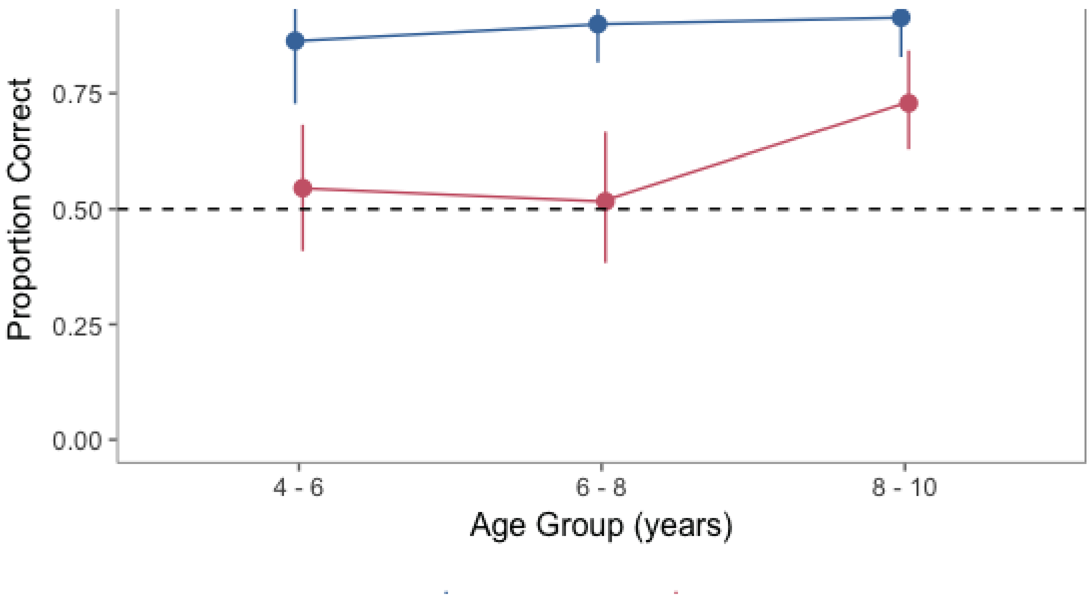
Mixed effects models: Is it time to go Bayesian by default?

babieslearninglanguage.blogspot.com/2018/02/mixed-effects-models-is-it-time-to-go.html



(tl;dr: Bayesian mixed effects modeling using brms is really nifty.)

Introduction: Teaching Statistical Inference?

How do you reason about the relationship between your data and your hypotheses? Bayesian inference provides a way to make normative inferences under uncertainty. As scientists – or even as rational agents more generally – we are interested in knowing the probability of some hypothesis given the data we observe. As a cognitive scientist I've long been interested in using Bayesian models to describe cognition, and that's what I did much of my graduate training in. These are custom models, sometimes fairly difficult to write down, and they are an area of active research. That's not what I'm talking about in this blogpost. Instead, I want to write about the basic practice of statistics in experimental data analysis.
Mostly when psychologists do and teach "stats," they're talking about frequentist statistical tests. Frequentist statistics are the standard kind people in psych have been using for the last 50+ years: t-tests, ANOVAs, regression models, etc. Anything that produces a p-value. P-values represent the probability of the data (or any more extreme) under the null hypothesis (typically "no difference between groups" or something like that). The problem is that this is not what we really want to know as scientists. We want the opposite: the probability of the hypothesis given the data, which is what Bayesian statistics allow you to compute. You can also compute the relative evidence for one hypothesis over another (the Bayes Factor).

Now, the best way to set psychology twitter on fire is to start a holy war about who's actually right about statistical practice, Bayesians or frequentists. There are lots of arguments here, and I see some merit on both sides. That said, there is lots of evidence that much of our implicit statistical reasoning is Bayesian. So I tend towards the Bayesian side on the balance <ducks head>. But despite this bias, I've avoided teaching Bayesian stats in my classes. I've felt like, even with their philosophical attractiveness, actually computing Bayesian stats had too many very severe challenges for students. For example, in previous years you might run into major difficulties inferring the parameters of a model that would be trivial under a frequentist approach. I just couldn't bring myself to teach a student a philosophical perspective that – while coherent – wouldn't provide them with an easy toolkit to make sense of their data.

The situation has changed in recent years, however. In particular, the BayesFactor R package by Morey and colleagues makes it extremely simple to do basic inferential tasks using Bayesian statistics. This is a huge contribution! Together with JASP, these tools make the Bayes Factor approach to hypothesis testing much more widely accessible. I'm really impressed by how well these tools work.

All that said, my general approach to statistical inference tends to rely less on inference about a particular hypothesis and more on parameter estimation – following the spirit of folks like Gelman & Hill (2007) and Cumming (2014). The basic idea is to fit a model whose parameters describe substantive hypotheses about the generating sources of the dataset, and then to interpret these parameters based on their magnitude and the precision of the estimate. (If this sounds vague, don't worry – the last section of the post is an example). The key tool for this kind of estimation is not tests like the t-test or the chi-squared. Instead, it's typically some variant of regression, usually mixed effects models.

Mixed-Effects Models

Especially in psycholinguistics where our experiments typically show many people many different stimuli, mixed effects models have rapidly become the de facto standard for data analysis. These models (also known as hierarchical linear models) let you estimate sources of random variation ("random effects") in the data across various grouping factors. For example, in a reaction time experiment some participants will be faster or slower (and so all data from those particular individuals will tend to be faster or slower in a correlated way). Similarly, some stimulus items will be faster or slower and so all the data from these groupings will vary. The lme4 package in R was a game-changer for using these models (in a frequentist paradigm) in that it allowed researchers to estimate such models for a full dataset with just a single command. For the past 8-10 years, nearly every paper I've published has had a linear or generalized linear mixed effects model in it.

Despite their simplicity, the biggest problem with mixed effects models (from an educational point of view, especially) has been figuring out how to write consistent model specifications for random effects. Often there are many factors that vary randomly (subjects, items, etc.) and many other factors that are nested within those (e.g., each subject might respond differently to each condition). Thus, it is not trivial to figure out what model to fit, even if fitting the model is just a matter of writing a command. Even in a reaction-time experiment with just items and subjects as random variables, and one condition manipulation, you can write

(1) rt ~ condition + (1 | subject) + (1 | item)

for just random intercepts by subject and by item, or you can nest condition (fitting a random slope) for one or both:

(2) rt ~ condition + (condition | subject) + (condition | item)

and you can additionally fiddle with covariance between random effects for even more degrees of freedom!

Luckily, a number of years ago, a powerful and clear simulation paper by Barr et al. (2013) came out. They argued that there was a simple solution to the specification issue: use the "maximal" random effects structure supported by the design of the experiment. This meant adding any random slopes that were actually supported by your design (e.g., if condition was a within-subject variable, you could fit condition by subject slopes). While this suggestion was quite controversial,* Barr et al.'s simulations were persuasive evidence that this suggestion led to conservative inferences. In addition, having a simple guideline to follow eliminated a lot of the worry about analytic flexibility in random effects structure. If you were "keeping it maximal" that meant that you weren't intentionally – or even inadvertently – messing with your model specification to get a particular result.

Unfortunately, a new problem reared its head in lme4: convergence. With very high frequency, when you specify the maximal model, the approximate inference algorithms that search for the maximum likelihood solution for the model will simply not find a satisfactory solution. This outcome can happen even in cases where you have quite a lot of data – in part because the number of parameters being fit is extremely high. In the case above, not counting covariance parameters, we are fitting a slope and an intercept across participants, plus a slope and intercept for *every participant* and for *every item*.

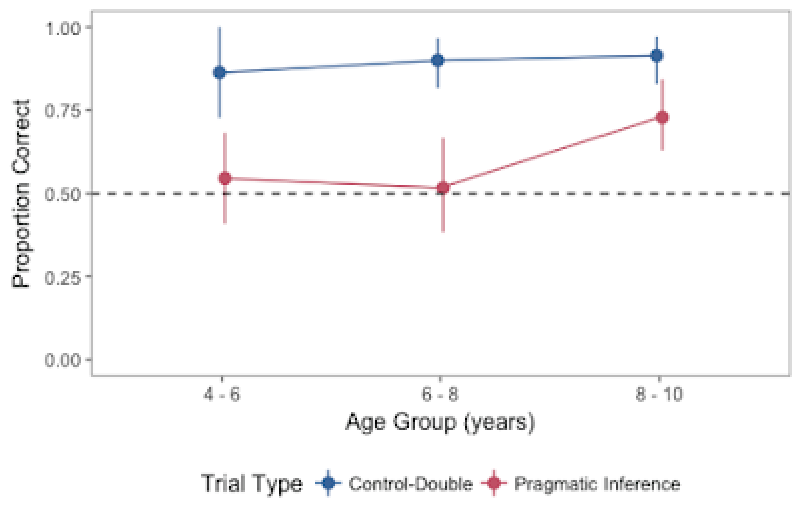
To deal with this, people have developed various strategies. The first is to do some black magic to try and change the optimization parameters (e.g., following these helpful tips). Then you start to prune random effects away until your model is "less maximal" and you get convergence. But these practices mean you're back in flexible-model-adjustment land, and vulnerable to all kinds of charges of post-hoc model tinkering to get the result you want. We've had to specify lab best-practices about the order for pruning random effects – kind of a guide to "tinkering until it works," which seems suboptimal. In sum, the models are great, but the methods for fitting them don't seem to work that well.

Enter Bayesian methods. For several years, it's been possible to fit Bayesian regression models using Stan, a powerful probabilistic programming language that interfaces with R. Stan, building on BUGS before it, has put Bayesian regression within reach for someone who knows how to write these models (and interpret the outputs). But in practice, when you could fit an lmer in one line of code and five seconds, it seemed like a bit of a trial to hew the model by hand out of solid Stan code (which looks a little like C: you have to declare your variable types, etc.). We have done it sometimes, but typically only for models that you couldn't fit with lme4 (e.g., an ordered logit model). So I still don't teach this set of methods, or advise that students use them by default.

brms?!? A worked example

In the last couple of years, the package brms has been in development. brms is essentially a front-end to Stan, so that you can write R formulas just like with lme4 but fit them with Bayesian inference.* This is a game-changer: all of a sudden we can use the same syntax but fit the model we want to fit! Sure, it takes 2-3 minutes instead of 5 seconds, but the output is clear and interpretable, and we don't have all the specification issues described above. Let me demonstrate.

The dataset I'm working on is an unpublished set of data on kids' pragmatic inference abilities. It's similar to many that I work with. We show children of varying ages a set of images and ask them to choose the one that matches some description, then record if they do so correctly. Typically some trials are control trials where all the child has to do is recognize that the image matches the word, while others are inference trials where they have to reason a little bit about the speaker's intentions to get the right answer. Here are the data from this particular experiment:



I'm interested in quantifying the relationship between participant age and the probability of success in pragmatic inference trials (vs. control trials, for example). My model specification is:

(3) correct ~ condition * age + (condition | subject) + (condition | stimulus)

So I first fit this with lme4. Predictably, the full desired model doesn't converge, but here are the fixed effect coefficients:

	beta	stderr	z	p
intercept	0.50	0.19	2.65	0.01
condition	2.13	0.80	2.68	0.01
age	0.41	0.18	2.35	0.02
condition:age	-0.22	0.36	-0.61	0.54

Now let's prune the random effects until the convergence warning goes away. In the simplified version of the dataset that I'm using here I can keep stimulus and subject intercepts and still get convergence when there are no random slopes. But in the larger dataset, the model won't converge unless i do *just* the random intercept by subject:

	beta	stderr	z	p
intercept	0.50	0.21	2.37	0.02
condition	1.76	0.33	5.35	0.00
age	0.41	0.18	2.34	0.02
condition:age	-0.25	0.33	-0.77	0.44

Coefficient values are decently different (but the p-values are not changed dramatically in this example, to be fair). More importantly, a number of fairly trivial things matter to whether the model converges. For example, I can get one random slope in if I set the other level of the condition variable to be the intercept, but it doesn't converge with either in this parameterization. And in the full dataset, the model wouldn't converge at all if I didn't center age. And then of course I haven't tweaked the optimizer or messed with the convergence settings for any of these variants. All of this means that there are a *lot* of decisions about these models that I don't have a principled way to make – and critically, they need to be made conditioned on the data, because I won't be able to tell whether a model will converge *a priori*!

So now I switched to the Bayesian version using brms, just writing brm() with the model specification I wanted (3). I had to do a few tweaks: upping the number of iterations (suggested by the warning messages from the output, changing to a Bernoulli model rather than binomial (for efficiency, again suggested by the error message), but this was very straightforward otherwise. For simplicity I've adopted all the default prior choices, but I could have gone more informative.

Here's the summary output for the fixed effects:

```
                      estimate  error    l-95% CI u-95% CI
intercept              0.54      0.48    -0.50     1.69
condition              2.78      1.43     0.21     6.19
age                    0.45      0.20     0.08     0.85
condition:age         -0.14      0.45    -0.98     0.84
```

From this call, we get back coefficient estimates that are somewhat similar to the other models, along with 95% credible interval bounds. Notably, the condition effect is larger (probably corresponding to being able to estimate a more extremal value for the logit based on sparse data), and then the interaction term is smaller but has higher error. Overall, coefficients look more like the first non-convergent maximal model than the second converging one.

The big deal about this model is not that what comes out the other end of the procedure is radically different. It's that it's *not* different. I got to fit the model I wanted, with a maximal random effects structure, and the process was almost trivially easy. In addition, and as a bonus, the CIs that get spit out are actually credible intervals that we can reason about in a sensible way (as opposed to frequentist confidence intervals, which are quite confusing if you think about them deeply enough).

Conclusion

Bayesian inference is a powerful and natural way of fitting statistical models to data. The trouble is that, up until recently, you could easily find yourself in a situation where there was a dead-obvious frequentist solution but off-the-shelf Bayesian tools wouldn't work or would generate substantial complexity. That's no longer the case. The existence of tools like BayesFactor and brms means that I'm going to suggest that people in my lab go Bayesian by default in their data analytic practice.

Thanks to Roger Levy for pointing out that model (3) above could include an age | stimulus *slope to be truly maximal. I will follow this advice in the paper.*

* Who would have thought that a paper about statistical models would be called "the cave of shadows"?
** Rstanarm did this also, but it covered fewer model specifications and so wasn't as helpful.