# Week 3: Longitudinal Data Analysis using Multilevel Modelling – Nonlinear Change

- Last week we talked about using multilevel models for longitudinal data analysis, now let's talk about how to deal with non-linear change over time
- All the good things about multilevel models are still relevant, now we're just focusing on modelling non-linear trajectories
    - This application of multilevel models is sometimes called "growth curve analysis"
- Here's an example that's a little more complicated than the ones we've looked at so far: these are data from an eye-tracking experiment where participants had to pick the picture that matched a spoken word.
        - They were faster to look at targets for high frequency (more familiar) words than for low frequency (less familiar) words
        - You can see there's a sort of S-shaped curve here, so how are we going to model this non-linear trajectory?
- To start, we have to choose a "functional form" – a mathematical function to describe that shape. To do that, we need to consider three factors:
        - The function must be adequate to the data – it must be able to take that shape
        - The function needs to have a property called "dynamic consistency"
        - And we need to think about what kind of prediction we want to do
    - It turns out that polynomials offer a pretty good solution to this problem, especially orthogonal polynomials. That's the approach I'll focus on in this lecture, but first let's talk about those three factors in a bit more detail
- The function must be able to take that shape – if your data have this kind of U-shape, a straight line isn't going to do it.
        - A quick way to check how well your function is fitting the data is to plot the residual error against the fitted values – the residuals should be about evenly distributed around 0 (how the red dots are).
        - If you see consistent deviations (as in the black dots), that means your function is missing some consistent aspects of the data
    - You can generate that plot quickly using the augment() function from the broom package (actually, for multilevel models, you need the broom.mixed package)
- Now let's talk about dynamic consistency. It will help to understand that there are two kinds of non-linearity
        - First is non-linear variables or predictors: for example time-squared – that's non-linear (quadratic) time, but notice that that the regression equation has the same form as a typical linear regression – the beta coefficients that are being estimated are still linear
        - Things are different when the parameters (the values that the model is trying to estimate) are themselves non-linearly related to the outcome, as in this equation
    - Dynamic consistency is when the model of the average (that's the group-level model) is equal to the average of the individual models
        - In multilevel models, the random effects (which correspond to individual-level models) have a mean of 0, so the average of those individual-level models will necessarily have 0 deviation from the group-level fixed effect estimates. In other words, multilevel models are always dynamically consistent

    - But this isn't always true once you go outside the multilevel modelling framework. If you fit a complex non-linear function to individual participants' data, then average those models together, you're not necessarily going to get a model of the group data
- Here's an example of what I mean:
        - the blue dots are the group-level data and the blue line is a model of those data – it fits nicely
        - the grey lines are individual participant models, they also fit well
        - but if I average the parameters of those individual participant models and plot it, I get the red line – that's not the group-level model and doesn't fit the group-level data
        - this is what lack of dynamic consistency looks like
    - and it's a big problem for statistical inference: if I compare the individual-level parameters from the left panel to the ones in the right panel (using something like a t-test for example), I'll be testing whether the *red lines* are different but those lines don't correspond to anything interesting
    - so if you fit non-linear functions to individual participants and try to compare the central tendencies of their parameter estimates, you'll be making non-sensical inferences
        - it is hypothetically possible to implement non-linear functions within a multilevel modelling framework, though I am not aware of any good packages for doing this
- It's maybe useful to step back and think about why we even use statistical models and how they are related to making predictions
        - One of the simplest statistical models is the mean and standard deviation
        - The point of them is to provide a compact quantitative description of the data
        - They're not making any new predictions and they're not falsifiable – they just describe the data. A mean can't be "wrong", though it can be useful or not so useful depending on the data
        - A good statistical model provides a useful quantitative description of the data
    - Useful for what? That's the role of the theoretical models. Theoretical models are how you make new predictions and they can be wrong. In fact, a theory being wrong – being falsified by the data – is an important step in scientific discovery.
        - Theoretical models can be quantitative, as in mathematical psychology or computational cognitive modelling, but they're still doing a different thing from a statistical model
        - The statistical model and the theoretical model need to be connected – the statistical model should provide information for testing the theoretical model – but they are not the same thing. And that's a good thing because the data can inform different theoretical models
- Polynomials are a pretty good solution to these different constraints
        - They can model any smooth curve shape (this is closely related to Taylor's theorem)
        - They are dynamically consistent
        - Their main weakness is that they are bad at capturing asymptotic behaviour. In general, this isn't too bad because we usually care about the changing part of the data, not the asymptotic part. It helps if you can avoid modelling long flat sections (the asymptotes). And it is important not to try to extrapolate from polynomial curve fits.
    - If you're going to use polynomials, you'll need to choose a polynomial order – quadratic, cubic, fourth-order, whatever

        - With some experience, you can just look at the curve shape and have a pretty good guess about the right order
        - You can also take a statistical approach: include terms that improve the model fit
        - Alternatively you can use a theoretical approach: your predictions will typically be about particular kinds of effects so you can focus on polynomial terms that correspond to those effects
- One more thing about polynomials: the terms of a natural polynomial are usually correlated and on different scales – this means that you'll have collinear predictors and can have estimation problems (left figure)
        - Orthogonal polynomials are just a transformation that centres and scales them, so they are uncorrelated and on the same scale
- You do need to be careful about interpreting orthogonal polynomials: the intercept corresponds to the overall average rather than the y-intercept
- The linear term corresponds to the linear slope, with the pivot point in the centre
        - The quadratic term corresponds to the steepness of the rise and fall rate, which is useful for modelling U-shapes
        - The cubic and quartic terms also correspond to steepness around inflection points, though it gets hard to interpret those higher-order terms
- A quick reminder about random effects: we've talked about random intercepts and slopes, now that we're dealing with polynomials we're going to have random "slopes" for those polynomial terms
    - And following the strategy I described before, we'll start with a maximal random effect structure and simplify it if we run into convergence problems
- Ok, so now we have some ideas about how to model the data in our eye-tracking data example
- To start, let's create a third-order polynomial – the code_poly() function is a helper function for doing this. You give it your data frame, identify the predictor variable that needs to have polynomial versions, and the polynomial order you need. By default, it will make a graph that shows you how your raw predictor is related to the polynomials it created. (This function defaults to using orthogonal polynomials, but you can specify natural polynomials if you want those)
- You can see this added three variables ploy1, poly2, and poly3 to the data frame
- I've skipped ahead to fitting the full model, but you can build up the Condition effects if you want
        - The fixed effects are the three polynomial time terms and their interactions with Condition
        - Then we have random effects of Subject with random "slopes" for each of the polynomial terms. This allows individual subject variability in each of the curve shape components
        - And we have analogous random effects for subject-by-condition, since condition is within-subject (I'll come back to why I specified them this way in a minute)
        - Here are the fixed effects: you can see there is a significant Condition intercept effect and quadratic effect
- And you can plot the model fit – an important step when fitting complex models to make sure the fit is good and the inferences you want to make are supported by your model
- Now let's talk about the random effects. We specified two sets of random effects: subject-level and subject-by-condition level, for each time term. You can see values there: these would be the zeta values from the equations we talked about before
- In the model summary, what we see is the variance (standard deviation here) and covariance of these random effects – these values are being estimated and the individual unit-level random effects on the previous slide are constrained to have this variance-covariance structure with a mean of 0

    - This is why degrees of freedom in multilevel models are tricky to define – you have these inter-related parameters to estimate so they're not exactly "free"
- There's another aspect of the random effects I want to discuss. In the full model before, I used two random effects (subject-level and subject-by-condition).
        - Another approach would be to have random by-condition slopes of Condition; putting Condition on the left of the pipe. This would be a more straightforward extension of the random effect structures we used in the first two weeks
        - You can see it would produce virtually the same result (somewhat higher p-values, but the same result – significant intercept and quadratic effect of Condition)
- Using this approach has some advantages – it allows more flexible variance and covariance estimates
- But at the cost of more random effect parameter estimates. With two conditions (high vs. low) and three time terms, the difference is not too bad. But if you have more conditions (like a 2x2 design) and more time terms, the difference can be huge
        - In general, I think it's worth trying this left-side random effect structure. In practice, I often have data sets that can't support that level of complexity (flexibility) in the random effect structure and the other approach offers a reasonable way to simplify it without losing the most important elements
- When you run into convergence problems, polynomial models offer you several different options for simplifying the random effect structure
    - One option is to remove a higher-order polynomial random effect. For example, if I'm not too interested in the cubic term, I might be willing to make the simplifying assumption that subjects do not vary in that term
    - Another option is to remove the random effect correlations. This can be done manually by writing out each effect or by using a double-pipe notation
- This is also a good time to talk a little more about what the random effects are doing
        - When we treat participants as random effects, this captures the typical assumption that the participants are a random sample from some population and we want to generalise to that population
        - This graph (created by Tristan Mahr) shows three ways of thinking about individual variation:
            - "complete pooling" is one model for everyone. That's what you get from OLS regression and all of the residual error is just treated as the same kind of error
            - "no pooling" is each person has their own completely independent model. I've heard this called statistical "amnesia" because each participant's data is fit without any memory of the other participants. Now it's true that each participant is different from the others, but they're not completely unrelated – they come from the same population after all
            - "partial pooling" is what you get in multilevel models. Each participant has their own random effect, so they are different from others, but those random effects are constrained to come from one distribution (that is, participants are sampled from some single population)
- This produces an effect called "shrinkage": the individual participants' estimates "shrink" toward the group mean. This is a very useful property because each individual's performance is a mix of general population-level properties, their individual unique properties, and noise. We usually care about the population-level and individual-level stuff, but we can't know exactly how much noise is

contributing. But, we can use the performance of everyone else in the sample to *estimate* how much noise might be happening and what might be legit individual differences. This is exactly what partial pooling or shrinkage accomplishes
        - I recommend reading Tristan Mahr's and/or Michael Clark's explanations of these concepts to understand them better
- Keep in mind that the benefits of partial pooling and shrinkage depend on the assumption that the participants are sampled from some population and you want to generalise. That's not always the case.
    - For example, in neurological or neuropsychological case studies, you're trying to characterise the performance of each participant and make claims about that specific participant, not some population of participants.
    - In those cases, it can make more sense to treat participants as fixed effects
- Key points: this lecture covered two broad issues
    - Modelling non-linear change: I described some general considerations and focused on using polynomials. Polynomials have some very nice mathematical properties, but you have to be careful with interpretation and extrapolation when you use them
    - Random effect structure: as a general rule, you want to start with a maximal random effect structure and simplify it if needed.
        - Three simplification strategies: Putting categorical variables on the right side of the pipe, removing random correlations, and removing higher-order terms
    - The discussion of random effects also brought up the key concepts of partial pooling and shrinkage, which are important for understanding how multilevel models distinguish group-level effects, individual-level differences, and noise.