Week 1: Introduction to Multilevel Modelling

- Hello and welcome to Multivariate Statistics and Methodology with R. My name is Dan Mirman and I'll be teaching the Multilevel Modelling module, which is the first 5 weeks of the semester
    - Multilevel modelling is really just an extension of linear regression, which you learned about last semester in univariate statistics. But multilevel models allow you to capture nested or hierarchical data structures
- What do I mean by nested or hierarchical data structures?
    - Well, for example these would be clustered observations like measurements of multiple children that all come from the same family or classroom or neighbourhood
    - Or these could be repeated measurements of the same individual, like multiple questions on an exam or a within-subject experimental manipulation
    - A specific and important case of repeated measures is longitudinal data where the repeated measurements are spread out over time
  - As you can see, nested or hierarchical data are very common across different sub-disciplines of psychological science (they're actually very common in the behavioural sciences in general, as well as in various branches of biology and ecology). This makes multilevel modelling a very useful statistical method
- Here's one quick example of why this is useful: modelling gradual change
  - You can see from the graph that word learning was faster for high TP words than for low TP words
  - But when I tried a t-test on overall accuracy, it was only marginal.
  - A repeated measures ANOVA showed a main effect of Block, marginal effect of TP (that's the same thing as the t-test), and the interaction was not significant
  - Could do separate t-tests for each block, and they're significant only in block 4 (and marginal in block 5), but that's not a very compelling result
    - What we really need here is a model of the learning curve trajectories, which we can get from multilevel models
- Nested data have two key features that multilevel models can capture
    - Nested data are not independent. Taking a simple longitudinal example, a child that taller-than-average at time t, is likely to be taller-than-average at your next measurement time t+1
    - This non-independence is related to individual differences – that's a tall kid. So it's not just noise or a statistical nuisance; it's actually related to the phenomenon you're studying
    - Nesting can also happen at multiple levels: if you're measuring kids from the same family or hospital, then those height measurements have an additional layer of non-independence
  - The second key feature is that the clustered observations can be related by a continuous variable, like time.
    - When that happens, you want to model it as a continuous variable (for example, not as discrete measurement waves, but as measurements that were separated by a specific duration)
    - You might also be interested in the trajectories or shapes of change over the continuous course of that variable. In developmental or longitudinal contexts these are sometimes called "growth curves"
- So, multilevel models allow you to

        - correctly model the non-independence of observations
        - to simultaneously estimate group-level and individual-level effects
        - and to model trajectories of change
    - a quick note about nomenclature: “multilevel modelling” is part of a family of very closely related statistical methods that just have slight differences in terminology and implications about the data
        - in this course, I’ll mostly use “multilevel models” or I might slip up and say multilevel regression
        - they are also called hierarchical linear models or mixed effects models (also mixed effects regression and linear mixed models)
        - in longitudinal contexts with non-linear data they are also called growth curve models or growth curve analysis
        - I’m saying this because you’ll sometimes see these other terms and I want you to know that they are referring to the same kind of statistical method, just with a slightly different name
- Let’s start from a brief review of linear regression
    - Here you’ve got a relationship between outcome variable Y and predictor Time. Beta_0 is the “intercept”, which is the value of Y at Time=0. Beta_1 is the “slope”, which is the estimated change in Y per unit of Time
    - If we want to talk about individual observations Y of participant i at time j, we can write it like this, with the residual error epsilon_ij
    - In multilevel model terms, we would refer to beta_0 and beta_1 as “fixed effects”
    - And the error term as a “random effect”
- In multilevel models, we can think of that linear regression as a level-1 model
    - And the level-2 is a model of the level-1 parameters. That is to say, a model of beta_0i – the intercept for participant i, which is the population mean intercept (gamma_00) and that participant’s deviation from the population mean intercept (zeta_0i)
    - You could also imagine an effect of condition C on the intercept (gamma_0C)
    - And an analogous model for the level-1 parameter beta_1
- The fixed effects part of multilevel models is pretty similar to regular regression, the magic is in the random effects or residual errors
        - Because those zeta values are specific to a particular individual (or other kind of cluster, like family or school or whatever)
        - They are that individual’s unexplained deviation from the intercept and slope, so they reflect some aspect of their individual differences. We’ll talk more about this as we go on through the module
        - For now, I want to get that general idea and to know that these random effects require a lot of data to estimate.
        - In a sense, random effects are “the hard part” of multilevel modelling both conceptually (this is where you need to describe the nested structure of your data) and computationally (need lots of data to estimate).
- As a quick summary
        - Fixed effects are typically the things you’re studying, the reproducible properties of the world, like differences between nouns and verbs, effects of WM load or age, etc. The model will estimate unique, essentially unconstrained coefficients for each fixed effect condition
    - Random effects are the randomly sampled observational units over which you intend to generalise

        - So these might be particular nouns and verbs, particular individuals that happened to participate in your study, etc.
        - They reflect unexplained variance and that variance is assumed to follow a normal distribution with a mean of 0
- One more technical thing you need to know before we fit our first multilevel model: maximum likelihood estimation
        - This is a procedure for finding parameter estimates that maximize the likelihood of observing the actual data
        - For simple regression, ordinary least squares produces the MLE parameters by just solving an equation
        - This is not possible for multilevel models, so they use an iterative algorithm to gradually converge to MLE parameters. The algorithm is good, but it's not guaranteed to converge to MLE parameters, and we'll talk later about how to deal with convergence problems
    - For now, you need to know that the relevant goodness of fit measure is log-likelihood (often abbreviated LL)
        - Unlike R^2, log-likelihood is not inherently meaningful (it's not proportion of variance or anything like that)
        - But, changes or differences in LL are good measures of improvement in model fit.
        - In fact, -2 times the change in LL is distributed as chi-square with degrees of freedom corresponding to the number of added (or removed) parameters.
        - So it's easy to get a p-value for a comparison of two multilevel models, though the models have to have the same structure with parameters only added (or removed)
- Ok, that's the theoretical stuff, now let's talk about how to actually fit the models in R
        - You'll need the lme4 package
        - The key function is lmer() and it takes a formula, the data, and some options. Broadly similar to the lm() function for basic regression
        - Then you'll evaluate the model or models by comparing them, checking their parameter estimates, plotting model fits, and so on
        - Based on those evaluations, you will often want to improve or adjust the model in various ways and again do comparisons or evaluations.
        - So your analysis will often involve fitting multiple models for comparison or to get the random effects right
- Let's look at a concrete example
        - These are visual search response times and just from looking at the data we can guess that there will be two main effects: it takes more time to find the target when there are more distractors (effect of set size) and stroke survivors (aphasic group) are slower than the control group. It's not immediately clear whether the group difference gets bigger when there are more distractors – that would be a group-by-set size interaction
- Now let's fit the models using the lmer() function from the lme4 package
        - We can start with a null model: there's just one fixed effect -- an overall intercept for the RT (that's what the "1" stands for); but it has a more complex random effect structure – there are by-participant "random intercepts" (participants have different baseline RT) and "random slopes" (participants are differentially slower as the number of distractors increases.
        - The null model isn't very interesting, but it will serve as a baseline comparison for the next model: where we add a fixed effect of set size

        - The “1” is implied, so we don’t need to explicitly specify it unless there are no other terms in the formula. Also, if I’m feeling lazy, I’ll use F instead of writing out FALSE
        - Notice that the random effects stayed the same, so the only difference between the “null” model and this model is the addition of a single fixed effect
    - We can further build up the model by adding a fixed effect of Dx (that’s the group variable and will correspond to a baseline difference in RT) and the Dx-by-set size interaction
- Now that we’ve fit the models, we can compare them. The anova() function will do the comparison – this is a little confusing because the test is not an ANOVA, it’s a model comparison using change in log-likelihood and the chi-square distribution, as you can see from the output
- Here are the same model comparisons in a cleaner table. You can see the log likelihood values (logLik), the “deviance” is -2*logLik, and the Chisq column shows the change in the mode fit – that’s the chi squared statistic and the Df column has the number of added parameters. And from that you get the p-value.
        - So we can see that, compared to the null model, adding set size substantially improves model fit: response times are affected by number of distractors
    - Then adding effect of Diagnosis on intercept (vs.0) significantly improves model fit: stroke survivors respond more slowly than control participants do
    - But adding the interaction of set size and Diagnosis, i.e., effect of Diagnosis on slope (vs.1), does not significantly improve model fit: stroke survivors are not more affected by distractors than control participants are
- You can also use the summary() function to inspect the model
- You’ll notice that the fixed effects don’t have p-values
        - First, let’s just think about what those p-values would be: they are one-sample t-tests of whether the fixed effect estimate is different from 0, with the t-statistic equal to the estimate divided by the standard error of the estimate
    - The problem is that the degrees of freedom for that t-test are not simple to determine
        - The fixed effects are certainly free parameters, but the random effects are estimated under constraints (normal distribution with mean of 0)
        - However, the df *can* be estimated. The two most common estimations are Kenward-Roger and Satterthwaite and they are implemented in a few different packages
- One of the easiest to use is lmerTest: you just load the package and fit your model the same way, then the summary will contain Satterthwaite-approximated df and p-values
    - Here’s an example using the full model of the visual search data [scroll down to see the p-values in the summary]
- An important step in model evaluation is plotting the model fit
        - One way to do that is using the fitted() function. This works very nicely for overlaying observed data and model fits
- Another way is to use the effects package to get model estimates along with SE and confidence intervals.
- Then you can plot those values – this is particularly useful when you have complex models or missing data
- Some general advice
        - This semester you will be learning statistical methods that don't have "cookbook" recipes. You'll need to actively engage with the data and research question in order

to come up with a good model for answering the question, then to defend/explain that model.

- Practice is absolutely critical to learning how to do this. You can't learn it just from the lectures; you have to try it with real data. You will make mistakes, run into problems, etc. Identifying the mistakes and solving those problems is how you'll master this material.
    - We have made all of the example data sets and code available to you for exactly this reason.
- Come to our live Q&A sessions, do the lab exercises. If you're not sure, *try something* then try to figure out whether it worked or not. Ask questions when you're stuck -- we're here to help you learn, but it will only work if you engage in *active, hands-on learning.*