Chapter 12: Multilevel Models

Mark Andrews

Contents

Introduction

Multilevel models are a broad class of models that are applied to data that consist of sub-groups or clusters, including when these clusters are hierarchically arranged. Although they have been in existence for decades, they have become very widely used within the last 10 to 20 years due to computational advances. They are are now a major statistical modelling tool in the social sciences as well many other fields of research. A number of related terms are used to describe multilevel models: *hierarchical* models, *mixed effects* models, *random effects* models, and more. These terms are not strictly synonymous but do describe models that are all related to the general concept of a multilevel model. Here, we will prefer to use the term multilevel model as the main general term for these models. We will also use the term *hierarchical* model, at least a certain sense of the term, as essentially synonymous to multilevel model, and we'll use the term *mixed effect* models, or *mixed effect regression*, as the term for a particular widely used variant of the multilevel regression model.

As we will see, the defining feature of multilevel models is that they are *models of models*. In other words, for each cluster or sub-group in our data we create a statistical model, and then model how these statistical models vary across the clusters or sub-groups. We will begin our coverage of multilevel models by exploring *random effects* models. These are some of the simplest types of multilevel models, but yet they can make clear the key defining characteristics of the multilevel models generally. We then proceed to cover multilevel linear models, which are often referred to a *linear mixed effects* models. We will also describe how to perform Bayesian versions of these models.

Random effects models

Let us consider the following data set, which is on rat tumours.

```
rats_df <- read_csv('data/rats.csv',
                    col_types = cols(batch = col_character())
)
rats_df
#> # A tibble: 71 x 3
#>    batch     m     n
#>    <chr> <dbl> <dbl>
#>  1 1         0    20
#>  2 2         0    20
#>  3 3         0    20
#>  4 4         0    20
#>  5 5         0    20
#>  6 6         0    20
#>  7 7         0    20
#>  8 8         0    19
#>  9 9         0    19
#> 10 10        0    19
#> # ... with 61 more rows
```

This data set consists of data from $J = 71$ batches of rats. For each batch, we have the number of rats in it (`n`) and the number of rats in the batch that developed tumours (`m`). Let us begin by focusing in on a single batch.

```
rats_df_42 <- filter(rats_df, batch == '42')
rats_df_42
#> # A tibble: 1 x 3
#>   batch     m     n
#>   <chr> <dbl> <dbl>
#> 1 42        2    13
```

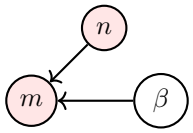
In this batch, out of 13 rats, the recorded number of tumours was 2. With these numbers alone, we can provide a simple statistical model of the tumour rate in batch 42. In this model, we can say that there is a fixed but unknown probability of a tumour in this batch, which we will denote by θ. In other words, each rat in the batch of size $n = 13$ has probability θ of developing a tumour and so the recorded number of tumours, $m = 2$, is a draw from a binomial distribution with parameter θ and size $n = 13$. In other words, our model is a binomial model:

$$m \sim \text{Binom}(\theta, n).$$

This is identical to the following binomial logistic regression model.

$$m \sim \text{Binom}(\theta, n), \quad \log\left(\frac{\theta}{1-\theta}\right) = \beta.$$

This binomial model can be represented by the following diagram.



This kind of diagram is known as a *Bayesian network*. It is a directed acyclic graph showing each variable or parameter in the model. The shaded nodes indicate that the corresponding variable is observed. From this diagram we see that the variable m is conditionally dependent on n, which is observed, and β, which is not observed and so must be inferred. While this diagram is very simple, it is useful to use it here to compare it to other models that we will use below.

We can implement this binomial logistic model in R using `glm`.

```
M <- glm(cbind(m, n-m) ~ 1,
         data = rats_df_42,
         family = binomial(link = 'logit')
)
```

Note that in this model, the left hand side of the `~` operator is `cbind(m, n-m)`, where `m` gives the number of rats in the batch that have tumours and `n -m` gives the number of rates in the batch who do not have tumours. The command `cbind` is used to stack vectors side by side as columns, and so `cbind(m, n-m)` creates a matrix with two columns. From this model `M`, we can see that our estimate of θ is as follows:

```
ilogit <- function(x) 1/(1 + exp(-x))
coef(M) %>% ilogit() %>% unname()
#> [1] 0.1538462
```

This is expected given that out of 13 rats in this batch, the recorded number of tumours was 2, which is a proportion of 0.154.

We can now easily extend this model to apply to all batches in our data set. In other words, for each of our J batches, where n_j is the batch's size, θ_j is its fixed but unknown probability of developing a tumour, and m_j is its recorded number of tumours, we have the following model:

$$m_j \sim \text{Binom}(\theta_j, n_j), \quad \log\left(\frac{\theta_j}{1-\theta_j}\right) = \beta_j.$$

This is implemented using `glm` as follows.

```
M <- glm(cbind(m, n-m) ~ 0 + batch,
         data = rats_df,
         family = binomial(link = 'logit')
)
```

Using `broom::tidy` and some `dplyr` and related tools, we can look at the estimates and confidence intervals for a random sample of 10 of the batches.

```
M_estimates <- tidy(M) %>%
  select(term, beta = estimate) %>%
  mutate(term = str_remove(term, 'batch'),
         theta = ilogit(beta)) %>%
  rename(batch = term)

M_estimates %>%
  sample_n(10) %>%
  arrange(beta) %>%
  mutate_at(vars(beta,theta), ~round(., 2))
#> # A tibble: 10 x 3
#>    batch   beta theta
#>    <chr>  <dbl> <dbl>
#>  1 5     -27.8   0
#>  2 13    -27.6   0
#>  3 29     -2.2   0.1
#>  4 23     -2.08  0.11
#>  5 37     -2.01  0.12
#>  6 48     -1.39  0.2
#>  7 58     -1.13  0.24
#>  8 59     -1.1   0.25
#>  9 64     -0.85  0.3
#> 10 70     -0.51  0.37
```

Although we have implemented a single `glm` model, this has effectively lead to J separate binomial models. The Bayesian network diagram for these models is shown in the diagram in Figure 1.

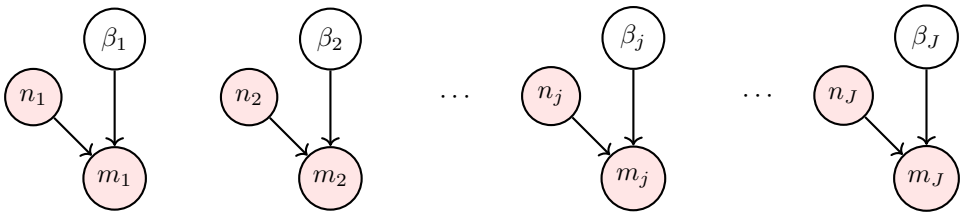


Figure 1: Inferring the log odds of a tumour β_j in each of the J batches is identical to J independent binomial models.

In other words, we have a model of the tumour rate for batch 1, another for batch 2, and so on. From this, we do not have a model of the distribution of tumour rates across all batches. We do not, for example, have a model that gives us the mean or standard deviation, or any other information, about the tumour rate across all possible batches in this experiment, of which our set of 71 batches are a sample. In order to obtain this model, we must perform a multilevel model.

A multilevel model extension of the binomial logistic regression model above is as follows.

$$\text{for } j \in 1 \dots J, \quad m_j \sim \text{Binom}(\theta_j, n_j), \quad \log\left(\frac{\theta_j}{1-\theta_j}\right) = \beta_j, \quad \beta_j \sim N(b, \tau^2).$$

The crucial added feature here is that the log odds of the tumour probabilities is being modelled as normally distributed with a mean of b and a standard deviation of τ. These dependencies are shown in the Bayesian network diagram in Figure 2.

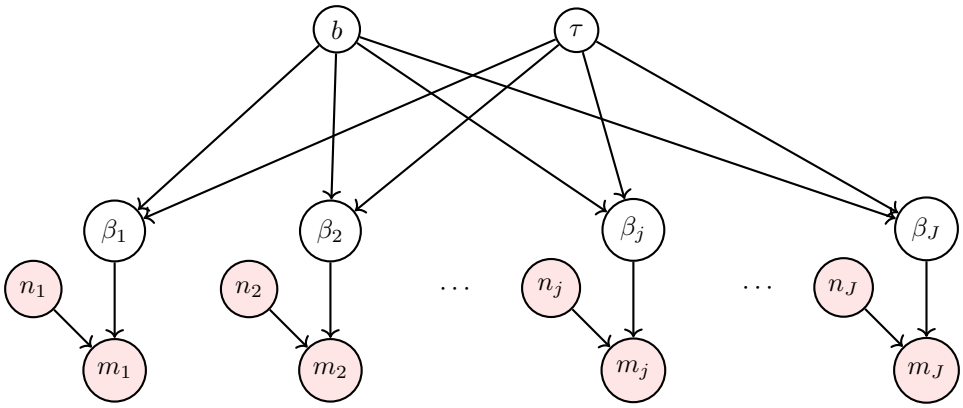


Figure 2: Inferring the log odds of a tumour β_j in each of the J batches is identical to J independent binomial models.

In this multilevel model, just as in the previous non-multilevel model, $\beta_1, \beta_2, \dots \beta_j \dots \beta_J$ have fixed but unknown values. However, in addition, these values are modelled as all drawn from the same normal distribution. The two important consequences of this are as follows. First, it provides a model of the *population* from which $\beta_1, \beta_2, \dots \beta_j \dots \beta_J$ are a sample. Given that each β_j effectively defines a model for a batch of rats, then the normal distribution from which $\beta_1, \beta_2 \dots \beta_j \dots \beta_J$ are drawn is a *model of models*. Amongst other things, this population model of the β's allows to predict the log odds, or probability, of a tumour for any future batch of rats, i.e. batch $J+1$. Second, because we are assuming that $\beta_1, \beta_2, \dots \beta_j \dots \beta_J$ are all drawn from the same normal distribution, this introduces constraints on the inference of the values of each β_j. In other words, to infer the value of β_j, the observed values of m_j and n_j are not the only relevant pieces of information. Now, the values of b and τ are also relevant, and because b and τ are also unknown, they

themselves must be inferred from $\beta_1, \beta_2, \ldots \beta_j \ldots \beta_J$. This effectively means that the inferences concerning $\beta_1, \beta_2, \ldots \beta_j \ldots \beta_J$ are inter-dependent and mutually constrain one another. We will explore both of these important general features of multilevel models below.

Given that we can rewrite $\beta_j \sim N(b, \tau^2)$ as $\beta_j = b + \xi_j$ where $\xi_j \sim N(0, \tau^2)$, we can rewrite the multilevel model as

$$\text{for } j \in 1 \ldots J, \quad m_j \sim \text{Binom}(\theta_j, n_j), \quad \log\left(\frac{\theta_j}{1-\theta_j}\right) = b + \xi_j, \quad \xi_j \sim N(0, \tau^2).$$

We can then implement this model using the `glmer` model that is part of the `lme4` package.

```
library(lme4)
M_ml <- glmer(cbind(m, n-m) ~ 1 + (1|batch),
              data = rats_df,
              family = binomial(link = 'logit')
)
```

Before we proceed to examine the results of this model, let us first describe the formula syntax. The right hand side of the formula is `1 + (1|batch)`. The first `1` indicates the intercept term of the model, which is b. This is identical to how we indicate an intercept only model using `lm`, `glm`, etc. For example, if were modelling a set of n values $y_1, y_2 \ldots y_n$ as $y_i \sim N(\mu, \sigma^2)$, we could write `lm(y ~ 1)`. The `(1|batch)` is a *random intercepts* statement. Specifically, in the case of `glmer`, it states that for each value of the `batch` variable we have a constant term, and that these constant terms are normally distributed from a *zero mean* normal distribution. As such, `(1|batch)` gives us the $\xi_j \sim N(0, \tau^2)$ for each of the J batches.

Let us look at the summary of this model.

```
summary(M_ml)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(m, n - m) ~ 1 + (1 | batch)
#>    Data: rats_df
#>
#>      AIC      BIC   logLik deviance df.resid
#>    319.9    324.4   -157.9    315.9       69
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.2392 -0.6230 -0.1055  0.4795  1.0253
#>
#> Random effects:
#>  Groups Name        Variance Std.Dev.
#>  batch  (Intercept) 0.4417   0.6646
#> Number of obs: 71, groups:  batch, 71
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -1.9369     0.1211     -16   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The first thing to note is the estimated value of b, which is listed as the estimate of the intercept term in the `Fixed effects` section. We see that this has the value of -1.937. The estimate of τ and τ^2, on the other hand, are given by the `Std.Dev.` and `Variance` terms for the `(Intercept)` for `batch` in the `Random effects` section. We see that the of τ is 0.665. As such, our model of the distribution of the log odds of the tumours is a normal distribution whose mean and standard deviation are estimated to be -1.937 and 0.665,

respectively. The translates into the probability distribution over the probabilities of tumours that we see in Figure 3.

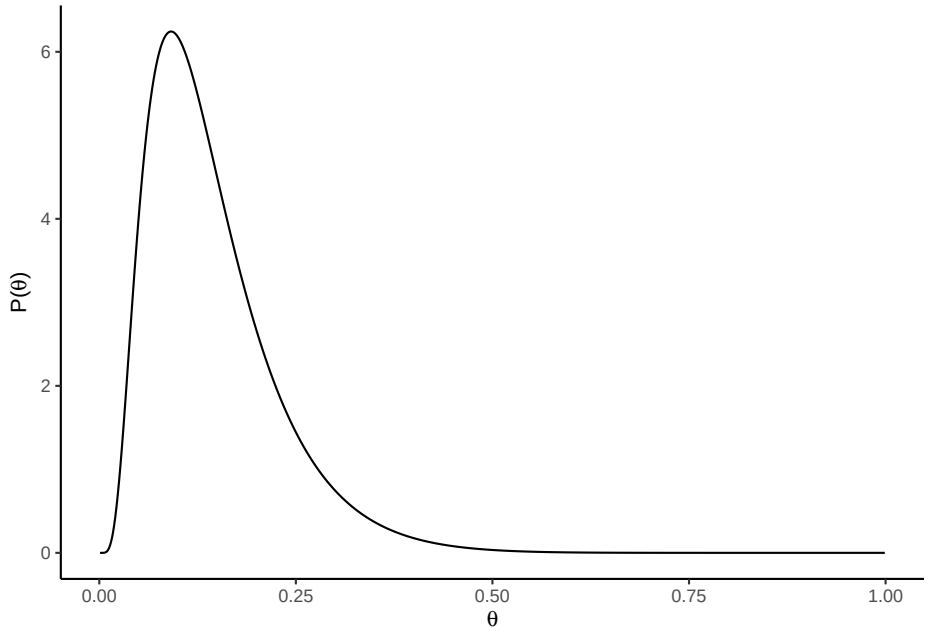


Figure 3: The estimate of the population distribution over θ for the rats tumour multilevel model.

Given that this normal distribution with mean $b = -1.937$ and standard deviation $\tau = 0.665$ is a model of the population from which the $\beta_1, \beta_2 \ldots \beta_j \ldots \beta_J$ are drawn, then we can make statements like the following. The expected value of the log odds of a tumour in a future batch of rats is -1.937 and with 95% probability, this log odds is between -3.239 and -0.634. Equivalently, the expected value of the probability of a tumour in a future batch of rats is 0.126 and with 95% probability, this probability is between 0.038 and 0.347.

From this model, we can also obtain the estimates of $\xi_1, \xi_2 \ldots \xi_j \ldots \xi_J$ from the model by using the `ranef` command.

```
ranef(M_ml)$batch %>%
  head()
#>    (Intercept)
#> 1   -0.6298720
#> 10  -0.6096908
#> 11  -0.6096908
#> 12  -0.5888596
#> 13  -0.5888596
#> 14  -0.5673326
```

We may obtain the estimates of b using the `fixef` command.

```
b <- fixef(M_ml)
```

We may then add on the estimates of b to get the estimates of $\beta_1, \beta_2 \ldots \beta_j \ldots \beta_J$.

```
b + ranef(M_ml)$batch %>%
  head()
#>    (Intercept)
#> 1    -2.566785
#> 10   -2.546604
#> 11   -2.546604
```

```
#> 12   -2.525773
#> 13   -2.525773
#> 14   -2.504246
```

We may obtain the estimates of $\beta_1, \beta_2 \dots \beta_j \dots \beta_J$ more directly by using the `coef` command.

```
M_ml_estimates <- coef(M_ml)$batch

M_ml_estimates %>%
  head()
#>    (Intercept)
#> 1    -2.566785
#> 10   -2.546604
#> 11   -2.546604
#> 12   -2.525773
#> 13   -2.525773
#> 14   -2.504246
```

Comparing these values to the corresponding values in the non-multilevel model, we can see how the estimates of $\beta_1, \beta_2 \dots \beta_j \dots \beta_J$ mutually constrain one another. This phenomenon is an example of *shrinkage*. In this model, it is easier to visualize this effect if we look at $\theta_1, \theta_2 \dots \theta_j \dots \theta_J$, which are simply the inverse logit transforms of $\beta_1, \beta_2 \dots \beta_j \dots \beta_J$. In Figure 4, we compare the estimates of $\theta_1, \theta_2 \dots \theta_j \dots \theta_J$ from the flat or non-multilevel model `M` against those of the multilevel model `M_ml`.

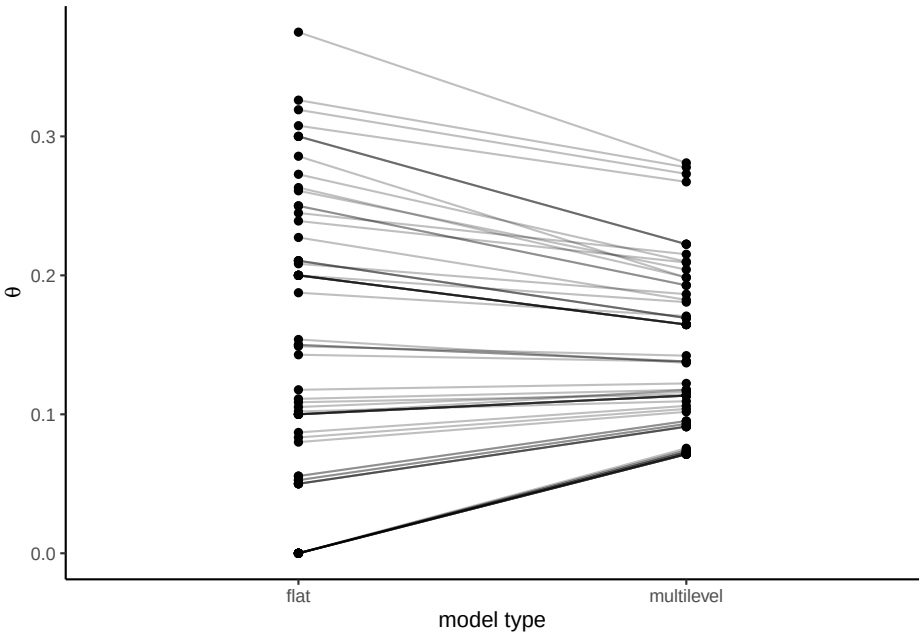


Figure 4: Estimates for $\theta_1, \theta_2 \dots \theta_j \dots \theta_J$ from the flat or non-multilevel model (left) and the multilevel model (right).

As we can see, the estimates for $\theta_1, \theta_2 \dots \theta_j \dots \theta_J$, and hence also for $\beta_1, \beta_2 \dots \beta_j \dots \beta_J$, in the case of the multilevel model are brought closer together towards the centre compared to the non-multilevel model.

To understand why this phenomenon occurs, let us compare the likelihood functions for the non-multilevel and the multilevel model. In the non-multilevel model, there are J unknowns: $\beta_1, \beta_1 \dots \beta_j \dots \beta_J$, and given

the observed data, the likelihood function is

$$\prod_{j=1}^{J} \mathrm{P}(m_j|n_j, \beta_j).$$

Maximising this function with respect to β_j is equivalent to maximizing $\mathrm{P}(m_j|n_j, \beta_j)$ with respect to β_j, which is $\hat{\beta}_j = \mathrm{logit}(m_j/n_j)$. On the other hand, in the multilevel model, there are $J+2$ unknowns: $\beta_1, \beta_1 \ldots \beta_j \ldots \beta_J$, b and τ, and the likelihood function is

$$\prod_{j=1}^{J} \mathrm{P}(m_j|n_j, \beta_j)\mathrm{P}(\beta_j|b, \tau).$$

Maximising this function with respect to β_j is equivalent to maximizing $\mathrm{P}(m_j|n_j, \beta_j)\mathrm{P}(\beta_j|b, \tau)$, which is a compromise between maximizing $\mathrm{P}(m_j|n_j, \beta_j)$ and maximizing $\mathrm{P}(\beta_j|b, \tau)$, with the latter being maximized as β_j becomes closer to b. Maximising the multilevel model's likelihood with respect to b is equivalent to maximizing $\prod_{j=1}^{J} \mathrm{P}(\beta_j|b, \tau)$ with respect to b, which occurs at $\hat{b} = \frac{1}{J}\sum_{j=1}^{J} \beta_j$. From this, we see that the estimate of each β_j is based on the value of m_j and n_j, but is also pulled towards the average of $\beta_1, \beta_1 \ldots \beta_j \ldots \beta_J$.

Normal random effects models

Let us now consider a new data set.

```
alcohol_df <- read_csv('data/alcohol.csv')
alcohol_df
#> # A tibble: 411 x 3
#>    country  year alcohol
#>    <chr>   <dbl>   <dbl>
#>  1 Russia   1985   13.3
#>  2 Russia   1986   10.8
#>  3 Russia   1987   11.0
#>  4 Russia   1988   11.6
#>  5 Russia   1989   12.0
#>  6 Chile    1990    9.43
#>  7 Ecuador  1990    8.4
#>  8 Greece   1990   12.5
#>  9 Russia   1990   12.3
#> 10 Russia   1991   12.7
#> # ... with 401 more rows
```

In this, we have the per capita average alcohol consumption in $J = 189$ countries in $K = 22$ different years, though we do not necessarily have data from each country in each year. Let us denote the per capita alcohol values by $y_1, y_2 \ldots y_i \ldots y_n$. For each y_i, we have an indicator variable $x_i \in 1 \ldots J$, which indicates the country that y_i corresponds to. An initial model for $y_1, y_2 \ldots y_i \ldots y_n$ could then be

$$y_i \sim N(\mu_{[x_i]}, \sigma^2), \quad \text{for } i \in 1 \ldots n,$$

where $\mu_1, \mu_2 \ldots \mu_j \ldots \mu_J$ are the country alcohol per capita consumption averages for the J countries. This model be represented by the Bayesian network diagram shown in Figure 5. This is a non-multilevel model because the alcohol consumption averages in each country are being modelled independently of those of other countries. In this model, however, we do assume that the standard deviations of the alcohol consumption values across all countries are the same, and given by σ, whose value is fixed but unknown. What this entails, therefore, is that this model is a one-way Anova model with the standard homogeneity of variance assumption.

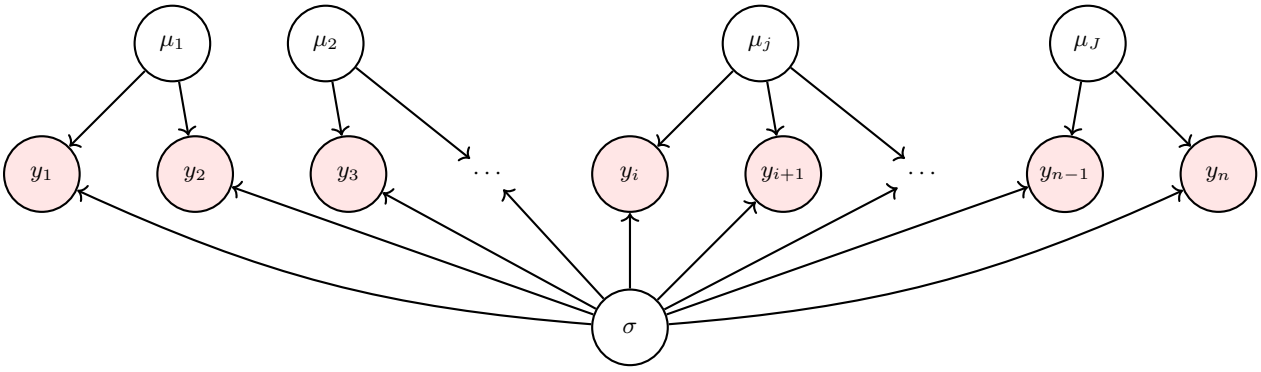


Figure 5: A non-multilevel model for average alcohol consumption across countries. The $y_1, y_2 \dots y_n$ variables are alcohol consumption averages for J different countries over different years. As depicted here, we are assuming that y_1 and y_2 represent averages for the same country, country $j = 1$, two different years. The overall average for this country is given by μ_1, whose value is unknown. This non-multilevel model is identical to a one-way anova model with J groups. The common standard deviation term σ is simply the homogeneity of variance assumption of the one-way anova.

A multilevel counterpart to the above model would be as follows.

$$y_i \sim N(\mu_{[x_i]}, \sigma^2), \quad \text{for } i \in 1 \dots n,$$
$$\mu_j \sim N(\phi, \tau^2), \quad \text{for } j \in 1 \dots J.$$

This model, which is depicted by a Bayesian network diagram in Figure 6, extends the previous one by assuming that the $\mu_1, \mu_2 \dots \mu_j \dots \mu_J$ are drawn from a normal distribution with mean ϕ and standard deviation of τ. Given that y_i can be rewritten as $y_i = \mu_{[x_i]} + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma^2)$, and that μ_j can be rewritten as $\mu_j = \phi + \xi_j$ where $\xi_j \sim N(0, \tau^2)$, we can rewrite the above model as

$$y_i = \phi + \xi_{[x_i]} + \epsilon_i, \quad \text{for } i \in 1 \dots n,$$

where each $\xi_j \sim N(0, \tau^2)$ and each $\epsilon_i \sim N(0, \sigma^2)$. Here, ϕ signifies the global average per capita alcohol consumption rate. Each ξ_j is the *random offset* of country j from ϕ, and each ϵ_i is the residual error for each observation. In this model, the residual error ϵ_i gives the random year by year deviation from the country x_i's average consumption rate.

We can implement this model using `lme4::lmer` as follows.

```
M_ml <- lmer(alcohol ~ 1 + (1|country),
             data = alcohol_df)
summary(M_ml)
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: alcohol ~ 1 + (1 | country)
#>    Data: alcohol_df
#>
#> REML criterion at convergence: 1918.2
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.4661 -0.1582 -0.0719  0.1642  5.9576
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  country  (Intercept) 22.208   4.713
#>  Residual              1.108   1.053
```

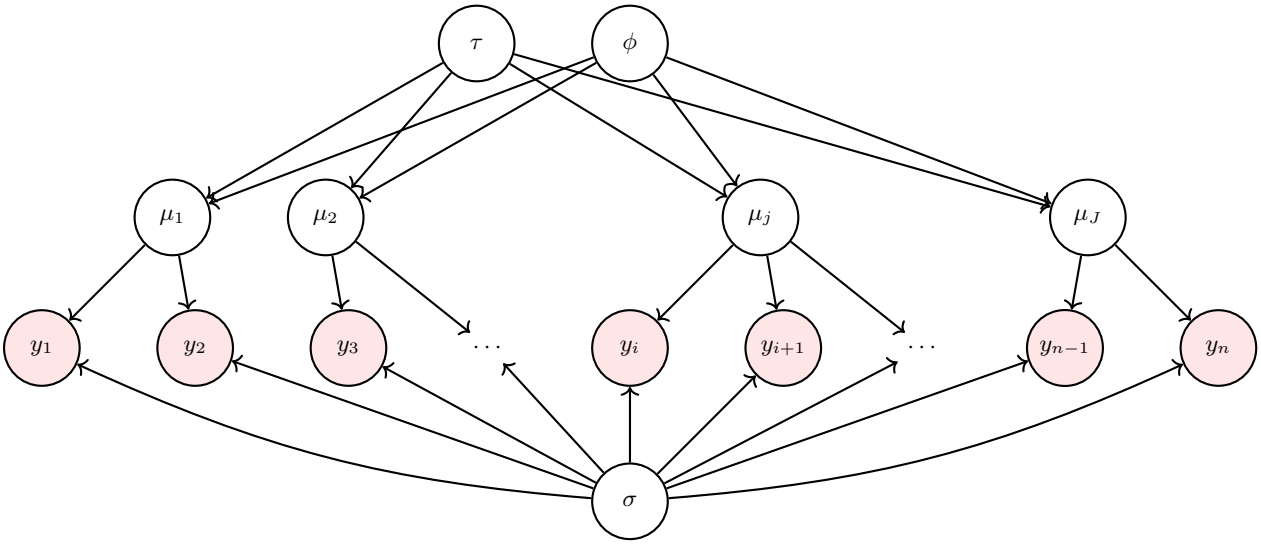


Figure 6: A multilevel model for average alcohol consumption across. This model is an extension of the non-multilevel model shown in Figure 5 by modelling the $\mu_1, \mu_2 \ldots \mu_j \ldots \mu_J$ as samples from a normal distribution with mean ϕ and standard deviation τ.

```
#> Number of obs: 411, groups:  country, 189
#>
#> Fixed effects:
#>             Estimate Std. Error t value
#> (Intercept)   6.6612     0.3469    19.2
```

As we can see from the `(Intercept)` estimate in the `Fixed effects` and the `Std.Dev.` for `country` in the `Random effects`, the normal distribution of the μ values has a mean of $\phi = 6.661$ and standard deviation of $\tau = 4.713$. The residual standard deviation σ is given by the `Std.Dev.` for `Residual` in the `Random effects`, and has the value of $\sigma = 1.053$.

Given the nature of the random effects model, i.e. each y_i is modelled as $y_i = \phi + \xi_{[x_i]} + \epsilon_i$, the variance of y is equal to $\tau^2 + \sigma^2$. The value

$$\frac{\tau^2}{\tau^2 + \sigma^2}$$

is known as the *intraclass correlation* (ICC), which takes on values between 0 and 1. Obviously, ICC tells us how much of the total variance in the data is due to variation between the countries. If the ICC is relatively high, and so τ^2/σ^2 is relatively high, the observed values *within* countries will be close together relative to the *between* country averages, and thus there will be relatively high clustering of the data. In this data, the ICC is 0.95.

One consequence of a high ICC, and specifically a high value of τ^2 relative to σ^2, is that shrinkage effects will be less. This can be seen in Figure 7 where we compare the estimates of each μ_j from the non-multilevel and multilevel models. In more detail, in a normal random effects model, the estimate of a μ_j will be approximately as follows (see Gelman and Hill (2007), p253, for details):

$$\mu_j \approx \frac{\frac{n_j}{\sigma^2}\bar{y}_j + \frac{1}{\tau^2}\bar{y}}{\frac{n_j}{\sigma^2} + \frac{1}{\tau^2}},$$

where $\bar{y}_j$ is the average of the y_i values corresponding to country j, and $\bar{y}$ is overall average of the y_i values. In general, when τ^2 is large relative to σ^2, the influence of the global average $\bar{y}$ on μ_j will be minimal. This will be especially the case as n_j gets larger.

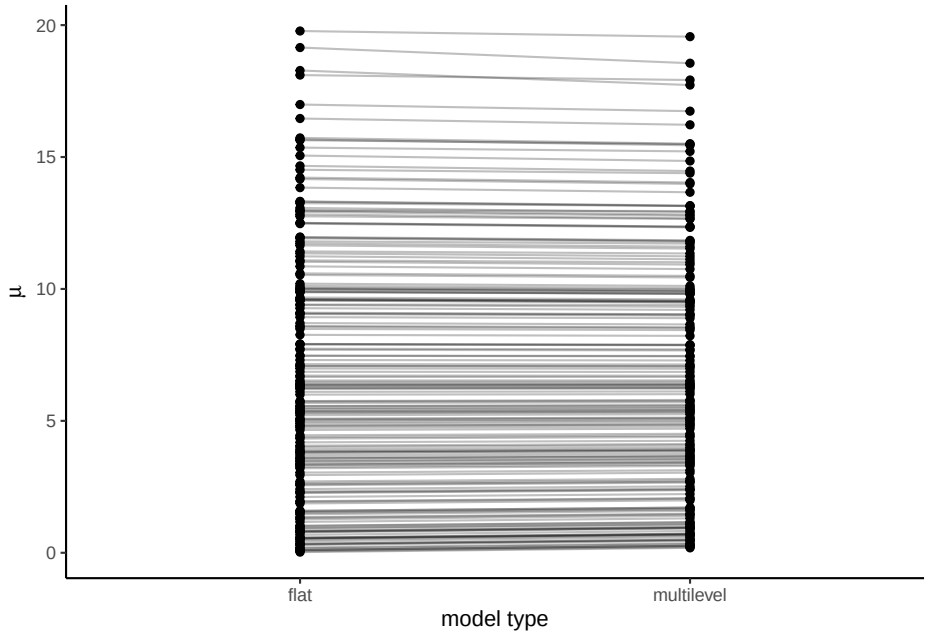


Figure 7: Estimates for $\mu_1, \mu_2 \dots \mu_j \dots \mu_J$ from the flat or non-multilevel model (left) and the multilevel model (right). Shrinkage effects are minimal due to the high value of τ^2 relative to σ^2.

Linear mixed effects models

We will now consider multilevel linear regression models. These are often referred to as linear mixed effects models, for reasons that will be clear after we describe them in more detail. As with random effects models, these models are best introduced by way of example. For this, we will use he `sleepstudy` data set from `lme4`, which provides the average reaction time for each person on each day of a sleep deprivation experiment that lasted 10 days.

```
sleepstudy <- lme4::sleepstudy %>%
  as_tibble()
sleepstudy
#> # A tibble: 180 x 3
#>    Reaction  Days Subject
#>       <dbl> <dbl> <fct>
#>  1     250.     0 308
#>  2     259.     1 308
#>  3     251.     2 308
#>  4     321.     3 308
#>  5     357.     4 308
#>  6     415.     5 308
#>  7     382.     6 308
#>  8     290.     7 308
#>  9     431.     8 308
#> 10     466.     9 308
#> # ... with 170 more rows
```

This data is displayed in Figure 8.

To begin our analysis, let us first focus on one arbitrarily chosen experimental subject, namely subject `350`.

```
sleepstudy_350 <- sleepstudy %>%
```

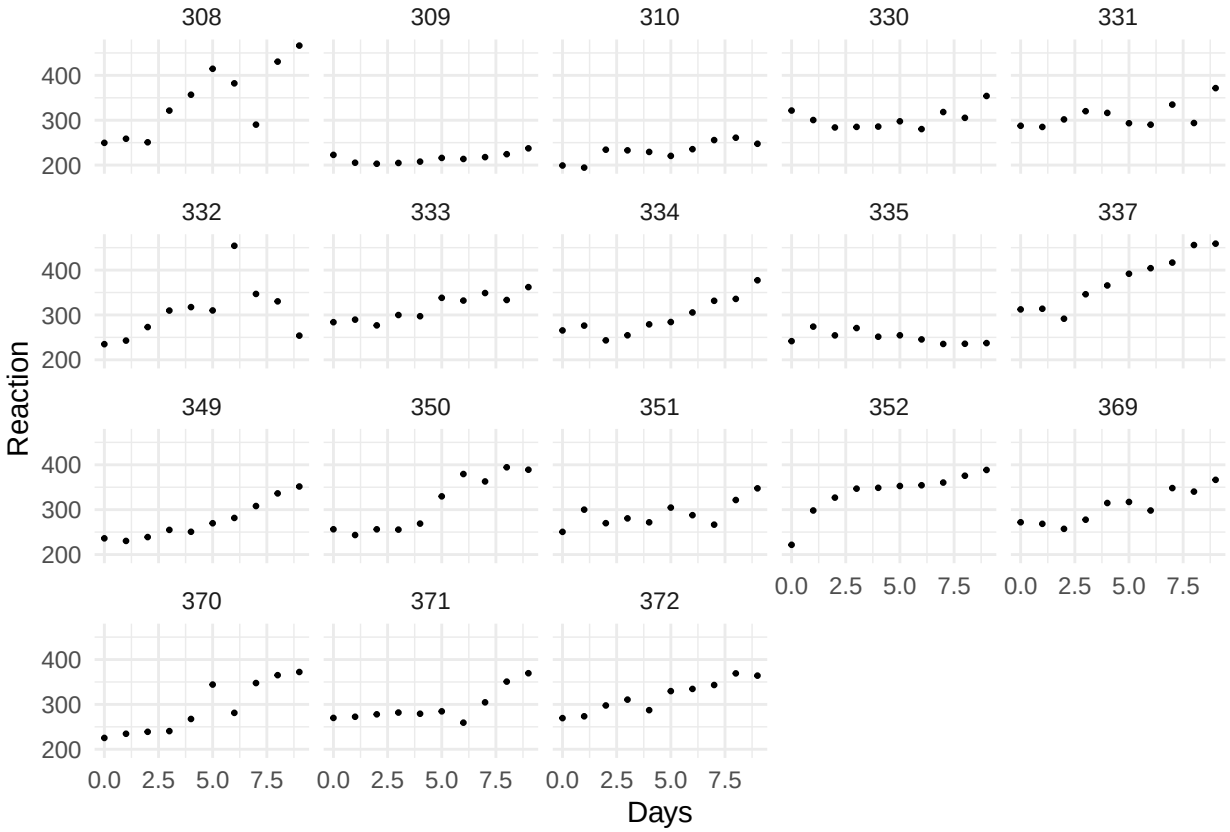


Figure 8: Each figure shows the average reaction time data from a subject in sleep deprivation on each day of the 10 day experiment.

```
  filter(Subject == 350)
sleepstudy_350
#> # A tibble: 10 x 3
#>    Reaction  Days Subject
#>       <dbl> <dbl> <fct>
#>  1     256.     0 350
#>  2     243.     1 350
#>  3     256.     2 350
#>  4     256.     3 350
#>  5     269.     4 350
#>  6     330.     5 350
#>  7     379.     6 350
#>  8     363.     7 350
#>  9     394.     8 350
#> 10     389.     9 350
```

The trend over time in this subject's average reaction time can be modelled using the following normal linear model:

$$y_d \sim N(\mu_d, \sigma^2), \quad \mu_d = \beta_0 + \beta_1 x_d, \quad \text{for } d \in 1 \dots n,$$

where y_d represents the subject's reaction time on their dth observation, and $x_d \in \{0, 2, \dots n = 9\}$ indicates the day when this observation happened. Using $\vec{\beta} = [\beta_0, \beta_1]^\intercal$, we can represent this model using a Bayesian network diagram as we do in Figure 9. In that figure, we provide two equivalent diagrams, with Figure 9b

using a plate notation that denotes a repetition of nodes within a bounding plate according to an index, which in this case is $d \in 1 \dots n$. This model is implemented in R as follows.

```
M_350 <- lm(Reaction ~ Days, data = sleepstudy_350)
```

The estimated values of the coefficients are as follows.

```
coef(M_350)
#> (Intercept)        Days
#>   225.83460    19.50402
```

Thus, we estimate that the average reaction time of subject `350` increases by 19.5 on each day of study. In addition, because the first day of the study was indicated by $x_i = 0$, this subject's average reaction prior to any sleep deprivation was 225.83.

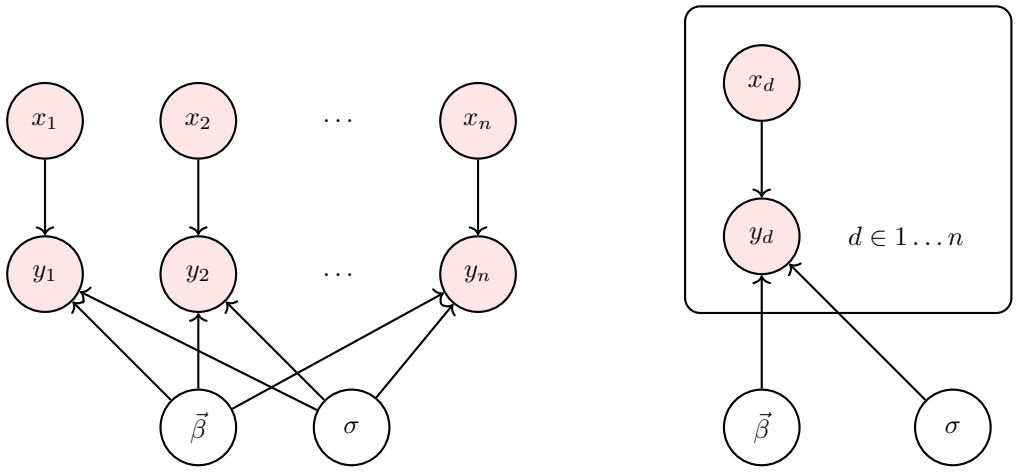


Figure 9: Two equivalent Bayesian network diagrams representing a normal linear model with one predictor variable. Diagram b) uses a compact plate notation whereby all variables within the plate are repeated for all values of the index i, which takes values from 1 to n.

Were we to provide a similar model for each subject in the experiment, whom we will index by $j \in 1 \dots J$, this would lead to J independent normal linear models. If we denote the average reaction time on observation d for subject j by y_{jd}, this set of models is as follows.

$$y_{jd} \sim N(\mu_{jd}, \sigma_j^2), \quad \mu_{jd} = \beta_{j0} + \beta_{j1} x_{jd}, \quad \text{for } j \in 1 \dots J, \text{ for } d \in 1 \dots n_j.$$

Note that here we have J data sets, one for each each subject in the experiment, and d is used to index the observations within each one. Thus, for the jth data-set, d ranges from 1 to n_j. This model is represented in a Bayesian network diagram in Figure 10a. If we assume that there is a common residual standard deviation term σ, rather than one per each of the J subjects, this model is identical to a varying intercept and varying slope linear model. These models are linear regression models whose intercepts and slopes vary according to a categorical variable, which in this case is the `Subject` variable. Using R, we can implement this model as follows.

```
M_flat <- lm(Reaction ~ 0 + Subject + Subject:Days, data = sleepstudy)
```

Note that the `0 + Subject` in the formula ensures that we obtain a separate intercept term for each subject, rather than one subject being the *base* level and all others being represented as offsets from this base. Likewise, the `Subject:Days` ensures that we obtain a separate slope for each subject. Formally, this model is equivalent to

$$y_{jd} \sim N(\mu_{jd}, \sigma^2), \quad \mu_{jd} = \beta_{j0} + \beta_{j1} x_{jd}, \quad \text{for } j \in 1 \dots J, \text{ for } d \in 1 \dots n_j,$$

and we have provided a Bayesian network diagram of it in Figure 10b. We can see that it is identical to that represented in 10a with the exception of the shared standard deviation σ. In other words, a (non-multilevel) varying intercept and varying slope model, where the intercepts and slopes vary by subject, is identical to a set of independent linear models, one per each subject, but with a single standard deviation term σ shared across all subjects.

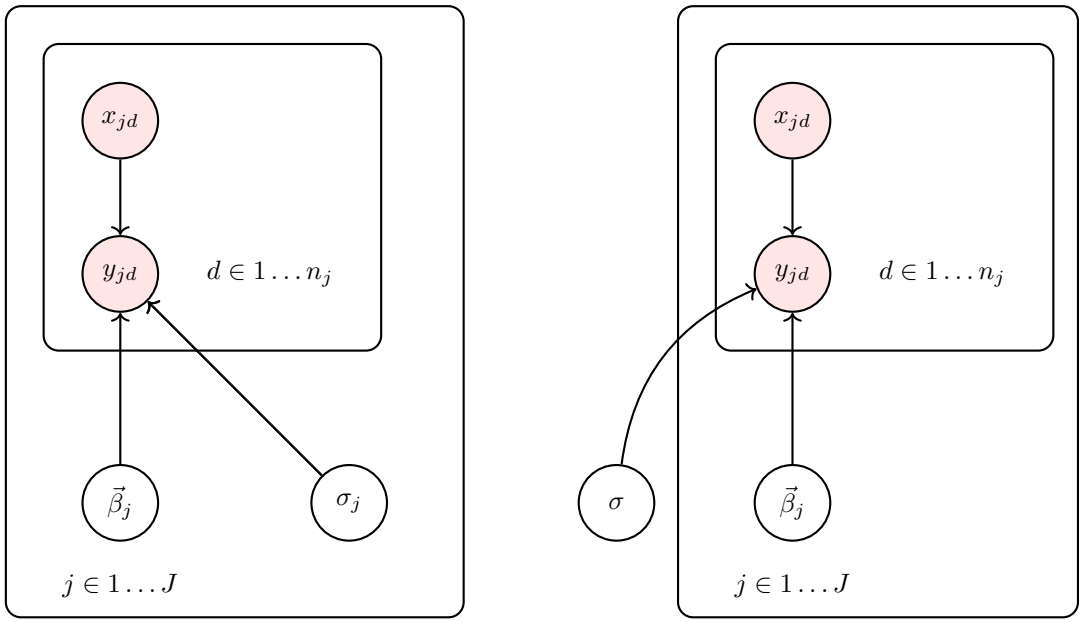


Figure 10: Bayesian network diagrams for a) a set of J independent normal linear models, and b) a varying slope and varying intercept linear model whereby the slope and intercept vary by a categorical variable with J levels.

Let us now consider a multilevel variant of this non-multilevel varying intercept and slope model. In this, we assume that the vector of coefficients $\vec{\beta}_j = [\beta_{j0}, \beta_{j1}]^\intercal$ is drawn from a multivariate Normal distribution with mean vector $\vec{b}$ and covariance matrix Σ. This model can be written as follows.

$$
\begin{aligned}
y_{jd} &\sim N(\mu_{jd}, \sigma), \quad \mu_{jd} = \beta_{j0} + \beta_{j1} x_{jd}, \quad \text{for } j \in 1 \dots J, \text{ for } d \in 1 \dots n_j,, \\
\vec{\beta}_j &\sim N(\vec{b}, \Sigma) \quad \text{for } j \in 1 \dots J,
\end{aligned}
$$

The Bayesian network diagram for this model is shown in Figure 11. As we can see, this is an extension of the Bayesian network diagram in Figure 10b, with the extension being that each $\vec{\beta}_j$ are modelled as functions of $\vec{b}$ and Σ.

We can rewrite this multilevel model in the following manner.

$$
\begin{aligned}
\text{for } i \in 1 \dots n, \quad & y_i \sim N(\mu_i, \sigma^2), \\
& \mu_i = \beta_{[s_i]0} + \beta_{[s_i]1} x_i, \\
\text{for } j \in 1 \dots J, \quad & \vec{\beta}_j \sim N(\vec{b}, \Sigma).
\end{aligned}
$$

Note that here the i index ranges over all values in the entire data-set, i.e. $i \in 1, 2 \dots n$, and each $s_i \in 1, 2 \dots J$ is an indicator variable that indicates the identity of the subject on observation i. This notation with a single subscript per observation and indicator variables is more extensible, especially for complex models. Using this new notation, given that $\vec{\beta}_j \sim N(\vec{b}, \Sigma)$, we can rewrite $\vec{\beta}_j$ as $\vec{\beta}_j = \vec{b} + \vec{\zeta}_j$ where $\vec{\zeta}_j \sim N(0, \Sigma)$. Substituting $\vec{b} + \zeta_j$ for $\vec{\beta}$, and thus substituting $b_0 + \zeta_{j0}$ and $b_1 + \zeta_{j1}$ for β_{j0} and β_{j1}, respectively, we have the following

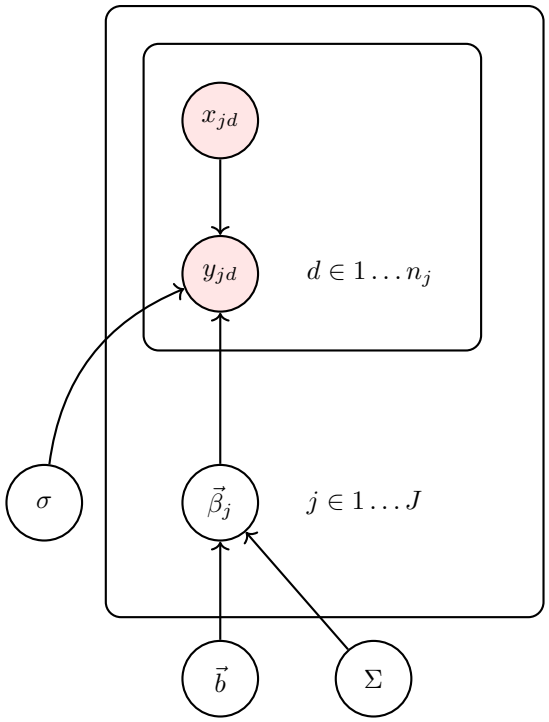


Figure 11: Bayesian network diagrams for a multilevel varying slopes and intercepts linear model.

model.

$$
\begin{aligned}
\text{for } i \in 1 \ldots n, \quad y_i &\sim N(\mu_i, \sigma^2), \\
\mu_i &= \beta_{[s_i]0} + \beta_{[s_i]1} x_i, \\
\mu_i &= b_0 + \zeta_{[s_i]0} + (b_1 + \zeta_{[s_i]1}) x_i, \\
&= \underbrace{b_0 + b_1 x_i}_{\text{fixed effects}} + \underbrace{\zeta_{[s_i]0} + \zeta_{[s_i]1} x_i}_{\text{random effects}}, \\
\text{for } j \in 1 \ldots J, \quad \vec{\zeta}_j &\sim N(0, \Sigma).
\end{aligned}
$$

As we can see from this, a multilevel normal linear model is equivalent to a non-multilevel model (the *fixed effects* models) plus a normally distributed random variation to the intercept and slope for each subject (the *random effects*). The fixed effects are sometimes known as *population level* effects: they apply to all observations. The random effects, on the other hand, vary across each different value of the grouping variable, which in this example is an individual participant in the experiment. Put another way, the fixed effects give the average effects in the population. The extent to which each individual varies around this average is given by the random effects. That the multilevel linear model can be described in terms of fixed and random effects is why these models are known as a *linear mixed effects model*.

We can implement this model in R using `lme4::lmer`.

```
M_ml <- lmer(Reaction ~ Days + (Days|Subject),
             data = sleepstudy)
```

The syntax here matches the fixed and random effects description of the model. The `Reaction ~ Days` tells us that the fixed effects model is a simple linear regression model with one predictor, and so with one intercept and one slope term. The `(Days|Subjects)` tells us that there is random variation to the slope for `Days` and implicitly there's also random variation to the intercept term. We could make the variation to the intercept term explicit by writing `(1 + Days|Subject)`, which is identical to `(Days|Subject)` because the `1 +` is included always by default just as it is included by default in fixed effects part, as it is in any R regression formula syntax.

The results of this model is obtained as follows.

```
summary(M_ml)
#> Linear mixed model fit by REML ['lmerMod']
#> Formula: Reaction ~ Days + (Days | Subject)
#>    Data: sleepstudy
#>
#> REML criterion at convergence: 1743.6
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -3.9536 -0.4634  0.0231  0.4634  5.1793
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev. Corr
#>  Subject  (Intercept) 612.10   24.741
#>           Days         35.07    5.922   0.07
#>  Residual             654.94   25.592
#> Number of obs: 180, groups:  Subject, 18
#>
#> Fixed effects:
#>             Estimate Std. Error t value
#> (Intercept)  251.405      6.825  36.838
#> Days          10.467      1.546   6.771
#>
#> Correlation of Fixed Effects:
#>      (Intr)
#> Days -0.138
```

The value of $\vec{b}$ is available under `Estimate` in the `Fixed effects`, and we can get these directly as follows.

```
b <- fixef(M_ml)
b
#> (Intercept)        Days
#>   251.40510    10.46729
```

Thus, the average effect of sleep deprivation on reaction time across all individuals is that their reaction time increases by 10.47 each day. Also, the average individual has an average reaction time of 251.41 on day 0 of the experiment, which means that this is the average reaction time of the average person generally. The values in the covariance matrix Σ and of the residual standard deviation σ can be obtained from the values provided under `Random effects`. These are available more directly as follows.

```
VarCorr(M_ml)
#>  Groups   Name        Std.Dev. Corr
#>  Subject  (Intercept) 24.7407
#>           Days         5.9221  0.066
#>  Residual             25.5918
```

Note that the covariance matrix is defined as follows.

$$\Sigma = \begin{bmatrix} \tau_0^2 & \tau_0\rho\tau_1 \\ \tau_0\rho\tau_1 & \tau_1^2 \end{bmatrix}.$$

We can obtain this directly as follows.

```
VarCorr(M_ml)$Subject %>%
  Matrix::bdiag() %>%
  as.matrix()
```

```
#>             (Intercept)      Days
#> (Intercept)  612.100158  9.604409
#> Days           9.604409 35.071714
```

The values of the standard deviations τ_0 and τ_1 are also available as follows.

```
s <- VarCorr(M_ml)$Subject %>% attr('stddev')
s
#> (Intercept)        Days
#>   24.740658    5.922138
```

The correlation matrix corresponding to Σ is obtained as follows.

```
P <- VarCorr(M_ml)$Subject %>% attr('correlation')
P
#>             (Intercept)       Days
#> (Intercept)  1.00000000 0.06555124
#> Days         0.06555124 1.00000000
```

From this, we have $\rho = \Sigma_{12} = \Sigma_{21} = 0.066$. More generally, given that

$$\Sigma = \begin{bmatrix} \tau_0^2 & \tau_0\rho\tau_1 \\ \tau_0\rho\tau_1 & \tau_1^2 \end{bmatrix} = \begin{bmatrix} \tau_0 & 0 \\ 0 & \tau_1 \end{bmatrix} \underbrace{\begin{bmatrix} 0 & \rho \\ \rho & 1 \end{bmatrix}}_{\text{corr. matrix}} \begin{bmatrix} \tau_0 & 0 \\ 0 & \tau_1 \end{bmatrix},$$

we can also obtain Σ as follows:

```
# create a diagonal matrix of the stddev
s <- diag(s)

Sigma <- s %*% P %*% s
Sigma
#>             [,1]      [,2]
#> [1,] 612.100158  9.604409
#> [2,]   9.604409 35.071714
```

The contours of the 2d normal distribution with mean vector $\vec{b}$ and covariance matrix Σ are shown in Figure 12.

The estimates of each $\vec{\beta}_j$ for $j \in 1 \ldots J$ can be obtained using the `coef` function.

```
coef(M_ml)$Subject %>%
  head()
#>     (Intercept)      Days
#> 308    253.6637 19.666262
#> 309    211.0064  1.847605
#> 310    212.4447  5.018429
#> 330    275.0957  5.652936
#> 331    273.6654  7.397374
#> 332    260.4447 10.195109
```

These estimates are superimposed on the contours of the 2d normal distribution in Figure 12. They are represented by the tip of the arrow corresponding to each subject. The base of the arrow, by contrast, represents the estimates of the coefficients in the non-multilevel varying-intercept and varying-slope model. As we can see, the estimates in the multilevel model are shrunk towards the centre of the $N(\vec{b}, \Sigma)$ distribution.

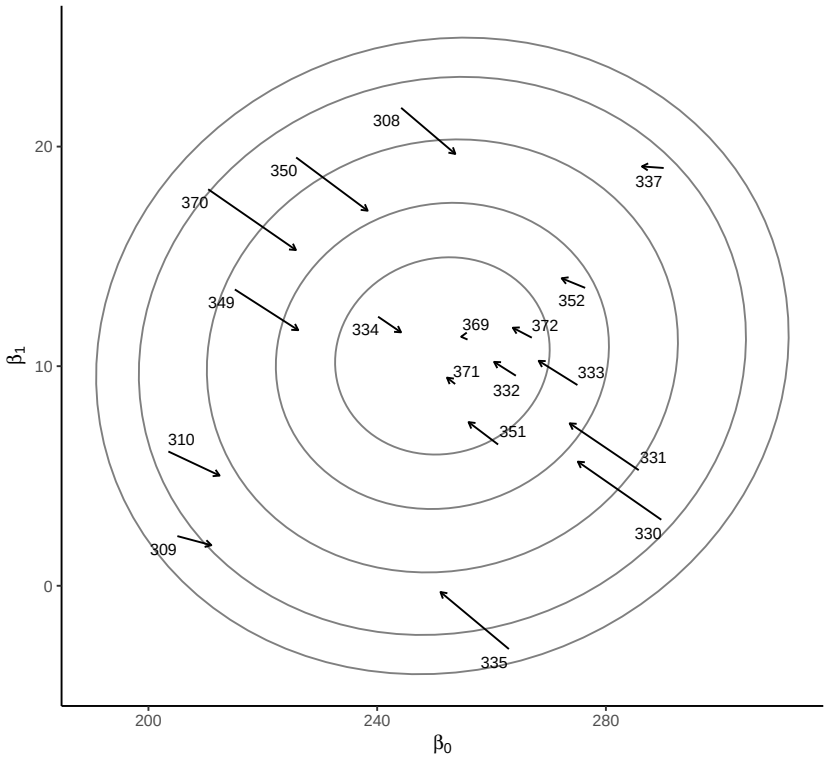


Figure 12: The contour plot shows the contours of the 2d normal distribution centered at $\vec{b}$ and whose covariance matrix is Σ. Superimposed on this are the estimates of the coefficients of each subject, both in the non-multilevel model (base of arrow) and in the multilevel model (tip of arrow). As we can see, the estimates of the coefficients in the multilevel are all shrunk towards the overall mean $\vec{b}$.

Inference

The multilevel model above, i.e.,

$$\begin{aligned} \text{for } i \in 1 \dots n, \quad & y_i \sim N(\mu_i, \sigma^2), \\ & \mu_i = \underbrace{b_0 + b_1 x_i}_{\text{fixed effects}} + \underbrace{\zeta_{[s_i]0} + \zeta_{[s_i]1} x_i}_{\text{random effects}}, \\ \text{for } j \in 1 \dots J, \quad & \vec{\zeta_j} \sim N(0, \Sigma), \end{aligned}$$

can be rewritten in matrix notation as follows:

$$\vec{y} = X\vec{b} + Z\vec{\gamma} + \vec{\epsilon}, \quad \vec{\gamma} \sim N(0, \Omega), \quad \vec{\epsilon} \sim N(0, I_n \sigma^2).$$

Here, $\vec{y}$ is an $n \times 1$ (column) vector. The matrix X is a $n \times 2$ matrix whose first column is a vector where each value is 1 and the second column is $\vec{x} = [x_1, x_2 \dots x_n]^\intercal$. The matrix Z is a $n \times 2J$ matrix. It is easiest to understand if viewed as the horizontal concatenation of two $n \times J$ matrices, Z^0 and Z^1:

$$Z = [Z^0, Z^1].$$

Row i and column j of Z^0 is Z^0_{ij}, which takes the value of 1 if and only if $s_i = j$. In other words, each row i of Z^0 is a J dimensional (row) vector whose values are all zero with the exception of one element, namely element s_i, whose value is 1. Likewise, in matrix Z^1, each row i is a J dimensional (row) vector whose values are all zero with the exception of one element, namely element s_i, whose value is x_i. The vector $\vec{\gamma}$ is $2J \times 1$ vector, and is easiest to understand at the vertical concatenation of two $J \times 1$ vectors:

$$\vec{\gamma} = \begin{bmatrix} \vec{\gamma}_0 \\ \vec{\gamma}_1 \end{bmatrix}.$$

The vector $\vec{\gamma}_0 = [\zeta_{1,0}, \zeta_{2,0} \ldots \zeta_{j,0} \ldots \zeta_{J,0}]^\intercal$, where $\zeta_{j,0}$ is the first element of $\vec{\zeta}_j$ from the original description of the model. Likewise, the vector $\vec{\gamma}_1 = [\zeta_{1,1}, \zeta_{2,1} \ldots \zeta_{j,1} \ldots \zeta_{J,1}]^\intercal$, where $\zeta_{j,1}$ is the second element of $\vec{\zeta}_j$. The $\vec{\gamma}$ is random vector with a $2J$ dimensional multivariate normal distribution with a zero mean vector and whose covariance matrix Ω has a $\Sigma_{[1,1]}$ as in its first J elements and $\Sigma_{2,2}$ and its remaining J elements. Its off-diagonal elements are zero except for elements $\Omega_{[j,J+j]}$ and $\Omega_{[J+j,j]}$, for each $j \in 1...J$, whose values are all $\Sigma_{[2,2]}$. The $n \times 1$ vector $\vec{\epsilon}$ is also a random vector distributed as a n dimensional multivariate normal distribution with zero mean vector and a covariance matrix $I_n\sigma^2$ where I_n is the n dimensional identity matrix. In other words, this is a diagonal matrix whose diagonal elements are all equal to σ^2.

From this, we have that $\vec{y}$ has a multivariate normal distribution:

$$\vec{y} \sim N(X\vec{b}, V), \quad V = Z\Omega Z^\intercal + I_n\sigma^2,$$

where V is a $n \times n$ covariance matrix. The density function for $\vec{y}$ is

$$\mathrm{P}(\vec{y}|X\vec{b}, V) = \frac{1}{(2\pi)^{n/2}|V|^{1/2}} e^{-\frac{1}{2}(\vec{y}-X\vec{b})^\intercal V^{-1}(\vec{y}-X\vec{b})},$$

and so the log the likelihood with respect to $\vec{b}$ and V is

$$L(\vec{b}, V|\vec{y}, X) \propto -\tfrac{1}{2}\left(\log|V^{-1}| + (\vec{y} - X\vec{b})^\intercal V^{-1}(\vec{y} - X\vec{b})\right).$$

If we knew V, then the maximum likelihood estimator of $\vec{b}$, $\hat{b}$, is

$$\hat{b} = (X^\intercal V^{-1} X)^{-1} X^\intercal V^{-1}\vec{y}.$$

Setting $\vec{b}$ in the likelihood function equal to $\hat{b}$ gives the *profile likelihood* function, which may be then maximized with respect to the parameters of V, which are ultimately based on the parameters Σ and σ.

The maximum likelihood estimate for variance parameters are biased, however. As an alternative to the likelihood function, the *restricted* (also known as, *residual*) maximum likelihood (REML) estimates are often used. These are obtained by maximizing the following variant of the likelihood function:

$$L_{\mathrm{reml}}(\vec{b}, V|\vec{y}, X) \propto -\tfrac{1}{2}\left(\log|V^{-1}| + (\vec{y} - X\vec{b})^\intercal V^{-1}(\vec{y} - X\vec{b}) + \log|X^\intercal V^{-1} X|\right).$$

Maximising this with respect to $\vec{b}$ leads to the same solution as above, but the estimates of the variance parameters are unbiased.

Varying intercepts or varying slopes only models

The above model allowed for random variation in both the intercepts and slopes but we can choose to have random variation in only one or the other. A varying intercept only multilevel model is defined as follows.

$$\begin{aligned} \text{for } i \in 1 \ldots n, \quad y_i &\sim N(\mu_i, \sigma^2), \\ \mu_i &= \beta_{[s_i]0} + b_1 x_i, \\ \text{for } j \in 1 \ldots J, \quad \beta_{j0} &\sim N(b_0, \tau_0^2), \end{aligned}$$

which can be rewritten, using the same reasoning as above,

$$\begin{aligned} \text{for } i \in 1 \ldots n, \quad y_i &\sim N(\mu_i, \sigma^2), \\ \mu_i &= b_0 + b_1 x_i + \zeta_{[s_i]0}, \\ \text{for } j \in 1 \ldots J, \quad \zeta_{j0} &\sim N(0, \tau_0^2). \end{aligned}$$

Using `lmer`, we would implement this as follows.

```
M_ml_vi <- lmer(Reaction ~ Days + (1|Subject),
                data = sleepstudy)
```

The fixed effects give us an estimate of the slope and intercept as before.

```
fixef(M_ml_vi)
#> (Intercept)        Days
#>   251.40510    10.46729
```

The random effects just provide a measure of standard deviation τ_0 for the random intercepts as well as residual standard deviation σ.

```
VarCorr(M_ml_vi)
#>  Groups   Name        Std.Dev.
#>  Subject  (Intercept) 37.124
#>  Residual             30.991
```

Absent here, compared to the varying intercepts and varying slopes model is the estimate for τ_1 and ρ. The estimates of the coefficients are obtained using `coef`, as before.

```
coef(M_ml_vi)$Subject %>%
  head()
#>     (Intercept)     Days
#> 308    292.1888 10.46729
#> 309    173.5556 10.46729
#> 310    188.2965 10.46729
#> 330    255.8115 10.46729
#> 331    261.6213 10.46729
#> 332    259.6263 10.46729
```

As we can see, all the subjects' slopes are identical and have the value of the estimate of b_1, which is 10.467. The lines corresponding to these coefficients are shown in Figure 13a.

The varying slope only multilevel model allows only the slopes to vary across subjects and it leaves the intercepts fixed. It is defined as follows.

$$\begin{aligned} \text{for } i \in 1 \dots n, \quad y_i &\sim N(\mu_i, \sigma^2), \\ \mu_i &= b_0 + \beta_{[s_i]1} + x_i, \\ \text{for } j \in 1 \dots J, \quad \beta_{j1} &\sim N(b_0, \tau_1^2), \end{aligned}$$

which can be rewritten

$$\begin{aligned} \text{for } i \in 1 \dots n, \quad y_i &\sim N(\mu_i, \sigma^2), \\ \mu_i &= b_0 + b_1 x_i + \zeta_{[s_i]1} x_i, \\ \text{for } j \in 1 \dots J, \quad \zeta_{j1} &\sim N(0, \tau_1^2). \end{aligned}$$

Using `lmer`, we would implement this as follows.

```
M_ml_vs <- lmer(Reaction ~ Days + (0+Days|Subject),
                data = sleepstudy)
```

The fixed effects give us an estimate of both the slope and intercept as with the previous models.

```
fixef(M_ml_vs)
#> (Intercept)        Days
#>   251.40510    10.46729
```

The random effect provide a measure of standard deviation τ_1 and σ.

```
VarCorr(M_ml_vs)
#>  Groups   Name Std.Dev.
#>  Subject  Days  7.260
#>  Residual      29.018
```

Absent here compared to the full model is the estimate for τ_0 and ρ. The estimates of the coefficients are obtained using `coef` as before.

```
coef(M_ml_vs)$Subject %>%
  head()
#>     (Intercept)       Days
#> 308    251.4051 20.0866918
#> 309    251.4051 -4.2326711
#> 310    251.4051 -0.8189202
#> 330    251.4051  9.1273878
#> 331    251.4051 10.6754843
#> 332    251.4051 11.5352979
```

Here, all the subjects' intercepts are identical and have the value of the estimate of b_0, which is 251.405. The lines corresponding to these coefficients are shown in Figure 13b.

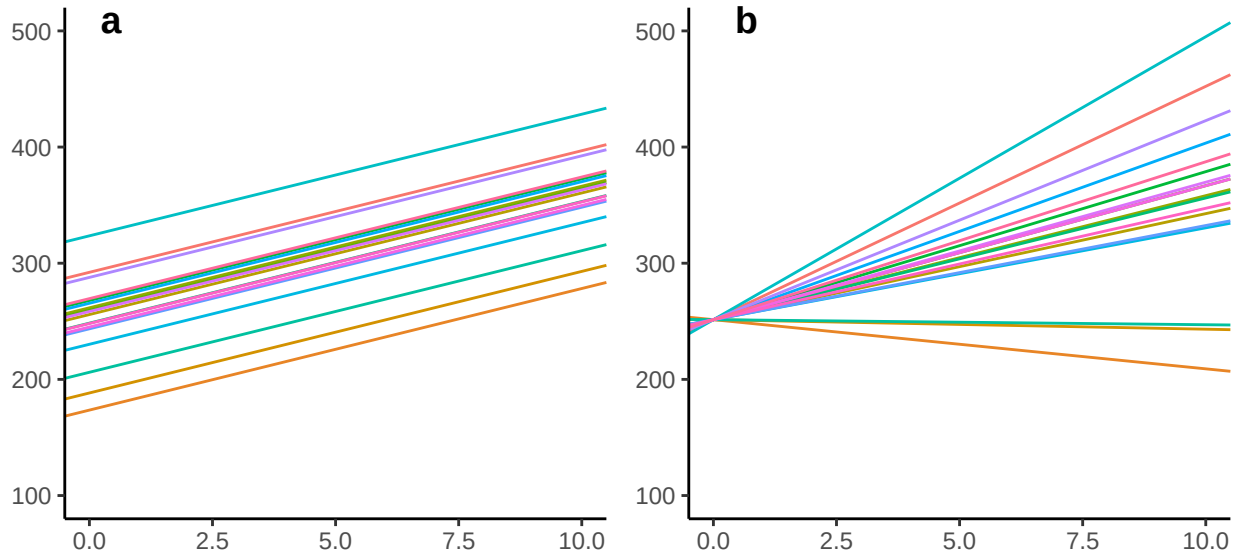


Figure 13: Lines of best fit for each data group in a varying intercepts only a) or varying slopes only b) multilevel linear model.

One final variant of the full model is where we allow for both varying slopes and intercepts but assume no correlation between each β_{j0} and β_{j1}. In other words, we assume that these are drawn from independent normal distributions.

$$\begin{aligned} \text{for } i \in 1 \dots n, \quad y_i &\sim N(\mu_i, \sigma^2), \\ \mu_i &= \beta_{[s_i]0} + \beta_{[s_i]1} x_i, \\ \text{for } j \in 1 \dots J, \quad \beta_{j0} &\sim N(b_0, \tau_0^2), \\ \beta_{j1} &\sim N(b_1, \tau_1^2), \end{aligned}$$

which is identical to each β_j vector being drawn from a diagonal covariance matrix, i.e. where $\rho = 0$.

Using `lmer`, we would implement this as follows.

```
M_ml_diag <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject),
                  data = sleepstudy)
```

We can obtain the same model using the following formula syntax.

```
M_ml_diag2 <- lmer(Reaction ~ Days + (Days||Subject),
                   data = sleepstudy)
```

The fixed effects give us an estimate of both the slope and intercept as with each of the previous models.

```
fixef(M_ml_diag2)
#> (Intercept)        Days
#>   251.40510    10.46729
```

The random effect provide a measure of the τ_0, τ_1 and σ standard deviations.

```
VarCorr(M_ml_diag2)
#>  Groups    Name        Std.Dev.
#>  Subject   (Intercept) 25.0513
#>  Subject.1 Days         5.9882
#>  Residual              25.5653
```

The only quantity that is absent here compared to the full model is the estimate for ρ. The estimates of the coefficients are obtained using `coef` as before.

```
coef(M_ml_diag2)$Subject %>%
  head()
#>     (Intercept)      Days
#> 308    252.9178 19.790783
#> 309    211.0312  1.868110
#> 310    212.2241  5.079492
#> 330    275.9240  5.498636
#> 331    274.3196  7.273348
#> 332    260.6271 10.158792
```

Here, both the intercepts and slopes vary across subjects.

Models for nested and crossed data

In all the examples considered thus far, our multilevel models had only two levels, which we can denote level 0 and level 1. For example, we had rats (level 0) within batches (level 1), observations of per capita alcohol consumption in different years (level 0) within different countries (level 1), individual observations of reaction times on different days (level 0) grouped by different experimental subjects (level 1). We can easily have groups within groups, and we usually refer to these models as *nested* multilevel models. Here, we will consider nested linear mixed effects models.

Let us consider the following data which is based on a subset of the `classroom` data made available in R package `WWGbook`.

```
classroom_df <- read_csv('data/classroom.csv')
classroom_df
#> # A tibble: 1,190 x 5
#>    mathscore   ses classid schoolid classid2
#>        <dbl> <dbl>   <dbl>    <dbl>    <dbl>
#>  1       480  0.46     160        1        1
#>  2       569 -0.27     160        1        1
#>  3       567 -0.03     160        1        1
#>  4       532 -0.38     217        1        2
#>  5       478 -0.03     217        1        2
#>  6       515  0.76     217        1        2
#>  7       503 -0.03     217        1        2
#>  8       509  0.2      217        1        2
#>  9       510  0.64     217        1        2
#> 10       473  0.13     217        1        2
#> # ... with 1,180 more rows
```

Each observation is of a child (level 0), and for each one we have their mathematics test score in their first grade (`mathscore`) and their home socioeconomic status (`ses`). There are n children in total, indexed by $i \in 1 \dots n$. Each child is in one of J classes (level 1, denoted by `classid`), and each class is within one of K schools (level 2, denoted by `schoolid`). This hierarchical arrangement of the data is shown in Figure 14.

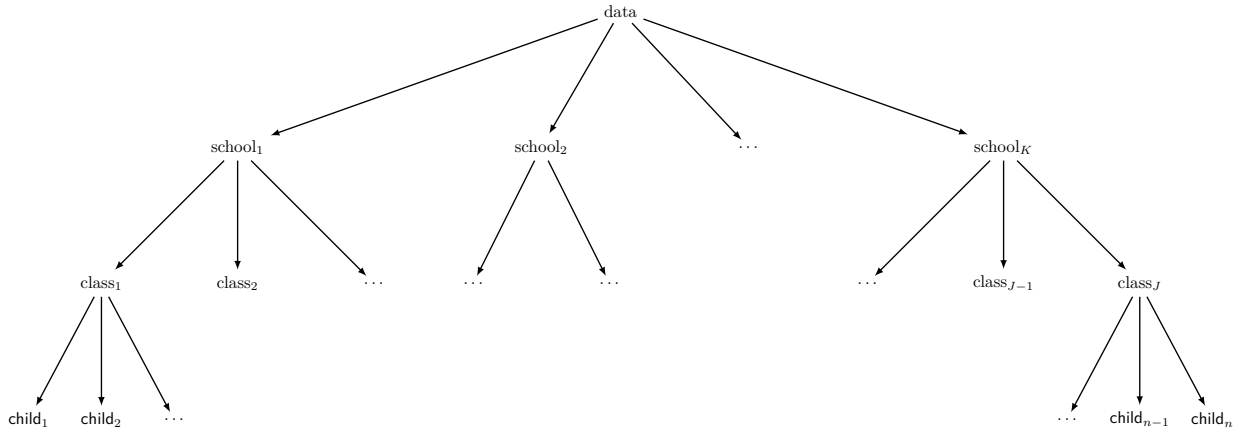


Figure 14: The hierarchical arrangement of the data in the `classroom_df` data set. Each one of the N children is in one of J classes and each one of the J classes is in one of K schools.

The relationship between `ses` and `mathscore` can vary across different schools, see Figure 15a. Furthermore, *within* each school, there may be variation in the effect of `ses` on `mathscore` across different classes, see Figure 15b. Formally, a three-level multilevel varying intercepts and slopes linear model corresponding to the above problem may be presented as follows.

$$\begin{aligned}
\text{for } i \in 1 \dots n, \quad & y_i \sim N(\mu_i, \sigma^2), \\
& \mu_i = \gamma_{[c_i]0} + \gamma_{[c_i]1} x_i, \\
\text{for } j \in 1 \dots J, \quad & \vec{\gamma}_j \sim N(\vec{\beta}_{[s_j]}, \Sigma_c), \\
\text{for } k \in 1 \dots K, \quad & \vec{\beta}_k \sim N(\vec{b}, \Sigma_s),
\end{aligned}$$

For each one of the n children, we have their `mathscore` variable y_i, their `ses` variable x_i, and an indicator variable $c_i \in 1 \dots J$. The indicator variable indicates which of the J different classes the child is a member of. For each of one of these J classes, we have another indicator variable $s_j \in 1 \dots K$. This indicates the school that class j is a member of. For example, if $s_j = k$, this means that class j is within school k. From this description, we see that there are J vectors of coefficients, i.e. $\vec{\gamma}_1, \vec{\gamma}_2 \dots \vec{\gamma}_J$, and each one corresponds to one of the J different classes in the data set. Each vector $\vec{\gamma}_j$ is a sample from a multivariate normal distribution centered at $\vec{\beta}_{[s_j]}$, where s_j is the school that class j is a member of. The covariance matrix of this multivariate normal distribution is Σ_c. The K different vectors $\vec{\beta}_1, \vec{\beta}_2 \dots \vec{\beta_K}$ are themselves drawn from another multivariate normal distribution whose center is $\vec{b}$ and whose covariance matrix is Σ_s.

Using the same reasoning as above, we may rewrite $\vec{\gamma}_j \sim N(\vec{\beta}_{[s_j]}, \Sigma_c)$ as $\vec{\gamma}_j = \vec{\beta}_{[s_j]} + \vec{\xi}_j$ where $\vec{\xi}_j \sim N(0, \Sigma_c)$. Likewise, we may rewrite $\vec{\beta}_k \sim N(\vec{b}, \Sigma_s)$ as $\vec{\beta}_k = \vec{b} + \vec{\zeta}_k$ where $\vec{\zeta}_k \sim N(0, \Sigma_s)$. Thus, we have

$$\begin{aligned}
\vec{\gamma}_j &= \vec{\beta}_{[s_j]} + \xi_j, \\
&= \vec{b} + \zeta_{[s_j]} + \xi_j.
\end{aligned}$$

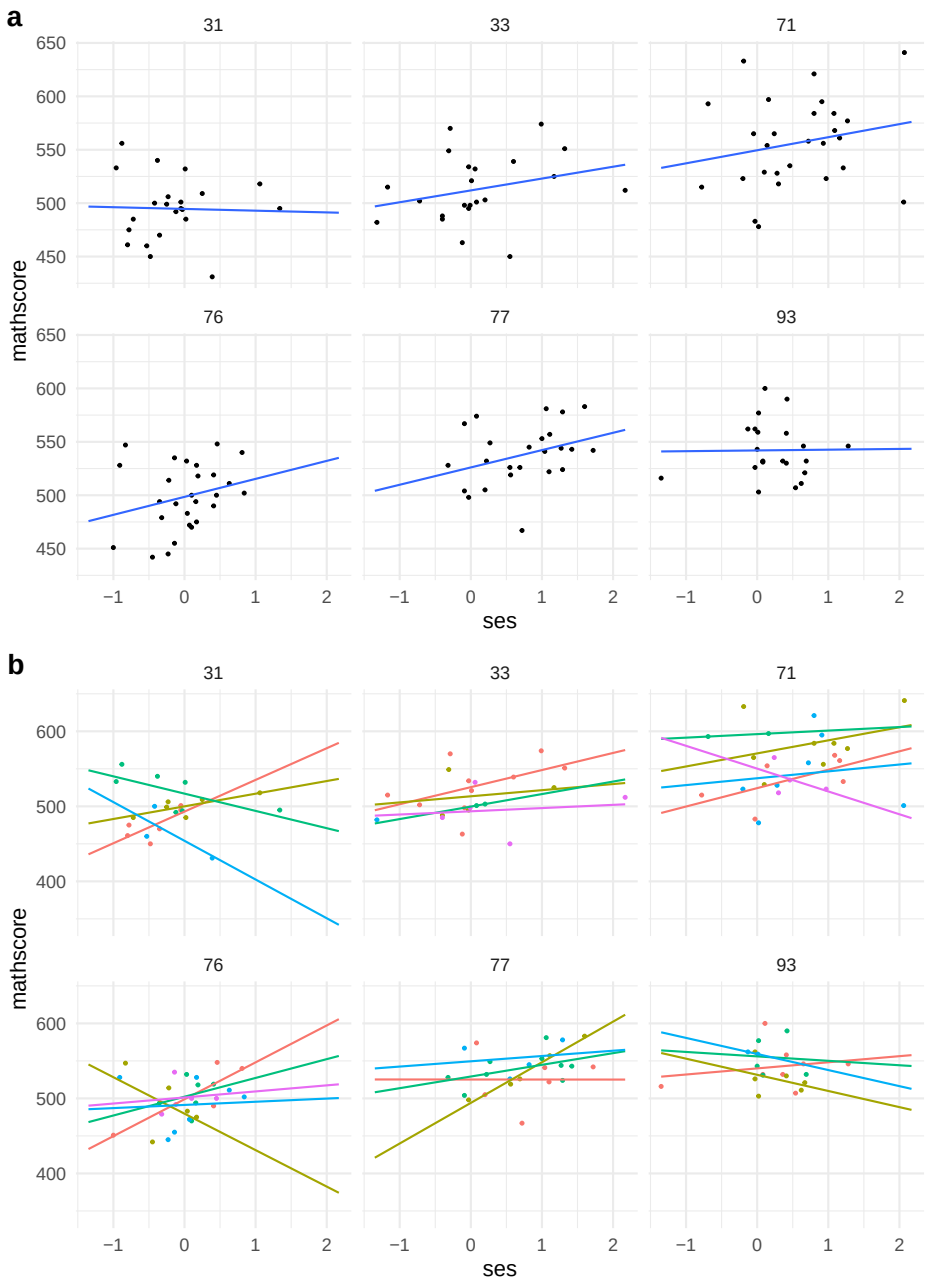


Figure 15: In each scatterplot in both subfigure a and b, the points represent individual children, and for each one, we have their mathematics test score plotted against their home socioeconomic status. Each scatterplot is from a different school. In subfigure a, the line of best fit for the children in the school is shown. In subfigure b, a separate line of best fit for each class within each school is shown.

This allows us to rewrite the original model as follows.

$$\begin{aligned}
\text{for } i \in 1 \dots n, \quad & y_i \sim N(\mu_i, \sigma^2), \\
& \mu_i = \underbrace{b_0 + b_1 x_i}_{\text{fixed effects}} + \underbrace{\zeta_{[z_i]0} + \zeta_{[z_i]1} x_i}_{\text{school random effects}} + \underbrace{\xi_{[c_i]0} + \xi_{[c_i]1} x_i}_{\text{class random effects}}, \\
\text{for } j \in 1 \dots J, \quad & \vec{\xi}_j \sim N(0, \Sigma_c), \\
\text{for } k \in 1 \dots K, \quad & \vec{\zeta}_k \sim N(0, \Sigma_s).
\end{aligned}$$

Here, for notational simplicity, we have used a new indicator variable $z_i = s_{[c_i]} \in 1 \dots K$ that indicates the school that child i is a member of.

Using `lmer`, we can implement this model as follows.

```
M_classroom <- lmer(mathscore ~ ses + (ses|classid) + (ses|schoolid),
                    data = classroom_df)
```

As we can see, we simply use random effects terms, one for random variation by school and the other for random variation by class. Before, we proceed, by examining the results of `VarCorr`, we see that there is a perfect correlation between the random slopes and random intercepts due to class.

```
VarCorr(M_classroom)
#>  Groups   Name        Std.Dev. Corr
#>  classid  (Intercept)  8.33529
#>           ses          0.91453 1.000
#>  schoolid (Intercept) 15.44468
#>           ses          7.24255 0.443
#>  Residual             32.90608
```

This indicates overparameterization in the model, and can be avoided by removing the possibility of a correlation between the slopes and intercepts for class.

```
M_classroom <- lmer(mathscore ~ ses + (ses||classid) + (ses|schoolid),
                    data = classroom_df)
```

Now, just like previous linear mixed effects models, our fixed effects will consist of a single slope and intercept term.

```
fixef(M_classroom)
#> (Intercept)         ses
#>   522.82481    11.20433
```

From this, we see that at a population level, every unit increase in `ses` leads to a 11.2 increase in `mathscore`. Information concerning the random effects are available from `VarCorr`.

```
VarCorr(M_classroom)
#>  Groups    Name        Std.Dev. Corr
#>  classid   (Intercept)  8.3964
#>  classid.1 ses          0.0000
#>  schoolid  (Intercept) 15.4398
#>            ses          7.3039  0.470
#>  Residual              32.8950
```

From this, amongst other things, we see there is a minimal random variation in the `ses` slopes across the different classes.

The variable `classid` unambiguously identified the class of children by giving each class in each school a unique identifier. In some cases, we do not have unique identifiers for nested groups. As an example, consider the `classid2` variable in the `classroom_df` data sets. Let us draw a sample of observations from `classroom_df`.

```
classroom_df %>%
  sample_n(10)
#> # A tibble: 10 x 5
#>    mathscore    ses classid schoolid classid2
#>        <dbl>  <dbl>   <dbl>    <dbl>    <dbl>
#>  1       524  0.290      41       14        1
#>  2       519 -0.03      102       26        2
#>  3       552 -0.03      149       53        2
#>  4       552 -0.61       42       75        2
#>  5       499 -0.24      117       55        3
#>  6       522 -0.05      223       29        2
#>  7       473  0.13      217        1        2
#>  8       526  0.04       75       97        1
#>  9       557  0.8        29       72        1
#> 10       478 -0.03      217        1        2
```

Unlike `classid`, `classid2` provides an identifier for each class that is *relative* to its school. Thus, we have class `1` in school `41`, and class `1` in school `97`. These are completely unrelated classes. In order to deal with variables like this, we must use an alternative formula syntax. For example, in order to have a varying intercepts only model, where the intercept varies by class and school, and assuming we are using the `classid2` variable, we use the following syntax in the formula to indicate that the values of `classid2` are relative to that of `schoolid`.

```
M_classroom_vi <- lmer(mathscore ~ ses + (1|schoolid/classid2),
                       data = classroom_df)
```

This is identical to the following model.

```
M_classroom_vi2 <- lmer(mathscore ~ ses + (1|schoolid) + (1|schoolid:classid2),
                        data = classroom_df)
```

The syntax in this second version makes it clear that we are effectively creating a unique identifier for the class by combining the values of `schoolid` and `classid2`. These two models, `M_classroom_vi` and `M_classroom_vi2`, are identical to that which would have been obtained had we used the unambiguous class identifier `classid` as in the following model.

```
M_classroom_vi3 <- lmer(mathscore ~ ses + (1|schoolid) + (1|classid),
                        data = classroom_df)
```

An alternative multilevel arrangement of data is known as a *crossed* structure. Again, this is easiest to illustrate by way of an example. Consider a type of cognitive psychology experiment known as a lexical decision task, which is used to understand the basic cognitive processes involved in reading words. In this experiment, multiple subjects are each shown multiple different letter strings, e.g. *dinosaur*, *morhet*, *children*, etc., and they have to respond as to whether the string is a word or not, and their reaction times to do so are also recorded. Thus, we would have subjects $s_1, s_2 \dots s_j \dots s_J$ being shown letter strings $w_1, w_2 \dots w_k \dots w_K$, and from each subject for each string, we have a response, e.g. a yes/no binary response, or reaction, or both. Over the entire experiment, our total set of responses could be denoted $r_1, r_2 \dots r_n$. These are our level 0 observations. Crucially, each response is cross classified as belonging to a particular subject and a particular letter string. See Figure 16.

In general, when dealing with crossed multilevel structures, we simply consider the random variation of slopes or intercepts according to more than one grouping variable. As an example, consider the British Lexicon Project data that we also considered in Chapter 3. This provides us with data from a large lexical decision experiment, and a small fraction of this data is provided in the file `data/blp-short2.csv`.

```
blp_df <- read_csv('data/blp-short2.csv')
blp_df
#> # A tibble: 662 x 6
```

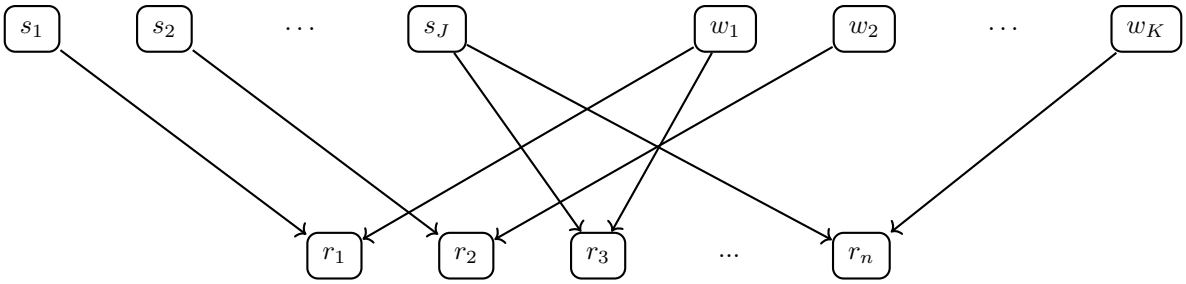


Figure 16: A *crossed* multilevel data arrangement, such as would arise in a lexical decision experiment. Each r_i is grouped under one of the $s_1, s_2 \ldots s_J$ and also one of the $w_1, w_2 \ldots w_K$.

```
#>    spelling participant    rt old20 nletters  freq
#>    <chr>          <dbl> <dbl> <dbl>    <dbl> <dbl>
#>  1 encode             4   470   2.2        6    93
#>  2 encode            46   599   2.2        6    93
#>  3 encode            78   403   2.2        6    93
#>  4 encode            72   975   2.2        6    93
#>  5 encode            32   504   2.2        6    93
#>  6 encode            52   588   2.2        6    93
#>  7 encode            68   418   2.2        6    93
#>  8 encode            10   718   2.2        6    93
#>  9 encode            54   742   2.2        6    93
#> 10 encode             6   776   2.2        6    93
#> # ... with 652 more rows
```

Here, each `rt` observation corresponds to a particular participant and a particular word (`spelling`). There are 58 distinct words and 78 distinct participants.

For simplicity, let us consider the multilevel linear model whereby average `rt` varies by both `participant` and `spelling`. This model is as follows.

$$\begin{aligned} \text{for } i \in 1 \ldots n, \quad & y_i \sim N(\mu_i, \sigma^2), \\ & \mu_i = \beta_{[s_i]} + \gamma_{[w_i]}, \\ \text{for } j \in 1 \ldots J, \quad & \beta_j \sim N(b_s, \tau_s^2), \\ \text{for } k \in 1 \ldots K, \quad & \gamma_k \sim N(b_w, \tau_w^2). \end{aligned}$$

We can rewrite β_j as $\beta_j = b_s + \zeta_j$, where $\zeta_j \sim N(0, \tau_s^2)$, and rewrite γ_j as $\gamma_j = b_w + \xi_k$, where $\xi_k \sim N(0, \tau_w^2)$. This leads to

$$\begin{aligned} \text{for } i \in 1 \ldots n, \quad & y_i \sim N(\mu_i, \sigma^2), \\ & \mu_i = \nu + \zeta_{[s_i]} + \xi_{[w_i]}, \\ \text{for } j \in 1 \ldots J, \quad & \beta_j \sim N(0, \tau_s^2), \\ \text{for } k \in 1 \ldots K, \quad & \gamma_k \sim N(0, \tau_w^2), \end{aligned}$$

where $\nu = b_s + b_w$. This is implemented in `lmer` as follows.

```
M_blp <- lmer(rt ~ 1 + (1|participant) + (1|spelling),
              data = blp_df)
```

From this, we see the estimate of ν is as follows.

```
fixef(M_blp)
#> (Intercept)
#>    586.2724
```

The estimates of τ_s, τ_w, and σ are as follows.

```
VarCorr(M_blp)
#>  Groups      Name        Std.Dev.
#>  participant (Intercept)  77.738
#>  spelling    (Intercept)  64.128
#>  Residual                137.229
```

Group level predictors

In the examples thus far, the predictor variables were at the bottom level, level 0. For example, at level 0, we had children, and the values of the predictor variable `ses` were specific to each child. Consider the following two related data sets, which are based on the `MathAchieve` and `MathAchSchool` data sets, respectively, available in the `nlme` package:

```
mathachschool_df <- read_csv('data/mathachieveschool.csv')
mathach_df <- read_csv('data/mathachieve.csv')

mathach_df
#> # A tibble: 7,185 x 6
#>    pupil school minority sex       ses mathach
#>    <chr> <chr>  <chr>    <chr>   <dbl>   <dbl>
#>  1 p1    s1224  No       Female -1.53    5.88
#>  2 p2    s1224  No       Female -0.588  19.7
#>  3 p3    s1224  No       Male   -0.528  20.3
#>  4 p4    s1224  No       Male   -0.668   8.78
#>  5 p5    s1224  No       Male   -0.158  17.9
#>  6 p6    s1224  No       Male    0.022   4.58
#>  7 p7    s1224  No       Female -0.618  -2.83
#>  8 p8    s1224  No       Male   -0.998   0.523
#>  9 p9    s1224  No       Female -0.888   1.53
#> 10 p10   s1224  No       Male   -0.458  21.5
#> # ... with 7,175 more rows
mathachschool_df
#> # A tibble: 160 x 7
#>    school  size sector   pracad disclim himinty meanses
#>    <chr>  <dbl> <chr>     <dbl>   <dbl>   <dbl>   <dbl>
#>  1 s1224    842 Public     0.35   1.60        0  -0.428
#>  2 s1288   1855 Public     0.27   0.174       0   0.128
#>  3 s1296   1719 Public     0.32  -0.137       1  -0.42
#>  4 s1308    716 Catholic   0.96  -0.622       0   0.534
#>  5 s1317    455 Catholic   0.95  -1.69        1   0.351
#>  6 s1358   1430 Public     0.25   1.54        0  -0.014
#>  7 s1374   2400 Public     0.5    2.02        0  -0.007
#>  8 s1433    899 Catholic   0.96  -0.321       0   0.718
#>  9 s1436    185 Catholic   1     -1.14        0   0.569
#> 10 s1461   1672 Public     0.78   2.10        0   0.683
#> # ... with 150 more rows
```

In the `mathach_df` data, for each pupil in a school, we have variable related to their ethnic minority status, sex, home socioeconomic background, and their mathematical achieve score (`mathach`). Thus, in this data set, we have level 1 grouping variable, `school`, but all predictors are at level 0. The `mathachschool_df` data set, on the other hand, provides us with predictors related to the schools. For example, we have the school's size, whether it is public or private, etc., the proportion of students in the school on an academic track (`pracad`), a measure of the discrimination climate in the school (`disclim`), whether there is a high proportion of minority

students (`himinty`), and the mean socioeconomic status of the school (`meanses`). In order to use the school level and pupil level predictors together, we will join these two data frames by `school`.

```
mathach_df2 <- left_join(mathach_df,
                         mathachschool_df,
                         by = 'school')
```

Group level predictors present do not special difficulty for linear mixed effects models but can not be treated identically to individual observation level predictors. For example, just like in similar examples above, we use a multilevel varying slope and varying intercept model to look at how `ses` affects `mathach` scores and how this effect varies by `school`. The `ses` variable is a pupil level variable, and we would be able to treat other pupil level variable s in a similar way. We would not, however, be able to use group level predictors, e.g. `pracad`, `disclim`, etc., in a similar way. For example, we could not perform the following model.

```
# does not work
lmer(mathach ~ pracad + (pracad|school), data = mathach_df2)
```

This is simply because the effect of `pracad` on `mathach` can not vary by school: each school has exactly one value of `pracad`.

We still can easily use `pracad` in a linear mixed effect model, for example as follows.

```
# does work
# glp: group level predictor
M_glp <- lmer(mathach ~ pracad + (1|school), data = mathach_df2)
```

In this case, our model is that a pupil's `mathach` varies by the school's `pracad`, and there is variation across schools in the terms of the average `mathac`, with the latter effect being due to the random intercepts term. In this model, the fixed effects coefficients are as follows.

```
fixef(M_glp)
#> (Intercept)      pracad
#>    8.384325    8.232278
```

We therefore see that a unit increase in `pracad` leads to a 8.23 increase in `mathach`. The random effects are as follows.

```
VarCorr(M_glp)
#>  Groups   Name        Std.Dev.
#>  school   (Intercept) 2.0614
#>  Residual             6.2565
```

From this, we see that the standard deviation in the intercept terms across schools is 2.06.

A more interesting situation involving group level predictors arises in the following situation. Assuming that our outcome variable is still `mathach`, we could model how this varies by the pupil's `ses`, and how this effect varies across schools, using a multilevel varying slopes and intercepts model. We could also use a school level predictor, for example, `himinty`, and model how the school-level slopes and intercepts for the `ses` effect vary by it.

Formally, this model would be as follows.

$$\begin{aligned} \text{for } i \in 1 \dots n \quad & y_i \sim N(\mu_i, \sigma^2), \\ & \mu_i = \beta_{[s_i]0} + \beta_{[s_i]1} x_i, \\ \text{for } j \in 1 \dots J \quad & \vec{\beta}_j \sim N(\vec{b}_0 + \vec{b}_1 z_j, \Sigma), \end{aligned}$$

where y_i is the `mathach` score and x_i is the `ses` score of pupil i, and z_j is the `himinty` score of school j. As

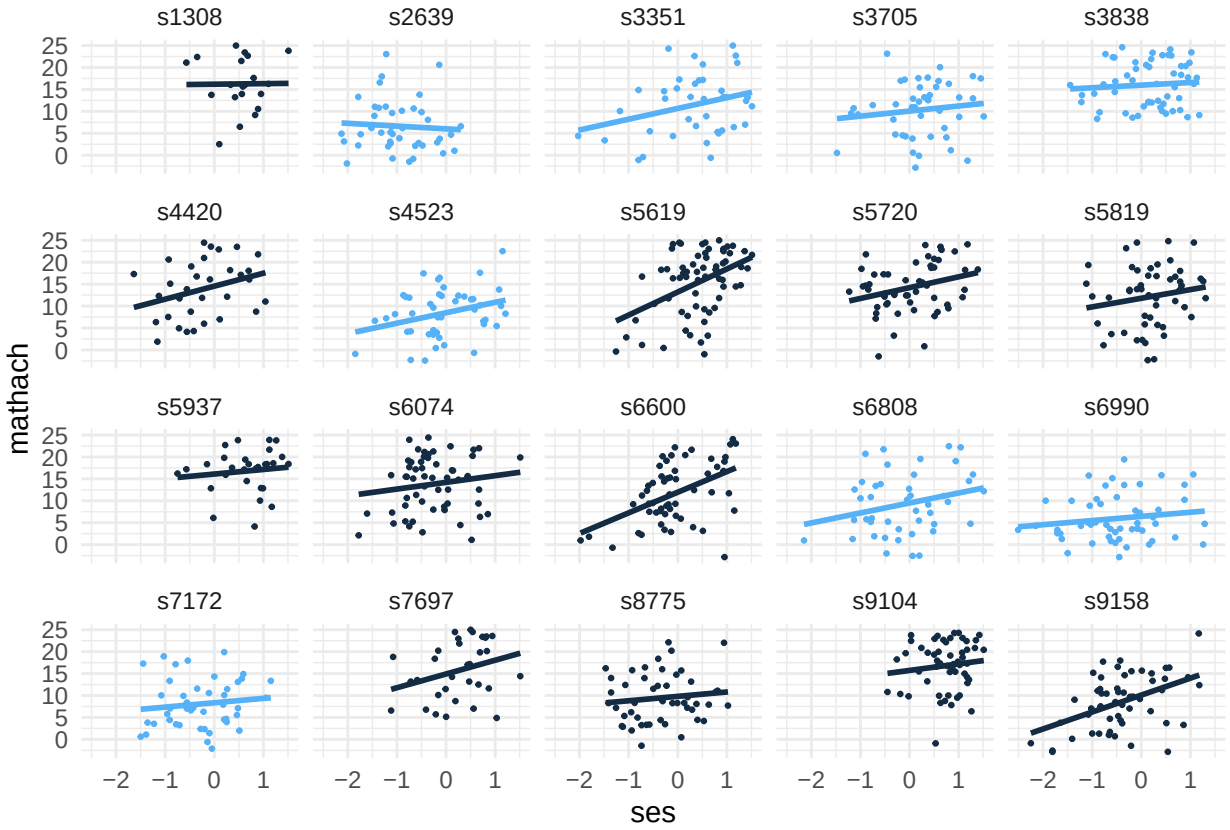


Figure 17: How mathematical achievement varies by a child's socioeconomic background across a sample of different school. Schools are colour coded to indicate whether they have high ethnic minority proportions (lighter colour) or not.

before, we can rewrite $\vec{\beta}_j$ as follows,

$$\begin{aligned} \vec{\beta}_j &= \vec{b}_0 + \vec{b}_1 z_j + \vec{\zeta}_j, \\ \begin{bmatrix} \beta_{j0} \\ \beta_{j1} \end{bmatrix} &= \begin{bmatrix} b_{00} \\ b_{01} \end{bmatrix} + \begin{bmatrix} b_{10} \\ b_{11} \end{bmatrix} z_j + \begin{bmatrix} \zeta_{j0} \\ \zeta_{j1} \end{bmatrix}, \end{aligned}$$

where

$$\vec{\zeta}_j = \begin{bmatrix} \zeta_{j0} \\ \zeta_{j1} \end{bmatrix} \sim N(0, \Sigma).$$

From this, we have

$$\begin{aligned} \text{for } i \in 1 \dots n \quad y_i &\sim N(\mu_i, \sigma^2), \\ \mu_i &= (b_{00} + b_{10} z_{[s_i]} + \zeta_{[s_i]0}) + (b_{01} + b_{11} z_{[s_i]} + \zeta_{[s_i]1}) x_i, \\ &= \underbrace{b_{00} + b_{01} x_i + b_{10} z_{[s_i]} + b_{11} z_{[s_i]} x_i}_{\text{fixed effects}} + \underbrace{\zeta_{[s_i]0} + \zeta_{[s_i]1} x_i}_{\text{random effects}}, \\ \text{for } j \in 1 \dots J \quad \vec{\zeta}_j &\sim N(0, \Sigma), \end{aligned}$$

From this we see that the role of the group level predictor $z_{[s_i]}$ interacts with the pupil level predictor x_i.

To implement this model in R, we used the joined data set `mathach_df2`. Note that `himinty` is coded as binary variable where a value of `1` means that there is a high proportion of ethic minorities in the school.

```
# glp: group level predictor
M_glp <- lmer(mathach ~ ses + himinty + ses:himinty + (ses|school),
              data = mathach_df2)
```

The standard deviations of variation in random intercepts and slopes, and their correlation is as follows.

```
VarCorr(M_glp)
#>  Groups   Name        Std.Dev. Corr
#>  school   (Intercept) 2.05931
#>           ses         0.60327  -0.285
#>  Residual             6.06828
```

The fixed effects coefficients are as follows:

```
fixef(M_glp)
#> (Intercept)         ses     himinty ses:himinty
#>  13.1542443   2.5436697  -1.8593332  -0.5760956
```

From this, we see that the role of `himinty` changing its value from 0 to 1 decreases the intercept by 1.86 and decreases the slope of the effect of `ses` by 0.58.

Bayesian multilevel models

In practical terms, Bayesian approaches to multilevel modelling differ from their classical counterparts only by their method of inference. In both cases, we begin by specifying a probabilistic model of the observed data. For example, if our data was the sleep deprivation and reaction time data from above, regardless of whether a Bayesian or a classical approach is being taken, a reasonable model of this data would the multilevel varying slopes and intercepts linear model that we have used already and that can be stated as follows:

$$\begin{aligned} \text{for } i \in 1 \dots n, \quad & y_i \sim N(\mu_i, \sigma^2), \\ & \mu_i = b_0 + b_1 x_i + \zeta_{[s_i]0} + \zeta_{[s_i]1} x_i, \\ \text{for } j \in 1 \dots J, \quad & \vec{\zeta_j} \sim N(0, \Sigma). \end{aligned}$$

where y_i, s_i, x_i are, respectively, the reaction time, subject index, and days of sleep deprivation corresponding to observation i. In general, how we specify this probabilistic model and the choices available to us are identical in both the Bayesian and classical approaches. Having specified the model, however, the Bayesian and classical approaches differ in terms of how they perform the inference of the unknown parameters. As we have seen in previous chapters, classical approaches infer estimators of these unknowns, for example, by maximum likelihood estimation, and then consider the sampling distribution of these estimators for hypothetical true values of the parameters. From this, we obtain p-values for hypothesis tests, null or otherwise, and confidence intervals, and related concepts. Bayesian approaches, on the other hand, first posit a *prior* probability distribution over the unknown parameters. In the above model, for example, we can parameterize it in terms of 6 unknowns:

$$b_0, b_1, \sigma, \tau_0, \tau_1, \rho,$$

where τ_0^2, τ_1^2 are the diagonal elements of Σ, and ρ is its off-diagonal element. Bayesian methods will therefore posit a probability distribution $\mathrm{P}(b_0, b_1, \sigma, \tau_0, \tau_1, \rho)$ over this six dimensional space. This prior is then combined, via Bayes' rule, with the *likelihood* function over the parameters to result in the posterior distribution. The posterior distribution tells us the probability that the true values of the parameters are a particular values, conditional on all the assumptions of the model.

If we accept default settings for priors and other quantities, performing a Bayesian multilevel linear model using `brms::brm` is identical to using `lme4::lmer`. For example, the following implements the Bayesian counterpart of the `lmer` based model that we used on Page 15, which clearly differs only by using the command `brm` instead of `lmer`:

```
M_bayes <- brm(Reaction ~ Days + (Days|Subject),
               data = sleepstudy)
```

```
#> Running /usr/local/lib/R/bin/R CMD SHLIB foo.c
#> make[1]: Entering directory '/tmp/RtmpjmDqdt'
#> gcc -I"/usr/local/lib/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"  -I"/usr/
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
#>                  from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown ty
#>   613 | namespace Eigen {
#>       | ^~~~~~~~~
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected
#>   613 | namespace Eigen {
#>       |                 ^
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file
#>    96 | #include <complex>
#>       |          ^~~~~~~~~
#> compilation terminated.
#> make[1]: *** [/usr/local/lib/R/etc/Makeconf:167: foo.o] Error 1
#> make[1]: Leaving directory '/tmp/RtmpjmDqdt'
```

The main results of this model may be obtained by `summary(M_bayes)`, or equivalently by just typing the name of the model `M_bayes`.

```
M_bayes
#>  Family: gaussian
#>   Links: mu = identity; sigma = identity
#> Formula: Reaction ~ Days + (Days | Subject)
#>    Data: sleepstudy (Number of observations: 180)
#> Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
#>          total post-warmup samples = 4000
#>
#> Group-Level Effects:
#> ~Subject (Number of levels: 18)
#>                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> sd(Intercept)          27.04      7.08    15.69    42.04 1.00     1750     2591
#> sd(Days)                6.57      1.49     4.19     9.96 1.00     1701     2055
#> cor(Intercept,Days)     0.08      0.29    -0.48     0.64 1.00     1146     1907
#>
#> Population-Level Effects:
#>           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> Intercept   250.95      7.25   236.35   264.98 1.00     1626     2342
#> Days         10.46      1.66     7.24    13.83 1.00     1486     1985
#>
#> Family Specific Parameters:
#>       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
#> sigma    25.91      1.58    23.11    29.11 1.00     3568     2920
#>
#> Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
#> and Tail_ESS are effective sample size measures, and Rhat is the potential
#> scale reduction factor on split chains (at convergence, Rhat = 1).
```

In this output, amongst other information, for each one of the six variables $b_0, b_1, \sigma, \tau_0, \tau_1, \rho$, we have the mean

of the posterior distribution (`Esimate`), the standard deviation of the posterior distribution (`Est.Error`), and the upper and lower limits of the 95% high posterior density interval (`l-95% CI`, `u-95% CI`).

The prior on `M_bayes` can be seen as follows:

```
M_bayes$prior
#>                         prior     class      coef   group resp dpar nlpar bound
#> 1                                     b
#> 2                                     b      Days
#> 3  student_t(3, 288.7, 59.3) Intercept
#> 4       lkj_corr_cholesky(1)         L
#> 5                                     L           Subject
#> 6      student_t(3, 0, 59.3)        sd
#> 7                                    sd           Subject
#> 8                                    sd      Days Subject
#> 9                                    sd Intercept Subject
#> 10     student_t(3, 0, 59.3)     sigma
```

From this, we see that a improper uniform prior was specified for the fixed effect slope parameter, i.e. b_1 in our formula above. For b_0, a non-standard Student t-distribution was used. This is centered at the overall median of the `Reaction` variable, and its scale is set to the median absolute deviation from the median (MAD). For the covariance matrix corresponding to Σ, the `lkj_corr_cholesky` prior with hyperparameter of `1` is used. When the hyperparameter is set to `1`, this effectively puts uniform prior on on the correlation matrix. For τ_0, τ_1, and σ a half t-distribution, centered at 0, and with a scale equal to the MAD of `Reaction` is used. Although it is not stated explicitly that a half t-distribution is used, this is the case because of the nature of these variable all being defined on the positive real line.

In general, we may use the command `brm` in place of `lmer` to produce a Bayesian counterpart of any `lme4` based linear mixed effects model, assuming we accept the default priors set by `brm`. For example,

```
M_glp_bayes <- brm(mathach ~ ses + himinty + ses:himinty + (ses|school),
                   data = mathach_df2)
#> Running /usr/local/lib/R/bin/R CMD SHLIB foo.c
#> make[1]: Entering directory '/tmp/RtmpjmDqdt'
#> gcc -I"/usr/local/lib/R/include" -DNDEBUG   -I"/usr/local/lib/R/site-library/Rcpp/include/"  -I"/usr/
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:88,
#>                  from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: error: unknown ty
#>   613 | namespace Eigen {
#>       | ^~~~~~~~~
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: error: expected
#>   613 | namespace Eigen {
#>       |                 ^
#> In file included from /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Dense:1,
#>                  from /usr/local/lib/R/site-library/StanHeaders/include/stan/math/prim/mat/fun/Eigen
#>                  from <command-line>:
#> /usr/local/lib/R/site-library/RcppEigen/include/Eigen/Core:96:10: fatal error: complex: No such file
#>    96 | #include <complex>
#>       |          ^~~~~~~~~
#> compilation terminated.
#> make[1]: *** [/usr/local/lib/R/etc/Makeconf:167: foo.o] Error 1
#> make[1]: Leaving directory '/tmp/RtmpjmDqdt'
```

Very often, the results of the models in `brm` and `lmer` are very similar in terms of their practical conclusions. Consider, for example, the posterior means for the fixed effects from the Bayesian model, i.e.,

```
M_glp_bayes %>%
  summary() %>%
  extract2('fixed') %>%
  extract(,'Estimate')
#>   Intercept         ses     himinty ses:himinty
#>  13.1445391   2.5461065  -1.8431743  -0.5778964
```

These are very similar to the maximum (restricted) likelihood estimates from `lmer`:

```
M_glp %>% fixef()
#> (Intercept)         ses     himinty ses:himinty
#>  13.1542443   2.5436697  -1.8593332  -0.5760956
```

Bayesian multilevel models, however, assuming we are implementing them using a probabilistic programming language like Stan, on which `brms` is based, become easier to extend. Exploring how to do goes beyond the intended scope of this chapter, but we will cover examples of this in Chapter 17.

References

Gelman, Andrew, and Jennifer Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. New York, NY: Cambridge University Press.