Introduction to Linear Mixed Effects Modelling in R

Jacob Passfield

The following section works through the example given at *https://ourcodingclub.github.io/tutorials/mixed-models/*. The example serves as an introduction to the concepts of linear mixed effects modelling and the corresponding code syntax in R.

1 What is mixed effects modelling and why does it matter?

Ecological data is often complex. There may be different grouping factors (populations, species, site where data was collected). Sample sizes may be too small, which is a difficulty when trying to fit complicated models with many parameters. Finally our data points may not be independent (using quadrats within sites to collect data adds structure to our data since quadrats are nested within sites).

Mixed models deal with messy data; use all the data (regardless of the sample size, the structure of data and whether we have to fit many covariates) and saves degrees of freedom compared to standard linear models.

This document will over cover linear mixed models.

2 Exploring the data

We focus on a ficticious study system about dragons. We decided to train dragons and collected data on dragon intelligence `testScore`. Individuals with a range of body lengths were sampled across three sites in eight different moutain ranges.

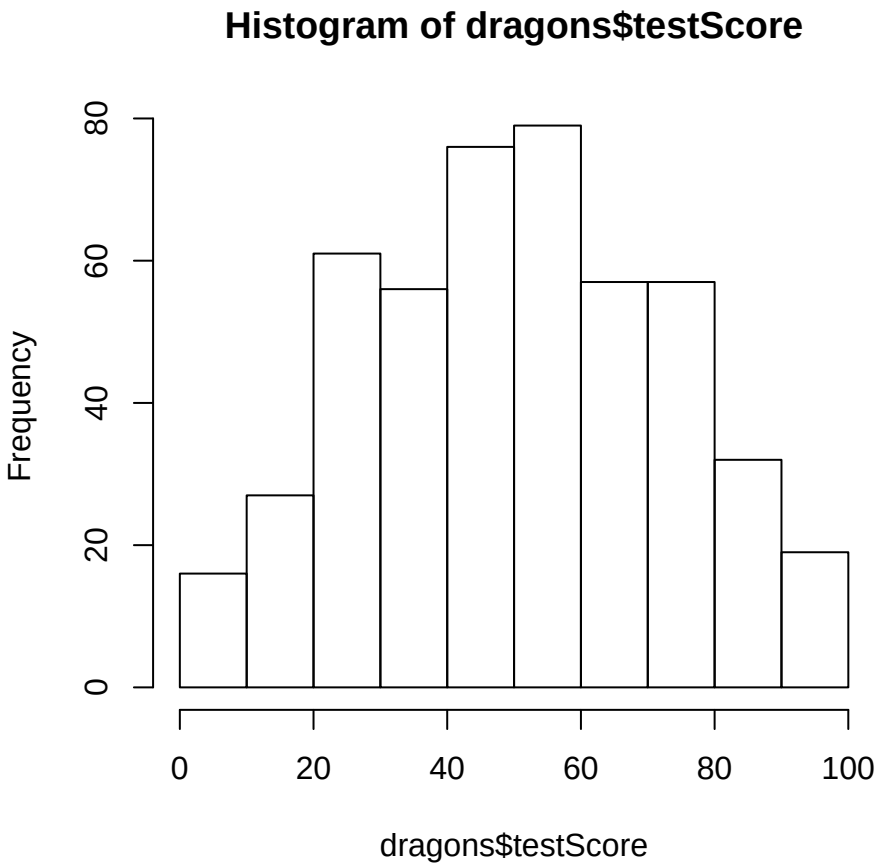
```
load("_data/dragons.RData")
head(dragons)
```

```
##   testScore bodyLength mountainRange  X site
## 1 16.147309   165.5485      Bavarian NA    a
## 2 33.886183   167.5593      Bavarian NA    a
## 3  6.038333   165.8830      Bavarian NA    a
## 4 18.838821   167.6855      Bavarian NA    a
## 5 33.862328   169.9597      Bavarian NA    a
## 6 47.043246   168.6887      Bavarian NA    a
```

We investigate how body length affects the test scores of the dragons.

The distibution of the response variable `testScore` is approximate to a normal distribution. We do not need to worry about the distribution of explanatory variables.

```
hist(dragons$testScore)
```



Standardising explantory variables (`bodyLength`) to have a mean of zero (centering) and standard deviation of one (scaling) ensures the estimated coefficients are all on the same scale so it is easier to compare effect sizes.

```
dragons$bodyLength2 <- scale(dragons$bodyLength, center=T, scale=T)
```

`scale()` centers the data (column mean subtracted from the values in the column) and scales it (centered column values divided by the column's standard deviation).

3 Fitting all the data into one analysis

Fitiing a linear model to our data (ignoring different sites and moutain ranges) is one way to analyse this data.

In the follwoing model `testScore` is the response and `bodyLength2` is the predictor (explanatory variable).

```
basic.lm <- lm(testScore ~ bodyLength2, data = dragons)
# lm(formula, data)
# formula here is reponse ~ predictor
# ~ seperates the left- and right-hand sides in a model formula
# the left-hand side is optional and one-sided formulae are used in some contexts
summary(basic.lm)
```

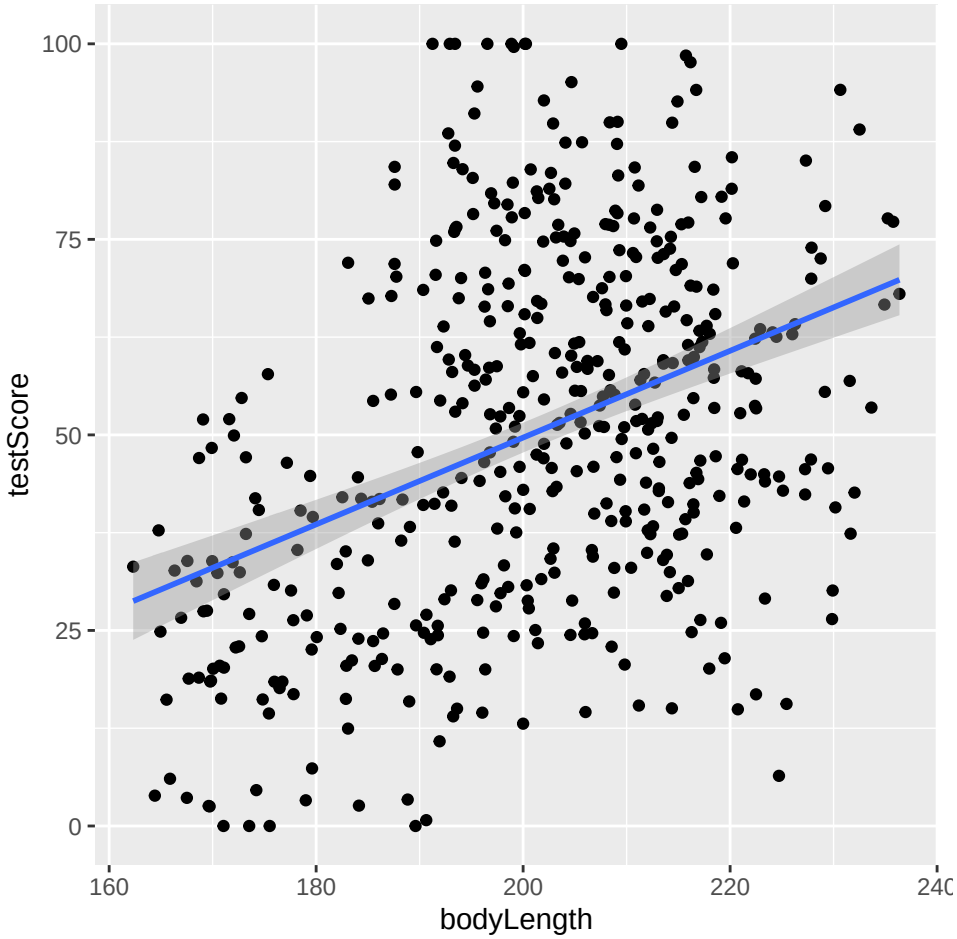
```
##
## Call:
## lm(formula = testScore ~ bodyLength2, data = dragons)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -56.962 -16.411  -0.783  15.193  55.200
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  50.3860     0.9676  52.072   <2e-16 ***
## bodyLength2   8.9956     0.9686   9.287   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.2 on 478 degrees of freedom
## Multiple R-squared:  0.1529, Adjusted R-squared:  0.1511
## F-statistic: 86.25 on 1 and 478 DF,  p-value: < 2.2e-16
```

Plotting data.

```
library(ggplot2)
(prelim_plot <- ggplot(dragons, aes(x=bodyLength, y=testScore))
   + geom_point() + geom_smooth(method="lm"))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# () around the whole ggplot code creates and shows the graph without the added step of
# typing prelim_plot after
# method="lm" fits a linear model
```

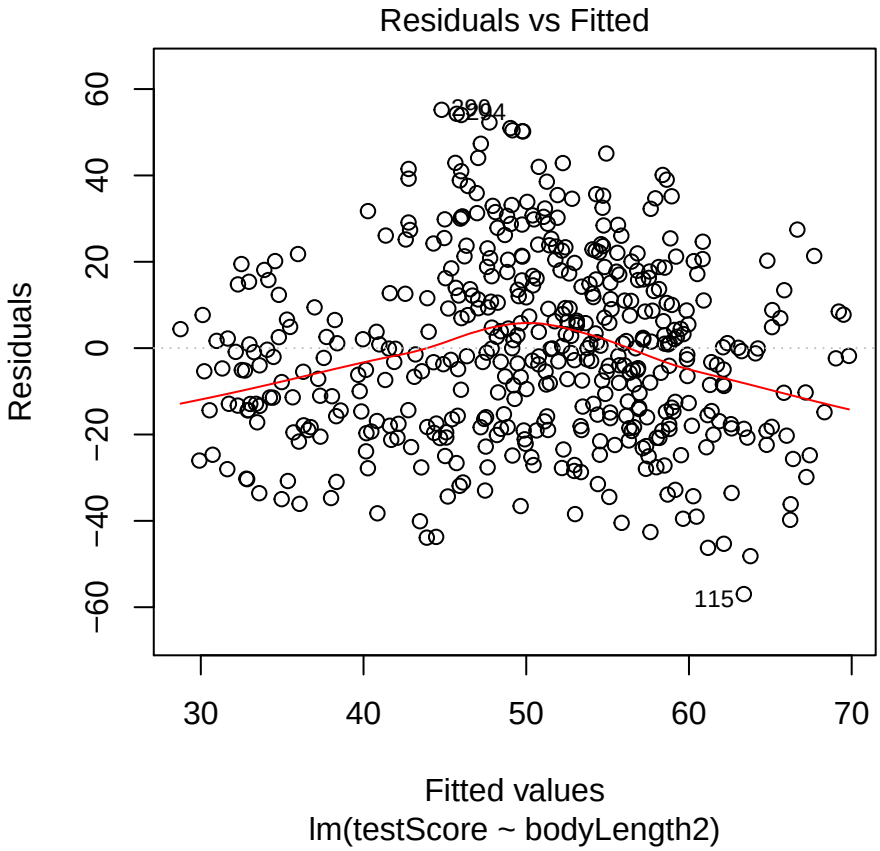
```
# geom_smooth() uses formula y ~ x
# geom_smooth() adds a regression line
# the grey zone is the 95% confidence level interval for predictions from "lm"
```

From the linear model and the plot, it appears that bigger dragons do better in the intelligence test. This is odd because size shouldn't affect the test scores.

Are the assumptions met?

Plotting the residuals should yield a red line that is nearly flat.

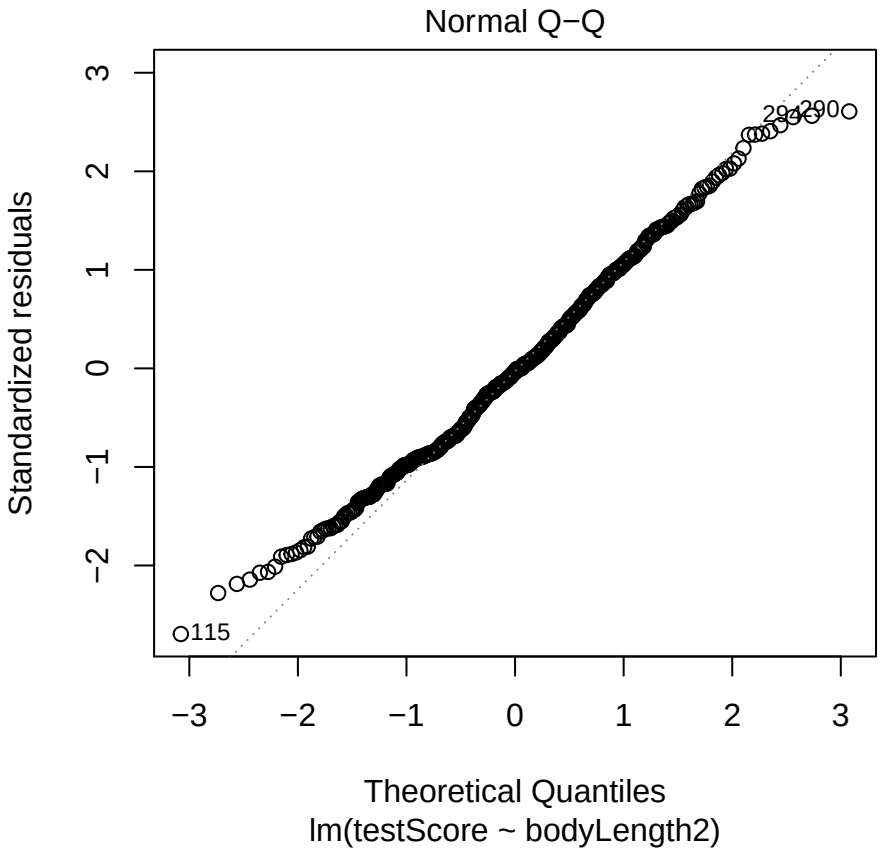
```
plot(basic.lm, which=1)
```




```
# which=1: a plot of residuals against fitted values
# line not perfectly straight but something they go along with in this example
# for own data be careful
# the bigger the sample size, the less of a trend expected
```

For a Q-Q plot the points should ideally fall along the diagonal dashed line.

```
plot(basic.lm, which=2)
```

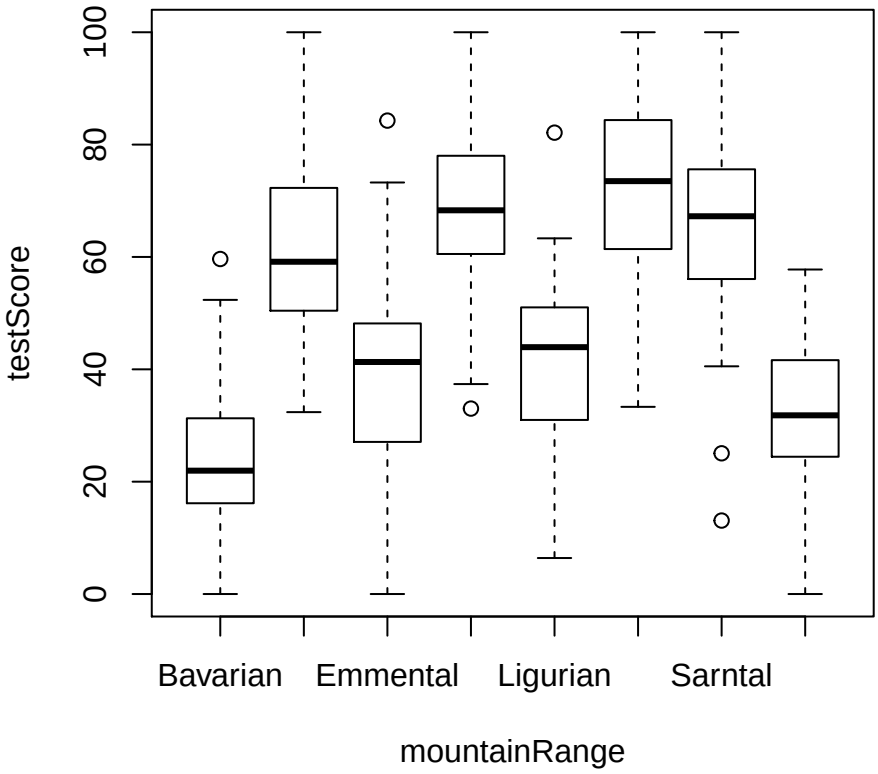



```
# which=2: a normal Q-Q plot
# points off at the extremes but happens often
# not too bad
```

Is our data independent?

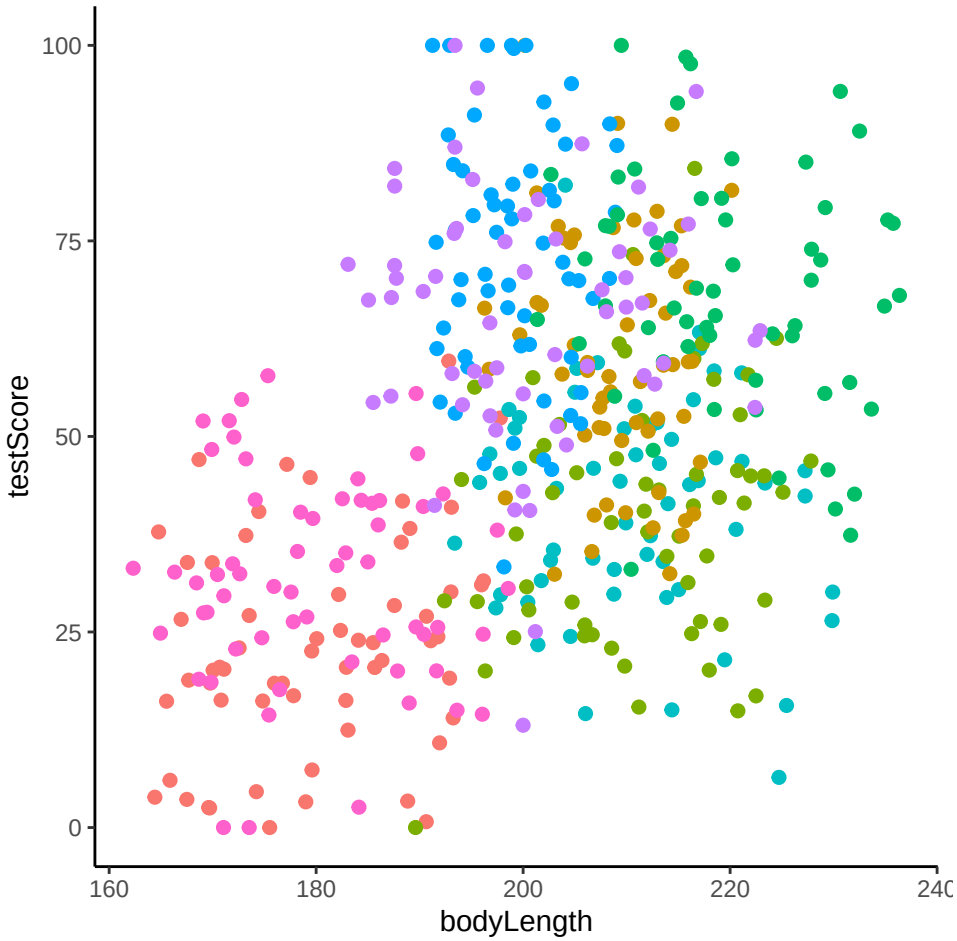
Since the data collected multiple samples from eight mountain ranges, it's expected that data from within each mountain range are more similar to each other than the data from different mountain ranges, meaning the samples are correlated.

```
# checking to see if data is correlated
boxplot(testScore ~ mountainRange, data = dragons)
```



Plotting and colouring points by mountain range.

```
(colour_plot <- ggplot(dragons, aes(x=bodyLength, y=testScore, colour=mountainRange)) +
    geom_point(size=2) + theme_classic() + theme(legend.position="none"))
```

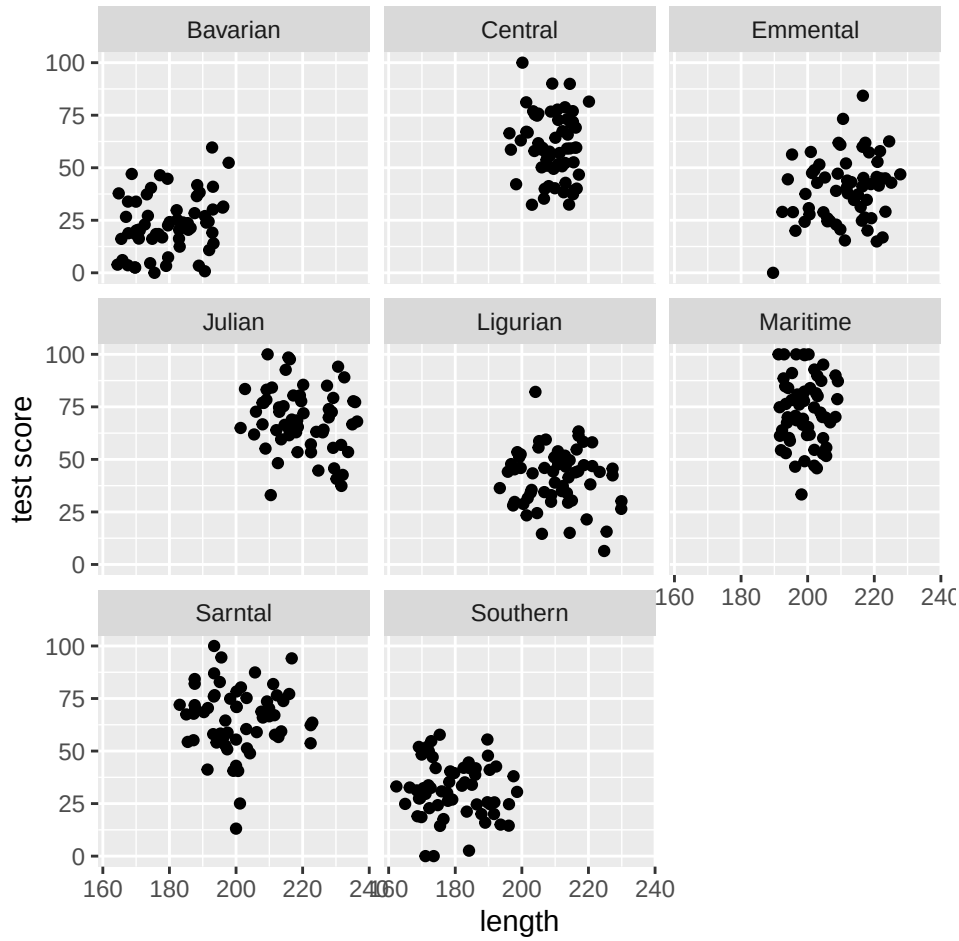


These two plots show that moutain ranges vary both in dragon body length and in their test scores. This confirms that our observations from within each of the ranges aren't independent. Ignoring this fact could lead to inaccurate conculsions and so we continue.

4 Running multiple analyses

We could run many seperate analyses and fit a regression for each of the mountain ranges.

```
# data split by mountain range
(split_plot <- ggplot(aes(bodyLength, testScore), data=dragons) + geom_point() +
    facet_wrap(~mountainRange) + xlab("length") + ylab("test score"))
```



We have eight analyses fogetting we have different sites in each mountain range, which aren't independent either. We could run an analysis for each site in each range separately.

We'd have to estimate a slope and intercept parameter for each regression. That's 2 parameters, 3 sites and 8 mountain ranges, which means 48 parameter estimates ($2*3*8 = 48$) and the sample size for each analysis would bo only 20 dragons per site.

Doing this severely decreases our sample sample whilst increasing chances of a Type 1 Error (falesly rejecting the null hypothesis) by carrying out multiple comparions.

5 Modifying the current model

We want to use all the data, but account for data coming from different mountain range (considering sites on hold for now).

```
# adding mountain range as a fixed effect to our basic.lm
mountain.lm <- lm(testScore ~ bodyLength2 + mountainRange, data=dragons)
summary(mountain.lm)
```

```
##
## Call:
## lm(formula = testScore ~ bodyLength2 + mountainRange, data = dragons)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -52.263  -9.926   0.361   9.994  44.488
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              23.3818     2.5792   9.065  < 2e-16 ***
## bodyLength2               0.2055     1.2927   0.159  0.87379
## mountainRangeCentral     36.5828     3.5993  10.164  < 2e-16 ***
## mountainRangeEmmental    16.2092     3.6966   4.385 1.43e-05 ***
## mountainRangeJulian      45.1147     4.1901  10.767  < 2e-16 ***
## mountainRangeLigurian    17.7478     3.6736   4.831 1.84e-06 ***
## mountainRangeMaritime    49.8813     3.1392  15.890  < 2e-16 ***
## mountainRangeSarntal     41.9784     3.1972  13.130  < 2e-16 ***
## mountainRangeSouthern     8.5196     2.7313   3.119  0.00192 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.96 on 471 degrees of freedom
## Multiple R-squared:  0.5843, Adjusted R-squared:  0.5773
## F-statistic: 82.76 on 8 and 471 DF,  p-value: < 2.2e-16
```

Now body length is not significant. The model above is estimating the difference in test scores between the mountain ranges. But we are not quantifying test scores for each specific mountain range: want to know whether body length affects test scores and so we want to control the variation coming from mountain ranges or, the random effects coming from mountain ranges. Time for mixed effects models.

6 Mixed effects models

A mixed model allows us to use all the data (higher smaple size) and account for the correlations between data coming from the sites and mountain ranges. Also, we'll estimate fewer parameters and avoid porblems with multiple comparisons that we'd encounter while using seperate regressions.

```
library(lme4)
```

6.1 Fixed and random effects

The difference between them has little to do with the variables themselves and more to do with your research question.

Fixed effects are variables that we expect will have on the response variable. In our example, we want to know how dragon body length impacts the dragon's test score. So body length is a fixed effect and test score is the dependent variable.

Random effects are usually grouping factors that we are trying to control. They're always categorical. We are not specifically interested in their impact on the response variable, but we know that they might be inlfuencing the patterns we see.

The data for our random effect is just a sample of all the possibilities. We can't collect data for every mountain where dragons live (time-consuming) so we generalise results to a whole population based on representative sampling. We don't care about estikating how much better pupils in school A have done to school B, but since their respective teachers might be a reason why their scores would be different, we'd like to account for this variation when we predict scores for pupils in school Z.

In our example, we are looking to control for the effects of mountain range. We have sampled only 8 so our data is just a sample of all the exisiting mountain ranges. Although we don't want to know the effect each specific mountain range has on the test score (we want our model to generalise to dragons from other mountain ranges) we know test scores from within ranges might be correlated so we want to control that.

6.1.1 More about random effects

Golden rule: want your random effects to have at least five levels. So if we wanted to control the effects of a dragon's sex on intelligence, we would fit sex as a fixed effect, not random (sex: male or female, a two level factor).

Estimating on few data points is very imprecise and so even though you could, there wouldn't be a lot confidence in it.

Random doesnt have much to do with mathematical randomness, but more to do with the grouping of variables. It's about making our models representative of our questions and getting better estimates.

Ask yourself: what are you trying to do? What are you trying to make predictions about? What is just variation (noise) that you need to control for?

6.2 Fitting our first mixed model

The response varibale is `testScore` and we are attempting explain part of the variation in test score through fitting body length as a fixed effect. However the response variable has some residual (unexplained) variation associated with mountain ranges. We model that unexplained variation through variance by using random effects.

Variation: dispersion. Variance: quantifies the dispersion.

Research question changes slightly to: is there an association between dragon's body length and the test score *after* controlling for the variation in mountain ranges?

```
# fit the random effect using (1|variableName)
mixed.lmer <- lmer(testScore ~ bodyLength2 + (1|mountainRange), data=dragons)
summary(mixed.lmer)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: testScore ~ bodyLength2 + (1 | mountainRange)
##    Data: dragons
##
## REML criterion at convergence: 3985.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.4815 -0.6513  0.0066  0.6685  2.9583
##
## Random effects:
##  Groups        Name        Variance Std.Dev.
##  mountainRange (Intercept) 339.7    18.43
##  Residual                  223.8    14.96
## Number of obs: 480, groups:  mountainRange, 8
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  50.3860     6.5517   7.690
## bodyLength2   0.5377     1.2750   0.422
##
## Correlation of Fixed Effects:
##             (Intr)
## bodyLength2 0.000
```

The random effect part tells you how much variance you find among levels of your grouping factor(s), plus the residual variance.

The fixed effect part is very similar to a linear model output: intercept and error; slope estimate and error.

Once mountain ranges is accounted for, it's obvious that dragon body length doesn't explain the differences in the test scores. Notice the model estimate is smaller than its(associated error. This means that the effect (slope) cannot be distinguised from zero.

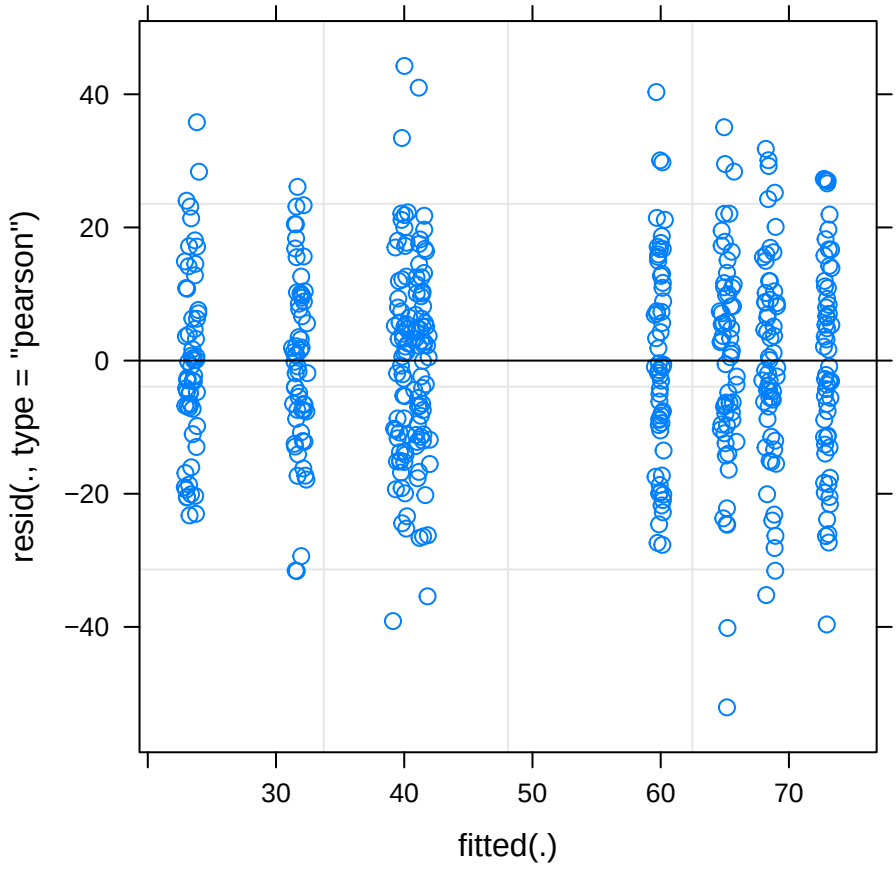
The random effect of the moutain range is meant to capture all the influences of moutain ranges on dragon test scores, regardless of whether we observe those influences explicity or not, or whether those influences are big or small or whether many tiny influences combined affect test scores, that's what we try to control for.

The variance for `mountainRange` = 339.7. Mountain ranges are clearly important: they explain a lot of variation. Variance of `mountainRange` divided by the total variance, $339.7/(339.7 + 223.8)$ is approximately `60%`.

The differences between mountain ranges explain `~60%` of the variance left over after the varaince is explained by our fixed effects.

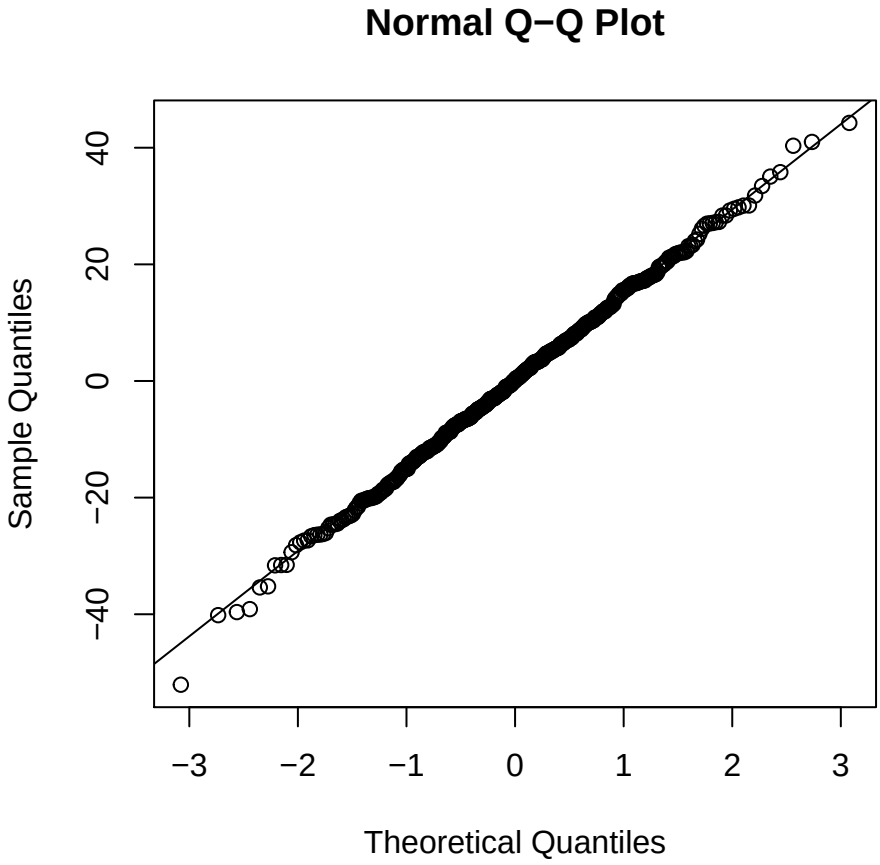
It's good practice to look at plots to check our assumptions.

```
plot(mixed.lmer)
```




```
# looks alright, no patterns evident
```

```
qqnorm(resid(mixed.lmer))
qqline(resid(mixed.lmer))
```




```
# points fall nicely onto the line
```

6.2.1 Types of random effects

A factor is any categorical independent variable.

Above we used `(1|mountainRange)` to fit our random effect. The factor is on the right of the `|` operator and is referred to as a grouping factor.

Random effects (factors) can be crossed or nested, depending on the relationship between the variables.

6.2.1.1 Crossed random effects

There are *hierarchical linear models* or *multilevel models*, but while all HLMs are mixed models, not all mixed models are hierarchical because you can have crossed (or partially crossed) random factors that do not represent levels in a hierarchy.

In our example, we monitor dragons (subject) across different mountain ranges (context). Since dragons can fly, we might observe the same dragon across different mountain ranges, but also, we might not see all the dragons visiting all the mountain ranges. Therefore, we can potentially observe every dragon in every mountain range (crossed) or observe observe some dragons across some of the mountain ranges (partially crossed). We then fit the identity of the dragon and mountain range as (partially) crossed random effects.

As a different example, an effect is (fully) crossed when all the subjects have experienced all the levels of that effect. For instance, if you had a fertilisation experiment on seedlings growing in a seasonal forest and took repeated measurements over time (say 3 years) in each seasons, you may want to have a crossed factor called `season` (Summer 1, Summer 2, Summer 3, Autumn 1, Autumn 2, …), ie. a factor for each season of each year. This grouping factor would account for the fact that all plants in the experiment, regardless of the fixed (treatment) effect (ie. fertilised or not), may have experienced a very hot summer in the second

year, or a very rainy spring in the third year, and those conditions could cause interference in the expected patterns.

You don't even need to have associated climate data to account for it! You just know that all observations from spring 3 may be more similar to each other because they experienced the same environmental quirks rather than because they're responding to your treatment.

lme4 handles partially and fully crossed factors well.

6.2.1.2 Nested random effects

Think Russian nesting dolls. We call these models hierarchical; often there's an element of scale or sampling stratification in there.

Using our fertilisation experiment. Say you have 50 seedlings in each bed, with 10 control and 10 experimental beds, 1000 seedlings altogether. Say you datat collecting once in each season in each of the 3 years. On each plant you measure the length of 5 leaves. So: 5 leaves $*$ 50 plants $*$ 20 beds $*$ 4 seasons $*$ 3 years $=$ 60000 measurements.

If you were to run analysis using a simple linear regression ('leaflength ~ treatment) you'd commit pseudoreplication (increasing your sampling size by using non-independent data). With a sampling size of 60000 you'd certainly get a significant effect of treatment which may not have any ecological meaning at all. It violates the assumption of independence of observations that is central to linear regression.

Now the nesting dolls come in. Leaves within a plant and plants within a bed may be more similar to each other (for genetic and environmental reasons). We can therefore add a random effect structure to account for this nesting: `leafLength ~ treatment + (1|Bed/Plant/Leaf)`.

This way the model will account for non independence in the data. The same leaves have been sampled repeatedly, mulitple leaves were measured on an individual and plants are grouped into beds which may recieve different amounts of sun.

With the crossed effects, for example, all the leaves have been measured in all seasons then your model would become: `leafLength ~ treatment + (1|Bed/Plant/Leaf) + (1|Season)`.

6.2.1.2.1 Implicit versus explicit nesting

Avoid implicit nesting.

To tackle this: look at our study where we not only collected data across multiple mountain ranges, but across several sites too.

```
str(dragons)
```

```
## 'data.frame':    480 obs. of  6 variables:
##  $ testScore    : num  16.15 33.89 6.04 18.84 33.86 ...
##  $ bodyLength   : num  166 168 166 168 170 ...
##  $ mountainRange: Factor w/ 8 levels "Bavarian","Central",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ X            : logi  NA NA NA NA NA NA ...
##  $ site         : Factor w/ 3 levels "a","b","c": 1 1 1 1 1 1 1 1 1 1 ...
##  $ bodyLength2  : num [1:480, 1] -2.21 -2.08 -2.19 -2.07 -1.93 ...
##   ..- attr(*, "scaled:center")= num 201
##   ..- attr(*, "scaled:scale")= num 16.2
```

Just like mountain ranges, we have to assume that data collected within our sites might be correlated and so we should include sites as an additional random effect in our model.

Our site variable is a three-level factor, with sites called a, b and c. The nesting of the site within the mountain range is implicit (sites are meaningless unless assigned to specific mountain ranges). To avoid confusion, we call a new variable, that is explicitly nested, `sample`.

```
dragons <- within(dragons, sample <- factor(mountainRange:site))
head(dragons)
```

```
##   testScore bodyLength mountainRange  X site bodyLength2     sample
## 1 16.147309   165.5485      Bavarian NA    a   -2.206233 Bavarian:a
## 2 33.886183   167.5593      Bavarian NA    a   -2.082204 Bavarian:a
## 3  6.038333   165.8830      Bavarian NA    a   -2.185605 Bavarian:a
## 4 18.838821   167.6855      Bavarian NA    a   -2.074419 Bavarian:a
## 5 33.862328   169.9597      Bavarian NA    a   -1.934145 Bavarian:a
## 6 47.043246   168.6887      Bavarian NA    a   -2.012538 Bavarian:a
```

We have 24 samples (8 mountain ranges $*$ 3 sites) and not just 3. Our sample is a 24-level factor and we should use that instead of using site in our models because each site belongs to a specific mountain range.

To sum up: for nested random effects, the factor appears only within a particular level of another factor (each site belongs to a specific mountain range and only to that range); for crossed effects, a given factor appears in more than one level of another factor (dragons appearing within more than one mountain range). Or you can just remember that if your random effects aren't nested, then they are crossed!

6.3 Fitting our second mixed model

Based on the above, using the following specification would be **wrong**, as it would imply that there are only three sites with observations at each of the 8 mountain ranges (crossed).

```
mixed.WRONG <- lmer(testScore ~ bodyLength2 + (1|mountainRange) +
                      (1|site), data = dragons)
# treats the two random effects as if they are crossed
summary(mixed.WRONG)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: testScore ~ bodyLength2 + (1 | mountainRange) + (1 | site)
##    Data: dragons
##
## REML criterion at convergence: 3980.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.5079 -0.6489  0.0138  0.6976  3.0851
##
## Random effects:
##  Groups        Name        Variance Std.Dev.
##  mountainRange (Intercept) 409.90   20.246
##  site          (Intercept)  10.52    3.243
##  Residual                   219.19   14.805
## Number of obs: 480, groups:  mountainRange, 8; site, 3
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   50.386      7.430   6.782
## bodyLength2   -2.600      1.641  -1.584
##
## Correlation of Fixed Effects:
##             (Intr)
## bodyLength2 0.000
```

Where it says `Numbers of obs: 480...`, the grouping of obsevrations is wrong: only 3 sites when we've

actually sampled 24 different locations.

We can fit a new model, one that takes into account both the differences between the mountain ranges, as well as the differences between the sites within those mountain ranges by using our sample variable.

Research question changes slightly again: is there an association between dragon's body length and the test score *after* controlling for variation in mountain ranges and sites within mountain ranges?

```
mixed.lmer2 <- lmer(testScore ~ bodyLength2 + (1|mountainRange) + (1|sample),
                    data = dragons)
summary(mixed.lmer2)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: testScore ~ bodyLength2 + (1 | mountainRange) + (1 | sample)
##    Data: dragons
##
## REML criterion at convergence: 3970.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2425 -0.6752 -0.0117  0.6974  2.8812
##
## Random effects:
##  Groups        Name        Variance Std.Dev.
##  sample        (Intercept)  23.09    4.805
##  mountainRange (Intercept) 327.56   18.099
##  Residual                  208.58   14.442
## Number of obs: 480, groups:  sample, 24; mountainRange, 8
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   50.386      6.507   7.743
## bodyLength2    0.831      1.681   0.494
##
## Correlation of Fixed Effects:
##             (Intr)
## bodyLength2 0.000
```

The model recognises that there are 24 samples spread across 8 ranges.

Here, we try to account for all the mountain-range-level and all the sit-level influences and hope our random effects have soaked up all these influences so we can control for them in the model.

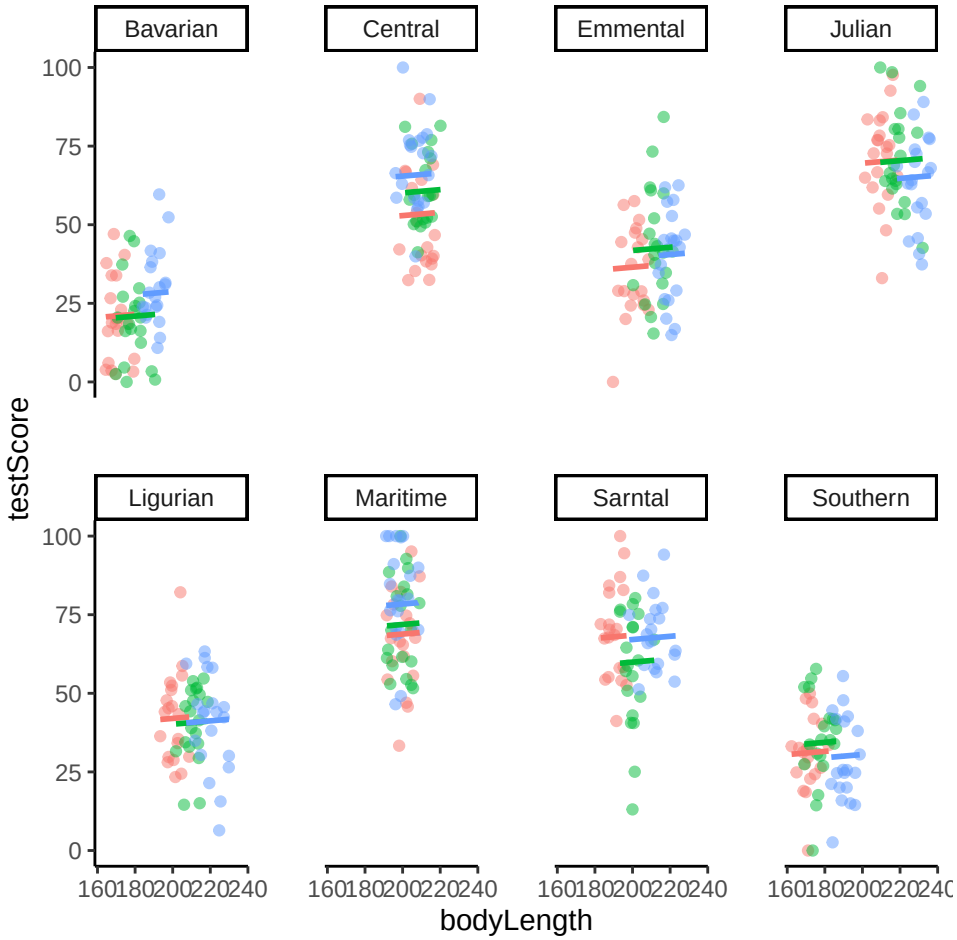
Could use the following syntax: `(1|mountainRange/site)` or even `(1|mountainRange) + (1|mountainRange:site)`.

It is advisable however to set out your variables properly and make sure nesting is stated explicitly within them, that way you don't have to remember to specify the nesting.

Plotting again we can see eight mountain ranges with three sites (different colour points) within them, with a line fitted through each site.

```
(mm_plot <- ggplot(dragons, aes(x=bodyLength, y=testScore, colour=site)) +
    facet_wrap(~mountainRange, nrow=2) +
   geom_point(alpha=0.5) +
   theme_classic() +
   geom_line(data=cbind(dragons, pred=predict(mixed.lmer2)), aes(y=pred), size=1) +
   # adding predicted line from mixed model
   theme(legend.position = "none", panel.spacing = unit(2,"lines"))
```

```
# adding space between panels
  )
```



6.3.1 Introducing random slopes

All the line on the above figure are parallel because we have only fitted random-intercept models. A random-intercept model allows the intercept to vary for each level of the random effects, but keeps the slope constant among them. In our example, using this model means that we expect dragons in all mountain ranges to exhibit the same relationship between body length and intelligence (fixed slope), although we acknowledge that some populations may be smarter or dumber to begin with (random intercept).

In life sciences, we often assume that not all populations show the exact same relationship, for instance if your study sites or populations are very far apart and have some relatively important environmental or genetic differences. Therefore, we often want to fit a random-slope and random-intercept model. Maybe the dragons in a very cold versus a very warm mountain range have evolved different body forms for heat conservation and may therefore be smart even if they're smaller than average.

We allow the random-slope and random-intercept by adding the fixed variable into the random effects brackets.

```
mixed.ranslope <- lmer(testScore ~ bodyLength2 + (1 + bodyLength2|mountainRange/site),
                       data = dragons)
```

```
## boundary (singular) fit: see ?isSingular
```

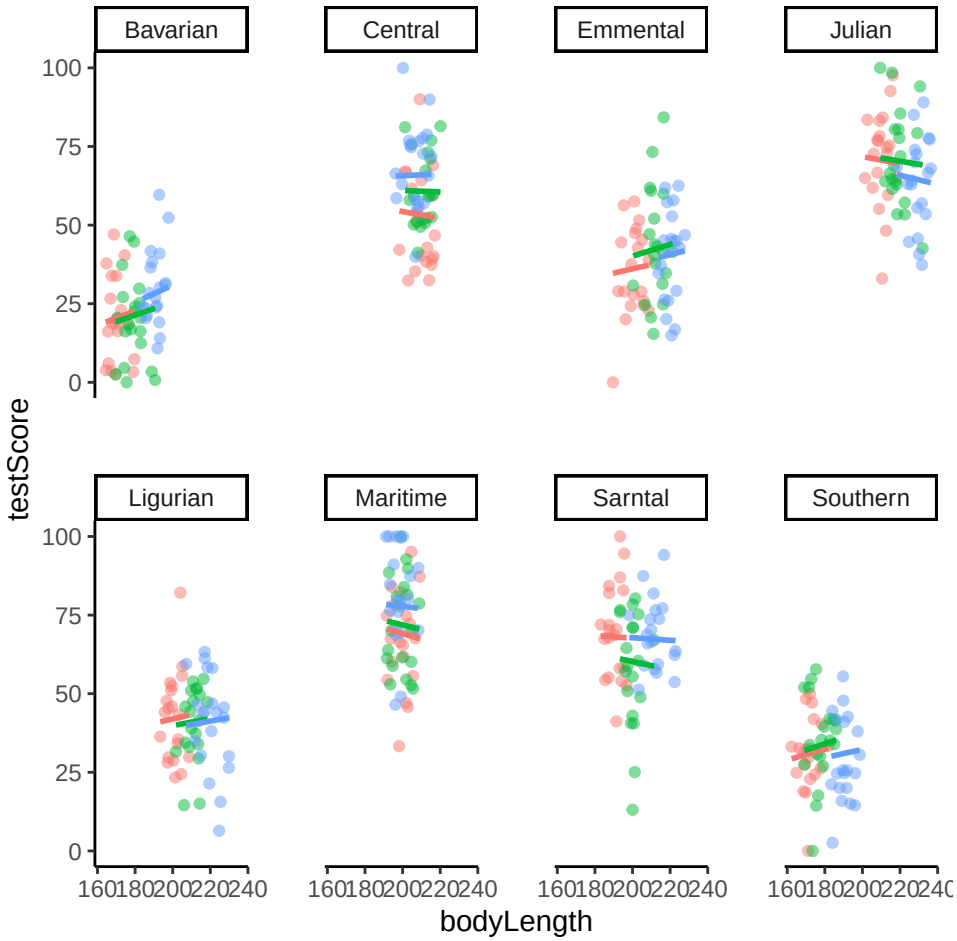
```
summary(mixed.ranslope)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: testScore ~ bodyLength2 + (1 + bodyLength2 | mountainRange/site)
##    Data: dragons
##
## REML criterion at convergence: 3968.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2654 -0.6737 -0.0200  0.6931  2.8432
##
## Random effects:
##  Groups              Name        Variance Std.Dev. Corr
##  site:mountainRange (Intercept)  19.8156  4.4515
##                     bodyLength2   0.7178  0.8472  1.00
##  mountainRange      (Intercept) 310.9691 17.6343
##                     bodyLength2   6.1119  2.4722  -1.00
##  Residual                       208.5025 14.4396
## Number of obs: 480, groups:  site:mountainRange, 24; mountainRange, 8
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  51.4263     6.3408   8.110
## bodyLength2   0.6691     1.8729   0.357
##
## Correlation of Fixed Effects:
##             (Intr)
## bodyLength2 -0.461
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see ?isSingular
```

Here we're modelling the intelligence of dragons as a function of body length, knowing that populations have differenct intelligence baselines and tha the relationship may vary among populations.

In the plot below, notice how the slopes for the different sites and mountain ranges are not parallel anymore.

```
(mm_plot <- ggplot(dragons, aes(x = bodyLength, y = testScore, colour = site)) +
      facet_wrap(~mountainRange, nrow=2) +
      geom_point(alpha = 0.5) +
      theme_classic() +
      geom_line(data = cbind(dragons, pred = predict(mixed.ranslope)), aes(y = pred),
                size = 1) +
   # adding predicted line from mixed model
      theme(legend.position = "none",
            panel.spacing = unit(2, "lines"))
)
```



Failing to account for the correlation in data might lead to misleading results. It seemed body length affected the test score unti we accounted for the variation coming from mountain ranges. Now we can see body length doesn't influence the test scores.

6.4 Presenting your model results

6.4.1 Plotting model predictions

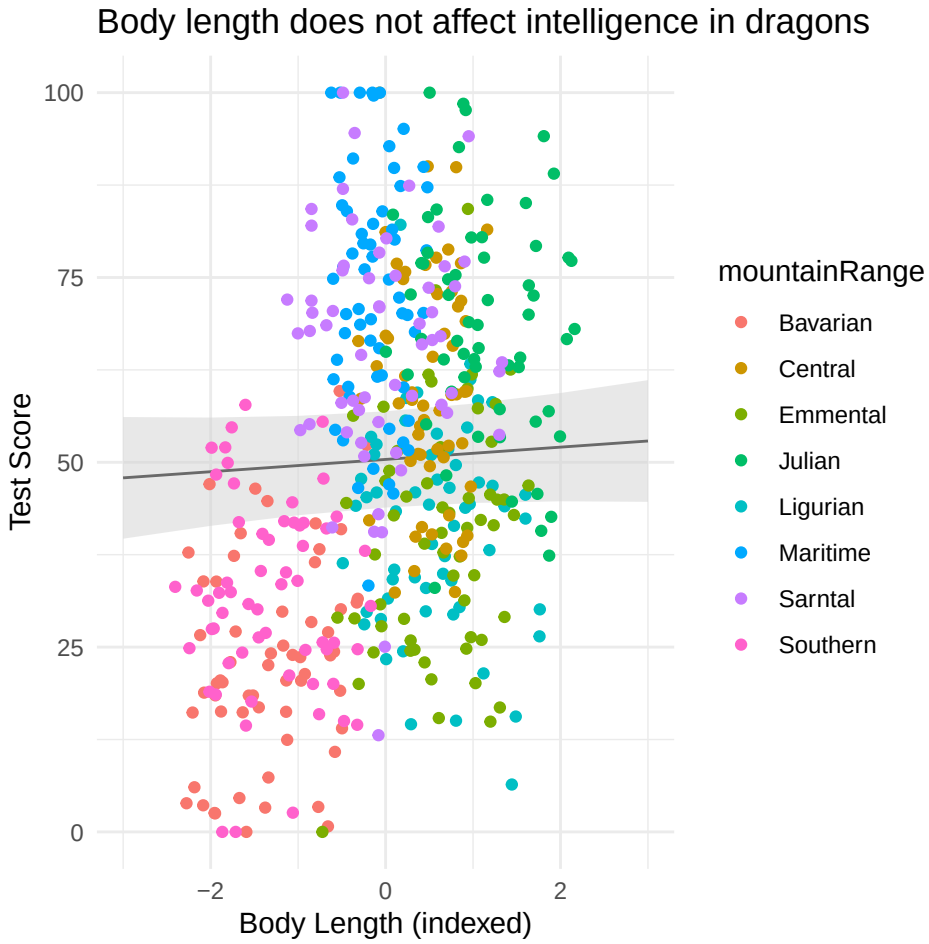
Often you will want to visualise your model as a regression line with some error around it, just like you would a simple linear model. However, ggplot2 stats options are not designed to estimate mixed-effect model objects correctly, so we will use the ggeffects package to help us draw the plots.

```
library(ggeffects)
# Install the package first if you haven't already, then load it

# Extract the prediction data frame
pred.mm <- ggpredict(mixed.lmer2, terms = c("bodyLength2"))
# This gives overall predictions for the model

# Plot the predictions
(ggplot(pred.mm) +
   geom_line(aes(x = x, y = predicted)) + # slope
   geom_ribbon(aes(x = x, ymin = predicted - std.error, ymax = predicted + std.error),
               fill = "lightgrey", alpha = 0.5) +  # error band
   geom_point(data = dragons, # adding the raw data (scaled values)
```

```
            aes(x = bodyLength2, y = testScore, colour = mountainRange)) +
   labs(x = "Body Length (indexed)", y = "Test Score",
        title = "Body length does not affect intelligence in dragons") +
   theme_minimal()
)
```

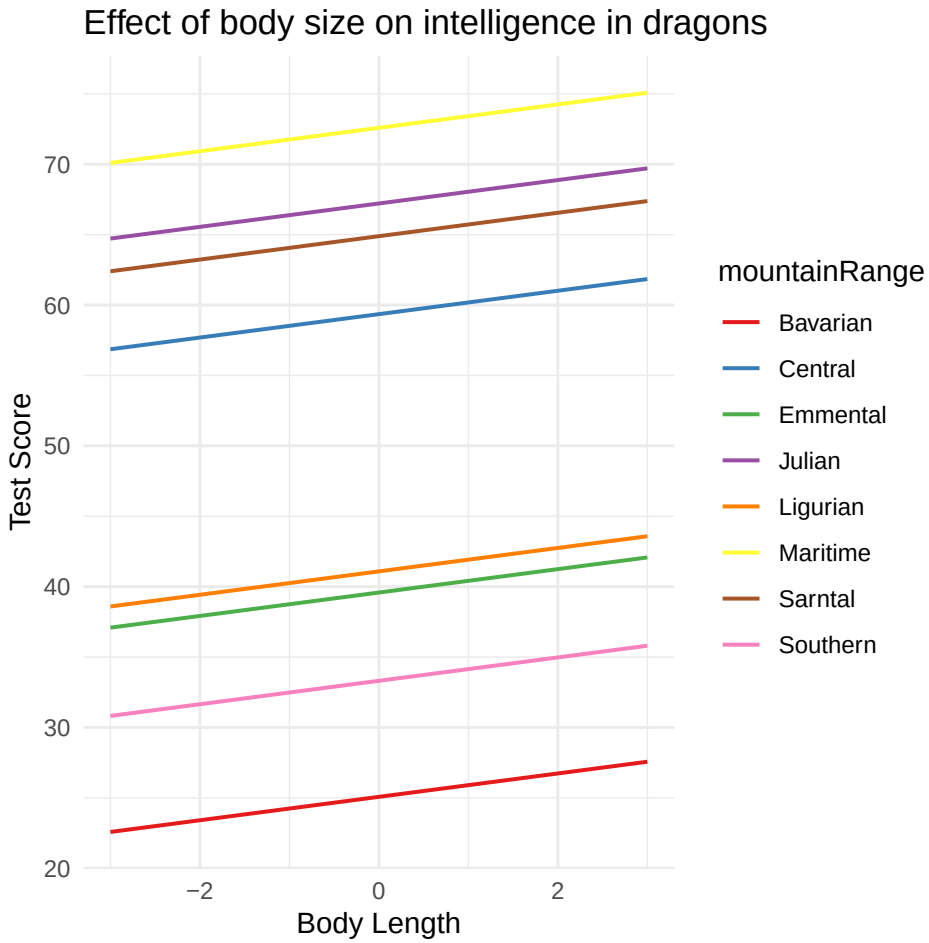


What if you want to visualise how the relationships vary according to different levels of random effects? You can specify `type = "re"` (for "random effects") in the ggpredict() function, and add the random effect name to the terms argument.

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------ tidyverse 1.3.0 --
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.3
## -- Conflicts --------------------------------------------------- tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x tidyr::unpack() masks Matrix::unpack()
```

```
ggpredict(mixed.lmer2, terms = c("bodyLength2", "mountainRange"), type = "re") %>%
   plot() +
   labs(x = "Body Length", y = "Test Score",
        title = "Effect of body size on intelligence in dragons") +
   theme_minimal()
```



You can clearly see the random intercepts and fixed slopes from this graph. When assessing the quality of your model, it's always a good idea to look at the raw data, the summary output, and the predictions all together to make sure you understand what is going on (and that you have specified the model correctly).