Bayesian Item Response Modelling in R with brms and Stan

Paul-Christian Bürkner
Aalto University, Department of Computer Science

Abstract

Item Response Theory (IRT) is widely applied in the human sciences to model persons' responses on a set of items measuring one or more latent constructs. While several R packages have been developed that implement IRT models, they tend to be restricted to respective prespecified classes of models. Further, most implementations are frequentist while the availability of Bayesian methods remains comparably limited. I demonstrate how to use the R package **brms** together with the probabilistic programming language Stan to specify and fit a wide range of Bayesian IRT models using flexible and intuitive multilevel formula syntax. Further, item and person parameters can be related in both a linear or non-linear manner. Various distributions for categorical, ordinal, and continuous responses are supported. Users may even define their own custom response distribution for use in the presented framework. Common IRT model classes that can be specified natively in the presented framework include 1PL and 2PL logistic models optionally also containing guessing parameters, graded response and partial credit ordinal models, as well as drift diffusion models of response times coupled with binary decisions. Posterior distributions of item and person parameters can be conveniently extracted and post-processed. Model fit can be evaluated and compared using Bayes factors and efficient cross-validation procedures.

Keywords: Item Response Theory, Bayesian Statistics, R, Stan, **brms**.

1. Introduction

Item Response Theory (IRT) is widely applied in the human sciences to model persons' responses on a set of items measuring one or more latent constructs (for a comprehensive introduction see Lord 2012; Embretson and Reise 2013; van der Linden and Hambleton 2013). Due to its flexibility compared to classical test theory, IRT provides the formal statistical basis for most modern psychological measurement. The best known IRT models are likely those for binary responses, which predict the probability of a correct answer depending on the item's difficulty and potentially other item properties as well as the participant's latent ability. The scope of IRT models is however much wider than this, and I will discuss several more interesting models in this paper.

Over the years, a multitude of software packages have been developed that implement IRT models. To date, most free and open source software in the field of IRT is written in the programming language R (R Core Team 2019), which has grown to become one of the primary languages for statistical computing. Examples for widely applied and actively maintained IRT specific R packages are **eRm** (Mair and Hatzinger 2007), **ltm** (Rizopoulos 2006), **TAM**

(Robitzsch, Kiefer, and Wu 2019), **mirt** (Chalmers 2012), **sirt** (Robitzsch 2019), and **psychotree** (Strobl, Kopf, and Zeileis 2015; Komboz, Zeileis, and Strobl 2018). Each of them supports certain classes of IRT models and related post-processing methods. Further, IRT models may also be specified in general purpose multilevel or structural equation modeling packages such as **lme4** (Bates, Mächler, Bolker, and Walker 2015b), **lavaan** (Rosseel 2012), or **blavaan** (Merkle and Rosseel 2015). I will provide a review and comparison of these package later on in Section 6.

In this paper, I present a Bayesian IRT framework based on the R package **brms** (Bürkner 2017; Bürkner 2018) and the probabilistic programming language Stan (Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li, and Ridell 2017). The propsed framework is quite extensive both in the models that can be specified and in the supported post-processing methods. Users can choose from over 40 built-in response distributions, which not only include standard IRT models such as binary, categorical or ordinal models, but also models for count data, response times or proportions, to name only a few available options. Users may also write their own custom response distributions not natively supported by **brms** for application in the proposed framework. The non-linear multilevel formula syntax of **brms** allows for a flexible yet concise specification of multidimensional IRT models, with an arbitrary number of person or item covariates and multilevel structure if required. Prior knowledge can be included in the form prior distributions, which constitute an essential part of every Bayesian model. Estimation is performed in Stan using MCMC sampling via adaptive Hamiltonian Monte Carlo (Hoffman and Gelman 2014; Stan Development Team 2019), an efficient and stable algorithm that works well in high dimensional, highly correlated parameter spaces. The flexbility of the proposed framework is not only helpful to the applied researcher who wishes to analyze their IRT data by means of one single package, but it is also provides an opportunity for more methodologically interested researchers who strive to develop new IRT models or model variants. **brms** could be a powerful and convenient tool to implement them in a Bayesian context. However, it arguably requires more work from the user at the start to familiarize themselves with the modeling syntax and post-processing options and probably has a much steeper learning curve than more specialized IRT packages.

This paper has three central purposes. First, it provides a thorough conceptual introduction to the proposed Bayesian IRT framework. Second, it demonstrates how this framework is implemented in statistical software. Third, based on several hands-on examples, it explains how the software can be used in practice to solve real-world questions. On the conceptual side, in Section 2, I substantially extend the work of De Boeck, Bakker, Zwitser, Nivard, Hofman, Tuerlinckx, and Partchev (2011), who initially opened up the road for the estimation of IRT models via multilevel models. However, they only considered generalized linear multilevel models and specifically focussed on binary data. I extend their framework in various directions, most notably to (a) a much larger number of response distributions, (b) non-linear IRT models, which do not make the assumption of the predictor term being of a (generalized) linear form, and (c) distributional IRT models, in which not only the main location parameter of the response distribution but also all other parameters may depend on item and person properties. On the software side, in Section 3 and 4, I introduce several new features in **brms** that have been implemented after the publication of its second paper (Bürkner 2018) to both support the presented framework in its entirety and provide several more specific features designed to make important IRT model classes possible within the framework. These features include the full integration of non-linear and distributional parameter pre-

dictions via a nested non-linear formula syntax, the implementation of several distributions designed for response times data, and extentions of distributions for ordinal data, for example for the purpose of modeling discrimination parameters. To help users applying the present framework and related software in practice, several hands-on examples are discussed in detail in Section 5. I provide a comparison of IRT supporting R packages in Section 6 and end with a conclusion in Section 7. All materials related to this paper are hosted on GitHub (`https://github.com/paul-buerkner/Bayesian-IRT-paper`).

2. Model description

The core of models implemented in **brms** is the prediction of the response y through predicting all K parameters ψ_k of the response distribution D. We write

$$y_n \sim D(\psi_{1n}, \psi_{2n}, \ldots, \psi_{Kn})$$

to stress the dependency on the n^{th} observation. In most R packages, the response distribution is called the model `family` and I adopt this term in **brms**. Writing down the model per observation n implies that we have think of the data in long rather than in wide format. That is, reponses to different items go in the same column of the data set rather than in different columns. The long format works well in combination with multilevel formula syntax and is arguably also more favourable from a programatical perspective (e.g., see Wickham and Grolemund 2016).

2.1. Response distributions

The response format of the items will critically determine which distribution is appropriate to model individuals' responses on the items. The possibility of using a wide range of response distributions within the same framework and estimating all of them using the same general-purpose algorithms is an impoartent advantage of Bayesian statistics. **brms** heavily exploits this advantage by offering a multitude of response distributions and even allowing the user to add their own. In this section, I will briefly review some common response distributions in IRT that are natively supported in the proposed framework.

If the response y is a binary success (1) vs. failure (0) indicator, the canonical family is the *Bernoulli* distribution with density

$$y \sim \text{Bernoulli}(\psi) = \psi^y (1 - \psi)^{1-y},$$

where $\psi \in [0, 1]$ can be interpreted as the success probability. Common IRT models than can be built on top of the bernoulli distribution are the 1, 2, and 3 parameter logistic models (1PL, 2PL, and 3PL models; Agresti 2010), which I will discuss in more detail in Sections 2.2 and 5.1.

If y consitutes a categorical response with $C > 1$ unordered categories, the *categorical* distribution is appropriate (Agresti 2010). It has the density

$$y \sim \text{categorical}(\psi_1, \ldots, \psi_C) = \prod_{c=1}^{C} \psi_c^{I_k(y)}$$

with cateory probabilities $P(y = c) = \psi_c > 0$ and $\sum_{c=1}^{C} \psi_c = 1$ where $I_c(y)$ is the indicator function which evaluates to 1 if $y = k$ and to 0 otherwise. For $C = 2$, the categorical distribution is equivalent to the Bernoulli distribution.

If y is an ordinal categorical response with C ordered categories, multiple possible response distributions are plausible (Agresti 2010; Bürkner and Vuorre 2019). They are all built on top of the categorical distribution but differ in how they define the category probabilities $P(y = c)$. The two most commonly applied ordinal families in IRT are the *cumulative model* and the *adjacent category model*. The cumulative model assumes

$$P(y = c) = F(\tau_c - \psi) - F(\tau_{c-1} - \psi)$$

where F is the cumulative distribution function (CDF) of a continuous unbounded distribution and τ is a vector of $C - 1$ ordered thresholds. If F is the standard logistic distribution, the resuling IRT model is called *graded response model* (GRM; Samejima 1997). Alternatively, one can use the adjacent category model, which, when combined with the logistic distribution, becomes the *partial credit model* (PCM; Rasch 1961). It assumes

$$P(y = c) = \frac{\exp\left(\sum_{j=1}^{c-1}(\psi - \tau_j)\right)}{\sum_{r=1}^{C}\exp\left(\sum_{j=1}^{r-1}(\psi - \tau_j)\right)}$$

with threshold vector τ whose element do not necessarily need to be ordered (Adams, Wu, and Wilson 2012). The PCM is widely applied in IRT for instance in various large scale assessment studies such as PISA (OECD 2017). I will provide hands-on examples of ordinal IRT models in Section 5.2.

If y consitutes a count variable without a natural upper bound (or an upper bound that is practically not reachable, for instance in dedicated speed tests), the *Poisson* distribution with density

$$y \sim \text{Poisson}(\psi) = \frac{\psi^y \exp(-\psi)}{y!},$$

or one of its various generalizations (e.g., see Shmueli, Minka, Kadane, Borle, and Boatwright 2005), may be an appropriate choice. In IRT, this leads to what is known as the *Rasch-Poisson-Counts model* (RPCM; Rasch 1960).

When items consist of a comparative judgement between C categorical alternatives on a continuous bounded scale, obtained responses are in a "proportion-of-total" (compositional) format (Hijazi and Jernigan 2009). That is, for each response category c, $y_c \in [0, 1]$ is the proportion of the total points that was assigend to that category so that $\sum_{c=1}^{C} y_c = 1$. If $C = 2$, the response $y = y_1$ on the first category can be modeled as *beta* distributed (as $y_2 = 1 - y_1$ is redundant). The mean-precision parameterization of the beta distribution has density

$$y \sim \text{Beta}(\psi_1 = \mu, \psi_2 = \phi) = \frac{y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}}{B(\mu\phi, (1-\mu)\phi)}$$

where B is the beta function. A multivariate generalization of the Beta family is the *Dirichlet* family, which can be used for compositional scores of more than two response categories (Hijazi and Jernigan 2009). On the full response vector $y = (y_1, ..., y_C)$ it has density

$$y \sim \text{Dirichlet}(\psi_1, \dots \psi_C, \psi_{C+1} = \phi) = \frac{1}{B((\psi_1, \dots, \psi_K)\phi)} \prod_{k=1}^{K} y_k^{\psi_k \phi - 1}.$$

Another important class of IRT models deals with response/reaction times, which tend to vary over items and persons in at least three ways: mean, variation, and right skewness of the responses. Accordingly, sufficiently flexible response distributions on reaction times are likely to require three parameters in order to capture these aspects. Two commonly applied 3-parameter distributions are the exponentially-modified Gaussian (exgaussian) distribution and the shifted lognormal distribution (Heathcote, Popiel, and Mewhort 1991; Wagenmakers and Brown 2007). Their densities are a little bit more involved and so I do not display them here, but they can be found for instance in Wagenmakers and Brown (2007) or when typing `vignette("brms_families")` in R. With the exgaussian distribution, we can directly parameterize the mean which simplifies interpreation of model parameters, at the expense of having a theoretically less justified model (Heathcote *et al.* 1991). I will provide a practical example of analyzing response times in an IRT context in Section 5.3.

Going one step further, it is often favorable to model persons' responses together with the corresponding response times in a joint process model. This not only to implies a more appropriate generative model for the data but may also foster theoretical understanding of the underlying processes (Ratcliff 1978; van der Maas, Molenaar, Maris, Kievit, and Borsboom 2011). One of these joint models, which can handle binary decisions together with their response times, is the Wiener drift diffusion model (Ratcliff 1978; van der Maas *et al.* 2011). Its parameters have meaning in the context of cognitive decision process described as a Wiener diffusion process with a drift towards one or the other binary choice alternative. The parameters of the four parameter drift diffusion model implemented in the presented framework are (1) the drift rate that describes a person's tendency towards one or the other two alternative, (2) the boundary separation that describes how much evidence needs to be accumulated until a decision is made, (3) the non-decision times that describes the time spend at processing the items and executing a motor response (i.e., everything non-decision related), and (4) the initial bias that describes persons tendency towards one of the two alternatives independent of the item properties. In IRT applications, it is common to fix the initial bias to 0.5, that is, to assume no initial bias towards one of the two alternatives (Molenaar, Tuerlinckx, van der Maas *et al.* 2015), which results in the three-parameter drift diffusion model. A more detailed discussion of the drift diffusion models is beyond the scope of the present paper, but can be found elsewhere (Ratcliff 1978; van der Maas *et al.* 2011; Molenaar *et al.* 2015). I will provide a practical example of fitting drift diffusion models to IRT data in Section 5.3.

2.2. Predicting distributional parameters

In the context of IRT, every distributional parameter ψ_k can written as a function $\psi_{kn} = f_k(\theta_{kp_n}, \xi_{ki_n})$ of person parameters θ_k and item parameters ξ_k, where p_n and i_n indicate the person and item, respectively, to which the n^{th} observation belongs[1]. In a regression

[1] A parameter may also be assumed constant across observations and thus be independent of person and

context, such models are often referred to as distributional regression models or as regression models of location, scale, and shape (Rigby and Stasinopoulos 2005) to stress the fact that all parameters of the distribution can be predicted, not just a single parameter – usually the mean of the distribution or some other measure of central tendency.

In addition to the response distribution itself, the exact form of the equations $\psi = f(\theta_p, \xi_i)$ (suppressing the indices k and n for simplicty) will critically define the meaning of the person and item parameters as well as the complexty of the model in general. In a linear model, f is the identity function and the relation between θ_p and ξ_i is linear and additive so that $\psi = \theta_p + \xi_i$. Unfortunately, such a model will not yield the desired results if ψ has natural range restrictions. For instance, if the response y is a binary success (1) vs. failure (0) indicator, and and we use the the Bernoulli response distribution, ψ can be interpreted as the success probability, which, by definition, must lie within the interval $[0, 1]$. However, a linear model $\psi = \theta_p + \xi_i$ may yield any real value and so is invalid when predicting probabilities. The solution for this problem is to use a non-linear function f appropriate to the scale of the predicted parameter ψ. This results in what is known as a *generalized linear model* (GLM). That is, the predictor term $\eta = \theta_p + \xi_i$ is still linear but transformed, as a whole, by a non-linear function f, which is commonly called 'response function'. For Bernoulli distributions, we can canonically use the logistic response function

$$f(\eta) = \text{logistic}(\eta) = \frac{\exp(\eta)}{1 + \exp(\eta)},$$

which yields values $f(\eta) \in [0, 1]$ for any real value η. As a result, we could write down the model of ψ as

$$\psi = \frac{\exp(\theta_p + \xi_i)}{1 + \exp(\theta_p + \xi_i)},$$

which is known as the Rasch or 1PL model (Bond and Fox 2013). Under the above model, we can interprete θ_p as the ability of person p in the sense that higher values of θ_p imply higher success probabilities regardless of the administered item. Further, we can interprete ξ_i as the easiness of item i as higher values of ξ_i imply higher success probabilities regardless of the person to which the item is administered. Note that most definitions of the Rasch model instead use $\theta_p - \xi_i$, in which case ξ_i becomes the item difficulty rather than the easiness. Clearly, both formulations are equivalent. In the present paper I generally use the easiness formulation as it naturally fits into the regression framework of **brms**.

In the context of IRT, GLMs already will carry us a long way, but at some point, their flexibility reaches a halt. A typical example of such a situation is when we stop assuming discriminations to be constant across items; an assumption that will often be violated in real world data (Andrich 2004). Instead, if we want to model varying item discrimations α_i, the predictor term becomes

$$\psi = f(\alpha_i(\theta_p + \xi_i)) = f(\alpha_i\theta_p + \alpha_i\xi_i).$$

The argument to f no longer forms a linear predictor as we now consider *products* of parameters. In the context of logistic models for dichotomous responses, we would refer to the

item parameters.

varying discrimination model as 2PL model (e.g., Andrich 2004). If persons have a non-zero probability γ_i of guessing the right answer of item i, independent of their abilities, this would yield the 3PL model, in my notation written as

$$\psi = f(\theta_p, \xi_i, \alpha_i, \gamma_i) = \gamma_i + (1 - \gamma_i)\, g(\alpha_i(\theta_p + \xi_i))$$

with g being some function to transform real values onto the unit interval (e.g., the logisitic function). The complexity of such a non-linear predictor may be arbitraily increased in theory, but of course needs to be carefully specified in order to yield an identifiable and interpretable model. Further, in the context of Bayesian IRT, prior distributions may additionally help to identify the model (see Section 2.4 for more details on priors).

2.3. Item and Person Covariates

A lot of research questions in the context of IRT do not simply require estimating person and item parameters but rather estimating the effects of person or item *covariates* (De Boeck *et al.* 2011), that is variables that vary across persons and/or items. De Boeck *et al.* (2011) differentiate covariates by their mode (person, item, or both) and the origin of the covariate as either internal (stems from item responses) or external (independent of the item responses). For instance, persons' age would be considered an external person covariate as it varies over persons but not over items and does not change its value according to item responses. Item type (e.g., figural, numeric, or verbal in case of typical intelligence test items) would be considered an external item covariate, while the number of previous items solved by a specific person at the time of administering a specific item would be an internal person by item covariate.

Regardless of the specific nature of the covariates, we may add them to the any linear predictor term η in the model so that it no longer only depends on individual person and item parameters, but also on a set of J covariates x_j:

$$\eta_{pi} = \theta_p + \xi_i + \sum_{j=1}^{J} b_j x_{jpi}$$

In the equation above, x_{jpi} is the value of the jth predictor for person p and item i. Of course, a person covariate is constant across items and an item covariate is constant across persons. I still index all covariates by both person and items, though, to shorten the notation without loss of generality.

A further differentiation of covariates may be made by considering over what mode (person, items, or both) the covariate effects are allowed to vary (i.e., interact with) in the model. For example, a persons' age varies between but not within persons, which implies that the *effect* of age may only vary across items. Conversely, the effect of an item covariate may only vary across persons as it is constant within each item. Extending the above notation for covariates, the regression coefficients b_j would then receive additional indices p or i (i.e., b_{jp} or b_{ji}) depending on whether the effect of the covariate is expected to vary over person or items.

For psychometric tests, it is essential to investigate differential item functioning (DIF; Holland and Wainer 2012). Items showing DIF have different properties for persons belonging to

different groups even if the persons have the same ability. Such items are problematic for the validity of the test as they hinder measurement equivalence and may lead to bias in the latent trait estimates (Millsap and Everson 1993; Holland and Wainer 2012). In turns out that DIF analysis can be performed by including and anlyzing specific person-by-item covariates. A detailed discussion about this approach is provided in De Boeck *et al.* (2011).

Depending on the nature of the covariates and over which mode their effects are assumed to vary, the full model may not be identified or at least hard to estimate and interprete. Thus, careful specification of covariates is critical to obtain sensible results. De Boeck *et al.* (2011) provide a thoughtful and thorough discussion of the use covariates in IRT models and I do not want to reiterate every detail, but simply note that all kinds of covariate models discussed in their paper may be specified in the here presented framework using the same formula syntax.

2.4. Prior distributions of person and item parameters

In Bayesian statistics, we are interested in the posterior distribution $p(\theta, \xi|y)$ of the person and item parameters given the data[2]. The posterior distribution is computed as

$$p(\theta, \xi|y) = \frac{p(y|\theta, \xi)\, p(\theta, \xi)}{p(y)}.$$

In the above equation $p(y|\theta, \xi)$ is the likelihood, $p(\theta, \xi)$ is the prior distribution and $p(y)$ is the marginal likelihood. The likelihood $p(y|\theta, \xi)$ is the distribution of the data given the parameters and thus relates the the data to the parameters. We may also describe the likelihood as the combination of response distribution and predictor terms discussed above. The prior distribution $p(\theta, \xi)$ describes the uncertainty in the person and item parameters before having seen the data. It thus allows to explicitely incorporate prior knowledge into the model. In practice, we will factorize the joint prior $p(\theta, \xi)$ into the product of $p(\theta)$ and $p(\xi)$ so that we can specify priors on person and items parameters independently. The marginal likelihood $p(y)$ serves as a normalizing constant so that the posterior is an actual probability distribution. Except in the context of specific methods (i.e., Bayes factors), $p(y)$ is rarely of direct interest.

In frequentist statistics, parameter estimates are usually obtained by finding those parameter values that maximise the likelihood. In contrast, Bayesian statistics estimate the full (joint) posterior distribution of the parameters. This is not only fully consistent with probability theory, but also much more informative than a single point estimate (and an approximate measure of uncertainty commonly known as 'standard error').

Obtaining the posterior distribution analytically is only possible in certain cases of carefully chosen combinations of prior and likelihood, which may considerably limit modelling flexibilty but yield a computational advantage. However, with the increased power of today's computers, Markov-Chain Monte-Carlo (MCMC) sampling methods constitute a powerful and feasible alternative to obtaining posterior distributions for complex models in which the majority of modeling decisions is made based on theoretical and not computational grounds. Despite all the computing power, these sampling algorithms are computationally very intensive and thus fitting models using full Bayesian inference is usually much slower than in point

[2]In IRT covariate models, the posterior distribution also includes the covariates' coefficients and all hyperparameters, but I keep this implicit in the equations to simplify the notation.

estimation techniques. However, advantages of Bayesian inference – such as greater modeling flexibility, prior distributions, and more informative results – are often worth the increased computational cost (Gelman, Carlin, Stern, Dunson, Vehtari, and Rubin 2013).

In the following, I will explain important aspects concerning the choice of priors for person parameters, although the same ideas apply to item parameters as well. A key decision when setting up an IRT model is whether we want person parameter (and/or items parameters) to share a common hierarchical prior or if we want to specify independent priors on each parameter. In the latter case, we would choose a prior and then fix its hyperparameters according to our understanding of the scale and prior knowledge about the parameter(s) to be estimated (Gelman *et al.* 2013). To make a concrete example, we can assume a normal distribution with mean 0 and standard deviation 3 for the person parameters of a Rasch model:

$$\theta_p \sim \text{Normal}(0, 3)$$

By definition of the normal distribution, we thus assume a-priori, that with 68% probability person parameters lie within $[-3, 3]$ and that with 97.5% probability person parameters lie within $[-6, 6]$ on the logit scale. Given the scale of the logistic response function, this prior can be considered weakly informative. That is, it restricts the parameters to a resonable range of values without strongly influencing the obtained posterior distribution. Of course, we don't need to restrict ourselves to normal distributions. Other prior distributions, such as a student-t distribution are possible as well, although assuming a normal distribution is arguably a good default choice (see also McElreath 2017).

A fundamentally different class of priors arises when assuming the person parameters to have the same underlying prior distribution with shared hyperparameters. Most commonly, a centered normal distribution is used so that

$$\theta_p \sim \text{Normal}(0, \sigma_\theta)$$

for all θ_p, which share a common standard deviation σ_θ. The latter is estimated as part of the model. Such a prior implies that parameters are shrunken somewhat towards their joint mean, a phenomenon also known as *partial pooling* (Gelman and Hill 2006). Partial pooling makes parameter estimates more robust as well as less influenced by extreme patterns and noise in the data (Gelman and Hill 2006). In the same way as for persons parameters, we may also partially pool item parameters so that

$$\xi_i \sim \text{Normal}(0, \sigma_\xi)$$

for all ξ_i, which now share a common standard deviation σ_ξ. It is common in IRT to partially pool person parameters (De Boeck *et al.* 2011) and I will follow this approach throughout this paper although a no pooling approach could be adopted in **brms** as well. If we decide to partially pool both person and item parameters, we have to amend the model slightly by adding an overall intercept parameter b_0 to the linear predictor, which then becomes $b_0+\theta_p+\xi_i$. We do this in order to catch average deviations from zero, which would otherwise no longer be appropriately modeled as both person and item parameters had been (soft) centered around zero by the prior. Such a formulation of IRT models via partially pooled person and/or item

parameters moves them into the framework of *generalized linear multilevel models* (GLMMs) and allows corresponding GLMM software to fit certain kinds of IRT models (De Boeck *et al.* 2011).

The above model formulation implies that person parameters estimated on different distributional parameters are assumed independent of each other, which turns out to be too restrictive an assumption in many applications. At best, we cannot be sure a priori of their independence. Thus, accounting for their possible dependence appears to be the safer choice. Statistically, correlated person parameters are modeled via a hierarchcial multivariate normal distribution in the form of

$$(\theta_{1p}, \ldots, \theta_{Kp}) \sim \text{Multinormal}(0, \boldsymbol{\Sigma}_\theta)$$

where θ_{kp} is the person parameter of person p used in the prediction of the distributional parameter ψ_k and $\boldsymbol{\Sigma}_\theta$ is the covariance matrix determining both the scale and the dependence structure of the person parameters. A covariance matrix tends to be relatively hard to interpret. Accordingly it is usually advantageous to decompose the covariance matrix into a correlation matrix capturing the dependence structure and a vector of standard deviations capturing the scales of the person parameters:

$$\boldsymbol{\Sigma}_\theta = \mathbf{D}(\sigma_{\theta 1}, \ldots, \sigma_{\theta K})\, \boldsymbol{\Omega}_\theta\, \mathbf{D}(\sigma_{\theta 1}, \ldots, \sigma_{\theta K})$$

In the above equation, $\boldsymbol{\Omega}_\theta$ denotes the correlation matrix and $\mathbf{D}(\sigma_{\theta 1}, \ldots, \sigma_{\theta K})$ denotes the diagonal matrix with standard deviations $\sigma_{\theta K}$ on the diagonal. Of course, the same argument applies to item parameters estimated on different distributional parameters so that we may want to model

$$(\xi_{1p}, \ldots, \xi_{Kp}) \sim \text{Multinormal}(0, \boldsymbol{\Sigma}_\xi)$$

and then decompose $\boldsymbol{\Sigma}_\xi$ into a correlation matrix $\boldsymbol{\Omega}_\xi$ and a vector of standard deviations $(\sigma_{\xi 1}, \ldots, \sigma_{\xi K})$ analogously to $\boldsymbol{\Sigma}_\theta$.

What remains to be specified are priors on the hyperparameters, that is, on the standard deviations and correlation matrices. In short, for standard deviations, I recommend priors whose densities have a mode at zero and fall off strictily monotonically for increasing parameter values. Examples for such priors are half-normal or half-cauchy priors. For correlation matrices, I recommend the LKJ prior (Lewandowski, Kurowicka, and Joe 2009), with which we can assign equal density over the space of valid correlation matrices if desired. More details on hyperparameters in **brms** and Stan are provided in Bürkner (2017), Bürkner (2018), and the Stan User's Manual (Stan Development Team 2019).

Lastly, I want to discuss priors on covariate effects. A special complexity in that context is that the scale of the coefficients depends not only on the (link-transformed) scale of the response variable but also on the scale of the covariates themselves (and possibly also on the dependency between covariates). Additionally, the choice of priors depends on the goal we want to achieve by their means, for instance, improving convergence, penalizing unrealisticly large values, or covariate selection (see also Gelman, Simpson, and Betancourt 2017). **brms** supports several covariate priors, ranging from completely flat "uninformative" priors (the current default), over weakly-informative priors for mild regularization and improving convergence to priors

intended for variable selection such as the horseshoe prior (Carvalho, Polson, and Scott 2010; Piironen, Vehtari *et al.* 2017). In general, setting priors is an active area of research and I hope that we can further improve our understanding of and recommendations for priors in the future.

3. Model specification in brms

In **brms**, specifying a GLMM of person and item parameters is done mainly via three arguments: `family`, `formula`, and `prior`. I will explain each of them in detail in the following.

3.1. Specifying the family argument

The model `family` specifies the response distribution as well as the response functions of the predicted distributional parameters. Following the convention of GLM theory, I do not specify the response function directly but rather its inverse, which is called the link function[3]. In **brms**, each response distribution has a dedicated primary parameter $\psi_1 = \mu$ that usually describes the mean of the distribution or some other measure of central tendency. This primary parameter is accompanied by a corresponding link function, which, as explained above, ensures that μ is on the scale expected by the distribution. In the **brms** framework, a `family` can be specified via

```
R> family = brmsfamily(family = "<family>", link = "<link>")
```

where `<family>` and `<link>` have to be replaced the the names of the desired response distribution and link function of μ, respectively. For binary responses, we could naturally assume a Bernoulli distribution and a `logit` function, which would then be passed to **brms** via

```
R> family = brmsfamily(family = "bernoulli", link = "logit")
```

The Bernoulli distribution has no additional parameters other than μ, but most other distributions do. Take, for instance, the normal distribution, which has two parameters, the mean μ and the residual standard deviation σ. The mean paramter μ can take on all real values and thus, using the identity link (i.e., no transformation at all) is a viable solution. If we assumed σ to be constant across observations, we would simply specify

```
R> family = brmsfamily(family = "gaussian", link = "identity")
```

If, however, we also modeled σ as depending on item and/or person parameters, we would need to think of a link function for σ as well. This is because σ is a standard deviation, which, by definition, can only take on positive values. A natural choice to restrict predictions to be positive is the log link function with the corresponding exponential response function, which is used as the default link for σ. To make this choice explicit, we write

[3]In my opinion, the convention of specifying link functions instead of response functions is unfortunate. I think it is more natural to transform linear predictors to the scale of the parameter via the response function, rather than transforming the parameter to the scale of the linear predictor.

```
R> family = brmsfamily(family = "gaussian", link = "identity",
R>                     link_sigma = "log")
```

An overview of available families in **brms** together with their distributional parameters and supported link functions is provided in `?brmsfamily`. Details about the parameterization of each family are given in `vignette("brms_families")`. If the desired response distribution is not available as a built-in family, users may specify their own custom families for use in **brms**. Details on custom families can be found by typing `vignette("brms_customfamilies")` in the console.

3.2. Specifying the formula argument

I will now discuss the `formula` argument of **brms**. Throughout this paper, I will assume the response variable to be named `y` and the person an item indicators to be named `person` and `item`, respectively. Of course, these names are arbitary and can be freely chosen by the user as long as the corresponding variables appear in the data set. If we just predict the main parameter μ of the response distribution (i.e., the mean or some other measure of central tendency), we just need a single R formula for the model specification. If we want to apply partial pooling to the person parameters but not to the item parameters, we would write

```
R> formula = y ~ 0 + item + (1 | person)
```

Instead, if we wanted to partially pool both person and item parameters, we would write

```
R> formula = y ~ 1 + (1 | item) + (1 | person)
```

Throughout this paper, I will model both person and item parameters via partial pooling as I believe it to be the more robust approach, which also scales better to more complex models (Gelman and Hill 2006). If partial pooling of items is not desired, the expression `1 + (1 | item)` has to be replaced by `0 + item`.

In standard R formula syntax, from which **brms** formula syntax inherits, covariates may be included in the model by adding their names to the formula. For instance, if we wanted to model an overall effect of a covariate `x`, we would write

```
R> y ~ 1 + x + (1 | item) + (1 | person)
```

Additionally, if we wanted the effect of `x` to vary over items, we would write

```
R> y ~ 1 + x + (1 + x | item) + (1 | person)
```

Modeling covariate effects as varying over persons can be done analogously. Interactions are specified via the `:` operator. That is, for covariates `x1` and `x2` we add `x1:x2` to the formula in order to model their interaction. We may also use `x1 * x2` as a convenient short form for `x1 + x2 + x1:x2`. As the data is expected to be in long format, the syntax for covariate effects is independent of the covariate type, that is, whether it is person or item related.

In most basic IRT models, only the mean of the response distribution is predicted while other distributional parameters, such as the residual standard deviation of a normal distribution, are assumed constant across all observations. Depending on the psychometric test, this may be too restrictive an assumption as items and persons not only differ in the mean response but also in other aspects, which are captured by additional parameters. To predict, multiple distributional parameters in **brms**, we need to specify one formula per parameter as follows:

```
R> formula = bf(
R>   y ~ 1 + (1 | item) + (1 | person),
R>   par2 ~ 1 + (1 | item) + (1 | person),
R>   par3 ~ 1 + (1 | item) + (1 | person),
R>   ...
R> )
```

The function `bf` is a shortform for `brmsformula`, which helps to set up complex models in **brms**. In the specification above, `par2` and `par3` are placeholders for the parameter names, which are specific to each response distribution, for instance, `sigma` in the case of the normal distribution. Covariates effects on such parameters may be included in the same way as described before.

The model formulation shown above implies that person and item parameters, respectively, of different distributional parameters are independent of each them to improve partial pooling across the whole model (see Section 2.4 for details). The solution implemented in **brms** (and currently unique to it) is to expand the `|` operator into `|<ID>|`, where `<ID>` can be any value. Person or item parameters with the same `ID` will then be modeled as correlated even though they appear in different R formulas. That is, if we want to model both person and item parameters as correlated across all distributional parameters, we choose some arbitray IDs, for instance `p` for person and `i` for item, and write

```
R> formula = bf(
R>   y ~ 1 + (1 |i| item) + (1 |p| person),
R>   par2 ~ 1 + (1 |i| item) + (1 |p| person),
R>   par3 ~ 1 + (1 |i| item) + (1 |p| person),
R>   ...
R> )
```

As discussed above, standard R formula syntax is designed to create additive predictors by splitting up the right-hand side of the `formula` in its unique terms separated from each other by `+` signs. This formulation is convenient and flexibile but it cannot be used to express non-linear predictors of arbitrary complexity. To achieve the latter, **brms** also features a second, more expressive way to parse R formulas. Suppose that the response `y` is related to some covariate `x` via a non-linear function `fun`. Further, suppose that the form of `fun` is determined by two parameters `nlpar1` and `nlpar2` which we need to estimate as part of the model fitting process. I will call them *non-linear parameters* to refer to the fact that they are parameters of a non-linear function. To complicate things, `nlpar1` and `nlpar2` are not necessarily constant across observations, but instead may vary across persons and item. That is, we need to specify a main non-linear formula as well as some additional linear formulas describing how

the non-linear parameters are predicted by person and item parameters. Basically, non-linear parameters are handled in the same way as distributional parameters. Suppose that `nlpar1` depends on both persons and items, while `nlpar2` just depends on the items. In **brms**, we can express this as

```
R> formula = bf(
R>   y ~ fun(x, nlpar1, nlpar2),
R>   nlpar1 ~ 1 + (1 | item) + (1 | person),
R>   nlpar2 ~ 1 + (1 | item),
R>   nl = TRUE
R> )
```

Using `nl = TRUE` is essential as it ensures that the right-hand side of the formula is taken literally instead of being parse via standard R formula syntax. Of course, we are not limited to one covariate and two non-linear parameters, but instead are able to specify any number of them in the formula. Further, the linear predictors of the non-linear parameters may contain all kinds of additive terms that I introduced above for usage with distributional parameters. This flexible combination of linear and non-linear formulas results in a model flexibility that, to my knowledge, is currently unmatched by any regression or IRT framework available in R or any other freely available programming language.

3.3. Specifying the prior argument

Prior specification is an essential part in the Bayesian workflow and **brms** offers an intuitive and flexible interface for convenient prior specification that can be readily applied to IRT models. In the following, I explain the syntax to specify priors in the proposed IRT framework. The priors I choose as examples below are not meant to represent any specific practical recommendations. Rather, the prior can only be understood in the context of the model it is a part of (Gelman *et al.* 2017). Accordingly, user-defined priors should always be chosen by keeping the model and relevant subject matter knowledge in mind. I will attempt to provide more ideas in this direction in Section 5.

The main function for the purpose of prior specification in **brms** is `set_prior`. It takes the prior itself in the form of a character string as well as additional arguments to define the parameters on which the prior should imposed. If we use partial pooling for item and/or person parameters, the normal prior on those parameters is automatically set and cannot be changed via the `prior` argument. However, we may change priors on the hyperparameters defining the covariance matrix of the person or item parameters that is on the standard deviations and correlation matrices. Suppose we want to define a half-Cauchy$(0, 5)$ prior on the standard devation σ_θ of the person parameters and an LKJ(2) prior on their correlation matrix $\mathbf{\Omega}_\theta$ across the whole model, then we write

```
R> prior = set_prior("cauchy(0, 5)", class = "sd", group = "person") +
R>   set_prior("lkj(2)", class = "cor", group = "person")
```

These priors will then apply to all distributional and non-linear parameters which vary across persons. As shown above, multiple priors may be combined via the `+` sign. Alternatively, `c()` or `rbind()` may be used to combine priors too. In Stan, and therefore also in **brms**, truncated

priors such as the half-Cauchy prior are implictely specified by imposing a hard boundary on the parameter, that is a lower boundary of zero for standard deviations, and then using the non-truncated version of the prior. Setting the hard boundary is done internally and so `"cauchy(...)"` will actually imply a half-Cauchy prior when used for a standard deviation parameter.

We can make priors specific to certain distributional parameters by means of the `dpar` argument. For instance, if we want a Gamma$(1, 1)$ prior on the person standard deviation of `dpar2` we write

```
R> prior = set_prior("gamma(1, 1)", class = "sd", group = "person",
R>                   dpar = "dpar2")
```

Analogously to distributional parameters, priors can be applied specifically to certain non-linear parameters by means of the `nlpar` argument.

If one chooses to *not* use partial pooling for the item parameters via formulas like `y ~ 0 + item + (1 | person)`, item parameters will be treated as ordinary regression coefficients and so their prior specification changes too. In this case, we are not limited to setting priors on all item parameters, but may also specify them differentially for certain items if desired. In **brms**, the class referring to regression coefficients if called `"b"`. That is, we can impose a Normal$(0, 3)$ prior on all item parameters via

```
R> prior = set_prior("normal(0, 3)", class = "b")
```

We may additionally set priors on the specific items. If, say, we know that `item1` will be relatively easy to answer correctly, we may encode this via a prior that has a mean greater than zero[4]. This could then look as follows:

```
R> prior = set_prior("normal(0, 3)", class = "b") +
R>   set_prior("normal(2, 3)", class = "b", coef = "item1")
```

Internally, **brms** will always search for the most specific prior provided by the user. If no user specified prior can be found, default priors will apply which are set to be very wide and can thus be considered non or weakly informative. Priors on the covariates can be specified in the same way as priors on non-hierarchical item parameters, that is via class `"b"`.

4. Parameter estimation and post-processing

The **brms** package uses Stan (Carpenter *et al.* 2017) on the back-end for the model estimation. Accordingly, all samplers implemented in Stan can be used to fit **brms** models. The flagship algorithm of Stan is an adaptive Hamiltonian Monte-Carlo (HMC) sampler (Betancourt, Byrne, Livingstone, and Girolami 2014; Betancourt 2017; Stan Development Team 2019), which represents a progression from the No-U-Turn Sampler (NUTS) by Hoffman and Gelman (2014). HMC-like algorithms produce posterior samples that are much less autocorrelated

[4]Rememember that **brms** uses the easiness formulation so that larger values mean higher probability of solving an item.

than those of other samplers such as the random-walk Metropolis algorithm (Hoffman and Gelman 2014; Creutz 1988). What is more, consequtive samples may even be anti-correlated leading to higher efficiency than completely independent samples (Vehtari, Gelman, Simpson, Carpenter, and Bürkner 2019). The main drawback of this increased efficiency is the need to calculate the gradient of the log-posterior, which can be automated using algorithmic differentiation (Griewank and Walther 2008), but is still a time-consuming process for more complex models. Thus, using HMC leads to higher quality samples but takes more time per sample than other typically applied algorithms. Another drawback of HMC is the need to pre-specify at least two parameters, which are both critical for the performance of HMC. The adaptive HMC Sampler of Stan allows setting these parameters automatically thus eliminating the need for any hand-tuning, while still being at least as efficient as a well tuned HMC (Hoffman and Gelman 2014). For more details on the sampling algorithms applied in Stan, see the Stan user's manual (Stan Development Team 2019) as well as Hoffman and Gelman (2014).

After the estimation of the parameters' joint posterior distribution, **brms** offers a wide range of post-processing options of which several are helpful in an IRT context. Below, I introduce the most important post-processing options. I will show their usage in hands-on examples in the upcoming sections. For a quick numerical and graphical summary, respectively, of the central model parameters, I recommend the `summary` and `plot` methods. The posterior distribution of person parameters (and, if also modeled as varying effects, item parameters) can be extracted with the `coef` method. The `hypothesis` method can be used to easily compute and evaluate parameter contrasts, for instance, when the goal is to compare the difficulty of two items or the ability of two persons. A visualization of the effects of item or person covariates is readily available via the `marginal_effects` method.

With the help of the `posterior_predict` method, **brms** allows drawing samples from the posterior predictive distribution. This not only allows to make predictions for existing or new data, but also enables the comparison between the actual response y and the response $\hat{y}$ predicted by the model. Such comparisons can be visualized in the form of posterior-predictive checks by means of the `pp_check` method (Gabry, Simpson, Vehtari, Betancourt, and Gelman 2019). Further, via the `log_lik` method, the pointwise log-likelihood can be obtained, which can be used, among others, for various cross-validation methods. One widely applied cross-validation approach is leave-one-out cross-validation (LOO-CV; Vehtari, Gelman, and Gabry 2017b), for which an approximate version is available via the `loo` method of the **loo** package (Vehtari *et al.* 2017b; Vehtari, Gelman, and Gabry 2017a). If LOO-CV is not an option or if the approximation fails, exact k-fold cross-valdiation is available via the `kfold` method. The cross-validation results can be further post-processed for the purpose of comparison, selection, or averaging of models. In these contexts, the `loo_compare`, `model_weights`, and `pp_average` methods are particularily helpful.

In addition to cross-validation based fit measures, the marginal likelihood (i.e., the denomintor in Bayes' theorem) and marginal likelihood ratios, commonly known as Bayes factors, can be used for model comparison, selection, or averaging as well (Kass and Raftery 1995). In general, obtaining the marginal likelihood of a model is a computationally demanding task (Kass and Raftery 1995). In **brms**, this is realized via bridgesampling (Meng and Wong 1996; Meng and Schilling 2002) as implemented in the **bridgesampling** package (Gronau and Singmann 2018). The corresponding methods are called `bridge_sampler` to obtain (log) marginal likelihood estimates, `bayes_factor` to obtain Bayes factors and `post_prob` to obtain posterior model

Anger	Gender	item	resp	id	btype	situ	mode	r2
20	M	S1WantCurse	no	1	curse	other	want	N
11	M	S1WantCurse	no	2	curse	other	want	N
17	F	S1WantCurse	perhaps	3	curse	other	want	Y
21	F	S1WantCurse	perhaps	4	curse	other	want	Y
17	F	S1WantCurse	perhaps	5	curse	other	want	Y
21	F	S1WantCurse	yes	6	curse	other	want	Y
39	F	S1WantCurse	yes	7	curse	other	want	Y
21	F	S1WantCurse	no	8	curse	other	want	N
24	F	S1WantCurse	no	9	curse	other	want	N
16	F	S1WantCurse	yes	10	curse	other	want	Y

Table 1: First ten rows of the `VerbAgg` data.

probabilities based on prior model probabilities and marginal likelihood estimates.

5. Examples

In this section, I am going to discuss several examples of advanced IRT models that can be fitted with **brms**. I will focus on three common model classes: binary, ordinal, and reaction time models, but the discussed principles also apply to other types of responses that can be analyzed by means of IRT.

5.1. Binary Models

To illustrate the application of **brms** to binary IRT models, I will use the `VerbAgg` data set (De Boeck and Wilson 2004), which is included in the **lme4** package (Bates *et al.* 2015b).

```
R> data("VerbAgg", package = "lme4")
```

This data set contains responses of 316 participants on 24 items of a questionnaire on verbal aggression. Several item and person covariates are provided. A glimpse of the data is given in Table 1 and more details can be found by typing `?lme4::VerbAgg`.

Let us start by computing a simple 1PL model. For reasons discussed in Section 2, I partially pool person and item parameters by specifying the model as

```
R> formula_va_1pl <- bf(r2 ~ 1 + (1 | item) + (1 | id))
```

To impose a small amount of regularization on the model, I set half-Normal$(0, 3)$ priors on the hierarchical standard deviations of person and items parameters. Given the scale of the logistic response function, this can be regarded as a weakly informative prior.

```
R> prior_va_1pl <-
R+   prior("normal(0, 3)", class = "sd", group = "id") +
R+   prior("normal(0, 3)", class = "sd", group = "item")
```

The model is then fit as follows:

```
R> fit_va_1pl <- brm(
R>   formula = formula_va_1pl,
R>   data = VerbAgg,
R>   family = brmsfamily("bernoulli", "logit"),
R>   prior = prior_va_1pl
R> )
```

To get a quick overview of the model results and convergence, we can summarize the main parameters nummerically using the `summary` method:

```
R> summary(fit_va_1pl)

 Family: bernoulli
  Links: mu = logit
Formula: r2 ~ 1 + (1 | item) + (1 | id)
   Data: VerbAgg (Number of observations: 7584)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~id (Number of levels: 316)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     1.39      0.07     1.25     1.54        895 1.00

~item (Number of levels: 24)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     1.20      0.19     0.89     1.62        537 1.00

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept    -0.16      0.26    -0.71     0.34        205 1.03

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

A graphical summary of the marginal posterior densities as well as the MCMC chains is obtained via

```
R> plot(fit_va_1pl)
```

and shown in Figure 1. Before interpreting the results, it is crucial to investigate whether the model fitting algortihm converged to its target, that is, the parameters' posterior distribution for fully Bayesian models. There are multiple ways to investigate convergence. We could

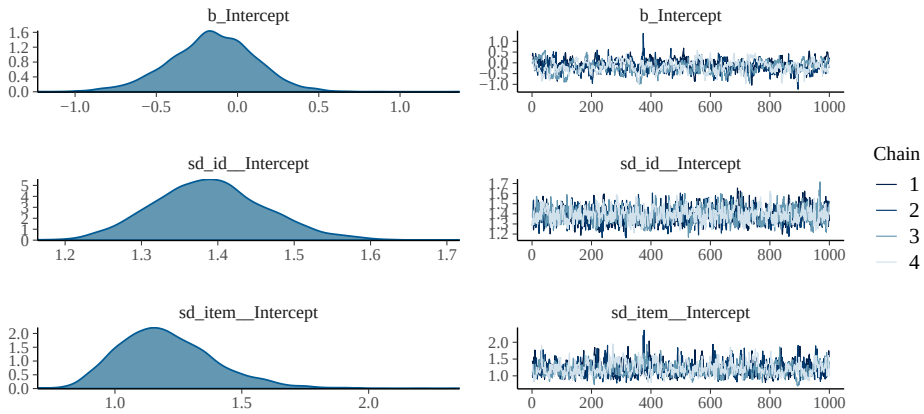

Figure 1: Summary of the posterior distribution of selected parameters obtained by model `fit_va_1pl`.

do so graphically by looking at trace plots (see the right-hand side of Figure 1) or more recently proposed rank plots (Vehtari *et al.* 2019). On that basis, we can interpret MCMC chains as having converged to the same target distribution, if the chains are mixing well individually (i.e., quickly jumping up and down) and are overlaying one another at the same time (Gelman *et al.* 2013). We may also investigate convergence numerically by mean of the scale reduction factor $\widehat{R}$ (Gelman and Rubin 1992; Gelman *et al.* 2013; Vehtari *et al.* 2019), which should be close to one (i.e., $\widehat{R} < 1.05$), and the effective sample size, which should be as large as possible but at least 400 to merely ensure reliable convergence diagnostics (Vehtari *et al.* 2019). The corresponding columns in the summary output are called `Rhat` and `Eff.Sample`. Convergence diagnostics for all model parameters can be obtained via the `rhat` and `neff_ratio` methods, respectively. Additionally, there are some diagnostics specific to (adaptive) HMC, which we can access using `nuts_params` and plotted via various options in `stanplot`. After investigating both the graphical and numerical indicators of convergence, we are confident that the model fitting algorithm succeeded so that we can start interpreting the results.

We see from the summary of the standard deviation parameters (named `sd(intercept)` in the output) that both persons and items vary substantially. Not all model parameters are shown in `summary` and `plot` to keep the output clean and readable and so we need to call other methods depending on what we are interested in. In IRT, this most likely includes the person and item parameters, which we can access via methods `coef` and `ranef` depending on whether or not we want to include overall effects (i.e., the global intercept for the present model) in the computation of the individual coefficients. This would typically be the case if we were interested in obtaining estimates of item difficulty or person ability. Otem and person parameters are displayed in Figure 2 and 3, respectively.

From Figure 2 it is clear that some items (e.g., the 4th item) are agreed on by a lot of individuals and thus have strongly positive easieness parameters, while other items (e.g., the 21th item) are mostly rejected and thus have a strongly negative easiness parameter. From Figure 3 we see that the person parameters vary a lot but otherwise show a regular pattern of blocks of persons getting very similar estimates. The latter is because, in the 1PL model, all items are assumed to have the same discrimination and are thus weighted equally. As a

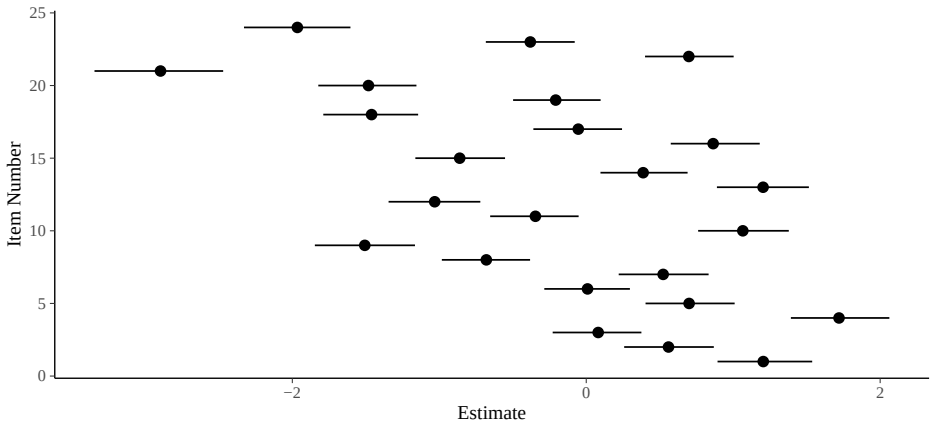

Figure 2: Posterior means and 95% credible intervals of item parameters as estimated by model `fit_va_1pl`.

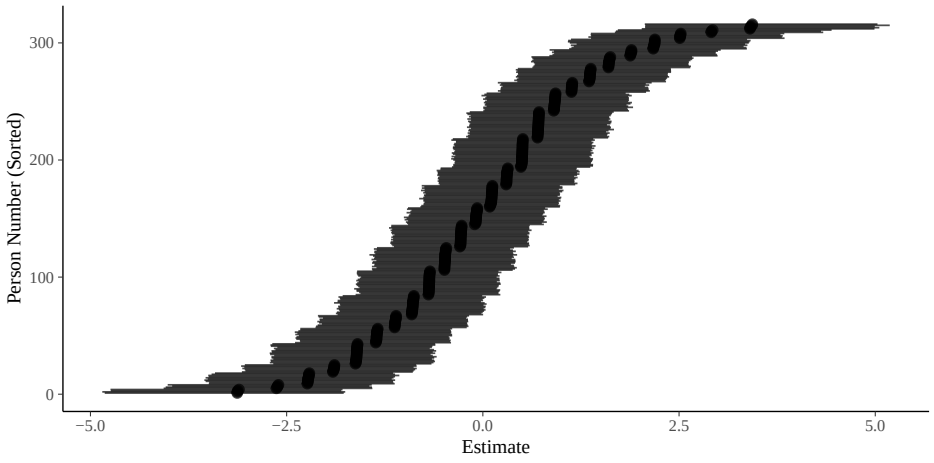

Figure 3: Posterior means and 95% credible intervals of person parameters (sorted) as estimated by model `fit_va_1pl`.

result, two persons endorsing the same number of items in total will receive the same estimate, regardless of which items they endorsed exactly. This assumption of equal discriminations is very restrictive and I will now investigate it in more detail. In a 2PL model, we would assume each item to have its own disrcimintation, which are to be estimated from the model along with all other parameters. Recall that mathematically, the 2PL model looks as follows:

$$P(y = 1) = \mu = \text{logistic}(\alpha_i(\theta_p + \xi_i))$$

Without any further restrictions, this model will likely not be identified (unless we were specifiying highly informative priors) because a switch in the sign of α_i can be corrected for by a switch in the sign of $\theta_p + \xi_i$ without a change in the overall likelihood. For this reason, I assume α_i to be positive for all items, a sensible assumption for the `VerbAgg` data set where a $y = 1$ always implies endorsing a certain verbally aggressive behavior. There are multiple ways to force α_i to be positive, one of which is to model it on the log-scale, that is to estimate $\log \alpha_i$ and then exponentiating the result to obtain the actual discrimination via $\alpha_i = \exp(\log \alpha_i)$.

```
R> formula_va_2pl <- bf(
R+   r2 ~ exp(logalpha) * eta,
R+   eta ~ 1 + (1 |i| item) + (1 | id),
R+   logalpha ~ 1 + (1 |i| item),
R+   nl = TRUE
R+ )
```

Above, I split up the non-linear model into two parts, `eta` and `logalpha`, each of which is in turn predicted by a linear formula. The parameter `eta` represents the sum of person parameter and item easiness, whereas `logalpha` represents the log discrimination. I modeled item easiness and discrimination as correlated by using `|i|` in both varying item terms (see Section 3). I impose weakly informative priors both on the intercepts of `eta` and `logalpha` (i.e., on the overall easiness and log discrimination) as well as on the standard deviations of person and item parameters.

```
R> prior_va_2pl <-
R+   prior("normal(0, 5)", class = "b", nlpar = "eta") +
R+   prior("normal(0, 1)", class = "b", nlpar = "logalpha") +
R+   prior("normal(0, 3)", class = "sd", group = "id", nlpar = "eta") +
R+   prior("normal(0, 3)", class = "sd", group = "item", nlpar = "eta") +
R+   prior("normal(0, 1)", class = "sd", group = "item", nlpar = "logalpha")
```

Finally, I put everything together and fit the model via

```
R> fit_va_2pl <- brm(
R>   formula = formula_va_2pl,
R>   data = VerbAgg,
R>   family = brmsfamily("bernoulli", "logit"),
R>   prior = prior_va_2pl,
R> )
```

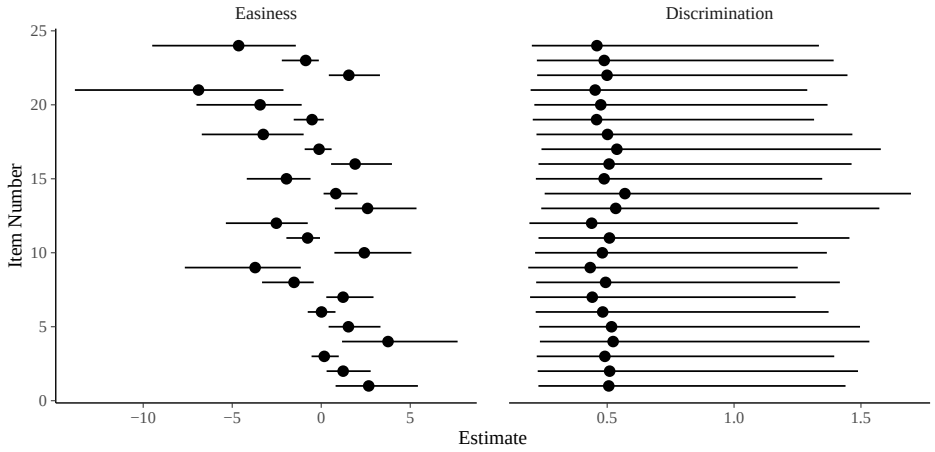

Figure 4: Posterior means and 95% credible intervals of item parameters as estimated by model `fit_va_2pl`.

The results of `summary` and `plot` indicate good convergence of the model and I don't show their outputs brevity's sake. Instead, I directly take a look at the item parameters in Figure 4. The discrimination estimates displayed on the right-hand have some considerable uncertainty, roughly between 0.3 and 1.2, but are overall very similar across items with posterior mean estimates of about 0.5. The easiness parameters displayed on the left-hand side still show a similar pattern as in the 1PL although their estimates are now more uncertain and spread out as a result of also estimating the discriminations.

The correlation between person parameters obtained by the two models turns out to be $r = 0.999$, so there is basically nothing gained from the 2PL model applied to this particular data set. In line with these results, model fit obtained via approximate leave-one-out cross-validation (LOO-CV) results in a LOOIC difference of $\Delta\text{LOOIC} = 4.32$ in favor of the 2PL model, which is very small both on an absolute scale and in comparison to its standard error $\text{SE} = 4.73$ depicting the uncertainty in the difference. Similarily, a model which assumes a constant discrimination across items, does not improve model fit noticably either ($\Delta\text{LOOIC} = 0.96$; $\text{SE} = 1.16$). For these reasons, I will continue to use the 1PL model in my further analysis of the data.

Modeling Covariates

When analysing the `VerbAgg` data set, I am not so much interested in the item and person parameters themselves, rather than in the effects of item and person covariates. I start by including only item covariates, in this case the behavior type (`btype`, with factor levels `curse`, `scold`, and `shout`), the situation type (`stype`, with factor levels `other` and `self`), as well as the behavior mode (`mode`, with factor levels `want` and `do`). Additionally, I assume the effect of `mode` to vary over persons, that is assume each person to have their own effect of `mode`. We specify this model in formula syntax as

```
R> r2 ~ btype + situ + mode + (1 | item) + (1 + mode | id)
```

This model assumes a varying intercept (i.e., baseline) and a varying effect of `mode` (i.e.,

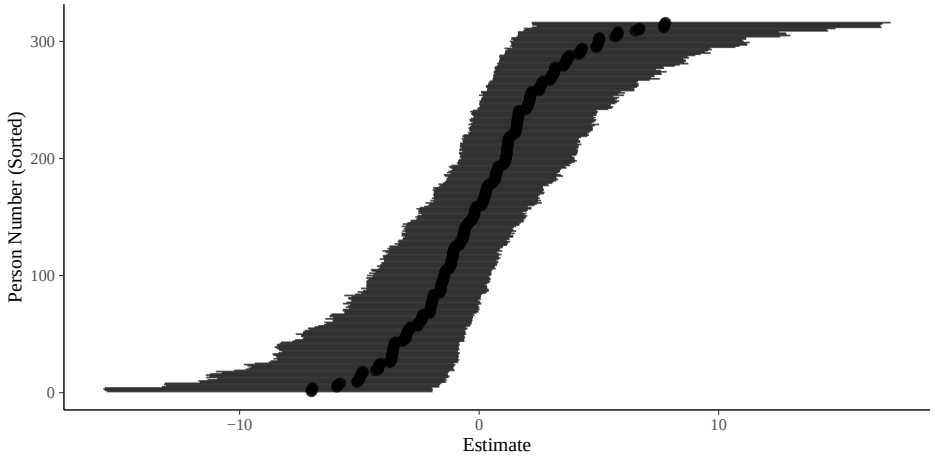

Figure 5: Posterior means and 95% credible intervals of person parameters (sorted) as estimated by model `fit_va_2pl`.

difference between `want` and `do`) per person. However, in this example, I am actually more interested in estimating varying effects of `want` and `do`, separately, in order to compare variation between these two modes. For this purpose, we slightly amend the formula, which now becomes

```
R> r2 ~ btype + situ + mode + (1 | item) + (0 + mode | id)
```

The notation `0 + mode` implies that each factor level of `mode` gets its own varying effect, instead of modeling the intercept and differences between factor levels. We are now ready to actually fit the model:

```
R> formula_va_1pl_cov1 <- bf(
R>   r2 ~ btype + situ + mode + (1 | item) + (0 + mode | id)
R> )
R> fit_va_1pl_cov1 <- brm(
R>   formula = formula_va_1pl_cov1,
R>   data = VerbAgg,
R>   family = brmsfamily("bernoulli", "logit"),
R>   prior = prior_va_1pl
R> )
```

As usual, a quick overview of the results can be obtained via

```
R> summary(fit_va_1pl_cov1)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: r2 ~ btype + situ + mode + (1 | item) + (0 + mode | id)
   Data: VerbAgg (Number of observations: 7584)
```

```
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~id (Number of levels: 316)
                     Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(modewant)             1.47      0.09     1.30     1.64       1814 1.00
sd(modedo)               1.67      0.10     1.48     1.87       1666 1.00
cor(modewant,modedo)     0.77      0.04     0.69     0.84       1680 1.00

~item (Number of levels: 24)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.46      0.09     0.31     0.67       1560 1.00

Population-Level Effects:
           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept      1.89      0.25     1.42     2.40       1835 1.00
btypescold    -1.13      0.24    -1.62    -0.64       1867 1.00
btypeshout    -2.24      0.25    -2.74    -1.74       2078 1.00
situself      -1.12      0.21    -1.53    -0.71       2088 1.00
modedo        -0.78      0.21    -1.20    -0.38       2170 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

From the summary output, we see that the behavior difference of the `do` and `want` behavior modes has a negative logit regression coefficient (b = -0.78, 95% CI = [-1.2, -0.38]), which implies that, holding other predictors constant, people are more likely to *want* to be verbally agressive than to actually *be* verbally agressive. However, although the direction of the effect is quite clear, its maginitude tends to be hard to interprete as it the regression coefficients are on the logit scale. To ease interpretation, we can transform and plot them on the original probability scale (see Figure 6) using a single line of code:

```
R> marginal_effects(fit_va_1pl_cov1, "mode")
```

Further, in the summary output, we see that both modes vary substantially over persons, with a little bit more variation in mode `do`. We may ask the question how likely it is, that the variation in `do` across persons is actually larger than the variation in `want`. Answering such a question in a frequentist framework would not be easy as the joint distribution of the two SD parameters is unlikely to be (bivariate) normal. In contrast, having obtained samples from the joint posterior distribution using MCMC sampling, as we did, computing the posterior distribution of the difference becomes a matter of computing the difference for each pair of posterior samples. This procedure of transforming posterior samples is automated in the `hypothesis` method of **brms**. For this particular question, we need to use it as follows:

```
R> hyp <- "modedo - modewant > 0"
R> hypothesis(fit_va_1pl_cov1, hyp, class = "sd", group = "id")
```

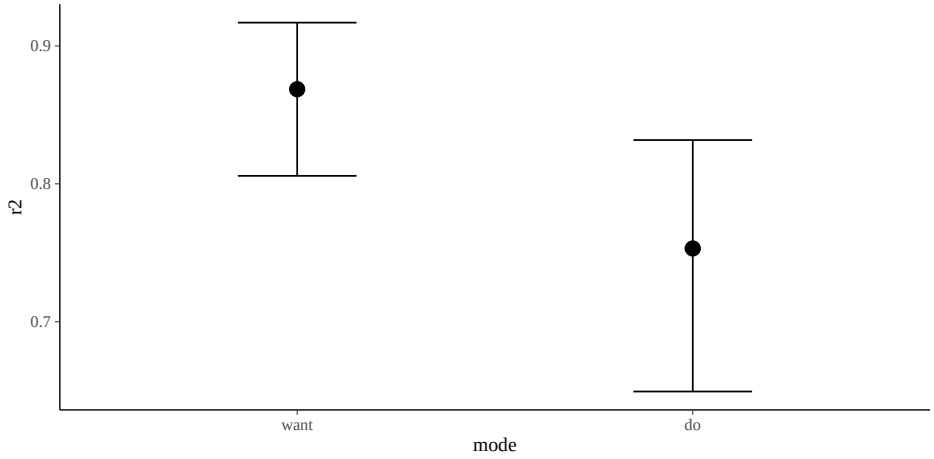

Figure 6: Expected probabilities of agreeing to an item in the VerbAgg data set as a function of the behavior mode conditioned on all other covariates being set to their reference categories.

```
             Hypothesis Estimate CI.Lower CI.Upper Post.Prob
1 (modedo-modewant) > 0      0.2     0.02     0.39      0.97
```

(output shortend for readability; CI denotes 90% the credibly interval). From the `Post.Prob` column we see that, given the model and the data, with 0.97 probability the SD of the `do` effects is higher than the SD of the `want` effects, although the expected SD difference of 0.2 (on the logit scale) is rather small.

Similarily to how we incorporate item covariates, we may also add person covariates to the model. In the `VerbAgg` data, we have information about the subjects' trait anger score `Anger` as measured on the Stat Trait Anger Expression Inventory (STAXI; Spielberger 2010) as well as about their `Gender`. Let us additionally assume `Gender` and `mode` to interact, that is allowing the effect of the behavior mode (`do` vs. `want`) to vary with the gender of the subjects. Further, I expect the individual item parameters to also vary with gender by replacing the term `(1 | item)` by `(0 + Gender | item)`. The complete model formula then looks as follows:

```
R> r2 ~ Anger + Gender + btype + situ + mode + mode:Gender +
R>   (0 + Gender | item) + (0 + mode | id)
```

We fit the model as usual with the `brm` function. Afterwards, we obtain a graphical summary of the effects of the newly added person covariates via

```
R> marginal_effects(fit_va_1pl_cov2, c("Anger", "mode:Gender"))
```

As visible on the left-hand side of Figure 7, increased trait anger is clearly associated with higher probabilities of agreeing to items in the `VerbAgg` data set. Also, as can be seen on the right-hand side of Figure 7, there is an interaction between behavior mode and gender. More specifically, women and men report wanting to be verbally aggressive by roughly the

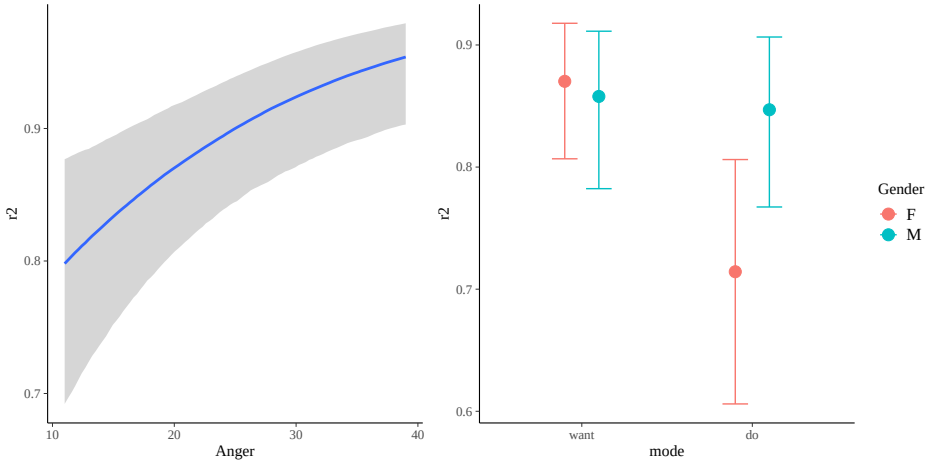

Figure 7: Expected probabilities of agreeing to an item in the VerbAgg data set as a function of the trait anger (left) and the interaction of behavior mode and subjects' gender (right) conditioned on all other categorical covariates being set to their reference categories and numerical covariates being set to their mean.

same probability, while men report actually being verbally aggressive with a much higher probability than women.

In all of the covariate models described above, there is no particular reasoning behind the choice of which item or person covariates are assumed to vary over persons or items, respectively, and which are assumed to be constant. We may also try to model multiple or even all item covariates as varying over persons and all person covariates as varying over items. In fact, this maximal multilevel approach may be more robust and conservative (Barr, Levy, Scheepers, and Tily 2013). In frequentist implementations of multilevel models, we often see convergence issues when using maximal multilevel structure (Bates, Kliegl, Vasishth, and Baayen 2015a). This has been interpreted by some as an indication of overfitting (Bates *et al.* 2015a) while others disagree (Barr *et al.* 2013). In any case, convergence issues seem to be a crude indicator of overfitting that I argue should not be blindly relied on. Fortunately, convergence of complex multilevel models turns out to be much less of a problem when using gradient based MCMC samplers such as HMC (Hoffman and Gelman 2014). For instance, when fitting a maximal multilevel structure of item and person covariates via the formula

```
R> r2 ~ 1 + Anger + Gender + btype + situ + mode +
R>   (1 + Anger + Gender | item) + (1 + btype + situ + mode  | id)
```

the **lme4** package indicates serious convergence issues while the **brms** model converges just fine (results not displayed here, see the supplementary R code for details). Of course, this is not to say that such a multilevel structure is necessarily sensible. However, being able to fit those models allows for more principled ways of testing afterwards *if* the assumed complexity is actually supported by the data, for instance via cross-validation or Bayes factors.

Modeling Guessing Parameters

A common aspect of binary item response data in IRT is that persons may be able to simply guess the correct answer with a certain non-zero probability. This may happen in a forced choice format where the correct answer is presented along with some distractors. As a result, the probability of correctly answering an item never falls below the guessing probability, regardless of the person's ability. For instance, when assuming all alternatives to be equally attractive in the absense of any knowledge about the correct answer, the guessing probability is 1 divided by the total number of alternatives. Such a property of the administered items needs to be taken into account in the estimated IRT model. The most commonly applied model in such a situation is the 3PL model[5]. Mathematically, the model can be expressed as

$$P(y = 1) = \mu = \gamma_i + (1 - \gamma_i) \times \text{logistic}(\alpha_i(\theta_p + \xi_i))$$

where γ_i represents the guessing probability of item i and all other parameters have the same meaning as in the 2PL model.

The items of the `VerbAgg` data set do not have a forced choice response format – and no right or wrong answers either – and so modeling guessing probabilities makes little sense for that data. For brevity's sake, I am not going to introduce another data set on which I apply 3PL models, but instead only focus on showing how to express such a model in **brms** without actually fitting the model.

Suppose we have adminstered forced choice items with 4 response alternatives of which only one is correct, then – under the assumption of equal probabilities of choosing one of the alternatives in case of guessing – we obtain a guessing probability of 25%. When modeling this guessing probability as known and otherwise following the recommendation presented in Section 2, we can write down the formula of the 3PL model as follows:

```
R> formula_va_3pl <- bf(
R+   r2 ~ 0.25 + 0.75 * inv_logit(exp(logalpha) * eta),
R+   eta ~ 1 + (1 |i| item) + (1 | id),
R+   logalpha ~ 1 + (1 |i| item),
R+   nl = TRUE
R+ )
R> family_va_3pl <- brmsfamily("bernoulli", link = "identity")
```

Above, I incorporated the logistic response function directly into the formula via `inv_logit`. As a result, the predictions of the overall sucess probabilities are already on the right scale and thus an additional usage of a link function is neither required nor reasonable. In other words, we have to apply the `identity` link function. Of course, we may also add covariates to all linear predictor terms of the model (i.e., to `eta` and `logalpha`) in the same way as demonstrated above for the 2PL model.

If we did not know the guessing probabilities, we can decide to estimate them along with all other model paramters. In **brms** syntax, the model then looks as follows:

[5]In addition to guessing probabilities, which increase the lower bound of success probability beyond 0, it is also possible that lapses decrease the upper bound of the sucess probability below 1. A binary model taking into account both guesses and lapses is referred to 4PL model. Arguably 4PL models more relevant for instance in psychophycis and less so in IRT. For that reason, I do not discuss it in more detail in this paper but want to point out that **brms** could also be used to fit 4PL models.

```
R> formula_va_3pl <- bf(
R+   r2 ~ gamma + (1 - gamma) * inv_logit(exp(logalpha) * eta),
R+   eta ~ 1 + (1 |i| item) + (1 | id),
R+   logalpha ~ 1 + (1 |i| item),
R+   logitgamma ~ 1 + (1 |i| item),
R+   nlf(gamma ~ inv_logit(logitgamma)),
R+   nl = TRUE
R+ )
```

There are some important aspects of this model specification that require further explanation. Since γ_i is a probability parameter, we need to restrict it between 0 and 1. One solution is to model γ_i on the logit scale via `logitgamma ~ 1 + (1 |i| item)` and then transform it back to the original scale via the `inv_logit` function, which exists both in **brms** and in Stan. I could have done this directly in the main formula but this would have implied doing the transformation twice, as `gamma` appears twice in the formula. For increased efficiency, I have defined both `gamma` and `logitgamma` as non-linear parameters and related them via `gamma ~ inv_logit(logitgamma)`. Passing the formula to `nlf` makes sure that the formula for `gamma` is treated as non-linear in the same way as setting `nl = TRUE` does for the main formula.

There are some general statistical problems with the 3PL model including estimated guessing probabilities, because the interpretability of the model parameters, in particular of the item difficulty and discrimination, suffers as a result (Han 2012). Accordingly, it may be more favorable to design items with known guessing probabilities in the first place.

5.2. Ordinal Models

When analysing the `VerbAgg` data using binary IRT models, I have assumed participants responses on the items to be a dichotomous `yes` vs. `no` decision. However, this is actually not entirely accurate as the actual responses were obtained on an ordinal three-point scale with the options `yes`, `perhaps`, `no`. In the former section, I have combined `yes` and `perhaps` into one response category, following the analysis strategy of De Boeck *et al.* (2011). In **brms**, we are not bounded to reducing the response to a binary decision but are instead able to use the full information in the response values by applying ordinal IRT models. There are multiple ordinal model classes (Agresti 2010; Bürkner and Vuorre 2019), one of which is the graded response model (GRM; see Section 2.1). As a reminder, when modeling the responses y via the GRM, we do not only have a predictor term η, but also a vector τ of $C - 1$ ordered latent thresholds, where C is the number of categories ($C = 3$ for the `VerbAgg` data). The GRM assumes that the observed ordinal responses arise from the categorization of a latent continuous variable, which is predicted by η. The thresholds τ indicate those latent values where the observable ordinal responses change from one to another category. An illustration of the model's assumptions is provied in Figure 8.

The model specification of the GRM, or for that matter of any ordinal model class, is highly similar to binary models. The only changes are that we switch out the binary variable `r2` in favor of the three-point ordinal variable `resp` and use the `cumulative` instead of the `bernoulli` family:

```
R> formula_va_ord_1pl <- bf(resp ~ 1 + (1 | item) + (1 | id))
```

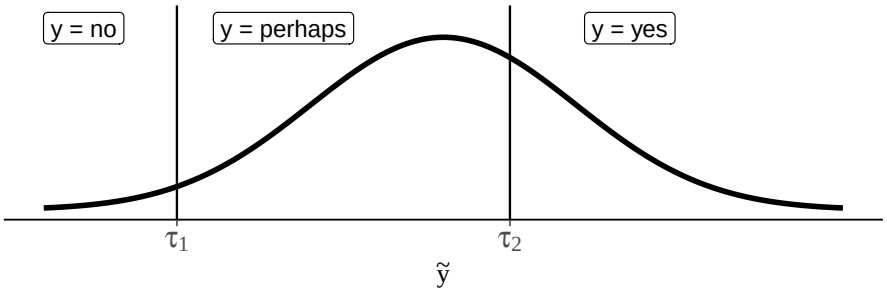

Figure 8: Assumptions of the graded response model when applied to the `VerbAgg` data. The area under the curve in each bin represents the probability of the corresponding event given the set of possible events for the latent variable $\tilde{y}$, which depends linearly on the predictor term η.

```
R> fit_va_ord_1pl <- brm(
R>   formula = formula_va_ord_1pl,
R>   data = VerbAgg,
R>   family = brmsfamily("cumulative", "logit"),
R>   prior = prior_va_1pl
R> )
```

The `summary` and `plot` output look very similar to the ones from the binary model except for that we now see two intercepts, which represent the ordinal thresholds. I do not show their outputs here for brevity's sake. Instead, let us focus on what exactly has changed in the estimation of the person parameters. As displayed on the left-hand side of Figure 9, person parameters estimated by the binary and those estimated by the ordinal model are largely in alignment with each other although we can observe bigger differences for larger values. The latter is to be expected since, in the ordinal model, I kept the two higher categories `perhaps` and `yes` separate thus increasing the information for larger but not so much for smaller person parameters. In accordance with this observation, we see that the person parameters whose precision has increased the most through the usage of an ordinal model are those with large mean values (see right-hand side of Figure 9). Taken together, we clearly gain something from correctly treating the response as ordinal, not only theoretically – `perhaps` is certainly something else than `yes` in most people's mind – but also statistically by increasing the precision of the estimates.

Similar to the binary case, one important extention to the standard GRM is to assume varying discriminations across items. The resulting generalized GRM is also a generalization of the binary 2PL model for ordinal responses. We have seen in Section 5.1 that discriminations were very similar across items in the binary case and I now want to take a look again when modeling the ordinal responses. We have to use slightly different formula syntax, though, as the non-linear syntax of **brms** cannot handle the ordinal thresholds in the way that is required when adding discrimination parameters. However, as having discrimination parameters in ordinal models is crucial for IRT, **brms** now provides a distributional parameter `disc` specifically for that purpose. We can predict this discrimination parameter using the distributional

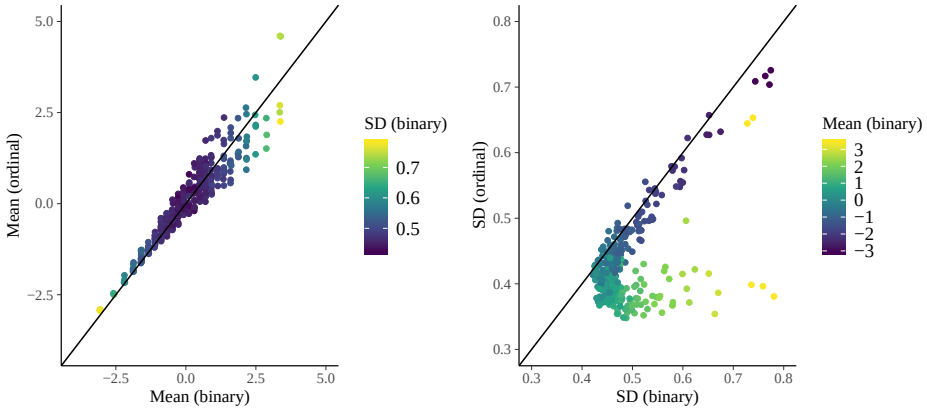

Figure 9: Relationship of person parameters estimated by the binary 1PL model and the ordinal graded response model. Posterior means are shown on the left-hand side and Posterior standard deviations are shown on the right-hand side.

regression framework[6]. By default, `disc` is modeled on the log-scale to ensure that the actual discrimination estimates are positive (see Section 5.1 for discussion in that issue). The model formula of the generalized GRM is given by

```
R> formula_va_ord_2pl <- bf(
R+   resp ~ 1 + (1 |i| item) + (1 | id),
R+   disc ~ 1 + (1 |i| item)
R+ )
```

We specify some weakly informative priors on the hierarchical standard deviations

```
R> prior_va_ord_2pl <-
R+   prior("normal(0, 3)", class = "sd", group = "id") +
R+   prior("normal(0, 3)", class = "sd", group = "item") +
R+   prior("normal(0, 1)", class = "sd", group = "item", dpar = "disc")
```

and finally fit the model:

```
R> fit_va_ord_2pl <- brm(
R>   formula = formula_va_ord_2pl,
R>   data = VerbAgg,
R>   family = brmsfamily("cumulative", "logit"),
R>   prior = prior_va_ord_2pl
R> )
```

A visualization of the item parameters can be found in Figure 10, in which we clearly see that discrimination does not vary across items in the GRM either.

[6]If `disc` is not predicted, it is automatically fixed to `1`.

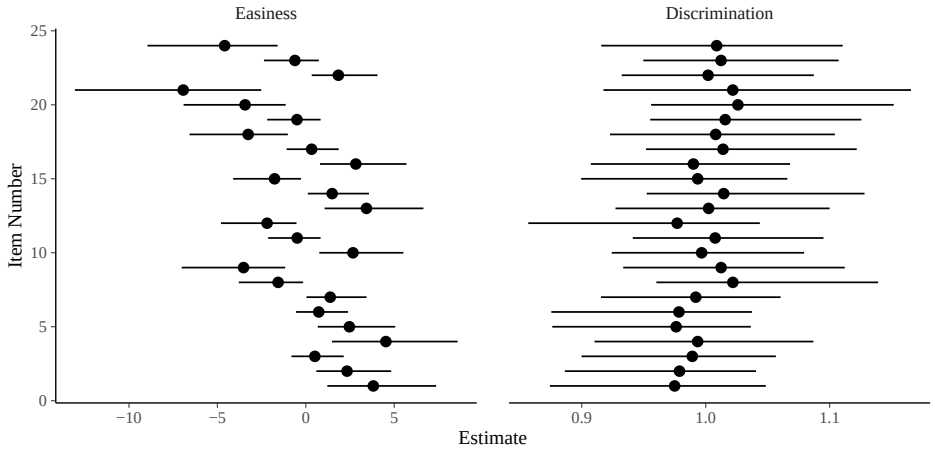

Figure 10: Posterior means and 95% credible intervals of item parameters as estimated by model `fit_va_ord_2pl`.

Having made the decision to stick to the GRM with constant discrimination, I again turn to the analysis of item and person covariates. These can be specified in the same way as for binary models. For instance, the GRM with both item and person covariates, and interaction between `mode` and `Gender`, as well as varying item parameters over `Gender` and varying person person parameters over `mode` would look as follows:

```
R> resp ~ Anger + Gender + btype + situ + mode + mode:Gender +
R>   (0 + Gender | item) + (0 + mode | id)
```

We can fit it the model as usual with the `brm` function and focus on the effect of trait `Anger` covariate in the following. First, let us compare the regression coefficients of `Anger` as obtained by the binary model and the GRM. We obtain $b_{\text{1PL}} = 0.06$ (95% CI = [0.02, 0.09]) for the 1PL model and $b_{\text{GRM}} = (0.07,$ 95% CI = [0.04, 0.11]) for the GRM, which are actually quite similar. Of course, this it not necessarily true in general and we cannot know for sure before having fitted both models. What will clearly be different are the predicted response probabilities as we now have three instead of two categories:

```
R> marginal_effects(fit_va_ord_cov1, effects = "Anger", categorical = TRUE)
```

As can be seen in Figure 11, increased trait anger is associated with higher probabilities of agreeing to items (`yes`) as compared to choosing `no` or `perhaps`. Although the plot may look like an interaction effect between `Anger` and the response variable `resp`, it really is just based on the single regression coefficient effecting the predicted probabilities of all response categories. Plotting predicted response probabilities instead of the response values themselves is recommend in ordinal models as the latter assumes equidistant categories, which is likely an invalid assumption for ordinal responses. That is, the perceived difference between `no` and `perhaps` in the participants' minds may be very different than the perceived difference between `perhaps` and `yes`.

This is also what leads us to another potential problem with the model assumptions, which is that the predictors are assumed to have a constant effect across all response categories. For

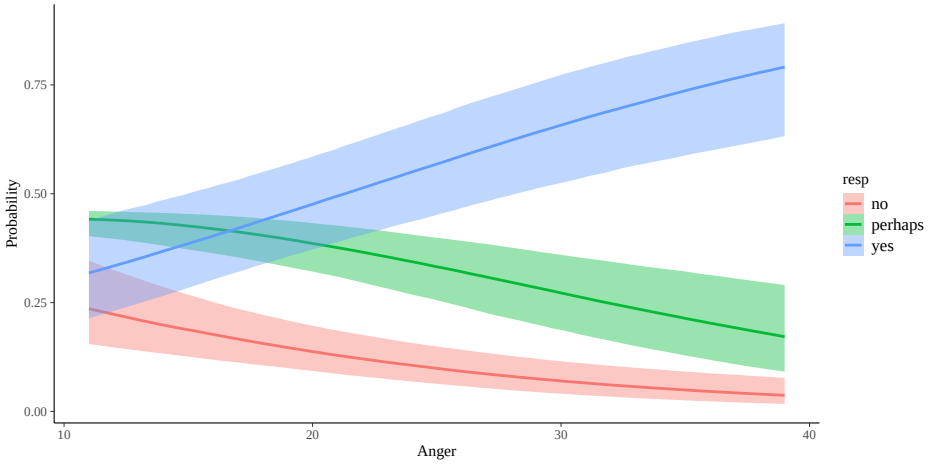

Figure 11: Expected probabilities of the three response categories in the `VerbAgg` data as a function of trait anger conditioned on all other categorical covariates being set to their reference categories and numerical covariates being set to their mean.

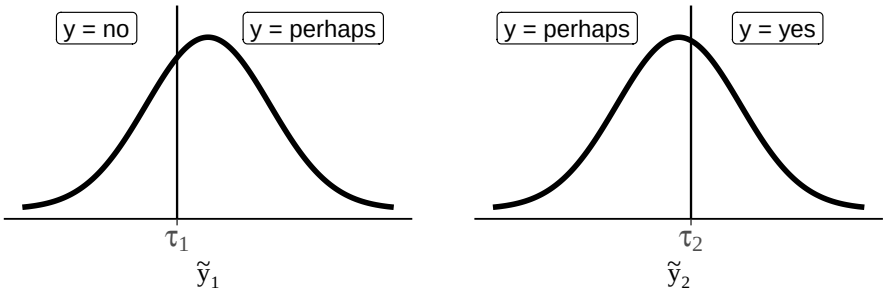

Figure 12: Assumptions of the partial credit model when applied to the `VerbAgg` data. The area under the curve in each bin represents the probability of the corresponding event given the set of possible events for the latent variables $\tilde{y}_1$ and $\tilde{y}_2$, respectively, which depend linearly on the predictor term η.

instance, it may very well be that `Anger` has little effect on the choice between `no` and `perhaps` but a much stronger one on the choice between `perhaps` and `yes`. This can be explicitely modeled and tested via what I call *category specific effects*, which imply estimating as many regression coefficients per category specific predictor as there are thresholds ($C - 1 = 2$ in our case). Unfortunately, we cannot reliably model category specific effects in the GRM as it may imply negative response category probabilities (Bürkner and Vuorre 2019). Instead, we have to use another ordinal model and I choose the *partial credit model* (PCM; Rasch 1961) for this purpose (see Section 2.1 for details). In the PCM, modeling category specific effects is possible because we assume not one but $C - 1$ latent variables which may have different predictor terms (see Figure 12 for an illustration).

Having selected an ordinal model class in which category specific effects are possible, all we need to do is wrap the covariate in `cs()` to estimate category specific effects. Suppose, we only want to model `Anger` as category specific, then we replace `Anger` with `cs(Anger)` in the

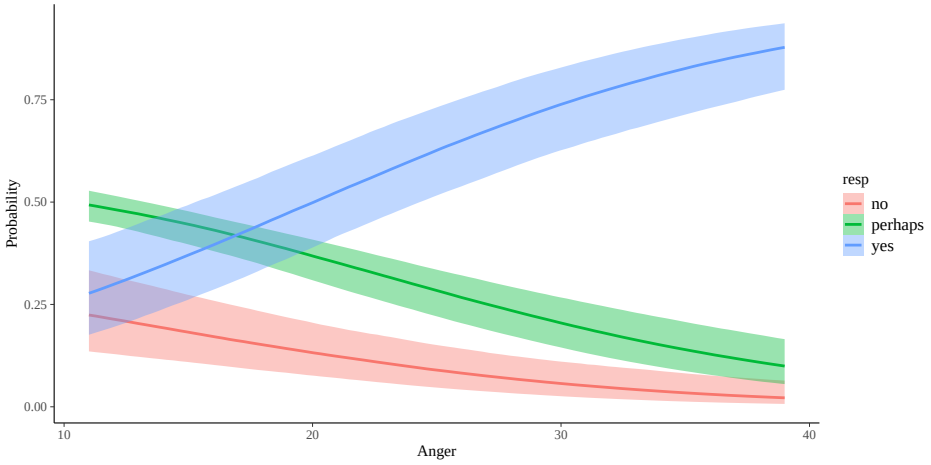

Figure 13: Expected response probabilities as predicted by model `fit_va_ord_cov2` as a function of trait anger conditioned on all other categorical covariates being set to their reference categories and numerical covariates being set to their mean.

model formula and leave the rest of the formula unchanged:

```
R> resp ~ cs(Anger) + Gender + btype + situ + mode + mode:Gender +
R>   (0 + Gender | item) + (0 + mode | id)
```

The model is then fitted with **brms** in the same way as the GRM except that we replace `family = brmsfamily("cumulative", "logit")` by `family = brmsfamily("acat", "logit")`. As the category specific coefficients for `Anger` on the logit-scale we obtain $b_{\text{PCM1}} = 0.03$ (95% CI = [0, 0.05]) and $b_{\text{PCM2}} = (0.1$, 95% CI = [0.07, 0.13]). That is, `Anger` seems to play a mucher stronger role in the decision between `perhaps` and `yes` than between `no` and `perhaps`. We may also visualize the effect via

```
R> marginal_effects(fit_va_ord_cov2, effects = "Anger", categorical = TRUE)
```

When we compare Figure 13 to Figure 11, we see that for higher `Anger` values a higher probability of choosing `yes` and a lower probability of choosing `perhaps` is predicted by the category specific PCM as compared to the basic GRM. This is also in accordance with the interpretation of the coefficients above.

5.3. Response Times Models

In this example, I will analyze a small data set of 121 subjects on 10 items measuring mental rotation that is shipped with the **diffIRT** package (Molenaar *et al.* 2015; see also van der Maas *et al.* 2011). The full data is described in Borst, Kievit, Thompson, and Kosslyn (2011). Each item consists of a graphical display of two 3-dimensional objects. The second object is either a rotated version of the first one or a rotated version of a different object. The degree of rotation (variable `rotate`) takes on values of 50, 100, or 150 and is constant for each item. Participants were asked whether the two objects are the same (yes/no) and the response is

person	item	time	resp	rotate
1	1	4.444	1	150
1	10	5.447	1	100
1	2	2.328	1	50
1	3	3.408	1	100
1	4	5.134	1	150
1	5	2.653	1	50
1	6	2.607	1	100
1	7	3.126	1	150
1	8	2.869	1	50
1	9	3.271	1	150

Table 2: First ten rows of the `rotation` data.

stored as either correct (1) or incorrect (0) (variable `resp`). The response time in seconds (variable `time`) was recorded as well. A glimpse of the data is provided in Table 2.

I will start by analyzing the response times, only, and use the exgaussian distribution for this purpose. Specifically, I am interested in whether the degree of rotation affects the mean, variation and right-skewness of the response times distribution. The effect of `rotate` can be expected to be smooth and monotonic (up to 180 degrees after which the effect should be declining as the objects become less rotated again) but otherwise of unknown functional form. In such a case, it could be beneficial to model the effect via some semi-parameteric methods such as splines or Gaussian processes (both of which is possible in **brms**), but this requires considerable more differentiated values of `rotate`. Thus, for this example, I will just treat `rotate` as a factor and use dummy coding with 50 degree as the reference category, instead of treating it as a continuous variable. Assuming all three parameters to vary over persons and items, we can write down the formula as

```
R> bform_exg1 <- bf(
R+   time ~ rotate + (1 |p| person) + (1 |i| item),
R+   sigma ~ rotate + (1 |p| person) + (1 |i| item),
R+   beta ~ rotate + (1 |p| person) + (1 |i| item)
R+ )
```

In theory, we could also model `rotate` as having a varying effect across persons (as `rotate` is an item covariate). However, as we are using a subset of only 10 items, modeling 9 varying effects per person, although possible, will likely result in overfitting. For larger data sets, this option could represent a viable option and deserves further consideration. Since both `sigma` (the standard deviation of the Gaussian component) and `beta` (the mean parameter of the exponential component representing the right skewness) can only take on positive values, I will use `log` links for both of them (this is actually the default but I want to make it explicite here). Together this results in the following model specification:

```
R> fit_exg1 <- brm(
R>   bform_exg1, data = rotation,
```

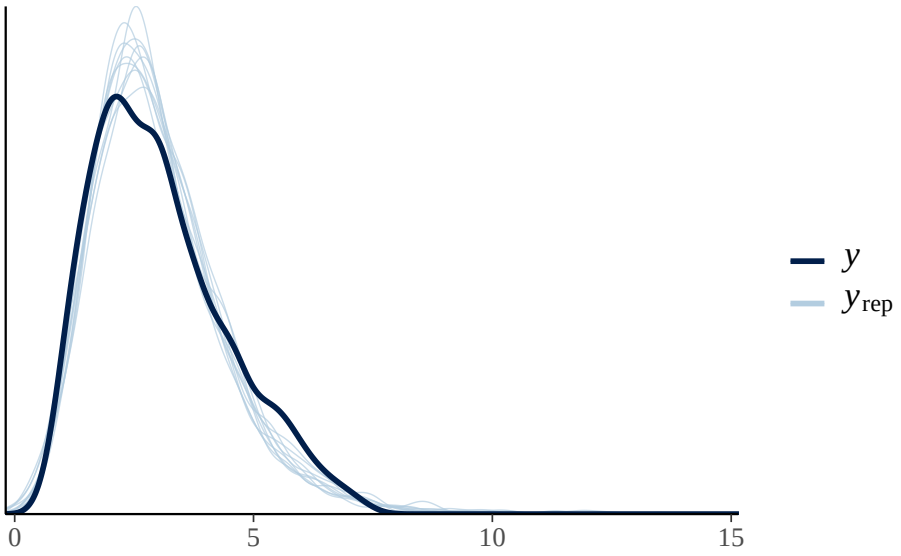

Figure 14: Posterior predictions of the exgaussian model `fit_exg1`.

```
R>   family = brmsfamily("exgaussian", link_sigma = "log", link_beta = "log"),
R>   chains = 4, cores = 4, inits = 0,
R>   control = list(adapt_delta = 0.99)
R> )
```

Increasing the sampling parameter `adapt_delta` reduces or ideally eliminates the number of "divergent transition" that indicate problems of the sampler exploring the full posterior distribution and thus bias the posterior estimates (Carpenter *et al.* 2017; Hoffman and Gelman 2014). From the standard outputs (not shown here), we can see that the model has converged well and produces reasonable posterior predictions (via `pp_check(fit_exg1)`; see Figure 14), so we can turn to investigating the effects of `rotate` on the model parameters:

```
R> marginal_effects(fit_exg1, "rotate", dpar = "mu")
R> marginal_effects(fit_exg1, "rotate", dpar = "sigma")
R> marginal_effects(fit_exg1, "rotate", dpar = "beta")
```

In Figure 15, we see that both the mean `mu` and the variation `sigma` increase with increasing degree of rotation, while the skewness `beta` rougly stays constant. The observation that mean and variation of response times increase simultaneously can be made in a lot of experiments and is discussed in Wagenmakers and Brown (2007).

The analysis of the response times is interesting, but does not provide a lot of insights into potentially underlying cognitive processes. For this reason, I will also use drift diffusion models to jointly model response times and the binary decisions. How the drift diffusion model looks exactly depends on several aspects. One is whether we deal with personality or ability tests. For personality tests, the binary response to be modeled is the actual *choice* between the two alternative, whereas for ability tests may want to rather use the *correctness* instead (Tuerlinckx and De Boeck 2005; van der Maas *et al.* 2011). Further, in the former

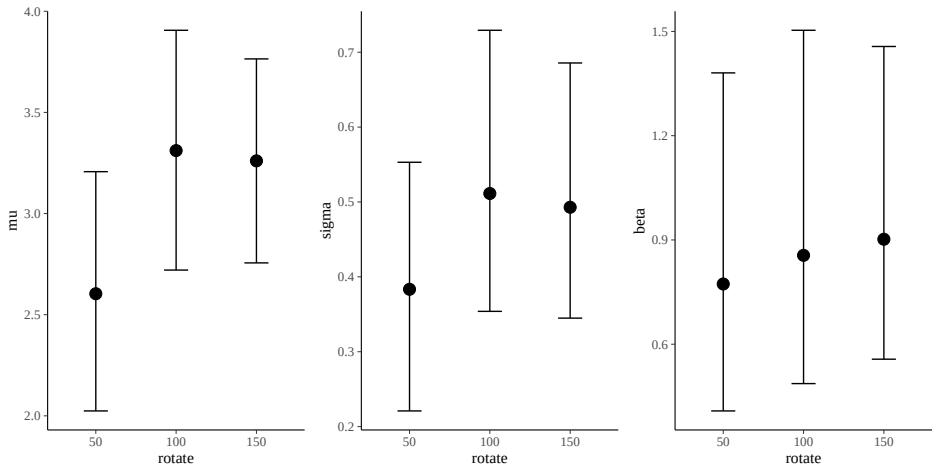

Figure 15: Parameters of the exgaussian model `fit_exg1` as a function of the degree of rotation.

case, person and item parameters may take on any real value and we combine them additively. In contrast, for ability tests, person and item parameters are assumed to be positive only and combined multiplicatively (van der Maas *et al.* 2011). The latter can also be expressed as an additive relationship on the log-scale. In the present example, we deal with data of an ability test and will use the described log-scale approach.

Again, my interest lies primarily with the effect of the degree of rotation. More specifically, I am interested in which ones of the three model parameters (drift rate, boundary separation, and non-decision time) are infludenced by the rotation. The fourth parameter, the initial bias, is fixed to 0.5 (i.e., no bias) to obtain the three-parameter drift diffuson model. Assuming all three predicted parameters to vary over persons and items, we write down the formula as

```
R> bform_drift1 <- bf(
R+   time | dec(resp) ~ rotate + (1 |p| person) + (1 |i| item),
R+   bs ~ rotate + (1 |p| person) + (1 |i| item),
R+   ndt ~ rotate + (1 |p| person) + (1 |i| item),
R+   bias = 0.5
R+ )
```

In Stan, drift diffusion models with predicted non-decision time are not only computationally much more demanding, but they also often require some manual specifcation of initial values. The easiest way is to set the intercept on the log-scale of `ndt` to a small value:

```
R> chains <- 4
R> inits_drift <- list(temp_ndt_Intercept = -3)
R> inits_drift <- replicate(chains, inits_drift, simplify = FALSE)
```

I will now fit the model. This may take some more time than previous models due to the complexity of the diffusion model's likelihood.

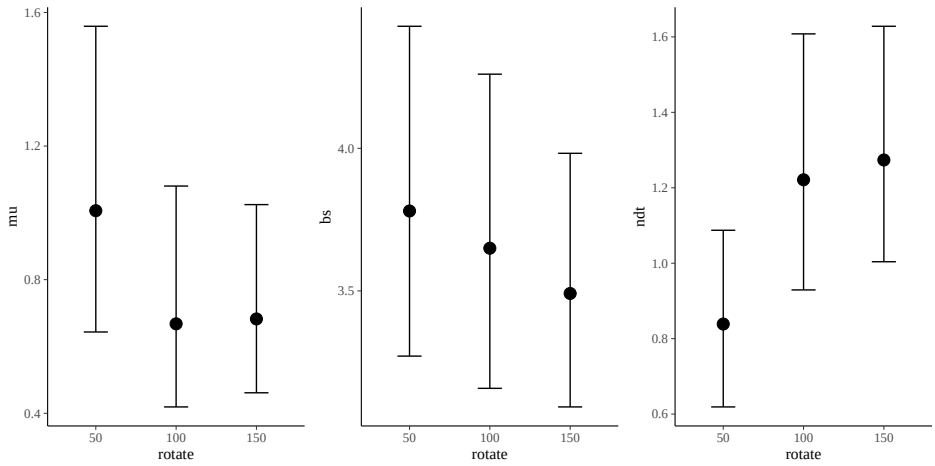

Figure 16: Parameters of the drift diffusion models as a function of the degree of rotation. The parameter `mu`, `bs`, and `ndf` represent the drift rate, boundary separation and non-decision time, respectively.

```
R> fit_drift1 <- brm(
R>   bform_drift, data = rotation,
R>   family = brmsfamily("wiener", "log", link_bs = "log", link_ndt = "log"),
R>   chains = chains, cores = chains,
R>   inits = inits_drift, init_r = 0.05,
R>   control = list(adapt_delta = 0.99)
R> )
```

From the standard outputs (not shown here), we can see that the model has converged well so we can turn to investigating the effects of `rotate` on the model parameters:

```
R> marginal_effects(fit_drift1, "rotate", dpar = "mu")
R> marginal_effects(fit_drift1, "rotate", dpar = "bs")
R> marginal_effects(fit_drift1, "rotate", dpar = "ndt")
```

As shown in Figure 16, both the drift rate and the non-decision time seem to be affected by the degree of rotation. The drift rate decreases slightly when increasing the rotation from 50 to 100 and roughly stays constant afterwards. Similarily, the non-decision time increases with increased rotation from 50 to 100 presumably as a result of the increased cognitive demand of processing the rotated objects (Molenaar *et al.* 2015).

In contrast, the boundary separation appears to be unaffected by the degree of rotation. Further, the standard deviation of the boundary separation across items (after controlling for the rotation), seems to be very small (SD = 0.05, 95%-CI = [0, 0.16]). We may also test this more formally by fitting a second model without item effects on the boundary separation, that is using the formula `bs ~ 1 + (1 |p| person)`, and then comparing the models for instance via approximate LOO-CV (method `loo`) or Bayes factors (method `bayes_factor`). The latter requires carefully specified prior distributions based on subject matter knowledge,

a topic which is out of the scope of the present paper.

6. Comparison of Packages

A lot of R packages have been developed that implement IRT models, each being more or less general in their supported models. In fact, for most IRT models developed in the statistical literature, we may actually find an R package implementing it. An overview of most of these packages is available on the Psychometrics CRAN task view (`https://cran.r-project.org/web/views/Psychometrics.html`). Comparing all of them to **brms** would be too extensive and barely helpful for the purpose of the present paper. Accordingly, I focus on a set of eight widely applied and actively maintained packages that can be used for IRT modeling. These are **eRm** (Mair and Hatzinger 2007), **ltm** (Rizopoulos 2006), **TAM** (Robitzsch *et al.* 2019), **mirt** (Chalmers 2012), **sirt** (Robitzsch 2019), **lme4** (Bates *et al.* 2015b), **lavaan** (Rosseel 2012), and **blavaan** (Merkle and Rosseel 2015). All of these packages are of high quality, user friendly, and well documented so I primarily focus my comparison on the features they support. A high level overview of the modeling options of each package can be found in Table 3 and more details are provided below.

eRm focuses on models that can be estimated using conditional maximum likelihood, a method only available for the 1PL model and PCM with unidimensional latent traits per person. Accordingly, its application is the most specialized among all the packages presented here. The **ltm**, **TAM**, and **mirt** packages all provide frameworks for fitting binary, categorical, and ordinal models using mostly marginal maximum likelihood estimation. They allow estimating discrimination parameters for all these model classes as well as 3PL or even 4PL models for binary responses. Of these three packages, **mirt** currently provides the most flexible framework with respect to both the models it can fit and the provided estimation algorithms. The package also comes with its own modeling syntax for easy specification of factor structure and parameter constraints. The **sirt** package, does not provide one single framework for IRT models but rather a large set of separate functions to fit special IRT models that complement and support other packages, in particular **mirt** and **TAM**. As a result, input and output structures are not consistent across model fitting functions within **sirt**, which makes it more complicated to switch between model classes. All of these IRT-specific packages have built-in methods for investigating and testing differential item functioning. In addition to these tools, a powerful approach for assessing differential item functioning via recursive partioning is implemented in **psychotree** (Strobl *et al.* 2015; Komboz *et al.* 2018) based on methods of the **psychotools** package (Zeileis, Strobl, Wickelmaier, Komboz, and Kopf 2018). It currently supports methods for dichotomous, categorical and ordinal models.

In contrast to the above packages, **lavaan** and **lme4** are not specifically dedicated to IRT modelling, but rather provide general frameworks for structural equation and multilevel models, respectively. Due to their generality and user-friendly interfaces, they have established themselves as the de facto standards in R when it comes to the frequentist estimation of these model classes. **lavaan** allows to fit multidimensional 1PL and 2PL binary, categorical and some ordinal IRT models using maximum likelihood or weighted least squares estimations. In addition, the **blavaan** package allows to fit **lavaan** models using Bayesian estimation methods. To date, not all **lavaan** models are available in **blavaan**, but I expect this to change in the future. **lme4** estimates multilevel models via marginal maximum likelihood estimation. While it is very flexible in the specification of covariates and multilevel structure, for instance for

the purpose of multidimensional IRT models, it neither supports 2PL (or more parameters) binary models, nor categorical or ordinal models.

brms is conceptually closest to **lme4** when it comes to the model specification and data structuring. These two packages expect the data to be in long format, that is all responses to be provided in the same column, while all other packages expect response to be in the form of a person $\times$ item matrix. Accordingly, the formula syntax also differs from the other packages in that we have to explicitely specify item and person grouping variables as they cannot be automatically identified from the data structure (see Section 3). The multilevel syntax of **lme4** and **brms** allows for an overall shorter model specification than the structural equation syntax of **lavaan** as items do not have to be targeted one by one. A drawback of the multilevel syntax is that constraining or fixing parameters is less intuitive and flexible than in the dedicated IRT packages or **lavaan** syntax.

What makes **brms** stand out is the combination of three key features. First, it extends the multilevel formula syntax of **lme4** to non-linear formulas of arbitrary complexity, which turns out to be very powerful for the purpose of IRT modelling (see Section 3). Second, it supports the widest range of response distributions of all the packages under comparison. This includes not only distributions for binary, categorical, and ordinal data, but also for response times, count, or even proportions to name only a few available options. Further, users may specify their own response distributions via the `custom_family` feature, fulfilling a similar purpose as `mirt::createItem` or `sirt::xxirt`. Third, not only the main location parameter but also all other parameters of the response distribution may be predicted by means of the non-linear multilevel syntax. In addition, multiple different response variables can be combined in a joint multivariate model in order to let person and/or item parameters inform each other, respectively, across response variables.

Another difference between **brms** and most of the other packages is that the former is fully Bayesian while the latter are mostly based point estimation methods. **TAM** and **mirt** support setting certain prior distributions on parameters but still perform estimation via optimization. **sirt** offers MCMC sampling only for 2PL and 3PL models with restrictive prior options and few built-in methods to post-process results. Similarily, **blavaan** can fit a subset of the models supported by **lavaan** using MCMC methods implemented in JAGS (Plummer 2013) or Stan (Carpenter *et al.* 2017) although the set of supported IRT models is currently much smaller than that of **brms** (see Table 3). While performing full Bayesian inference via MCMC sampling is often orders of magnitude slower than point estimation via maximum likelihood or least squares, the obtained information may be considered to be much higher: Not only do we get the posterior distribution of all model parameters, but also the posterior distribution of all quantities that can be computed on their basis (Gelman *et al.* 2013). For instance, the uncertainty in the parameters' posterior naturally propagates to the posterior predictive distributions, whose uncertainty can then be visualized along with the mean predictions. **brms** automates a lot of common post-processing tasks, such as posterior visualizations, predictions, and model comparison (see `methods(class = "brmsfit")` for a full list of options and the replication material of this paper for examples).

To what extent the increased information obtained via full Bayesian inference is worth the additional computational costs and corresponding waiting time depends on various factors related to the model, data, and goal of inference. For instance, if the model is relatively simple and there is a lot of data available to inform the model parameters, Bayesian and maximum likelihood estimates are unlikely to differ a lot unless strong prior information is

provided. Also, if the goal is to provide estimates in real time, for instance for the purpose of adaptive testing, full Bayesian inference may be too slow to be viable unless specifically tuned to such a task (e.g., see van der Linden and Ren 2015). I do not argue that a Bayesian approach to IRT is always superior, but instead want to point out its strengths (and also its weaknesses) so that users can make an informed decision as to when working with a Bayesian framework may improve the desired inference.

Similarily, while using general purpose frameworks for IRT such as those provided by **brms**, **lme4**, or **lavaan** may provide advantages in terms of modeling flexibility and consistency of model specification and post-processing, they clearly come with some disadvantages. Among others, such general frameworks are likely to require more work from the user at the start to familiarize themselves with the interface in order to fit the desired models as compared to packages with specific built-in function for common model classes. At the same time, post-processing methods of specialized software may be easier and more directly applicable to common use-cases, thus lowering the requirements in the actual coding expertise of users. For instance, the specification and post-processing of standard 1PL or 2PL models is more straightforward in dedicated IRT software and users only interested in such models may get reliable solutions faster this way. In other words, when introducing more and more general frameworks, the goal is not to render more specialized software irrelevant, but to provide an alternative for consistent model building an evaluation with a larger scope than specialized software is intended for.

7. Conclusion

In this paper, I have introduced a general framework for fitting Bayesian IRT models in R via **brms** and Stan. Within this framework, a wide range of IRT models can be specified, estimated, and post-processed in a consistent manner, without the need to switch between packages to obtain results for different IRT model classes. To my knowledge, the flexibility of the proposed framework is currently unmatched by any openly available IRT software. I have demonstrated its usefulness in examples of binary, ordinal, and response times data, although the framework entails a lot of other IRT model classes.

The advanced formula syntax of **brms** further enables the modeling of complex non-linear relationships between person and item parameters and the observed responses. However, the flexibility of the framework does not free the user from specifying reasonable models for their data. Just because a model can be estimated without problems does not mean it is also sensible from a theoretical perspective or provides valid inference about the effects under study. Tools for model comparison and selection as provided by **brms** may help in guiding users' decision, but should not be a substitute for clear theoretical reasoning and subject matter knowledge to guide model development and evaluation.

Taking a Bayesian perspective on specification, estimation, and post-processing of statistical models helps in building and fitting more complex and realistic models, but it is not the only reason for adopting it. As Bayesian statistics is fully embedded into probability theory, we can quantify uncertainty of any variable of interest using probability and make decisions by averaging over that uncertainty. Thus, we no longer have to fall back on premature binary decision making on the basis of, say, frequentist p-values or confidence intervals. As such, Bayesian inference is not just another estimation method but a distinct statistical framework

to reason from data using probabilistic models.

8. Acknowledgements

I would like to thank my colleagues of the Stan Development Team for creating, maintaining, and continuously improving Stan, which forms the basis for the success of **brms**. Further, I want to thank Marie Beisemann and Alexander Robitzsch for valuable comments on earlier versions of the paper. Finally, I would like to thank all the users who reported bugs or had ideas for new features, thus helping to further improve **brms**.

References

Adams RJ, Wu ML, Wilson M (2012). "The Rasch rating model and the disordered threshold controversy." *Educational and Psychological Measurement*, **72**(4), 547–573. doi:10.1177/0013164411432166.

Agresti A (2010). *Analysis of ordinal categorical data*. John Wiley & Sons. doi:10.1002/9780470594001.

Andrich D (2004). "Controversy and the Rasch model: a characteristic of incompatible paradigms?" *Medical Care*, **42**(1), 17–116.

Barr DJ, Levy R, Scheepers C, Tily HJ (2013). "Random effects structure for confirmatory hypothesis testing: Keep it maximal." *Journal of memory and language*, **68**(3), 255–278. doi:10.1016/j.jml.2012.11.001.

Bates D, Kliegl R, Vasishth S, Baayen H (2015a). "Parsimonious mixed models." *arXiv preprint*. URL https://arxiv.org/abs/1506.04967.

Bates D, Mächler M, Bolker B, Walker S (2015b). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48. doi:10.18637/jss.v067.i01.

Betancourt M (2017). "A Conceptual Introduction to Hamiltonian Monte Carlo." *arXiv preprint*. URL https://arxiv.org/pdf/1701.02434.pdf.

Betancourt M, Byrne S, Livingstone S, Girolami M (2014). "The Geometric Foundations of Hamiltonian Monte Carlo." *arXiv preprint*. URL https://arxiv.org/abs/1410.5110.

Bond TG, Fox CM (2013). *Applying the Rasch model: Fundamental measurement in the human sciences*. Psychology Press.

Borst G, Kievit RA, Thompson WL, Kosslyn SM (2011). "Mental rotation is not easily cognitively penetrable." *Journal of Cognitive Psychology*, **23**(1), 60–75. doi:10.1080/20445911.2011.454498.

Bürkner PC (2018). "Advanced Bayesian Multilevel Modeling with the R Package brms." *The R Journal*, **10**(1), 395–411. doi:10.32614/RJ-2018-017.

Bürkner PC (2017). "brms: An R Package for Bayesian Multilevel Models using Stan." *Journal of Statistical Software*, **80**(1), 1–28. doi:10.18637/jss.v080.i01.

Bürkner PC, Vuorre M (2019). "Ordinal Regression Models in Psychology: A Tutorial." *Advances in Methods and Practices in Psychological Science*, **2**(1), 77–101. URL 10.1177/2515245918823199.

Carpenter B, Gelman A, Hoffman M, Lee D, Goodrich B, Betancourt M, Brubaker MA, Guo J, Li P, Ridell A (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software*, **76**(1), 1–32. doi:10.18637/jss.v076.i01.

Carvalho CM, Polson NG, Scott JG (2010). "The horseshoe estimator for sparse signals." *Biometrika*, **97**(2), 465–480. doi:10.1093/biomet/asq017.

Chalmers RP (2012). "mirt: A Multidimensional Item Response Theory Package for the R Environment." *Journal of Statistical Software*, **48**(6), 1–29. doi:10.18637/jss.v048.i06.

Creutz M (1988). "Global Monte Carlo Algorithms for Many-Fermion Systems." *Physical Review D*, **38**(4), 1228–1238. doi:10.1103/PhysRevD.38.1228.

De Boeck P, Bakker M, Zwitser R, Nivard M, Hofman A, Tuerlinckx F, Partchev I (2011). "The estimation of item response models with the lmer function from the lme4 package in R." *Journal of Statistical Software*, **39**(12), 1–28. doi:10.18637/jss.v039.i12.

De Boeck P, Wilson M (2004). *Explanatory item response models*. John Wiley & Sons. doi:10.1007/978-1-4757-3990-9.

Embretson SE, Reise SP (2013). *Item response theory*. Psychology Press.

Gabry J, Simpson D, Vehtari A, Betancourt M, Gelman A (2019). "Visualization in Bayesian workflow." *Journal of the Royal Statistical Society A (Statistics in Society)*, **182**(2), 389–402. doi:10.1111/rssa.12378.

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013). *Bayesian Data Analysis (3rd Edition)*. Chapman and Hall/CRC. doi:10.1201/b16018.

Gelman A, Hill J (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Gelman A, Rubin DB (1992). "Inference from iterative simulation using multiple sequences (with discussion)." *Statistical Science*, **7**(4), 457–511.

Gelman A, Simpson D, Betancourt M (2017). "The prior can often only be understood in the context of the likelihood." *Entropy*, **19**(10), 555–567. doi:10.3390/e19100555.

Griewank A, Walther A (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Siam.

Gronau QF, Singmann H (2018). *bridgesampling: Bridge Sampling for Marginal Likelihoods and Bayes Factors*. R package version 0.6-0, URL https://CRAN.R-project.org/package=bridgesampling.

Han KT (2012). "Fixing the c Parameter in the Three-Parameter Logistic Model." *Practical Assessment, Research & Evaluation*, **17**(1), 1–24.

Heathcote A, Popiel SJ, Mewhort D (1991). "Analysis of response time distributions: An example using the Stroop task." *Psychological Bulletin*, **109**(2), 340–347. doi:10.1037/0033-2909.109.2.340.

Hijazi RH, Jernigan RW (2009). "Modelling compositional data using Dirichlet regression models." *Journal of Applied Probability & Statistics*, **4**(1), 77–91.

Hoffman MD, Gelman A (2014). "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo." *The Journal of Machine Learning Research*, **15**(1), 1593–1623.

Holland PW, Wainer H (2012). *Differential item functioning*. Routledge.

Kass RE, Raftery AE (1995). "Bayes factors." *Journal of the American Statistical Association*, **90**(430), 773–795.

Komboz B, Zeileis A, Strobl C (2018). "Tree-Based Global Model Tests for Polytomous Rasch Models." *Educational and Psychological Measurement*, **78**(1), 128–166. doi:10.1177/0013164416664394.

Lewandowski D, Kurowicka D, Joe H (2009). "Generating Random Correlation Matrices Based on Vines and Extended Onion Method." *Journal of Multivariate Analysis*, **100**(9), 1989–2001. doi:10.1016/j.jmva.2009.04.008.

Lord FM (2012). *Applications of item response theory to practical testing problems*. Routledge.

Mair P, Hatzinger R (2007). "Extended Rasch modeling: The eRm package for the application of IRT models in R." *Journal of Statistical Software*, **20**(9), 1–20. doi:10.18637/jss.v020.i09.

McElreath R (2017). *rethinking: Statistical Rethinking Course and Book Package*. R package version 1.59, URL https://github.com/rmcelreath/rethinking.

Meng XL, Schilling S (2002). "Warp bridge sampling." *Journal of Computational and Graphical Statistics*, **11**(3), 552–586. doi:10.1198/106186002457.

Meng XL, Wong WH (1996). "Simulating ratios of normalizing constants via a simple identity: a theoretical exploration." *Statistica Sinica*, **6**(4), 831–860.

Merkle EC, Rosseel Y (2015). "blavaan: Bayesian structural equation models via parameter expansion." *arXiv preprint arXiv:1511.05604*.

Millsap RE, Everson HT (1993). "Methodology review: Statistical approaches for assessing measurement bias." *Applied psychological measurement*, **17**(4), 297–334. doi:10.1177/014662169301700401.

Molenaar D, Tuerlinckx F, van der Maas HL, *et al.* (2015). "Fitting diffusion item response theory models for responses and response times using the R package diffIRT." *Journal of Statistical Software*, **66**(4), 1–34. doi:10.18637/jss.v066.i04.

OECD (2017). "PISA 2015: Technical Report." URL http://www.oecd.org/pisa/data/2015-technical-report/.

Piironen J, Vehtari A, *et al.* (2017). "Sparsity information and regularization in the horseshoe and other shrinkage priors." *Electronic Journal of Statistics*, **11**(2), 5018–5051. doi:10.1214/17-EJS1337SI.

Plummer M (2013). ***JAGS**: Just Another Gibs Sampler*. URL http://mcmc-jags.sourceforge.net/.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rasch G (1960). *Studies in Mathematical Psychology: I. Probabilistic models for some intelligence and attainment tests.* Nielsen & Lydiche.

Rasch G (1961). "On general laws and the meaning of measurement in psychology." In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, volume 4, pp. 321–333. University of California Press Berkeley, CA.

Ratcliff R (1978). "A theory of memory retrieval." *Psychological review*, **85**(2), 59–108. doi:10.1037/0033-295X.85.2.59.

Rigby RA, Stasinopoulos DM (2005). "Generalized Additive Models for Location, Scale and Shape." *Journal of the Royal Statistical Society C (Applied Statistics)*, **54**(3), 507–554.

Rizopoulos D (2006). "ltm: An R package for Latent Variable Modelling and Item Response Theory Analyses." *Journal of Statistical Software*, **17**(5), 1–25. doi:10.18637/jss.v017.i05.

Robitzsch A (2019). *sirt: Supplementary item response theory models*. R package version 3.3-26, URL https://CRAN.R-project.org/package=sirt.

Robitzsch A, Kiefer T, Wu M (2019). *TAM: Test analysis modules*. R package version 3.1-45, URL https://CRAN.R-project.org/package=TAM.

Rosseel Y (2012). "lavaan: An R Package for Structural Equation Modeling." *Journal of Statistical Software*, **48**(2), 1–36. doi:10.18637/jss.v048.i02.

Samejima F (1997). "Graded Response Model." In *Handbook of Modern Item Response Theory*, pp. 85–100. Springer-Verlag.

Shmueli G, Minka TP, Kadane JB, Borle S, Boatwright P (2005). "A useful distribution for fitting discrete data: revival of the Conway–Maxwell–Poisson distribution." *Journal of the Royal Statistical Society C (Applied Statistics)*, **54**(1), 127–142. doi:10.1111/j.1467-9876.2005.00474.x.

Spielberger CD (2010). "State-Trait anger expression inventory." *The Corsini Encyclopedia of Psychology*.

Stan Development Team (2019). *Stan Modeling Language: User's Guide and Reference Manual*. URL http://mc-stan.org/manual.html.

Strobl C, Kopf J, Zeileis A (2015). "Rasch Trees: A New Method for Detecting Differential Item Functioning in the Rasch Model." *Psychometrika*, **80**(2), 289–316. doi:10.1007/s11336-013-9388-3.

Tuerlinckx F, De Boeck P (2005). "Two interpretations of the discrimination parameter." *Psychometrika*, **70**(4), 629–650. doi:10.1007/s11336-000-0810-3.

van der Linden WJ, Hambleton RK (2013). *Handbook of modern item response theory*. Springer-Verlag.

van der Linden WJ, Ren H (2015). "Optimal Bayesian adaptive design for test-item calibration." *Psychometrika*, **80**(2), 263–288.

van der Maas HL, Molenaar D, Maris G, Kievit RA, Borsboom D (2011). "Cognitive psychology meets psychometric theory: On the relation between process models for decision making and latent variable models for individual differences." *Psychological review*, **118**(2), 339–356. doi:10.1037/a0022749.

Vehtari A, Gelman A, Gabry J (2017a). "Pareto smoothed importance sampling." *arXiv preprint*. URL https://arxiv.org/abs/1507.02646.

Vehtari A, Gelman A, Gabry J (2017b). "Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC." *Statistics and Computing*, **27**(5), 1413–1432. doi:10.1007/s11222-016-9696-4.

Vehtari A, Gelman A, Simpson D, Carpenter B, Bürkner PC (2019). "Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of MCMC." *arXiv preprint*. URL https://arxiv.org/abs/1903.08008.

Wagenmakers EJ, Brown S (2007). "On the linear relation between the mean and the standard deviation of a response time distribution." *Psychological Review*, **114**(3), 830–841. doi:10.1037/0033-295X.114.3.830.

Wickham H, Grolemund G (2016). *R for data science: import, tidy, transform, visualize, and model data*. O'Reilly Media, Inc.

Zeileis A, Strobl C, Wickelmaier F, Komboz B, Kopf J (2018). *psychotools: Infrastructure for Psychometric Modeling*. R package version 0.5-0, URL https://CRAN.R-project.org/package=psychotools.

Affiliation:

Paul-Christian Bürkner
Aalto University, Department of Computer Science
Konemiehentie 2, 02150 Espoo, Finland
E-mail: paul.buerkner@gmail.com
URL: https://paul-buerkner.github.io/

	Package							
Feature	eRm	ltm	TAM	mirt	sirt	(b)lavaan	lme4	brms
1-PLM	yes	yes	yes	yes	yes	yes	yes	yes
2-PLM	no	yes	yes	yes	yes	yes	no	yes
3-PLM	no	yes	yes	yes	yes	no	no	yes
4-PLM	no	no	no	yes	yes	no	no	yes
PCM	yes	yes	yes	yes	yes	no	no	yes
GRM	no	yes	no	yes	yes	yes	no	yes
CM	no	no	yes	yes	no	no	no	yes
LM	no	no	no	no	yes	yes	yes	yes
CoM	no	no	no	no	no	no	yes	yes
RTM	no	no	no	no	no	limited	limited	yes
PrM	no	no	no	no	no	no	no	yes
Multidimensional	no	no	yes	yes	yes	yes	yes	yes
Covariates	yes	yes	yes	yes	yes	yes	yes	yes
Constraints	no	yes	yes	yes	yes	yes	limited	limited
Latent classes	no	no	yes	yes	yes	no	no	no
Mixtures	no	no	yes	yes	yes	no	no	yes
Copulas	no	no	limited	no	limited	no	no	no
Splines	no	no	no	yes	yes	no	no	yes
Multilevel	no	no	no	yes	limited	limited	yes	yes
Joint models	no	no	no	yes	no	yes	no	yes
Imputation	no	no	yes	yes	yes	no	no	yes
Customizable	no	no	no	yes	yes	no	no	yes
Estimator	CML	MML	MML,JML	MML	various	various	MML	AHMC

Table 3: Overview of modeling options in IRT supporting packages. Abbreviations: x-PLM = x-parameter logistic models; PCM = partial credits models; GRM = graded response models; CM = categorical models; LM = linear models; CoM = count data models; RTM = response times models; PrM = Proportion models (i.e., Beta and Dirichlet models); CML = conditional maximum likelihood; MML = marginal maximum likelihood; JML = joint maximum likelihood; WLS = weighted least squres; AHMC = adaptive Hamiltonian Monte-Carlo.