# METABETA
# A FAST NEURAL MODEL FOR BAYESIAN MIXED-EFFECTS REGRESSION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Hierarchical data with multiple observations per group is ubiquitous in empirical sciences and is often analyzed using mixed-effects regression. In such models, Bayesian inference gives an estimate of uncertainty but is analytically intractable and requires costly approximation using Markov Chain Monte Carlo (MCMC) methods. Neural posterior estimation shifts the bulk of computation from inference time to pre-training time, amortizing over simulated datasets with known ground truth targets. We propose `metabeta`, a transformer-based neural network model for Bayesian mixed-effects regression. Using simulated and real data, we show that it reaches stable and comparable performance to MCMC-based parameter estimation at a fraction of the usually required time.

## 1 INTRODUCTION

Much of the data we work with has a hierarchical structure that naturally clusters into subgroups. When predicting the efficacy of a drug, for example, there may be subpopulation-specific effects in addition to effects on the population as a whole. When building a movie recommendation system, some films may be universally popular, yet individual preferences still matter. When looking at plant growth, the same fertilizer may perform well in one field but poorly in another due to local conditions. These challenges can be addressed using mixed-effects models, which provide a principled framework for capturing both overall trends (fixed effects) and group-specific deviations (random effects). Mixed-effects models have been widely adopted across disciplines – including ecology, psychology, and education – and are by now considered a standard approach for analyzing hierarchical data (Gelman & Hill, 2007; Harrison et al., 2018; Gordon, 2019; Yu et al., 2022).

In many such applications, we would like to estimate the parameters of a mixed-effects model in a Bayesian manner, enabling the incorporation of prior knowledge and the explicit quantification of uncertainty (Figueroa-Zúñiga et al., 2013; Gelman et al., 2013). However, closed-form solutions are generally unavailable even for the simplest cases, necessitating computationally expensive approximate inference methods such as Markov Chain Monte Carlo (MCMC, Metropolis et al., 1953). From a practitioner's perspective, this is undesirable as MCMC often entails prohibitively long inference times, even for moderately sized datasets.

In this work, we introduce `metabeta`, a probabilistic transformer-based neural network model, that is designed to efficiently approximate Bayesian inference for mixed-effects regression. It is trained via neural posterior estimation (Rezende & Mohamed, 2015; Gordon et al., 2018; Wildberger et al., 2023; Hollmann et al., 2025), amortizing computation costs over many simulated hierarchical datasets with available ground truth parameters. We demonstrate that `metabeta` achieves accuracy comparable to Hamiltonian Monte Carlo (HMC), which is the gold-standard MCMC method for Bayesian mixed-effects regression (Neal, 2011; Betancourt, 2018; Bürkner, 2018; Capretto et al., 2022). Importantly, our model reduces inference time by orders of magnitude, thereby greatly broadening the range of feasible applications for Bayesian mixed-effects regression. To further facilitate `metabeta`'s adoption for rapid deployment and plug-and-play compatibility, we provide open-source Python code for our implementation and plan to release a package with pretrained models that integrates seamlessly with existing analysis pipelines (Bürkner, 2018; Abril-Pla et al., 2023).

### 1.1 Related Work

Many methods for neural posterior estimation (NPE) have been proposed in recent years: `TabPFN` (Müller et al., 2021; Hollmann et al., 2025) is a transformer-based model that efficiently estimates a one-dimensional histogram-like posterior over outcomes $\mathbf{y}$. Its training data is simulated using random graphs, granting it intricate dependence-structure and excellent generalization for non-linear probabilistic prediction. Rezende & Mohamed (2015) pioneered the use of conditional normalizing flows (Papamakarios et al., 2021; Kobyzev et al., 2021) for NPE. Together with Gordon et al. (2018), they laid the groundwork for `BayesFlow`, a seminal framework combining transformer-based models and normalizing flows (Radev et al., 2020; 2023). In this framework, model parameters are sampled from a fixed prior, observations are sampled from a likelihood, and a neural network learns to map the observations to a joint posterior over model parameters. `BayesFlow` has been extended to hierarchical Bayesian models with separate networks for each level of hierarchy (Habermann et al., 2024), and to non-linear mixed-effects models optimized for cell biology and pharmacology (Arruda et al., 2023). In both cases, the priors are fixed and a new model has to be trained if a different prior is desired. This off-loads the amortization process to potential end-users, which at best nullifies the runtime advantage of NPE for practical purposes.

Our contribution consists of three aspects: (1) Our model is trained on simulations with varying data ranges and varying parameter priors, explicitly incorporating prior information into posterior estimation; (2) it deploys post-hoc refinements of posterior means and credible intervals using importance sampling (Tokdar & Kass, 2010) and conformal prediction (Vovk et al., 2022); (3) we aim to release a trained version of our model for data practitioners.

## 2 Methods

We briefly formalize mixed-effects regression (Section 2.1) and define a synthetic distribution over hierarchical datasets representative of scenarios practitioners care about (Section 2.2). We then present a neural network architecture that takes an entire dataset and priors as inputs and returns posterior distributions over all regression parameters (Section 2.3). This model is trained on synthetic datasets with available ground truth to perform accurate posterior inference (Section 2.4). In a final post-training step, we refine the model's outputs using importance sampling and conformal prediction (Section 2.5). All our code is implemented in `PyTorch` 2.7.1 (Paszke et al., 2019) and openly available at `https://github.com/user/censored-for-review`.

### 2.1 Mixed-Effects Regression

Mixed-effects regression extends traditional regression by explicitly accounting for within-group dependency in hierarchical data (Gelman & Hill, 2007; Brown, 2021; Fahrmeir et al., 2013). To model this dependency, mixed-effects regression distinguishes between two effect types:

- **Fixed effects** $\boldsymbol{\beta} \in \mathbb{R}^d$ capture the general, group-independent relation between predictor variables $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$ and the regression output variable $\mathbf{y}_i \in \mathbb{R}^{n_i}$.
- **Random effects** $\boldsymbol{\alpha}_i \in \mathbb{R}^q$ capture additional, group-specific variations for $q \leq d$ predictors. For each group $i = 1, \ldots, m$, we treat $\boldsymbol{\alpha}_i$ as samples from $\mathcal{N}_q(\mathbf{0}, \mathbf{S})$.

This yields the model:

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\boldsymbol{\alpha}_i + \boldsymbol{\varepsilon}_i \,, \tag{1}$$

with independent additive noise $\boldsymbol{\varepsilon}_i \sim \mathcal{N}_{n_i}(\mathbf{0}, \sigma_\varepsilon^2 \mathbf{I}_{n_i})$. The random effect predictor matrix $\mathbf{Z}_i$ is typically a submatrix of $\mathbf{X}_i$. Note, that the $n_i$ observations are *conditionally independent* given some fixed $\boldsymbol{\alpha}_i$ but *marginally dependent* over $\boldsymbol{\alpha}_i$:

$$\mathbf{y}_i|\boldsymbol{\alpha}_i \sim \mathcal{N}_{n_i}(\mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\boldsymbol{\alpha}_i,\ \sigma_\varepsilon^2\mathbf{I}_{n_i}) \quad \Longrightarrow \quad \mathbf{y}_i \sim \mathcal{N}_{n_i}(\mathbf{X}_i\boldsymbol{\beta},\ \mathbf{Z}_i\mathbf{S}\mathbf{Z}_i^\top + \sigma_\varepsilon^2\mathbf{I}_{n_i}).$$

The goal of Bayesian mixed-effects modeling is to obtain posteriors for all unobserved global $(\boldsymbol{\beta}, \sigma_\varepsilon^2, \mathbf{S})$ and local $(\boldsymbol{\alpha}_i)$ regression parameters, conditioned on the observed predictors, outcomes and priors of the global parameters.

### 2.2 Data Simulation and Preprocessing

To train our neural posterior estimator, we simulate hierarchically structured datasets using `PyTorch` as shown in Figure 1A.

*Priors*: For each dataset, we sample multi-dimensional priors – that is, for $q$ random effect variances $\sigma_i$ there are $q$ half-normal priors $\mathcal{HN}(\tau^2_{\sigma_i})$.

*Regression parameters*: (1) $q + 1$ variance parameters are sampled from half-normal distributions, the first $q$ being the random effect variances, and the last one the noise variance. (2) Then, $m \times q$ random effect vectors are sampled from a diagonal Gaussian, and $d$ fixed effects are sampled from another diagonal Gaussian. (3) Independent noise is sampled from a half-normal distribution.

*Observations*: The predictors $\mathbf{x}_{ij}$ are sampled independently from several distributions – including normal, Student-$t$, continuous uniform, Bernoulli, negative binomial, and scaled Beta distributions – with randomly chosen parameters. A correlation matrix is sampled from a Lewandowski-Kurowicka-Joe distribution and its lower triangular is multiplied with the predictors to induce correlation structure (Lewandowski et al., 2009). The random effects predictors are set to $\mathbf{z}_{ij} = \mathbf{x}_{ij}$ for $j \leq q$ and 0 otherwise. Predictors and parameters are passed through equation 1 and noise is added to generate outcomes $\mathbf{y_i}$ for each group $i$.

Further details on the simulation procedure can be found in Appendix A. We simulate separate mixed-effects datasets for training, validation, and testing. For the latter, we use `PyMC` (Abril-Pla et al., 2023) to estimate all posteriors with HMC (with 4 chains, 1000 tuning iterations, and 1000 draws). For the HMC model specification, we provide the true priors and the generative model used for simulation. We occasionally encountered divergence and strong outliers for some HMC chains, which may strongly affect performance statistics. For a fair comparison, we choose the chain with the fewest outliers identified by the median absolute deviation statistic (MAD, Hampel, 1974; Leys et al., 2013) for any given dataset.

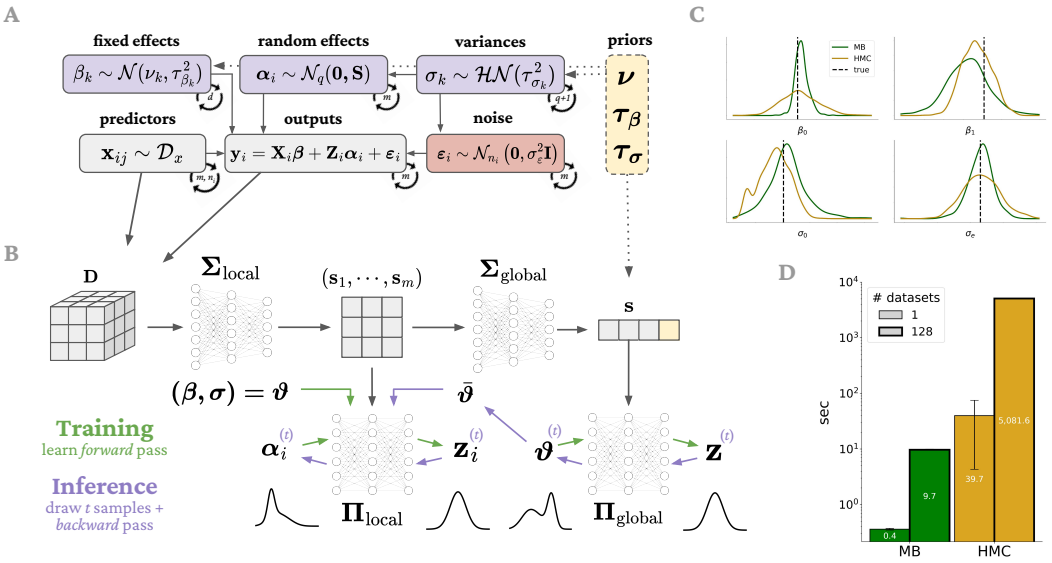


**Figure 1:** (A) *Dataset Simulation*. Given a set of priors, we sample regression parameters and noise in a cascading way. Predictors are sampled from various distributions for training and from real datasets for testing, and outcomes are generated according to equation 1. (B) *Model Pipeline*. Observed data are summarized locally (per group) and globally (across groups). During training, the posterior networks learn the forward mapping from the true regression parameters to a simple multivariate base distribution, conditioned on the respective summaries and priors. During inference, we draw *k* samples from the base distribution, and apply the implicitly learned backward mapping to them, approximating sampling from the unknown target posterior. (C) *Example Posteriors.* Kernel density estimates from the posterior samples of `metabeta` (MB) and Hamiltonian Monte Carlo (HMC) on a toy dataset. (D) *Compute Time.* For test sets with $d = 5$, $q = 1$, $m \leq 30$ and $n_i \leq 70$, our model takes several orders of magnitude less time to compute in comparison to HMC. Computation time was measured on a MacBook Air M2 with 24GB of RAM.

### 2.3 Model Architecture

The model architecture takes inspiration from `BayesFlow` (Radev et al., 2020; Habermann et al., 2024) and `TabPFN` (Hollmann et al., 2025) and has two main parts: (1) a *summary network* that computes a maximally informative dataset statistic over observations, and (2) a *posterior network* that uses the summary and priors to propose a joint posterior over regression parameters. Both are trained end-to-end. Since mixed-effect datasets are hierarchically structured, we use two summary and posterior networks, one for the global parameters (fixed effects and variance parameters) and one for the local parameters (group-specific random effects). The training and inference pipeline is visualized in Figure 1B. Data preprocessing is detailed in Appendix B and Appendix C.

#### Summary Network

Datasets vary in the number of groups and observations per group. A summary network $f_\Sigma$ extracts information for the posterior by pooling over all instances in a dataset. Since the data is structured hierarchically, it needs to be summarized accordingly over all exchangeable instances: In a first step, we pool over observations per group, generating $m$ local summaries $\mathbf{s}_1, \ldots, \mathbf{s}_m$. In a second step, we pool the local summaries over groups, generating a global summary $\mathbf{s}$. For summarization, we opted for a set transformer (Lee et al., 2019). Our implementation consists of multiple transformer encoder blocks (Vaswani et al., 2017), followed by averaging over the resulting sequence of transformer outputs. This yields the important property of permutation invariance, i.e. the summary stays the same regardless of the input ordering along the sequence dimension. The local and global summary network both consist of 4 transformer encoder blocks with 128 units, equally large feedforward layers, 8 attention heads, 1% dropout and GELU activations (Hendrycks & Gimpel, 2023).

#### Posterior Network

Posterior networks $f_\Pi$ take the dataset summaries and priors as inputs and propose a joint posterior for a set of parameters. Inference on global and local parameters is separated as proposed by Heinrich et al. (2023). Inference on global parameters $\boldsymbol{\vartheta} = (\boldsymbol{\beta}, \mathbf{S}, \sigma_\varepsilon^2)$ is conditioned on the global summary and the parameter priors. Inference on local variables $(\boldsymbol{\alpha}_i)$ is conditioned on the separate local summaries and the global parameters (the true ones during training, and the inferred ones during validation). We opted for a normalizing flow (Papamakarios et al., 2021) as our posterior network:

A normalizing flow learns an invertible mapping from a $d$-dimensional random variable $\mathbf{z}_n$ with a complex distribution to a $d$-dimensional random variable $\mathbf{z_0}$ with a regular distribution (e.g. a multivariate normal). The flow consists of a finite composition $T$ of continuously differentiable and invertible transforms $T_i$ with triangular Jacobians, $T = T_n \circ \cdots \circ T_1$. For some random variable $\mathbf{z}_0 \sim \mathcal{N}_d(\mathbf{0}, \mathbf{I})$, we model $T(\mathbf{z}_n) = \mathbf{z}_0 \iff T^{-1}(\mathbf{z}_0) = \mathbf{z}_n$ with $p_n(\mathbf{z}_n) = p_0(\mathbf{z}_0)\ |\det J_T(\mathbf{z}_0)|$. Each invertible transform $T_i$ is parameterized by a neural network that takes part of the current hidden state $\mathbf{z}_t$ and the summary $\mathbf{s}$ as inputs. Because of their efficiency, we opted for conditional affine coupling as our normalizing flow architecture (Dinh et al., 2014; 2017). For the base distribution we use a diagonal multivariate location-scale *t* distribution with learnable parameters for each dimension (Alexanderson & Henter, 2020). For both posterior networks, we use 4 affine coupling flow blocks parameterized by MLPs with three 128-unit feedforward layers, skip connections (He et al., 2016), 1% dropout and ReLU activations.

### 2.4 Learning

To calculate the loss for the global parameters, we use the forward Kullback-Leibler divergence between the unknown true posterior $p(\boldsymbol{\vartheta}|\mathbf{s})$ and its flow-based approximation $\hat{p}(\boldsymbol{\vartheta}|\mathbf{s}) := p_n(\mathbf{z}_n|\mathbf{s})$,

$$\ell_\Pi(\boldsymbol{\vartheta}, \mathbf{s}) \propto -\mathbb{E}_{\boldsymbol{\vartheta},\mathbf{s}}\left[\log \hat{p}(\boldsymbol{\vartheta}|\mathbf{s})\right] = -\mathbb{E}_{\boldsymbol{\vartheta},\mathbf{s}}\left[\log p_0(T(\boldsymbol{\vartheta}|\mathbf{s})) + \log|\det J_T(\boldsymbol{\vartheta}|\mathbf{s})|\right],$$

where $T$ and thereby the approximation $\hat{p}(\boldsymbol{\vartheta}|\mathbf{s})$ depend on the posterior network. Since the summary $\mathbf{s}$ of data $\mathbf{D}$ is itself depending on the summary network, the end-to-end loss can be written as

$$\ell_{\Pi,\Sigma}(\boldsymbol{\vartheta}, \mathbf{D}) \propto -\mathbb{E}_{\boldsymbol{\vartheta},\mathbf{D}}\left[\log \hat{p}(\boldsymbol{\vartheta}|f_\Sigma(\mathbf{D}))\right] = -\mathbb{E}_{\boldsymbol{\vartheta},\mathbf{D}}\left[\log p_0(T(\boldsymbol{\vartheta}|f_\Sigma(\mathbf{D})) + \log|\det J_T(\boldsymbol{\vartheta}|f_\Sigma(\mathbf{D}))|\right].$$

The objective for the local parameters is completely analogous. We average the local losses over groups and add the result to the global loss. The expectation is approximated by averaging over the

batch. Model weights are updated using the Schedule-Free AdamW optimizer (Defazio et al., 2024). We train separate models for different numbers of fixed effects and random effects until convergence, which requires between $10^5$ and $10^6$ training sets in our case.

### 2.5 Post-Hoc Refinement

#### Importance Sampling

Learning $\hat{p}$ by minimizing the forward Kullback-Leibler Divergence naturally forces $\hat{p}$ to be positive wherever $p$ is positive, making $\hat{p}$ mass-covering (Jerfel et al., 2021). Thus, we can use importance sampling to improve posterior estimation (Tokdar & Kass, 2010; Dax et al., 2023). For each sample $\boldsymbol{\vartheta}_k \sim \hat{p}(\boldsymbol{\vartheta}|\mathbf{D})$ we assign an importance weight,

$$w_k = \frac{p(\mathbf{D}|\boldsymbol{\vartheta}_k)p(\boldsymbol{\vartheta}_k)}{\hat{p}(\boldsymbol{\vartheta}_k|\mathbf{D})},$$

which is well-defined, as $\hat{p}$ is only zero if the numerator is zero. We use the weights to refine statistics of the samples (e.g. the posterior mean or empirical CDFs). Since we have two posterior networks and the local posterior is conditioned on the global estimates, we perform alternating importance sampling for both. For more details, please see Appendix D.

#### Calibration with Conformal Prediction

Uncertainty quantification is a hallmark of Bayesian inference, making the fidelity of the approximate posterior's credible intervals a critical concern. Posterior samples can be used to calculate empirical quantiles and thus also intervals that contain $c\%$ of the posterior density. Due to the mass-covering property of $\hat{p}$, the learned posteriors tend to be too wide – i.e. the true parameter is inside the $c\%$ credible interval in more than $c\%$ of the cases. This is a commonly known issue of normalizing flows (Chen et al., 2025; Dheur & Taieb, 2025). Conformal prediction (Vovk et al., 2022; Shafer & Vovk, 2008; Angelopoulos & Bates, 2022) is a general-purpose method that constructs distribution-free prediction sets $\hat{C}_\alpha$ such that $\mathbb{P}(\boldsymbol{\vartheta} \in \hat{C}_\alpha) \geq 1 - \alpha$. To construct $\hat{C}_\alpha$, we use a calibration set to calculate the difference between the true $\boldsymbol{\vartheta}$ and the closest border of the proposed $1 - \alpha$ credible interval $C_\alpha$. The $1 - \alpha$ quantile of these differences is then added to the proposed interval borders, widening them if the value is positive and narrowing them otherwise. Importantly, this does not require retraining but efficiently refines credible intervals post-hoc.

## 3 Results

We test our model against HMC on a toy dataset with highly constrained parameters and uncorrelated normal data (Section 3.1), in-distribution on held-out synthetic test sets of varying difficulty (Section 3.2), and out-of-distribution on semi-synthetic data where predictors $\mathbf{X}$ are taken from four real datasets (Section 3.3). This approach has the following considerable benefits over evaluation on purely real data: (1) The regression models are always correctly specified, (2) we know the ground truth parameters and can thus evaluate parameter recovery and coverage, (3) we can compare the results to in-distribution test data and gauge how well the model transfers to realistic predictors (Lueckmann et al., 2021; Ward et al., 2022). We use the following evaluation metrics: We quantify *parameter recovery* with Pearson's correlation $r$ and $\mathrm{RMSE}$ between the true parameters and posterior means. We check the fidelity of credible intervals using coverage errors: A $\mathrm{CE}(\alpha) = 0.05$ means that the model's $1 - \alpha$ credible interval $C_\alpha$ is on average 5% too wide. Coverage error are estimated using the difference between the relative frequency of the true parameter being inside $C_\alpha$ and the credible interval's nominal probability mass $(1 - \alpha)$:

$$\mathrm{CE}(\alpha) = \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left(\vartheta^{(b)} \in C_\alpha^{(b)}\right) - (1 - \alpha).$$

Finally, we plot posterior predictive distributions (Gelman et al., 2013), which visualize how much the posterior predictive samples $\tilde{\mathbf{y}}_\mathbf{t} \sim p(\mathbf{y}|\hat{\boldsymbol{\vartheta}}_\mathbf{t})$ match the actual $\mathbf{y}$. All metrics are calculated for both `metabeta` and HMC posterior samples. A brief comparison of required computation time is reported in Figure 1D.

### 3.1 TOY EXAMPLE

To gauge if the pipeline works for both our model and HMC, we first test both on a toy example with $d = 2$ and $q = 1$. Regression parameters and observed data all fall inside small ranges, and the observed single predictor is sampled from a standard normal distribution. Result figures can be found in Appendix E. Both models reach almost perfect parameter recovery correlation for fixed effects ($r_{\texttt{MB}} = 1.000$ vs. $r_{\texttt{HMC}} = 0.996$), random effects ($r_{\texttt{MB}} = 0.987$ vs. $r_{\texttt{HMC}} = 0.974$), and for variance parameters ($r_{\texttt{MB}} = 0.997$ vs. $r_{\texttt{HMC}} = 0.996$). The same pattern arises for recovery error for fixed effects ($\text{RMSE}_{\texttt{MB}} = 0.020$ vs. $\text{RMSE}_{\texttt{HMC}} = 0.197$), random effects ($\text{RMSE}_{\texttt{MB}} = 0.097$ vs. $\text{RMSE}_{\texttt{HMC}} = 0.136$), and for variance parameters ($\text{RMSE}_{\texttt{MB}} = 0.027$ vs. $\text{RMSE}_{\texttt{HMC}} = 0.029$). Posterior coverage is good for `metabeta` ($\text{CE}_{\texttt{MB}} = 0.021$), whereas HMC's marginal posterior for the variance parameters tend to be slightly too wide ($\text{CE}_{\texttt{HMC}} = 0.146$). Example kernel density estimates of the posteriors for a single regression dataset are plotted in Figure 1C: While the modes often agree, the posterior shapes vary between our model and HMC. Overall, both models perform well on the toy problem with a slight advantage for `metabeta`. This shows that the pipeline is in principle correctly specified for both approaches.

### 3.2 IN-DISTRIBUTION TESTS

We ran a sweep of tests to check how well our model can handle different mixed-effect datasets in terms of the problem size (determined by $d$ and $q$), numbers of total observations ($n = \sum_{i=1}^{m} n_i$) and signal to noise ratio, $\text{SNR} = \mathbb{V}(\mathbf{y} - \boldsymbol{\varepsilon})/\mathbb{V}(\boldsymbol{\varepsilon})$. The test sets were constructed using the same simulation pipeline as the training sets, but with different random seeds. Table 1 shows that our model's performance is most strongly affected by the problem size in terms of RMSE. Both relatively low signal to noise ratio and low $n$ affect `metabeta`'s performance slightly, but none is systematically more impactful than the other. However, its Pearson correlations and coverage are relatively robust to both problem size and dataset properties. HMC, on the other hand, tends to produce more outliers for larger $n$ and SNR. Accordingly, its coverage varies more strongly over datasets. Overall, our model appears to have comparably stable performance and outperforms HMC in most test cases.

**Table 1:** Performance evaluation for `metabeta` and HMC on synthetic test sets. Test set properties are number of fixed effects $d$, number of random effects $q$, relative sample size $n$, and signal to noise ratio SNR. The symbols ◯ resp. ∘ indicate top resp. bottom 50% of the test set, sorted by either $n$ or SNR. The evaluation metrics are Pearson's correlation-coefficient $r$, root mean squared error RMSE, and coverage error $\text{CE}(\alpha)$ averaged over $\alpha \in \{0.05, 0.1, 0.2, 0.32, 0.5\}$. All metrics are averaged across regression parameters. Bold formatting indicates better performance.

| | | | | metabeta | | | HMC | | |
|---|---|---|---|---|---|---|---|---|---|
| $d$ | $q$ | property | split | $r$ | RMSE | CE | $r$ | RMSE | CE |
| 3 | 1 | $n$ | ◯ | **0.993** | **0.508** | **0.010** | 0.799 | 19.879 | 0.057 |
| | | | ∘ | **0.995** | **0.546** | -0.071 | 0.991 | 1.475 | **0.012** |
| | | SNR | ◯ | **0.997** | **0.277** | 0.044 | 0.796 | 20.389 | **0.022** |
| | | | ∘ | 0.991 | 0.689 | **-0.021** | **0.998** | **0.679** | 0.129 |
| 5 | 2 | $n$ | ◯ | **0.994** | **0.502** | **0.015** | 0.828 | 10.703 | 0.024 |
| | | | ∘ | **0.968** | **1.064** | -0.085 | 0.840 | 5.458 | **-0.077** |
| | | SNR | ◯ | **0.973** | **0.794** | **0.000** | 0.794 | 11.879 | -0.056 |
| | | | ∘ | **0.988** | **0.729** | **0.009** | 0.946 | 1.899 | 0.090 |
| 8 | 3 | $n$ | ◯ | **0.968** | **1.234** | 0.013 | 0.897 | 3.096 | **0.001** |
| | | | ∘ | **0.975** | **1.212** | -0.078 | 0.899 | 2.962 | **-0.053** |
| | | SNR | ◯ | **0.955** | **1.094** | **0.009** | 0.836 | 4.058 | -0.066 |
| | | | ∘ | **0.966** | **1.309** | **0.020** | 0.956 | 1.516 | 0.086 |
| 12 | 5 | $n$ | ◯ | **0.972** | **1.339** | **0.008** | 0.887 | 37.148 | -0.028 |
| | | | ∘ | **0.961** | 1.699 | -0.082 | 0.946 | **1.662** | **-0.025** |
| | | SNR | ◯ | **0.945** | **1.405** | **-0.006** | 0.885 | 37.040 | -0.054 |
| | | | ∘ | **0.970** | **1.587** | **0.014** | 0.928 | 3.993 | 0.067 |

**Table 2:** Performance evaluation for `metabeta` and HMC on semi-synthetic test sets using real predictors $\mathbf{x_{ij}}$ and simulated regression parameters. The evaluation metrics are Pearson's correlation-coefficient $r$, root mean squared error RMSE, and coverage error $\mathrm{CE}(\alpha)$ averaged over $\alpha \in \{0.05, 0.1, 0.2, 0.32, 0.5\}$, separately per type of parameter. Bold formatting indicates better performance.

| | | metabeta | | | HMC | | |
|---|---|---|---|---|---|---|---|
| Source | Parameters | $r$ | RMSE | CE | $r$ | RMSE | CE |
| `MathAchieve` | $\boldsymbol{\beta}$ | **0.999** | **0.754** | **0.023** | 0.992 | 1.966 | 0.030 |
| | $\boldsymbol{\sigma}$ | **0.996** | **0.316** | **0.010** | 0.980 | 0.778 | 0.164 |
| | $\boldsymbol{\alpha}$ | **0.987** | **0.832** | **0.041** | 0.956 | 1.518 | 0.060 |
| `Exam` | $\boldsymbol{\beta}$ | **0.999** | **0.554** | 0.057 | 0.995 | 1.548 | **0.054** |
| | $\boldsymbol{\sigma}$ | 0.994 | **0.407** | **0.009** | **0.996** | 0.219 | 0.183 |
| | $\boldsymbol{\alpha}$ | **0.986** | **0.786** | **0.031** | 0.972 | 1.108 | 0.099 |
| `Gscmv` | $\boldsymbol{\beta}$ | **0.999** | **0.596** | **0.007** | 0.881 | 6.634 | -0.020 |
| | $\boldsymbol{\sigma}$ | **0.988** | **0.639** | **-0.009** | 0.477 | 30.083 | 0.071 |
| | $\boldsymbol{\alpha}$ | **0.964** | **1.430** | 0.016 | 0.889 | 2.434 | **-0.009** |
| `SleepStudy` | $\boldsymbol{\beta}$ | **1.000** | **0.092** | **0.147** | 0.996 | 1.292 | 0.194 |
| | $\boldsymbol{\sigma}$ | **0.996** | **0.310** | **0.050** | 0.687 | 9.709 | 0.169 |
| | $\boldsymbol{\alpha}$ | **0.975** | **1.065** | **0.024** | 0.972 | 1.166 | 0.115 |

### 3.3 Out-of-Distribution Tests

We gathered 4 canonical datasets that are often used for demonstration purposes of mixed-effects regression: (1) `MathAchieve` ($d = 5, q = 1$), available in `nlme` (Pinheiro et al., 1999), (2) `Exam` ($d = 4, q = 1$) and (3) `Gcsmv` ($d = 3, q = 1$), both available in `mlmRev` (Bates & Bolker, 2020), and (4) `sleepstudy` ($d = 2, q = 2$), available in `lme4` (Bates et al., 2015). All contain numerical and categorial predictors, have varying numbers of groups and observations per group, as well as different ranges for $\mathbf{X}$ and $\mathbf{y}$. For each, we built a semi-synthetic test set using real predictors $\mathbf{X}$, simulated regression parameters and resulting outcomes $\mathbf{y}$. Table 2 lists model performance for all datasets for each type of parameter. Our model has the clear advantage: In $11/12$ cases our model has the better correlation with the true parameters, in $12/12$ cases it has the lower RMSE and in $10/12$ cases it has the better coverage. HMC particularly struggles with the variance parameters, which do not pose a substantial problem for `metabeta`. Our model again has considerably fewer outliers (Figure 2A) and better coverage (Figure 2B), which is reflected in its credible intervals (Figure 3A). Both models produce appropriate posterior predictive distributions (Figure 3B). Overall, this suggests that the simulated predictors $\mathbf{X}$, on which our model is trained, approximate the structure of realistic datasets well. This makes our model useful for analyzing the type of datasets practitioners care about.

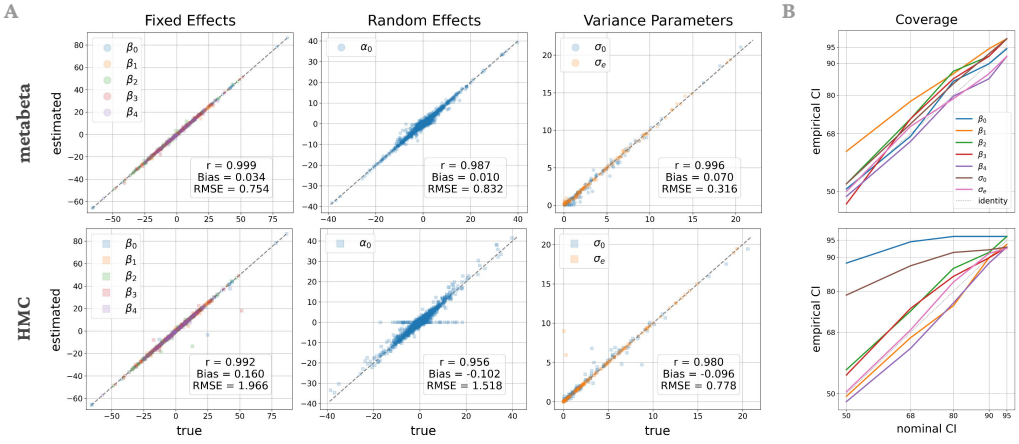


**Figure 2:** Results based on `MathAchieve`. Remaining results are depicted in Appendix E. (A) *Parameter Recovery.* Our model outperforms HMC on average in terms of r, bias and RMSE for all parameter types, and has fewer outliers. (B) *Coverage.* Our model's posterior credible intervals are on average more faithfully tuned.

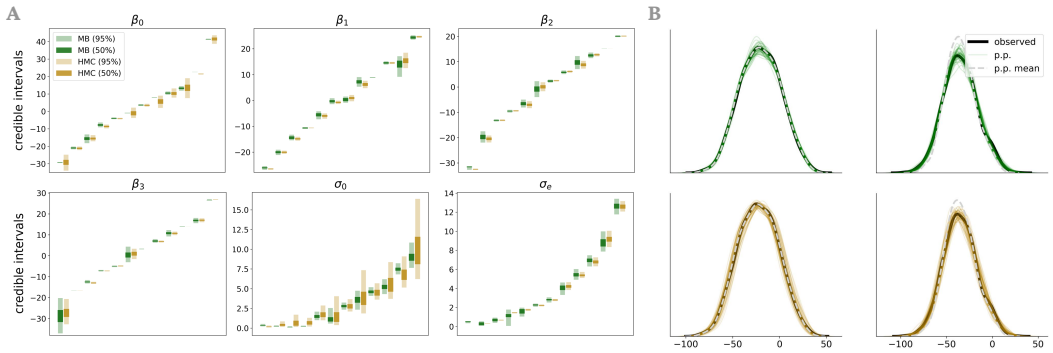


**Figure 3:** Results based on `MathAchieve` (A) *Credible Intervals*. 95% and 50% credible intervals for `metabeta` and HMC, compared over different parameter values. Note that discrepancies in width are mirrored in the coverage plot of Figure 2B: HMC has poorer coverage for $\beta_0$ and $\sigma_0$ and its credible intervals are on average wider than `metabeta`'s for both parameters. The plot for $\beta_4$ is omitted due to space constraints. (B) *Posterior Predictive*. Observed regression outputs (black) plotted against samples from the posterior predictive (colored) and its mean (grey) for both models. Curves based on kernel density estimates over data points, separately for two randomly chosen datasets.

## 4 Discussion

In this paper we present `metabeta`, a probabilistic transformer-based model that performs efficient approximate Bayesian inference for mixed-effects regression. We trained `metabeta` on simulated datasets with varying ranges for predictors, regression parameters, and outcomes. Most importantly, these datasets incorporate varying priors and we condition the model outputs on them, which not only amortizes the high computational costs encountered when using MCMC for parameter estimation, but also generalizes previous neural posterior estimation (NPE) techniques that are trained on a fixed prior. We show that our model has favorable and robust performance on in-distribution and out-of-distribution test sets, based on real hierarchical datasets practitioners care about. In each experiment, we compare the results of our model with Hamiltonian Monte Carlo (HMC), the gold-standard MCMC method for Bayesian mixed-effects regression, and show that `metabeta` often outperforms HMC in terms of accuracy, stability, and fidelity of credible intervals – all at a fraction of the time required for parameter estimation with HMC.

The high speed and explicit incorporation of priors opens new avenues for Bayesian mixed-effects regression: Analysts can now specify multiple priors simultaneously and check how robust the model posteriors are to varying a priori assumptions. Furthermore, it is straightforward to extend our model to a mixture of experts by passing the same dataset multiple times with different permutations of the design matrix columns, and then aggregating the resulting back-permuted posterior samples (Hollmann et al., 2025).

### 4.1 Limitations and Outlook

Each trained version of `metabeta` is currently tailored to the size of the regression problem in terms of the number of fixed effects ($d$) and the number of random effects ($q$). This means there are potentially as many model versions as combinations of $d$ and $q$. A single model with an upper bound to $d$ and $q$ is possible with our setup, but will never perform as well as a model that is specialized for a given problem size. Since a single snapshot of the weights requires about 20mb of storage, pulling each model from an online server on-demand seems to be a manageable alternative to a single model with on average worse performance.

Our choice of model architecture trades of posterior expressivity for computation speed: Other normalizing flow methods like Neural Spline Flows (Durkan et al., 2019), Flow Matching (Wildberger et al., 2023) or TarFlow (Zhai et al., 2025) offer more flexible posterior shapes, but posterior sampling is considerably more expensive than for Affine Coupling Flows. The relative simplicity of affine coupling posteriors can be seen as implicit regularization, preventing overly irregular quantification of regression parameter uncertainty.

A noteworthy outlook for `metabeta` is the incorporation of attention over predictors, in addition to the already present attention over samples (Müller et al., 2021). While this may introduce significant overhead for the summary network, it promises to handle dependencies between predictors in a more explicit and robust manner.

Finally, NPE models generally suffer when test data are strongly out of distribution in comparison to training data. While there are ways to amend this (Ward et al., 2022), a future iteration of `metabeta` will be trained directly on realistic hierarchical datasets generated by an LLM (Borisov et al., 2022; Wang et al., 2024; Jagadish et al., 2024; 2025).

### 4.2 Conclusion

Our model brings Bayesian mixed-effects regression closer to practical usability in real-world applications. In its current form, it already enables rapid prototyping – practitioners can quickly test different model specifications and validate findings using conventional tools if needed. Our analyses highlight that `metabeta` is immediately applicable to such use cases. Looking ahead, we envision scaling our model to larger regression problems, incorporating attention over predictors and training on more realistic, LLM-generated hierarchical data. This would open the door to entirely new applications of Bayesian mixed-effects regression that are currently out of reach.

## Reproducibility Statement

We include source code for our model, weight snapshots, and instructions to run data generation, training and testing procedures in the supplementary materials. Test datasets including MCMC samples are also contained there. Training data are omitted due to space constraints but can be quickly generated with the provided code.

## REFERENCES


Oriol Abril-Pla, Virgile Andreani, Colin Carroll, Larry Dong, Christopher J. Fonnesbeck, Maxim Kochurov, Ravin Kumar, Junpeng Lao, Christian C. Luhmann, Osvaldo A. Martin, Michael Osthege, Ricardo Vieira, Thomas Wiecki, and Robert Zinkov. PyMC: A modern, and comprehensive probabilistic programming framework in Python. *PeerJ Computer Science*, 9:e1516, September 2023. ISSN 2376-5992. doi: 10.7717/peerj-cs.1516.

Simon Alexanderson and Gustav Eje Henter. Robust model training and generalisation with Studentising flows, July 2020.

Anastasios N. Angelopoulos and Stephen Bates. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification, December 2022.

Jonas Arruda, Yannik Schälte, Clemens Peiter, Olga Teplytska, Ulrich Jaehde, and Jan Hasenauer. An amortized approach to non-linear mixed-effects modeling based on neural posterior estimation, August 2023.

Douglas Bates and Martin Maechler and Ben Bolker. mlmRev: Examples from Multilevel Modelling Software Review, April 2020.

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting Linear Mixed-Effects Models Using **lme4**. *Journal of Statistical Software*, 67(1), 2015. ISSN 1548-7660. doi: 10.18637/jss.v067.i01.

Michael Betancourt. A Conceptual Introduction to Hamiltonian Monte Carlo, July 2018.

Vadim Borisov, Kathrin Sessler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language Models are Realistic Tabular Data Generators. In *The Eleventh International Conference on Learning Representations*, September 2022.

Violet A. Brown. An Introduction to Linear Mixed-Effects Modeling in R. *Advances in Methods and Practices in Psychological Science*, 4(1):2515245920960351, January 2021. ISSN 2515-2459, 2515-2467. doi: 10.1177/2515245920960351.

Paul-Christian Bürkner. Advanced Bayesian Multilevel Modeling with the R Package brms. *The R Journal*, 10(1):395, 2018. ISSN 2073-4859. doi: 10.32614/RJ-2018-017.

Tomás Capretto, Camen Piho, Ravin Kumar, Jacob Westfall, Tal Yarkoni, and Osvaldo A. Martin. Bambi: A Simple Interface for Fitting Bayesian Linear Models in Python. *Journal of Statistical Software*, 103:1–29, August 2022. ISSN 1548-7660. doi: 10.18637/jss.v103.i15.

Tianyu Chen, Vansh Bansal, and James G. Scott. Conditional diffusions for amortized neural posterior estimation. In *The 28th International Conference on Artificial Intelligence and Statistics*, February 2025.

Maximilian Dax, Stephen R. Green, Jonathan Gair, Michael Pürrer, Jonas Wildberger, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Neural Importance Sampling for Rapid and Reliable Gravitational-Wave Inference. *Physical Review Letters*, 130(17):171403, April 2023. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.130.171403.

Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The Road Less Scheduled. *Advances in Neural Information Processing Systems*, 37: 9974–10007, December 2024.

Victor Dheur and Souhaib Ben Taieb. Multivariate Latent Recalibration for Conditional Normalizing Flows, May 2025.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *arXiv: Learning*, October 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations*, February 2017.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural Spline Flows, December 2019.

Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian Marx. *Regression: Models, Methods and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-34332-2 978-3-642-34333-9. doi: 10.1007/978-3-642-34333-9.

Jorge I. Figueroa-Zúñiga, Reinaldo B. Arellano-Valle, and Silvia L. P. Ferrari. Mixed beta regression: A Bayesian perspective. *Computational Statistics & Data Analysis*, 61:137–147, May 2013. ISSN 0167-9473. doi: 10.1016/j.csda.2012.12.002.

Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge Univ. Press, Cambridge, 23rd printing edition, 2007. ISBN 978-0-521-68689-1 978-0-521-86706-1.

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Texts in Statistical Science Series. CRC Press, Taylor & Francis Group, Boca Raton London New York, third edition edition, 2013. ISBN 978-1-4398-4095-5.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-Learning Probabilistic Inference for Prediction. In *International Conference on Learning Representations*, September 2018.

Katherine R. Gordon. How Mixed-Effects Modeling Can Advance Our Understanding of Learning and Memory and Improve Clinical and Educational Practice. *Journal of Speech, Language, and Hearing Research : JSLHR*, 62(3):507–524, March 2019. ISSN 1092-4388. doi: 10.1044/2018_JSLHR-L-ASTM-18-0240.

Daniel Habermann, Marvin Schmitt, Lars Kühmichel, Andreas Bulling, Stefan T. Radev, and Paul-Christian Bürkner. Amortized Bayesian Multilevel Models, August 2024.

Frank R. Hampel. The Influence Curve and its Role in Robust Estimation. *Journal of the American Statistical Association*, 69(346):383–393, June 1974. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.1974.10482962.

Xavier A. Harrison, Lynda Donaldson, Maria Eugenia Correa-Cano, Julian Evans, David N. Fisher, Cecily E. D. Goodwin, Beth S. Robinson, David J. Hodgson, and Richard Inger. A brief introduction to mixed effects modelling and multi-model inference in ecology. *PeerJ*, 6:e4794, May 2018. ISSN 2167-8359. doi: 10.7717/peerj.4794.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Lukas Heinrich, Siddharth Mishra-Sharma, Chris Pollard, and Philipp Windischhofer. Hierarchical Neural Simulation-Based Inference Over Event Ensembles. *Transactions on Machine Learning Research*, October 2023. ISSN 2835-8856.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs), June 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, January 2025. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-024-08328-6.

Akshay K Jagadish, Julian Coda-Forno, Mirko Thalmann, Eric Schulz, and Marcel Binz. Human-like category learning by injecting ecological priors from large language models into neural networks. *arXiv preprint arXiv:2402.01821*, 2024.

Akshay K Jagadish, Mirko Thalmann, Julian Coda-Forno, Marcel Binz, and Eric Schulz. Meta-learning ecological priors from large language models explains human learning and decision making. *arXiv preprint arXiv:2509.00116*, 2025.

Ghassen Jerfel, Serena Wang, Clara Wong-Fannjiang, Katherine A. Heller, Yian Ma, and Michael I. Jordan. Variational refinement for importance sampling using the forward Kullback-Leibler divergence. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 1819–1829. PMLR, December 2021.

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43 (11):3964–3979, November 2021. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2020.2992934.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 3744–3753. PMLR, May 2019.

Daniel Lewandowski, Dorota Kurowicka, and Harry Joe. Generating random correlation matrices based on vines and extended onion method. *Journal of Multivariate Analysis*, 100(9):1989–2001, October 2009. ISSN 0047259X. doi: 10.1016/j.jmva.2009.04.008.

Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, July 2013. ISSN 00221031. doi: 10.1016/j.jesp.2013.03.013.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking Simulation-Based Inference. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pp. 343–351. PMLR, March 2021.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953. ISSN 0021-9606. doi: 10.1063/1.1699114.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers Can Do Bayesian Inference. In *International Conference on Learning Representations*, October 2021.

Radford M. Neal. *MCMC Using Hamiltonian Dynamics*. May 2011. doi: 10.1201/b10905.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019.

José Pinheiro, Douglas Bates, and R Core Team. Nlme: Linear and Nonlinear Mixed Effects Models, November 1999.

Stefan T. Radev, Ulf K. Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Kothe. BayesFlow: Learning Complex Stochastic Models With Invertible Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1452–1466, 2020. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2020.3042395.

Stefan T. Radev, Marvin Schmitt, Lukas Schumacher, Lasse Elsemüller, Valentin Pratz, Yannik Schälte, Ullrich Köthe, and Paul-Christian Bürkner. BayesFlow: Amortized Bayesian Workflows With Neural Networks, July 2023.

Danilo Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1530–1538. PMLR, June 2015.

Glenn Shafer and Vladimir Vovk. A Tutorial on Conformal Prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008. ISSN 1533-7928.

Surya T. Tokdar and Robert E. Kass. Importance sampling: A review. *WIREs Computational Statistics*, 2(1):54–60, January 2010. ISSN 1939-5108, 1939-0068. doi: 10.1002/wics.56.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer International Publishing, Cham, 2022. ISBN 978-3-031-06648-1 978-3-031-06649-8. doi: 10.1007/978-3-031-06649-8.

Yuxin Wang, Duanyu Feng, Yongfu Dai, Zhengyu Chen, Jimin Huang, Sophia Ananiadou, Qianqian Xie, and Hao Wang. HARMONIC: Harnessing LLMs for Tabular Data Synthesis and Privacy Protection. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024.

Daniel Ward, Patrick Cannon, Mark Beaumont, Matteo Fasiolo, and Sebastian Schmon. Robust Neural Posterior Estimation and Statistical Model Criticism. *Advances in Neural Information Processing Systems*, 35:33845–33859, December 2022.

Jonas Bernhard Wildberger, Maximilian Dax, Simon Buchholz, Stephen R. Green, Jakob H. Macke, and Bernhard Schölkopf. Flow Matching for Scalable Simulation-Based Inference. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023.

Zhaoxia Yu, Michele Guindani, Steven F. Grieco, Lujia Chen, Todd C. Holmes, and Xiangmin Xu. Beyond t test and ANOVA: Applications of mixed-effects models for more rigorous statistical analysis in neuroscience research. *Neuron*, 110(1):21–35, January 2022. ISSN 1097-4199. doi: 10.1016/j.neuron.2021.10.030.

Shuangfei Zhai, Ruixiang Zhang, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Angel Bautista, Navdeep Jaitly, and Josh Susskind. Normalizing Flows are Capable Generative Models, June 2025.

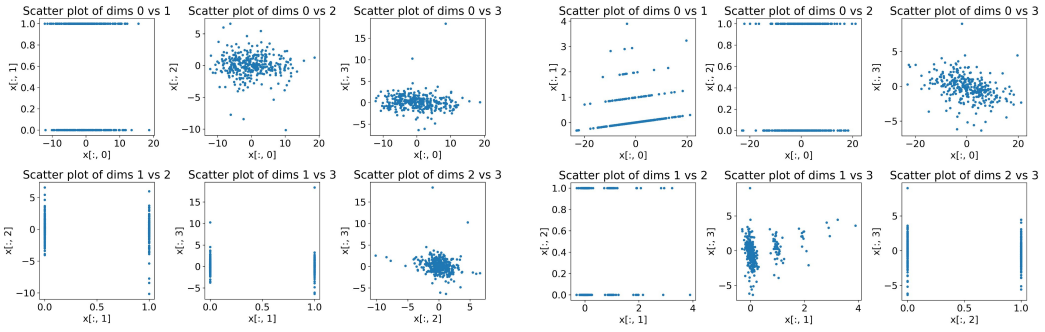


**Figure 4:** Scatter plots of sampled synthetic predictors for two datasets.

# A Dataset Simulation

## A.1 Predictors

We sample $n_i$ observations of predictor $j$, $\mathbf{x_{ij}}$, from the following distributions: Normal, StudentT, Uniform, Bernoulli, NegativeBinomial, ScaledBeta; with respective probability $(0.10, 0.40, 0.05, 0.25, 0.10, 0.10)$, making student $t$ distributed predictors the most likely ones. For each dataset, the parameters of these distributions are randomly sampled such that the resulting outcome $\mathbf{y}$ has a bounded variance (i.e. does not explode). Correlation is induced by sampling $\mathbf{L}\mathbf{L}^\top = \mathbf{R} \sim \text{LKJ}(10)$ and multiplying $\mathbf{L}$ with the design matrix $\mathbf{X}$. For binary variables $z$, we induce correlation with another variable $v$ using the following code snippet:

```
z = torch.randn_like(v)
z = r * v + (1 - r**2)**0.5 * z
probs = torch.sigmoid(z)
z = torch.bernoulli(probs)
```

An example of generated training data is visualized in Figure 8.

## A.2 Priors

Priors for parameters are sampled using the following code snippet:

```
nu_ffx = D.Uniform(-20, 20).sample((batch_size, cfg.max_d))
tau_ffx0 = D.Uniform(0.1, 30).sample((batch_size, 1))
tau_ffx1 = D.Uniform(0.1, 20).sample((batch_size, cfg.max_d-1))
tau_ffx = torch.cat([tau_ffx0, tau_ffx1], dim=-1)
tau_rfx = D.Uniform(0.1, 10).sample((batch_size, cfg.max_d))
tau_eps = D.Uniform(1e-3, 10).sample((batch_size, ))
```

For the toy example, $\nu$ is kept 0, $\tau_\beta$ is bounded below 5, and $\tau_\sigma$ is bounded below 1.

# B Standardization

Before entering the neural model, all observable data is normalized to zero mean and unit standard deviation over groups and observations. To keep the dependence structure intact, we also analytically standardize the regression parameters during training and un-standardize them after sampling, using the following equalities:

$$\beta_k^* = \beta_k \frac{\sigma_{x_k}}{\sigma_y}$$

$$\alpha_{ik}^* = \alpha_k \frac{\sigma_{z_k}}{\sigma_y} \sim \mathcal{N}\left(0, \sigma_k^{*2} = \sigma_k^2 \frac{\sigma_{z_k}^2}{\sigma_y^2}\right)$$

$$\sigma_\varepsilon^{*2} = \frac{\sigma_\varepsilon^2}{\sigma_y^2}$$

where $\sigma_{x_k}$ resp. $\sigma_y$ are the $k$th predictor's resp. the outcome's standard deviation, and $\beta_k^*$ is the $k$th slope after z-standardizing predictors and outcomes. The intercepts require special care:

$$\beta_0^* = \frac{\beta_0 + \sum_{k=1}^{d} \mu_{x_k}\beta_k - \mu_y}{\sigma_y}$$

$$\alpha_{i0}^* = \frac{\alpha_{i0} + \sum_{k=1}^{q} \mu_{z_k}\alpha_{ik}}{\sigma_y} = \frac{\sum_{k=0}^{q} \mu_{z_k}\alpha_{ik}}{\sigma_y} \sim \mathcal{N}\left(0, \sigma_{i0}^{*2}\right),$$

where $\mu_{x_k}$ is the mean of the $k$th predictor over all observations. Due to the sum term in the latter,

$$\sigma_{i0}^{*2} = \mathbb{V}(\alpha_{i0}) + \mathbb{V}(\sum_{k=1}^{q} \mu_{z_k}\alpha_{ik}) + 2 \cdot \mathrm{Cov}(\alpha_{i0}, \sum_{k=1}^{q} \mu_{z_k}\alpha_{ik}).$$

While training is greatly sped up using standardized predictors, un-standardization of intercept parameters may suffer from error propagation and large differences in $\sigma_y$ can cause numerical instability.

## C Data Representation and Embedding

Group-membership is represented implicitly by a separate tensor dimension, e.g. $\mathbf{X}$ has the shape $(batch, m, n, d)$. For `PyTorch` dataloader compatiblity, all tensors are zero-padded and corresponding masks are stored. To spread the learning signal evenly across the network, all slope-related variables are randomly permuted separately per regression dataset, using the same permutation for $\mathbf{X}, \mathbf{Z}, \boldsymbol{\beta}, \boldsymbol{b}_i$, and $\mathbf{S}$.

Observable data is concatenated along the last dimension to $\mathbf{D} = [\mathbf{y}, \mathbf{X}, \mathbf{Z}]$, and linearly projected to a higher-dimensional space (e.g. 128 dimensions). Since mixed-effects regression must be permutation invariant (wrt. to groups and observations per group), no positional encoding or explicit group identity information is passed as input, and instead group identity is represented implicitly by a separate tensor dimension, e.g. $\mathbf{X}$ has the shape $(batch, m, n, d)$.

## D Alternating Importance Sampling

For numerical stability, we compute

1. $\log w_i \leftarrow \log p(\mathbf{D}|\boldsymbol{\vartheta}_i) + \log p(\boldsymbol{\vartheta}_i) - \log q(\boldsymbol{\vartheta}_i|\mathbf{D})$
2. $\log w_i \leftarrow \min(\log w_i, \log w^\dagger)$, where $\log w^\dagger$ is the 98th percentile over $i$
3. $w_i \leftarrow \exp(\log w_i - \max_j \log w_j)$, such that $w_i \leq 1$ for all $i$
4. $\tilde{w}_i \leftarrow \frac{w_i}{\frac{1}{s}\sum_{i=1}^{s} w_i}$ such that $\sum_{i=1}^{s} \tilde{w}_i = s$.

Since we have two approximate posteriors (one for the global parameters, one for the random effects), we have two sets of samples which require separate importance weights (IW). For the global parameters posterior, the numerator can either use the *marginal* likelihood,

$$p(\mathbf{D}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta}) = \prod_{i=1}^{m} p\big(\mathbf{y}_i|\boldsymbol{\beta}, \mathbf{S}, \sigma_\varepsilon^2\big)p(\boldsymbol{\beta})p(\mathbf{S})p\big(\sigma_\varepsilon^2\big),$$

or the *conditional* likelihood,

$$p(\mathbf{D}|\boldsymbol{\vartheta})p(\boldsymbol{\vartheta}) = \prod_{i=1}^{m} p\big(\mathbf{y}_i|\mathbf{b}_i, \boldsymbol{\beta}, \sigma_\varepsilon^2\big)p(\mathbf{b}_i|\mathbf{S})p(\mathbf{S})p(\boldsymbol{\beta})p\big(\sigma_\varepsilon^2\big).$$

The marginal likelihood is more appropriate, because the global posterior does not receive any explicit information about the random effects, i.e. it is not conditioned on them.

However, calculating the marginal likelihood is inefficient, as it requires a matrix inversion for each sample. Empirically, parameters recovery also suffers from using marginal likelihood IW. Instead, we plug in the posterior mean of the random effects for the conditional likelihood IW. The IW for the random effects posterior is calculated accordingly, this time using the importance-weighted means of the global parameters. We alternate the two steps 3 times, starting with the local samples.

# E Result Figures

## E.1 Toy Example

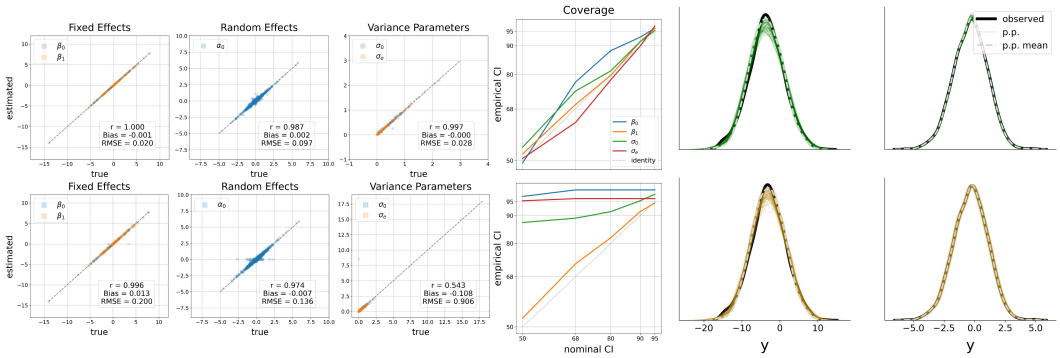


**Figure 5:** Results based on the toy example. Descriptors are the same as in Figure 2 and Figure 3B.

## E.2 Exam

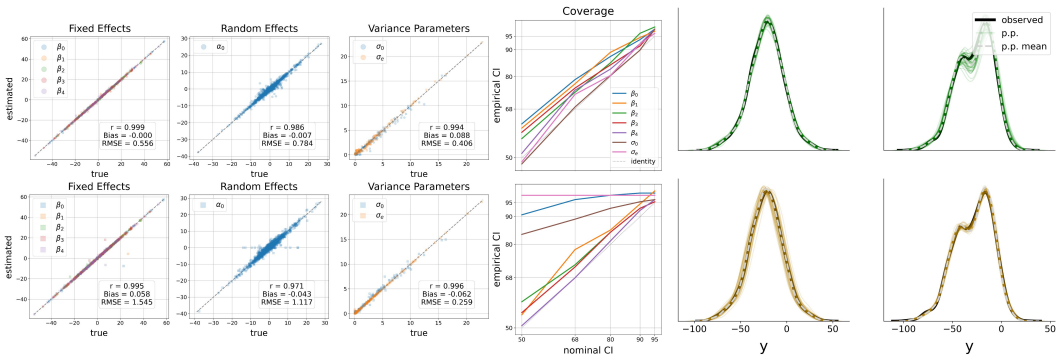


**Figure 6:** Results based on `Exam`. Descriptors are the same as in Figure 2 and Figure 3B.

### E.3 GCSEMV

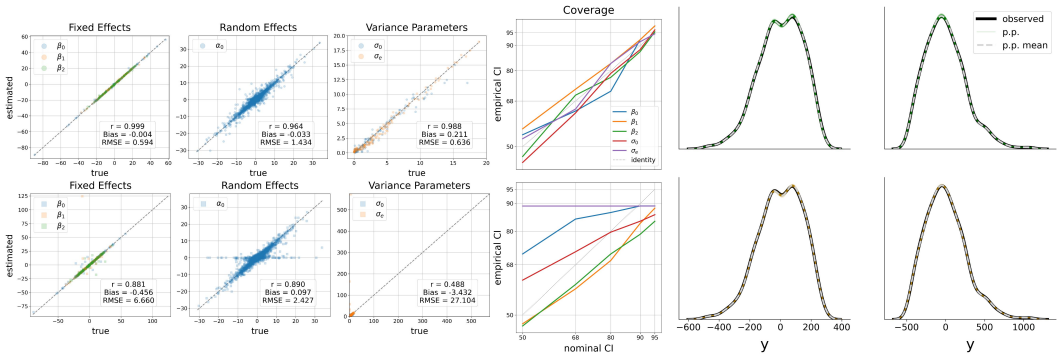


**Figure 7:** Results based on `Gcsemv`. Descriptors are the same as in Figure 2 and Figure 3B.

### E.4 SLEEPSTUDY

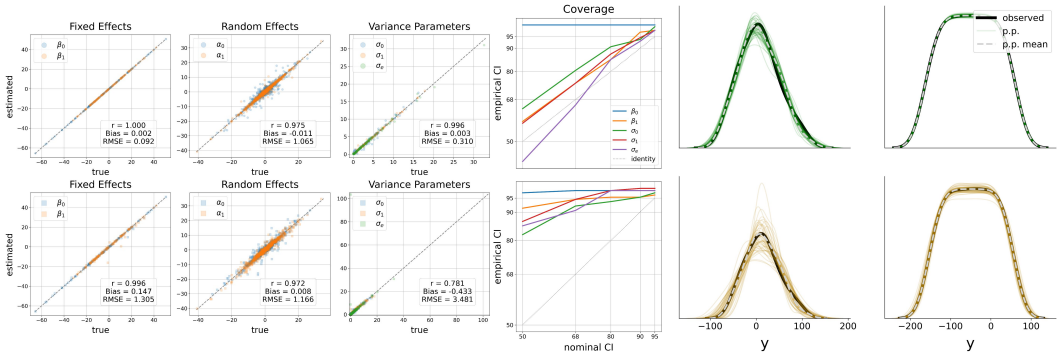


**Figure 8:** Results based on `Sleepstudy`. Descriptors are the same as in Figure 2 and Figure 3B.