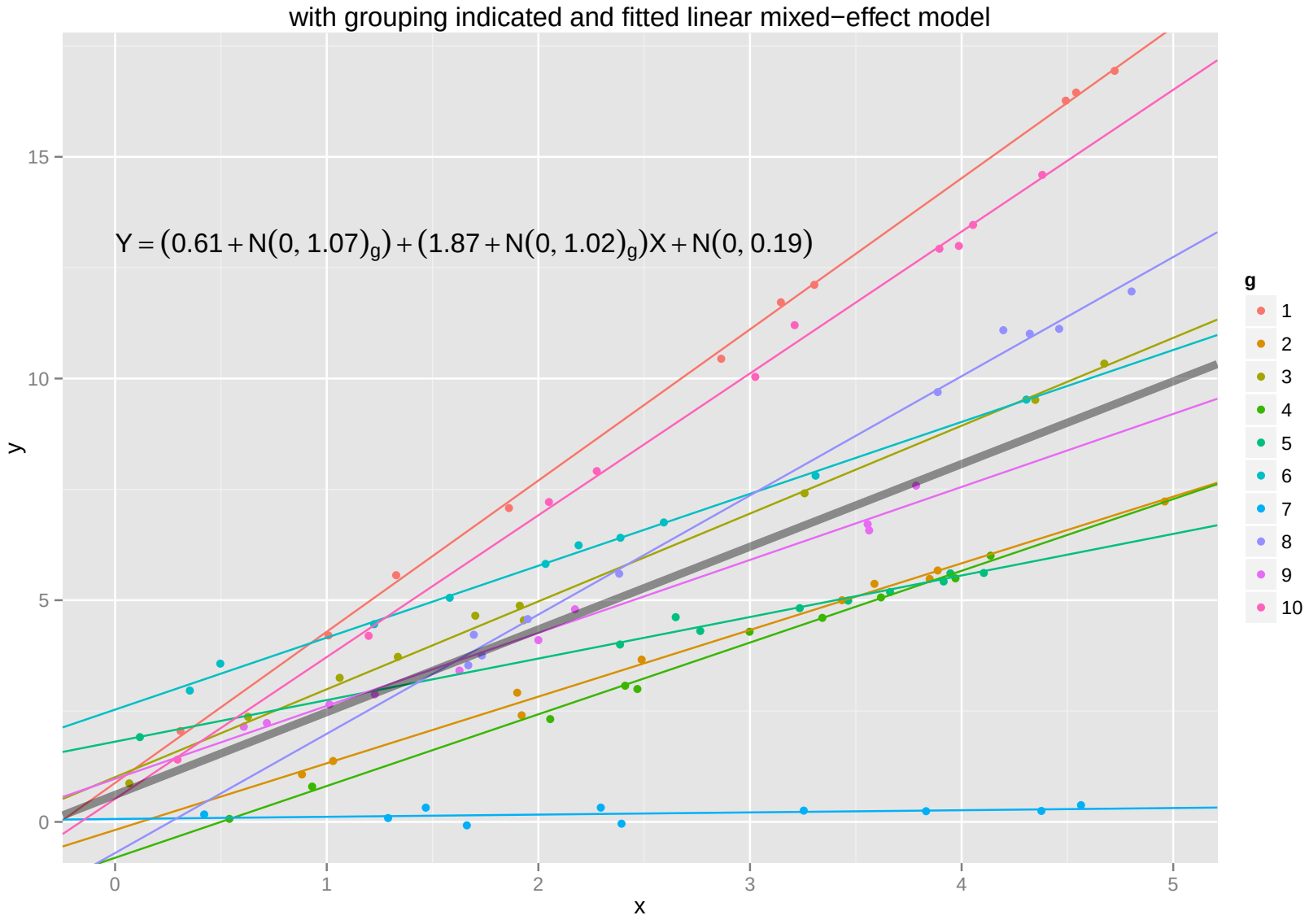
```
lme3 <- lmer(y ~ x + (x | g), data = dat)
fixef(lme3)
```

```
## (Intercept)           x
##      0.6097      1.8654
```

```
VarCorr(lme3)
```

```
##  Groups   Name        Std.Dev. Corr
##  g        (Intercept) 1.071
##           x           1.018    -0.08
##  Residual             0.181
```

# Visualizing the fit



# Packages for fitting LME

The two most commonly used R packages for fitting linear mixed-effect models are `nlme` and `lme4`.

- **`nlme`**: older package, comes with R, very stable; can fit linear and non-linear mixed-effect models; allows fitting of various covariance structures for random effects and residual errors.
- **`lme4`**: newer package, does not come with R; can fit linear, generalized linear, and nonlinear mixed-effect models; can also fit models with *crossed random effects*; does not currently allow fitting various covariance structures for residual errors; uses different computations than `nlme` that makes it better for larger data.

# lme4

For today's workshop we'll use `lme4`.

The syntax is a little easier to learn than `nlme`.

It's worth repeating: `lme4` does not currently provide facilities for modeling different variance-covariance structures for residuals, $\boldsymbol{R_i}$. Residual errors are assumed to be Normally distributed with mean 0 and variance $\sigma^2$.

By the way, the "4" in `lme4` refers to the technical fact that `lme4` was programmed using the S4 system in R.

# Data in Long Format

In R, LME data need to be in *long* format. That is, we have one record per subject per each measure of the dependent variable.

```
##   Reaction Days Subject
## 1    249.6    0     308
## 2    258.7    1     308
## 3    250.8    2     308
## 4    321.4    3     308
## 5    356.9    4     308
## 6    414.7    5     308
```

The dependent variable is `Reaction`. Notice we have one record per subject per each measure of `Reaction`. Here, `Subject` is the random effect.

# Fitting a model with `lme4`

Assuming one predictor and one level of grouping, say repeated measures on subjects.

**Random intercept**:

```
lme01 <- lmer(dv ~ x1 + (1 | g), data=df)
```

**Random slope**:

```
lme02 <- lmer(dv ~ x1 + (-1 + x1 | g), data=df)
```

**Correlated random slope and intercept**:

```
lme03 <- lmer(dv ~ x1 + (x1 | g), data=df)
```

**Uncorrelated random slope and intercept**:

```
lme04 <- lmer(dv ~ x1 + (x1 || g), data=df)
```

Of course we can have multiple predictors and random effects.
(NOTE: the ||-syntax works only for numeric predictors)

# Fitting a model with `lme4`

Assuming one predictor and two levels of grouping, say teachers within schools:

**Random intercept**:

```
lme01 <- lmer(dv ~ x1 + (1 | sch/tch), data=df)
```

**Or:** `lme01 <- lmer(dv ~ x1 + (1 | sch) + (1 | sch:tch), data=df)`

**Random slope**:

```
lme02 <- lmer(dv ~ x1 + (-1 + x1 | sch/tch), data=df)
```

**Correlated random slope and intercept**:

```
lme03 <- lmer(dv ~ x1 + (x1 | sch/tch), data=df)
```

**Uncorrelated random slope and intercept**:

```
lme04 <- lmer(dv ~ x1 + (x1 || sch/tch), data=df)
```

Again we can have multiple predictors and random effects. (NOTE: the ||-syntax works only for numeric predictors)

# Selected `lme4` extractor functions

Once we fit a model, `lme4` has several functions for extracting and viewing model information:

- `summary()` - summary of fitted LME
- `fixef()` - estimated fixed effect coefficients, $\beta$
- `ranef()` - predicted random effects, $\boldsymbol{b_i}$
- `coef()` - coefficients for *each group*, $\beta$ and $\beta + \boldsymbol{b_i}$
- `VarCorr()` - estimated variance parameters, $\boldsymbol{D}$ and $\boldsymbol{R_i}$
- `resid()` - residuals, or estimated random errors, $\epsilon_i$

In addition, the function `plot(allEffects())` from the `effects` package can quickly help us visualize the model.

Let's go to R!

# Why no p-values in the `lme4` summaries?

Recall that p-values in the coefficient summary of fitted linear models are the probability of getting a test statistic as large (or larger) if the coefficient was indeed 0.

Also recall that p-values are determined using null reference distributions. Under the null hypothesis, the test statistic has a known distribution.

In LME models, the null reference distribution is technically not known, at least not for unbalanced data. Thus the `lme4` authors elected to not output p-values based on a distribution that is not actually the distribution of the test statistic.

# How do I know if a coefficient is "significant"?

Quick, common-sense way: look at the `t value`.

If absolute value of `t value` is greater than 2, then the coefficient is likely significant. In other words, the coefficient estimate is more than 2 standard errors away from 0.

Note:
`t value` = `Estimate` / `Std. Error`

Of course ask yourself if the result is practically significant as well.

# How do I know if a coefficient is "significant"?

Another way is to compute 95% confidence intervals using the `confint()` function. If the interval contains 0, it is not significant. Further, confidence intervals give an indication of coefficient size and variability.

Say your fitted model object is `lme1`. Two ways to compute confidence intervals are as follows:

`confint(lme1)`

*computes a likelihood profile and finds the appropriate cutoffs based on the likelihood ratio test*

`confint(lme1, method="boot")`

*parametric bootstrapping ($B = 500$) with confidence intervals computed from the bootstrap distribution*

# But I want p-values!

The `lmerTest` package provides the kind of p-values SAS provides (based on Satterthwaite's approximations). Just load it, run `lmer` as usual and call `summary` on the object.

```
library(lmerTest)
m <- lmer(dv ~ x1 + (x1 | g), data=df)
summary(m)
```

A column of p-values is included in the summary output. Again, they're *approximate*.

Let's go back to R!

# Diagnostic plots

Once we fit a LME, we should assess our assumptions:

1. within-group errors (ie, residuals) are normally distributed, centered at 0 and have constant variance
2. random effects are normally distributed, centered at 0 and have constant variance

`lme4` provides a `plot()` method to help graphically check the variance assumptions of residuals. Say you have a fitted model object named `lme1`. The basic syntax is as follows:

```
plot(lme1, form=)
```

Where the `form=` argument is an optional formula specifying the desired type of plot.

# Examples of the `form=` argument

Say we fit the following LME:

```
lme1 <- lmer(y ~ x + (x | id), data=df)
```

**standardized residuals vs fitted values**:

```
plot(lme1)
```

**standardized residuals versus fitted values by x**:

```
plot(lme1, form = resid(.) ~ fitted(.) | x)
```

**box-plots of residuals by id**:

```
plot(lme1, form = id ~ resid(.))
```

# Diagnostic plots continued

**To check constant variance of random effects**:

```
plot(ranef(lme1))
```

*Note: if only one random effect was fit, this produces a qq plot.*

**To assess normality of residuals**:

```
qqnorm(resid(lme1))
```

**To assess normality of random effects**:

```
lattice::qqmath(ranef(lme1))
```

# Two other plots of interest

**plot predicted random effects for each level of a grouping factor**

```
lattice::dotplot(ranef(lme1))
```

Also known as a caterpillar plot. Good for checking if there are levels of a grouping factor with extremely large or small predicted random effects.

**To assess model fit**

```
plot(lme1, y ~ fitted(.) | id, abline = c(0,1))
```

Points lying on a diagonal line can provide indication of a good model fit.

Let's go back to R!

# Making predictions using model

The predict function allows you to make predictions with and without random effects.

**with random effects**

```
predict(lme1)
```

**without random effects; also known as marginal predictions**

```
predict(lme1, re.form=NA)
```

From the lme4 documentation for predict: "*There is no option for computing standard errors of predictions because it is difficult to define an efficient method that incorporates uncertainty in the variance parameters.*"

# Bootstrap predictions

Bootstrapping allows us to resample our data (with replacement), fit the same model, and make predictions.

If we bootstrap 1000 times, this means resample our data 1000 times, each time fitting a model and making predictions.

When done, we'll have 1000 predictions, which we can use to estimate a confidence interval.

The `bootMer` function in `lme4` automates some of this for us.

# Using the bootMer function

Let's say we fit the following lme4 model and want to make marginal predictions for thiouracil at week 5 with a confidence interval.

```
lmm1 <- lmer(wt ~ treat + weeks + (1 | subject),
             data=ratdrink)
nd <- data.frame(treat="thiouracil", weeks=5)
```

The following will perform 1000 bootstrap predictions and return a percentile CI.

```
b.out <- bootMer(x = lmm1,
                 FUN = function(x)predict(x, newdata=nd,
                                          re.form=NA),
                 nsim = 1000, .progress = "txt")
quantile(b.out$t, probs = c(0.025, 0.975))
```

Let's go back to R!

# REML vs. ML

If you look closely at the summary output for a model fit with `lme4`, you'll see the message: `Linear mixed model fit by REML`.

REML stands for Restricted Maximum-Likelihood. It provides unbiased estimates for the variance parameters.

By setting `REML=FALSE` in the `lmer()` function, we can estimate parameters using ML, Maximum Likelihood. But the variance estimates will be biased downwards.

The choice of ML vs REML affects model comparisons.

# Model Comparisons

We often compare models to see if a smaller model fits as well as a larger model.

We can compare models either by hypothesis testing or by selection criteria (such as AIC or BIC).

When it comes to hypothesis testing, the choice of ML vs REML estimation is important:

> *To compare two models fit by REML, each must have the same fixed effects.*
>
> *To compare two models fit by ML, one must be nested within the other.*

The function to compare models is `anova()`. Its default behavior is to refit the models with ML before comparing. Specify `refit=FALSE` to suppress refitting.

# Comparing nested models

Say we fit two models:

```
lme1 <- lmer(y ~ x + z + x:z + (1 | g), data=df)
lme2 <- lmer(y ~ x + z + (1 | g), data=df)
```

The following refits the models with ML and compares them via Likelihood Ratio hypothesis test:

```
anova(lme1, lme2)
```

Notice that `lme2` is nested in `lme1`. That is, `lme2` is a special case of `lme1` with interaction coefficient(s) for `x:z` equal to 0.

The null hypothesis states the models are equal. Rejecting the null means we prefer the larger model.

# Model comparison with AIC

AIC (Akaike Information Criterion) is a criterion-based approach to model building. It is a measure of "goodness" we try to optimize. We want to minimize AIC.

To see AIC:
`extractAIC(lme1)`

Among several models, the one with lower AIC is usually preferred. This approach requires *no hypothesis testing*, involves no p-values, and doesn't care whether you used REML or ML.

Is it better than hypothesis testing? That's up to you. Whatever you do, expect to do some experimentation and iteration to find better models. Also expect to use a great deal of subjective judgment.

# Comparing models with different random effects

Random effect parameters are measures of variances. Variance is greater than or equal to 0.

When testing whether or not to include a random effect in a model, we're testing that its variance is 0. In that case our hypothesis test involves values lying on the boundary of the parameter space.

The result is that the standard hypothesis test (ie, a likelihood ratio test) is *conservative*. The p-value is too high.

If you have a small p-value (say $< 0.001$), that's not a problem.

If you have a p-value close to significance, (say about 0.10) you may want to consider calculating a corrected p-value using a *mixture of chi-square distributions*.

# A mixture of chi-square distributions

A likelihood ratio test (LRT) statistic has a chi-square distribution with degrees of freedom equal to the difference in parameters between two models.

Let's say our LRT is on 2 degrees of freedom. The null distribution of this test statistic is NOT a chi-square with 2 degrees of freedom since our null value (variance $= 0$) is on the boundary of the parameter space.

It's been suggested that a 50:50 mixture, $0.5\chi^2_{df} + 0.5\chi^2_{df-1}$, can serve as a reference null distribution for computing the p-value. But this is still only an approximation.

We'll demonstrate this in the R script. Let's go!

# References


Faraway, J. (2006). *Extending the Linear Model with R*. Chapman and Hall/CRC.

Fox, J. & Weisberg, S. (2015). *Mixed-Effects Models in R: an appendix to An R Companion to Applied Regression*.

Galecki, A. and Burzykowski T. (2013). *Linear Mixed-Effect Models Using R*. Springer.

Pinheiro, J. & Bates, D. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer.

West, B., Welch, K., & Galecki, A. (2015) *Linear Mixed Models*. Chapman and Hall/CRC.

GLMM FAQ:
http://bbolker.github.io/mixedmodels-misc/glmmFAQ.html

# Thanks for coming today!

For help and advice with statistics, contact us to set up an appointment: statlab@virginia.edu

Sign up for more workshops or see past workshops: http://data.library.virginia.edu/training/

Register for the Research Data Services newsletter: http://data.library.virginia.edu/newsletters/