Supplementary Materials for "Re-Evaluating Machine Learning for MRP Given the Comparable Performance of (Deep) Hierarchical Models"

A Models and Inference

This appendix provides information on the model specification noted in the main text as well as brief derivations of the new technical contributions and their importance to efficient estimation.

The main paper presents an example with an intercept and two random effects. It is often convenient to represent the model in a "general design" notation (Zhao et al. 2006). Using the results in Goplerud (2022), I present a model with an arbitrary number of random effects using both the general design (Equation A.1) and Gelman and Hill (2006) notation (Equation A.2). In both cases, I use conditionally conjugate Inverse-Wishart priors on the variance components $\boldsymbol{\Sigma}_j$, a flat prior on the fixed effects $\boldsymbol{\beta}$, and the conventional normal prior on the random effects. $\boldsymbol{z}^b_{i,j}$ represents the covariates for individual i for random effect j, e.g. $\boldsymbol{z}^b_{i,j} = 1$ for a random intercept. Each random effect $j \in \{1, \cdots, J\}$ has g_j levels (e.g., 50 states) and a dimensionality of d_j (e.g., $d_j = 1$ for a random intercept). Thus, $\boldsymbol{\Sigma}_j$ is a $d_j \times d_j$ symmetric matrix. Each observation i distributed as a binomial random variable with n_i trials and y_i successes; in this paper, $n_i = 1$ as the outcome is binary.

$$\begin{aligned}
&y_i|\boldsymbol{\beta}, \boldsymbol{\alpha} \sim \text{Binom}(n_i, p_i), \quad p_i = \frac{\exp(\psi_i)}{1+\exp(\psi_i)}, \quad \psi_i = \boldsymbol{x}_i^T\boldsymbol{\beta} + \boldsymbol{z}_i^T\boldsymbol{\alpha} \\
&\boldsymbol{\alpha}_j|\boldsymbol{\Sigma}_j \sim N\left(\boldsymbol{0}, \boldsymbol{I}_{g_j} \otimes \boldsymbol{\Sigma}_j\right), \quad \boldsymbol{\Sigma}_j \sim \text{IW}(\nu_j, \boldsymbol{\Phi}_j), \quad p(\boldsymbol{\beta}) \propto 1 \\
&\boldsymbol{z}_{i,j} = \boldsymbol{m}_{i,j} \otimes \boldsymbol{z}^b_{i,j}, \quad \boldsymbol{\alpha}^T = [\boldsymbol{\alpha}_1^T, \cdots, \boldsymbol{\alpha}_J^T], \quad \boldsymbol{z}_i^T = [\boldsymbol{z}_{i,1}^T, \cdots, \boldsymbol{z}_{i,J}^T]
\end{aligned} \tag{A.1}$$

$$\begin{aligned}
y_i|\boldsymbol{\beta}, \{\{\boldsymbol{\alpha}_{j,g}\}_{g=1}^{g_j}\}_{j=1}^{J} \sim \text{Binom}(n_i, p_i), \quad p_i = \frac{\exp(\psi_i)}{1+\exp(\psi_i)}, \quad \psi_i = \boldsymbol{x}_i^T\boldsymbol{\beta} + \sum_{j=1}^{J}\left[\boldsymbol{z}^b_{i,j}\right]^T\boldsymbol{\alpha}_{j,g[i]} \\
\boldsymbol{\alpha}_{j,g}|\boldsymbol{\Sigma}_j \sim N(\boldsymbol{0}_{d_j}, \boldsymbol{\Sigma}_j), \quad \boldsymbol{\Sigma}_j \sim \text{IW}(\nu_j, \boldsymbol{\Phi}_j) \quad \forall(j,g), \quad p(\boldsymbol{\beta}) \propto 1
\end{aligned} \tag{A.2}$$

As posed, this problem is difficult to estimate even with variational techniques. A key move in Goplerud (2022) is to use Polya-Gamma data augmentation (Polson, Scott and Windle 2013) to augment the posterior with one Polya-Gamma variable ω_i for each observation (diagonally stacked into $\boldsymbol{\Omega}$). Polson, Scott and Windle (2013) note the following result for Polya-Gamma variables where a Polya-Gamma variable ω_i has two parameters b, c and the identity below holds for any positive a and b.

$$\frac{\exp(\psi)^a}{[1+\exp(\psi)]^b} = 2^{-b}\int \exp(s\psi - \psi^2/2\omega) f_{PG}(\omega|b,0)d\omega, \quad s = a - b/2 \tag{A.3a}$$

$$\omega \sim PG(b,c) \coloneqq \omega = \frac{1}{2\pi^2}\sum_{k=1}^{\infty}\frac{Z_k}{(k-1/2)^2 + c^2/(4\pi^2)}, \quad Z_k \overset{i.i.d.}{\sim} \text{Gamma}(b,1) \tag{A.3b}$$

After augmenting the likelihood for each observation with its corresponding Polya-Gamma variable, the relevant portion of the augmented posterior is shown below.

$$p(\boldsymbol{y}, \boldsymbol{\Omega}|\boldsymbol{\alpha}, \boldsymbol{\beta}) \propto \exp\left(\boldsymbol{s}^T[\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{\alpha}] - \frac{1}{2}\left[\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{\alpha}\right]^T \boldsymbol{\Omega}\left[\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{\alpha}\right]\right)\prod_{i=1}^{N} f_{PG}(\omega_i|n_i,0) \tag{A.4}$$

This is amenable to a Gibbs Sampler for all coefficients, including the Polya-Gamma random variables. Goplerud (2022) notes that this also means it is amenable to closed form mean-field variational inference with no further assumptions nor integration required. This result holds for any number and configuration of random effects, slopes, etc. I repeat the relevant result and algorithm (Result 1 and Algorithm 1) below. Please see that paper for full details and extensive simulations on this algorithm.

Result A.1 (Existence of CAVI (Scheme I from Goplerud 2022)). *Consider the factorization assumption:*

Scheme I: "Strong Factorization" — $\mathcal{X}_1 = q(\boldsymbol{\beta})\prod_{j=1}^{J} q(\boldsymbol{\alpha}_j)q(\boldsymbol{\Sigma})q(\boldsymbol{\Omega})$

For the model in Equation A.1 and for each choice of $\mathcal{X}_k$ above, each step of the CAVI algorithm can be implemented exactly in closed form, with no additional assumptions. For each $\mathcal{X}_k$, the optimal approximation for $q(\boldsymbol{\beta}, \boldsymbol{\alpha})$ is multivariate normal, $q(\boldsymbol{\Sigma})$ is the product of J independent Inverse-Wishart densities, and $q(\boldsymbol{\Omega})$ is the product of N independent Polya-Gammas.

This paper uses the following algorithm (Algorithm 1 from Goplerud 2022) for inference on the above model. Note that it immediately allows any of the traditional deep MRP models (i.e., adding many additional random intercepts and/or slopes) to be fit easily.

The remainder of this Appendix outlines the various technical extensions to Goplerud (2022) in this paper: the Huang-Wand prior, SQUAREM, Parameter-Expansion, and splines.

A.1 Huang-Wand Prior

The choice of prior for a hierarchical model is a non-trivial task. Existing non-Bayesian software typically employs a flat improper prior (e.g., the Laplace approximation in `lme4`; Bates et al. 2015), although this has been shown to have some theoretical and empirical problems (e.g., Gelman 2006; Chung et al. 2015). A popular choice in the case of a single variance parameter is the half-t prior popularized by Gelman (2006) that has a tractable form. Huang and Wand (2013) generalize this by allowing it govern multivariate variance

Algorithm A.1 CAVI from Goplerud (2022)

Set Priors of Inverse-Wishart: $\{\nu_j, \mathbf{\Phi}_j\}_{j=1}^J$; **Set Number of Iterations**: T

Initialize Variational Parameters: $\{\tilde{b}_i, \tilde{c}_i\}_{i=1}^N$ (for Polya-Gamma); $\tilde{\boldsymbol{\mu}}_\beta, \tilde{\mathbf{\Lambda}}_\beta, \tilde{\boldsymbol{\mu}}_\alpha, \tilde{\mathbf{\Lambda}}_\alpha$ (for $\boldsymbol{\beta}, \boldsymbol{\alpha}$); $\{\tilde{\nu}_j, \tilde{\mathbf{\Phi}}_j\}_{j=1}^J$ (for $\mathbf{\Sigma}_j$)

For t in $1, \cdots, T$

1. Update Polya-Gammas - $q\left(\{\omega_i\}_{i=1}^N\right)$: $\tilde{b}_i = n_i, \quad \tilde{c}_i = \sqrt{E_{q(\boldsymbol{\alpha},\boldsymbol{\beta})}\left[(\boldsymbol{x}_i^T\boldsymbol{\beta} + \boldsymbol{z}_i^T\boldsymbol{\alpha})^2\right]}$
2. Update $q(\boldsymbol{\beta}) \sim N(\tilde{\boldsymbol{\mu}}_\beta, \tilde{\mathbf{\Lambda}}_\beta)$:

$$\tilde{\mathbf{\Lambda}}_\beta = \left(\sum_{i=1}^N E_{q(\omega_i)}[\omega_i]\boldsymbol{x}_i\boldsymbol{x}_i^T\right)^{-1}, \quad \tilde{\boldsymbol{\mu}}_\beta = \tilde{\mathbf{\Lambda}}_\beta \boldsymbol{X}^T\left(\sum_{i=1}^N \left(y_i - \frac{n_i}{2}\right) - E_{q(\omega_i)}[\omega_i]\cdot \boldsymbol{z}_i^T E_{q(\boldsymbol{\alpha})}[\boldsymbol{\alpha}]\right)$$

3. Update $q\left(\boldsymbol{\alpha}_j\right) \sim N(\tilde{\boldsymbol{\mu}}_{\alpha,j}, \tilde{\mathbf{\Lambda}}_{j,\alpha})$, where $\boldsymbol{T}_j$ stacks the block diagonal expectation of the precision on the random effects ($\mathbf{\Sigma}_j^{-1}$):

$$\tilde{\mathbf{\Lambda}}_{\alpha,j} = \left(\boldsymbol{T}_j + \sum_{i=1}^N E_{q(\omega_i)}[\omega_i]\boldsymbol{z}_{i,j}\boldsymbol{z}_{i,j}^T\right)^{-1}, \quad \boldsymbol{T}_j = E_{q(\mathbf{\Sigma}_j)}\left[\boldsymbol{I}_{g_j} \otimes \mathbf{\Sigma}_j^{-1}\right]$$

$$\tilde{\boldsymbol{\mu}}_{\alpha,j} = \tilde{\mathbf{\Lambda}}_{\alpha,j}\boldsymbol{Z}_j^T\left[\sum_{i=1}^N \left(y_i - \frac{n_i}{2}\right) - E_{q(\omega_i)}[\omega_i]\cdot\left(\boldsymbol{x}_i^T E_{q(\boldsymbol{\beta})}[\boldsymbol{\beta}] + \sum_{\ell:\{1,\cdots,J\}\setminus j} \boldsymbol{z}_{i,\ell}^T E_{q(\boldsymbol{\alpha}_\ell)}[\boldsymbol{\alpha}_\ell]\right)\right]$$

4. Update $q\left(\{\mathbf{\Sigma}_j\}_{j=1}^J\right)$: $\tilde{\nu}_j = \nu_j + g_j, \quad \tilde{\mathbf{\Phi}}_j = \mathbf{\Phi}_j + \sum_{g=1}^{g_j} E_{q(\boldsymbol{\alpha}_{j,g})}\left[\boldsymbol{\alpha}_{j,g}\boldsymbol{\alpha}_{j,g}^T\right]$
5. Check for convergence, evaluate ELBO (see Goplerud 2022).

parameters (e.g., for a random slope and intercept). Their prior is shown below in both its mixture and marginal formulations (p. 441) where $\mathbf{\Sigma}$ is a positive definite matrix of dimensionality $p \times p$.

$$\begin{aligned}&\mathbf{\Sigma}|\{a_k\}_{k=1}^p \sim \text{InverseWishart}\left(\nu + p - 1, \quad 2\nu \cdot \text{diag}(1/a_1, \cdots, 1/a_p)\right); \\ &a_k \sim \text{InverseGamma}(1/2, 1/A_k^2); \quad \forall k \in \{1, \cdots, p\}\end{aligned} \tag{A.5a}$$

$$p(\mathbf{\Sigma}) \propto |\mathbf{\Sigma}|^{-(\nu+2p)/2} \cdot \prod_{k=1}^p \left[\nu(\mathbf{\Sigma}^{-1})_{kk} + 1/A_k^2\right]^{-(\nu+p)/2} \tag{A.5b}$$

They note some desirable properties; first, the marginal distributions on the standard deviations of $\mathbf{\Sigma}$ have the half-t formulation proposed by Gelman (2006). Further, if $\nu = 2$, then the prior implies a uniform distribution on the correlations between any two components of the prior. Second, for larger A_k, the prior can be made increasingly less informative while still remaining proper. Thus, it provides a way to stabilize the model slightly while not heavily distorting the posterior inferences.

In terms of variational inference, Huang and Wand (2013) show it can be easily put

into a coordinate ascent variational framework. Specifically, if one modifies the factorization assumption in Result A.1 above to assume independence between $\mathbf{\Sigma}_j$ and $\{a_{j,k}\}_{k=1}^{d_j}$, the resulting approximate posterior has an Inverse-Wishart distribution on $\mathbf{\Sigma}_j$ and independent Inverse-Gamma distributions on $\{a_{j,k}\}$ because of the conditional conjugacy. Result A.2 presents the updates for an algorithm with this prior.

Result A.2 (CAVI with Huang-Wand Prior)**.** *Consider the model in Equation A.1 where the prior on $\mathbf{\Sigma}_j$ is replaced with a Huang-Wand prior (Equation A.5) with hyper-parameters ν_j and $A_{j,k}$. The augmented posterior can be expressed as: $p\left(\boldsymbol{\beta}, \boldsymbol{\alpha}, \{\omega_i\}_{i=1}^N, \{\mathbf{\Sigma}_j, \{a_{j,k}\}_{k=1}^{d_j}\}_{j=1}^J | \boldsymbol{y}\right)$. Consider the following proposed factorization:*

$$\mathcal{X}_{\text{HW}} = q(\boldsymbol{\beta}) \prod_{j=1}^{J} \left[\left[q(\boldsymbol{\alpha}_j)) q(\{a_{j,k}\}_{k=1}^{d_j}) \right] q(\mathbf{\Sigma}_j) \right] q(\mathbf{\Omega})$$

Each step of CAVI can be implemented in closed form with no additional assumptions. The approximating distributions on $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$, $\mathbf{\Omega}$, $\mathbf{\Sigma}_j$ have the same distributional form as in Result A.1. The factorization on $q(\{a_{j,k}\})$ is the product of independent Inverse-Gamma densities.

The algorithm can be estimated as follows: Replace Step 4 in Algorithm A.1 as follows:

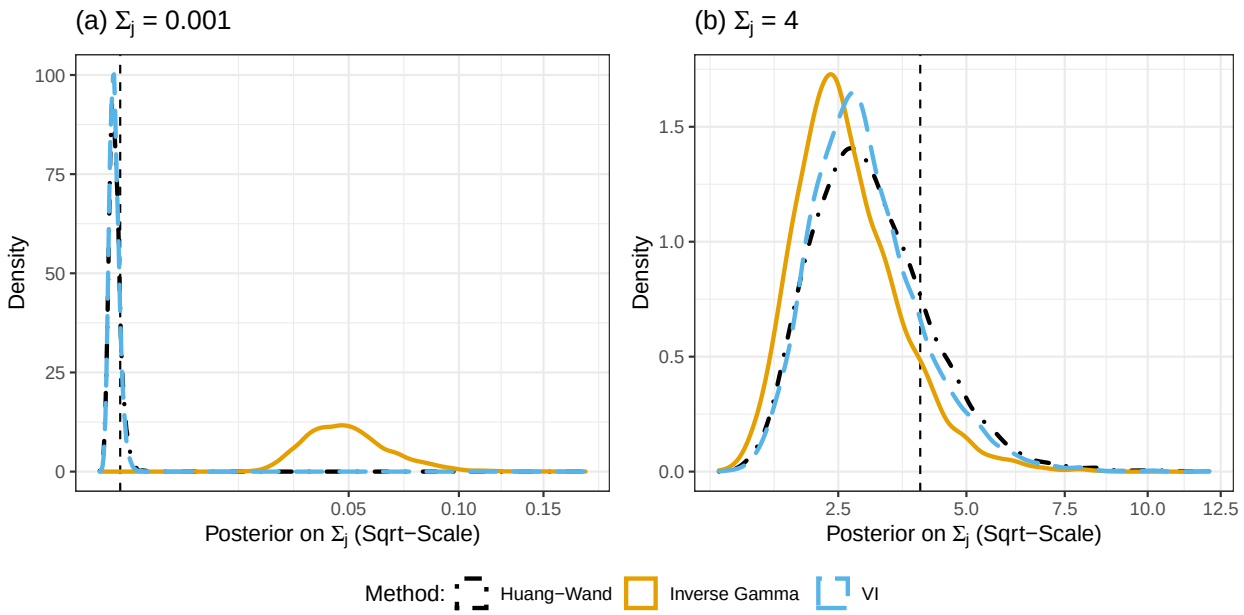
- *Update $q(\{\mathbf{\Sigma}_j\}_{j=1}^J)$:* $\tilde{\nu}_j = \nu_j + d_j - 1 + g_j$; $\tilde{\mathbf{\Phi}}_j = 2\nu_j \cdot \text{diag}\left(E_{q(a_{j,k})}[1/a_{j,k}]\right) + \sum_{g=1}^{g_j} E_{q(\boldsymbol{\alpha}_{j,g})}[\boldsymbol{\alpha}_{j,g}\boldsymbol{\alpha}_{j,g}^T]$
- *Update $q(\{a_{j,k}\})$:* $q(a_{j,k}) \sim \text{InverseGamma}(\tilde{a}_{j,k}, \tilde{b}_{j,k})$

$$\tilde{a}_{j,k} = \frac{\nu_j + d_j}{2} \quad \text{and} \quad \tilde{b}_{j,k} = 1/A_{j,k}^2 + \nu_j \left[E_{q(\mathbf{\Sigma}_j)}[\mathbf{\Sigma}_j^{-1}]\right]_{k,k}$$

Given the tight coupling of $\mathbf{\Sigma}_j$ and $a_{j,k}$ and the relatively cheap cost of the updates, the accompanying software performs this update a handful of times at each iteration, i.e. approximating optimizing both simultaneously. In all applications in this paper, I set $\nu_j = 2$ and $A_{j,k} = 5$. To illustrate the importance of this prior and the reasonableness of the variational approximation, consider the following stylized example: Conditional on the random effects $\boldsymbol{\alpha}_{j,g}$, does the Huang-Wand prior, its variational approximation, or the default Inverse-Wishart in Goplerud (2022) (i.e., $\nu = d + 1, \mathbf{\Phi} = \boldsymbol{I}$ or $a_0 = 1$, $b_0 = 1/2$ for the Inverse-Gamma) capture the true value well? To test this, I considered a scenario where $\mathbf{\Sigma}_j = 0.001$ (i.e., very small) and $\mathbf{\Sigma}_j = 4$ (i.e., large). I drew twenty samples from the corresponding normal, i.e. $g_j = 20$, and estimated the posterior or variational approximation. Figure A.1 shows the results.

The mis-calibration of the Inverse-Wishart prior is obvious when $\mathbf{\Sigma}_j = 0.001$, the resulting posterior is far too large—implying considerable under-shrinkage/regularization on the random effect estimates. This corroborates the concern in the main text that, especially for irrelevant random effects, the Inverse-Wishart prior will perform poorly. By contrast, the Huang-Wand prior is reasonably well-calibrated in both scenarios. The variational approximation is also rather good; it can correctly shrink irrelevant random effects by setting $\mathbf{\Sigma}_j$ to a small value, while closely approximating the true value for large ones.

Figure A.1: Comparison of Posterior Estimates



The solid orange line shows the posterior estimates with an Inverse-Gamma(1, 0.5) prior. The dot-dashed black line shows the posterior from the Huang-Wand prior; the dashed blue line shows the variational approximation.

A.2 SQUAREM

The models used in this paper employ a simple acceleration technique that maintains the monotonic convergence. Originally developed for Expectation-Maximization algorithms, SQUAREM (Varadhan and Roland 2008) is a squared iterative method that proceeds as follows in the application to variational inference. Noting that one can group all of the variational parameters into a block called $\boldsymbol{\theta}$, the SQUAREM algorithm can be adapted from Table 1 in Varadhan and Roland (2008) as shown below.

The above algorithm has many desirable properties; first, the final backtracking step (Step 7) ensures that it cannot decrease the objective and thus maintains monotonic convergence of the variational algorithm. Second, it is rather inexpensive to compute—it only requires the evaluation of the objective function at the proposed parameter vector $\boldsymbol{\theta}^*$ after performing three steps of the algorithm. Thus, the implementation of SQUAREM attempts to accelerate the model every three steps. In practice, it usually succeeds after no more than a few backtracking steps.

One additional modification is required for the above application: Given that many of the variational parameters are bounded (e.g., positive or symmetric matrices), I transform all parameters to live on an unbounded scale before applying SQUAREM. This ensures that all proposed $\boldsymbol{\theta}^*$ are valid regardless of the choice of α. For positive-constrained parameters, I take the log; for matrices, I take either the Cholesky decomposition or the LU decomposition and perform element-by-element SQUAREM where elements that are constrained to be positive are logged.

Algorithm A.2 SQUAREM (from Table 1 of Varadhan and Roland 2008)

1. Begin with initial parameters $\boldsymbol{\theta}^{(0)}$
2. Perform one step of CAVI (i.e., one step from Algorithm A.1) to get $\boldsymbol{\theta}^{(1)}$.
3. Perform a second step of CAVI (i.e., one step from Algorithm A.1 using $\boldsymbol{\theta}^{(1)}$ as initial values) to get $\boldsymbol{\theta}^{(2)}$.
4. Define the following quantities: $\boldsymbol{r} = \boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(0)}; \boldsymbol{v} = (\boldsymbol{\theta}^{(2)} - \boldsymbol{\theta}^{(1)}) - \boldsymbol{r}$
5. Calculate the step-length for SQUAREM α using the following formula:

$$\alpha = \min\left(-||\boldsymbol{r}||_2/||\boldsymbol{v}||_2, -1\right)$$

6. Propose a new $\boldsymbol{\theta}^*$ such that $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(0)} - 2\alpha\boldsymbol{r} + \alpha^2\boldsymbol{v}$
7. Evaluate whether the new $\boldsymbol{\theta}^*$ increases the objective (ELBO). If it does, set $\boldsymbol{\theta}^*$ as the new parameter estimates. If not, propose a new $\alpha \leftarrow (\alpha - 1)/2$ and try again. Note that if $\alpha = -1$, then $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(2)}$.

A.3 Parameter-Expansion

Another way to improve the speed of estimation is parameter expansion. It proceeds as follows: Note that Equation A.1 assumes a mean-zero prior on the random effect $\boldsymbol{\alpha}_{j,g}$: $\boldsymbol{\alpha}_{j,g} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_j)$. Goplerud (2022) provides the following definition of a parameter expansion below where the random effects are linearly transformed, and all other parameters adjusted, such that the log-posterior remains unchanged.

Definition A.1 (Expansions for Hierarchical Models)**.** *Define a set of expansion parameters $\boldsymbol{\xi}$ that consists, for each j, of a mean shift $\boldsymbol{\mu}_j \in \mathbb{R}^{d_j}$ and a scale shift $\boldsymbol{R}_j \in \mathbb{R}^{d_j \times d_j}$ such that $\boldsymbol{R}_j$ is invertible. I use superscript ^{X} to denote the "expanded" parameters. The mapping between $\boldsymbol{\theta}^X$ and $\boldsymbol{\theta}$ for a fixed $\boldsymbol{\xi}$ is denoted as $t_{\boldsymbol{\xi}}(\boldsymbol{\theta}^X)$ and listed below. $\boldsymbol{M}_j$ is a $p \times d_j$ matrix such that $[\boldsymbol{M}_j]_{a,b} = 1$ if the covariate corresponding to $[\boldsymbol{z}_{i,j}]_b$ is the same as the covariate for $[\boldsymbol{x}_i]_a$. All other elements of $\boldsymbol{M}_j$ are zero. For simplicity, assume that each element of $\boldsymbol{z}_i$ corresponds to some variable in $\boldsymbol{x}_i$, i.e. that each column of $\boldsymbol{M}_j$ has exactly one non-zero element.*

$$\left[\boldsymbol{\beta}, \boldsymbol{\alpha}, \{\boldsymbol{\Sigma}_j\}_{j=1}^J, \boldsymbol{\Omega}\right] = t_{\boldsymbol{\xi}}([\boldsymbol{\beta}^X, \boldsymbol{\alpha}^X, \{\boldsymbol{\Sigma}_j^X\}_{j=1}^J, \boldsymbol{\Omega}^X]) = \begin{cases} \boldsymbol{\beta} = \boldsymbol{\beta}^X + \sum_{j=1}^J \boldsymbol{M}_j \boldsymbol{R}_j \boldsymbol{\mu}_j \\ \boldsymbol{\alpha}_{j,g} = \boldsymbol{R}_j \left(\boldsymbol{\alpha}_{j,g}^X - \boldsymbol{\mu}_j\right) \\ \boldsymbol{\Sigma}_j = \boldsymbol{R}_j \boldsymbol{\Sigma}_j^X \boldsymbol{R}_j^T \\ \boldsymbol{\Omega} = \boldsymbol{\Omega}^X \end{cases}$$

The augmented model is listed below for an important special case treated in detail ("Mean Expansion") in the empirical analysis. The full expansion ("Translation Expansion") is also listed.

- *Mean Expansion: Assume all* $\boldsymbol{R}_j = \boldsymbol{I}_{d_j}$.

$$\ln p(y_i|\omega_i, \boldsymbol{\beta}^X, \boldsymbol{\alpha}^X) \propto \boldsymbol{s}^T[\boldsymbol{X}\boldsymbol{\beta}^X + \boldsymbol{Z}\boldsymbol{\alpha}^X] - 1/2[\boldsymbol{X}\boldsymbol{\beta}^X + \boldsymbol{Z}\boldsymbol{\alpha}^X]^T\boldsymbol{\Omega}[\boldsymbol{X}\boldsymbol{\beta}^X + \boldsymbol{Z}\boldsymbol{\alpha}^X]$$
$$p(\boldsymbol{\beta}^X) \propto 1, \quad \boldsymbol{\alpha}^X_{j,g}|\boldsymbol{\Sigma}^X_j, \sim N\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}^X_j\right), \quad p(\boldsymbol{\Sigma}^X_j) \sim IW(\nu_j, \boldsymbol{\Phi}_j)$$

- *Translation Expansion:*

$$\ln p(y_i|\omega_i, \boldsymbol{\beta}^X, \boldsymbol{\alpha}^X) \propto \boldsymbol{s}^T[\boldsymbol{X}\boldsymbol{\beta}^X + \boldsymbol{Z}\boldsymbol{R}\boldsymbol{\alpha}^X] - 1/2[\boldsymbol{X}\boldsymbol{\beta}^X + \boldsymbol{Z}\boldsymbol{R}\boldsymbol{\alpha}^X]^T\boldsymbol{\Omega}[\boldsymbol{X}\boldsymbol{\beta}^X + \boldsymbol{Z}\boldsymbol{R}\boldsymbol{\alpha}^X]$$
$$\boldsymbol{R} = \text{blockdiag}\left(\{\boldsymbol{I}_{g_j} \otimes \boldsymbol{R}_j\}_{j=1}^J\right), \quad p(\boldsymbol{\beta}^X) \propto 1, \quad \boldsymbol{\alpha}^X_{j,g}|\boldsymbol{\Sigma}^X_j \sim N\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}^X_j\right)$$
$$p(\boldsymbol{\Sigma}^X_j) \sim IW(\nu_j, \boldsymbol{R}_j^{-1}\boldsymbol{\Phi}_j\boldsymbol{R}_j^{-T})$$

With this definition, Goplerud (2022) applied a procedure known as PX-VB (Parameter-Expanded Variational Bayes; Jaakkola and Qi 2007) to improve convergence. His restatement of the result from Jaakkola and Qi (2007) is provided below:

Lemma A.1 (Parameter Expanded Variational Bayes - Jaakkola and Qi 2007)**.** *Given some factorization assumption* $\mathcal{X}$*, the following procedure converges no slower than the associated CAVI algorithm and maintains a monotonic improvement of the* ELBO*.*

1. *Perform one step of CAVI (e.g., Algorithm A.1, Steps 1-4) giving* $q(\boldsymbol{\theta})$ *and* $\text{ELBO}_{q(\boldsymbol{\theta})}$.
2. *Noting* $q(\boldsymbol{\theta}) \sim^d q(\boldsymbol{\theta}^X)$ *when* $\boldsymbol{\xi} = \boldsymbol{\xi}_{\text{Null}}$ *and thus* $\text{ELBO}^{X-\boldsymbol{\xi}_{\text{Null}}}_{q(\boldsymbol{\theta})} = \text{ELBO}_{q(\boldsymbol{\theta})}$*, maximize the* $\text{ELBO}^{X-\boldsymbol{\xi}}_{q(\boldsymbol{\theta})}$ *over* $\boldsymbol{\xi}$*.*

$$\hat{\boldsymbol{\xi}} = \arg\max_{\boldsymbol{\xi}} \text{ELBO}^{X-\boldsymbol{\xi}}_{q(\boldsymbol{\theta})} = \arg\max_{\boldsymbol{\xi}} E_{q(\boldsymbol{\theta})}[\ln p^X(\boldsymbol{y}, \boldsymbol{\theta}|\boldsymbol{\xi})] - E_{q(\boldsymbol{\theta})}[\ln q(\boldsymbol{\theta})]$$

   Note that $\text{ELBO}^{X-\hat{\boldsymbol{\xi}}}_{q(\boldsymbol{\theta})} \geq \text{ELBO}_{q(\boldsymbol{\theta})}$.

3. *Apply the reduction function to recover a distribution on the original, non-expanded space. Equivalently, transform* $q(\boldsymbol{\theta})$ *by applying a change-of-variables using* $t_{\hat{\boldsymbol{\xi}}}(\boldsymbol{\theta})$.

$$q'(\boldsymbol{\theta}) = \int t_{\hat{\boldsymbol{\xi}}}(\boldsymbol{\theta}) q(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

   Note that $\text{ELBO}_{q'(\boldsymbol{\theta})} = \text{ELBO}^{X-\hat{\boldsymbol{\xi}}}_{q(\boldsymbol{\theta})}$ *and* $\text{ELBO}_{q'(\boldsymbol{\theta})} \geq \text{ELBO}_{q(\boldsymbol{\theta})}$.

Using this result, Goplerud (2022) assumes that there is only a mean-expansion, i.e. $\boldsymbol{\alpha}^X_{j,g}$ has some non-zero mean, and derives a closed-form update for the parameter expansion step. This involves centering each random effect to be mean-zero and adjusting the corresponding fixed effects to keep the expected linear predictor constant.

This paper extends this work further by analyzing a PX-VB method for the translation expansion where $\boldsymbol{\alpha}^X_{j,g}$ is not only given a non-zero mean but is multiplied by some matrix $\boldsymbol{R}_j$. Assume that the random effects have been adjusted to be mean zero and thus the only expansion term is $\{\boldsymbol{R}_j\}_{j=1}^J$. Examining the objective (ELBO) and collecting

terms—given the factorization (Scheme I from Goplerud (2022)) assumed in this paper–results in the following objective. I use $\boldsymbol{\rho}$ to denote the stacked vectorized $\boldsymbol{R}_j$ matrices: $\boldsymbol{\rho}^T = \left[\text{vec}(\boldsymbol{R}_1)^T, \cdots \text{vec}(\boldsymbol{R}_J)^T\right]$. To build a more tractable algorithm, I also update the mean parameter of the variational distribution on $\boldsymbol{\beta}$ ($\tilde{\boldsymbol{\mu}}_\beta$) simultaneously to the expansion parameters. For simplicity, I focus on the random effects in Equation A.1, but splines are included similarly.

$$\begin{aligned}
\text{ELBO}_{q(\boldsymbol{\theta})}^{X-\boldsymbol{\xi}} \propto \quad & \boldsymbol{s}^T\left(\boldsymbol{X}\tilde{\boldsymbol{\mu}}_\beta + \boldsymbol{B}\boldsymbol{\rho}\right) - \frac{1}{2}\left(\boldsymbol{X}\tilde{\boldsymbol{\mu}}_\beta + \boldsymbol{B}\boldsymbol{\rho}\right)^T E_{q(\boldsymbol{\Omega})}[\boldsymbol{\Omega}]\left(\boldsymbol{X}\tilde{\boldsymbol{\mu}}_\beta + \boldsymbol{B}\boldsymbol{\rho}\right) + \\
& -\frac{1}{2}\boldsymbol{\rho}^T\left(\sum_{i=1}^N E_{q(\omega_i)}[\omega_i]\boldsymbol{K}_i^T\boldsymbol{K}_i\right)\boldsymbol{\rho} + \\
& \sum_{j=1}^J -\nu_j \cdot \ln|\boldsymbol{R}_j| - \frac{1}{2}\text{tr}\left(\boldsymbol{R}_j^{-1}\boldsymbol{\Phi}_j\left[\boldsymbol{R}_j^{-1}\right]^T E_{q(\boldsymbol{\Sigma}_j)}[\boldsymbol{\Sigma}_j^{-1}]\right) \\
& \text{where} \quad \boldsymbol{b}_{ij} = E_{q(\boldsymbol{\alpha}_{j,g[i]})}[\boldsymbol{\alpha}_{j,g[i]}] \otimes \boldsymbol{z}_{i,j}^b \quad \boldsymbol{b}_i^T = [\boldsymbol{b}_{i1}^T, \cdots, \boldsymbol{b}_{iJ}^T] \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_1^T \\ \cdots \\ \boldsymbol{b}_N^T \end{bmatrix} \\
& \boldsymbol{L}_{ij}^T\boldsymbol{L}_{ij} = \text{Var}(\boldsymbol{\alpha}_{j,g[i]}); \quad \boldsymbol{k}_{ij} = \left(\boldsymbol{L}_{ij} \otimes [\boldsymbol{z}_{ij}^b]^T\right) \quad \boldsymbol{K}_i = [\boldsymbol{k}_{i1}, \cdots, \boldsymbol{k}_{iJ}]
\end{aligned} \tag{A.6}$$

Note that the third line ($-\nu_j \ln|\boldsymbol{R}_j| \cdots$) represents the contribution of the prior; if this did not exist, then there is a closed-form update for the expansion parameters $\boldsymbol{\rho}$. However, for most choices of prior, this is intractable. Thus, I rely on a "one-step-late" idea for parameter expansion; Van Dyk and Tang (2003) apply a similar logic in the parameter-expanded EM case. The idea is to take the gradient but evaluate the gradient with respect to the intractable prior term as its *null* value, i.e. $\boldsymbol{R}_j = \boldsymbol{I}_{d_j}$. By setting the modified gradient equal to zero, one obtains the following update for $(\boldsymbol{\rho}, \tilde{\boldsymbol{\mu}}_\beta)$ where $\boldsymbol{0}_p$ is a vector of zeros with the dimensionality of $\boldsymbol{\beta}$.

$$\begin{aligned}
\hat{\boldsymbol{\rho}}, \hat{\tilde{\boldsymbol{\mu}}}_\beta \triangleq \quad & \boldsymbol{0} = [\boldsymbol{X}\boldsymbol{B}]^T\boldsymbol{s} - \\
& \left[[\boldsymbol{X}\boldsymbol{B}]^T E_{q(\boldsymbol{\Omega})}[\boldsymbol{\Omega}][\boldsymbol{X}\boldsymbol{B}] + \begin{pmatrix} \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times\sum_j d_j^2} \\ \boldsymbol{0}_{\sum_j d_j^2 \times p} & \sum_{i=1}^N E_{q(\omega_i)}[\omega_i]\boldsymbol{K}_i^T\boldsymbol{K}_i \end{pmatrix}\right]\begin{pmatrix} \tilde{\boldsymbol{\mu}}_\beta \\ \boldsymbol{\rho} \end{pmatrix} + \\
& \left[\boldsymbol{0}_p^T, \quad \left\{\text{vec}\left(-\nu_j\boldsymbol{I} + E_{q(\boldsymbol{\Sigma}_j)}[\boldsymbol{\Sigma}_j^{-1}]\boldsymbol{\Phi}_j\right)^T\right\}_{j=1}^J\right]^T
\end{aligned} \tag{A.7}$$

However, because of the one-step-late approximation, it is not guaranteed that these proposed updates $(\hat{\boldsymbol{\rho}}, \hat{\tilde{\boldsymbol{\mu}}}_\beta)$ will improve the objective. Thus, to ensure monotonic convergence, if it does not increase the objective, the software performs a few steps of numerical optimization (e.g., L-BFGS-B). Given availability of an analytic gradient and the modest size of the problem (the size of $\tilde{\boldsymbol{\mu}}_j$ plus $\sum_j d_j^2$), this is fairly inexpensive. In the case of a Huang-Wand prior, better performance is obtained by profiling out the $\tilde{b}_{j,k}$ parameters and performing

parameter expansion on this objective.

A.4 Splines

Consider the case of a single spline on a covariate x_i such as presidential two-party vote share. I follow Ruppert, Wand and Carroll (2003)'s presentation of splines as a hierarchical model as shown below. I present the simplest case of truncated linear functions as deviations from a globally linear trend; other extensions (e.g., to penalized B-splines; Eilers and Marx 1996) are straightforward.

The accompanying software follows the common strategy of using $\max(N_x/4, 35)$ knots where N_x is the number of unique values of $\{x_i\}$ and the knots placed at equally spaced quantiles of the distribution of $\{x_i\}$ (e.g., Ruppert, Wand and Carroll 2003); user-defined choices are possible.

$$
\begin{aligned}
&y_i \sim \text{Bern}(p_i); \quad p_i = \frac{\exp(\psi_i)}{1+\exp(\psi_i)}; \quad \psi_i = \beta_0 + \beta_1 x_i + \sum_{k=1}^{K} \gamma_k \left(x_i - \kappa_k\right)^+; \\
&\gamma_k \sim N(0, \sigma_\gamma^2); \quad p(\beta_0, \beta_1) \propto 1; \quad \sigma_\gamma^2 \sim p_0(\sigma_\gamma^2); \quad (x_i - \kappa_k)^+ = \begin{cases} 0 & \text{if} \quad x_i < \kappa_k \\ x_i - \kappa_k & \text{if} \quad x_i \geq \kappa_k \end{cases}
\end{aligned}
\tag{A.8}
$$

Note that as $\sigma_\gamma^2 \to 0$, i.e. the estimated random effect variance declines, the model collapses to one with a simple linear effect on the predictor x_i. As σ_γ^2 becomes large, the estimated effect becomes increasingly "wiggly". As in a normal random effect model, the data (and prior) thus determines the smoothness of the effect on x_i. Given that this model has one additional hierarchical term, it fits into the general design framework noted above (Equation A.1). If a spline on a second variable w_i were desired, it would be governed by a different variance parameter (e.g., $\sigma_{\gamma'}^2$) and enter as a second additional hierarchical term.

One important extension is to allow "factor-by-curve" splines, i.e. interactions between a spline and a categorical factor (Ruppert, Wand and Carroll 2003). This would allow, for example, the effect of presidential vote share to vary by education in a smooth fashion. In the most common formulation (again see Ruppert, Wand and Carroll 2003), the linear "fixed" component of the spline is not regularized and thus a baseline category is omitted. As that may over-fit, I present a slightly modified version where (i) the linear component is regularized and (ii) the smooth components share a variance component. The linear predictor for the binomial model is shown below. I focus on a single continuous covariate x_i and some factor j that has g_j levels where $z_i \in \{1, \cdots, g_j\}$ denotes the value for observation i.

$$\psi_i = \beta_0 + \beta_1 x_i + \sum_{k=1}^{K} \left[\gamma_{k,\text{Global}}(x_i - \kappa_k)^+\right] + \sum_{g=1}^{G_j} I(z_i = g) \left[(\alpha_{0,g} + \alpha_{1,g} x_i) + \sum_{k=1}^{K} \gamma_{k,g} (x_i - \kappa_k)^+\right];$$
$$\gamma_k \sim N(0, \sigma^2_{\gamma,\text{Global}}); \quad [\alpha_{0,g}, \alpha_{1,g}]^T \sim N(\mathbf{0}, \mathbf{\Sigma}_\alpha); \quad \gamma_{k,g} \sim N(0, \sigma^2_\gamma)$$
$$p(\beta_0, \beta_1) \propto 1; \quad \sigma^2_{\gamma,\text{Global}} \sim p_0(\sigma^2_{\gamma,\text{Global}}); \quad \mathbf{\Sigma}_\alpha \sim p_0(\mathbf{\Sigma}_\alpha); \quad \sigma^2_\gamma \sim p_0(\sigma^2_\gamma) \tag{A.9}$$

The model consists, therefore, of (a) a linear effect on x_i, (b) deviations from this linear effect by group g as a random slope/intercept combination, (c) a smooth effect on x_i, and (d) deviations from this smooth effect by group g. This formulation scales nicely to the case where the same continuous covariate (e.g., two-party vote share) has factor-by-curve for multiple factors (e.g., by education and ethnicity). The implementation is shown below.

- Add a random slope for the continuous variable x_i to each of the grouping factors, e.g. `(1 + x | g) + (1 + x | g2)`, etc.
- Add a global spline term corresponding to $\{\gamma_{k,\text{Global}}\}_{k=1}^K$. This adds one additional random intercept.
- Add derivations from the global spline term for each group for each of the interacting factors. This adds an additional random intercept for each grouping factor.

Thus, it is clear that this can be fit into the standard architecture of Goplerud (2022) for variational inference in logistic binomial models. The results are similar to existing work on variational inference for splines. In this paper, I assume independence between each of the spline-based random effects. I save for future explorations more detailed factorization schemes that, for example, allow dependencies between the deviations of the spline terms.

A.5 Importance of Acceleration Techniques

This section briefly illustrates the key need for acceleration techniques when the Huang-Wand prior is employed. The main issue when using this prior is that the model may require many more iterations to reach convergence. While there is some additional cost to estimating the Huang-Wand prior, the major concern is that estimation with this prior can often require hundreds of additional iterations to converge.

Thus, techniques that reduce the number of iterations considerably can be highly desirable—even if they increase the cost per iteration. As the discussion above notes, the cost of using SQUAREM is dominated by a handful of extra evaluations of the ELBO and a small number of matrix decompositions to extrapolate parameters on an unbounded scale; this is relatively inexpensive overall. The parameter expansion can be somewhat more expensive; however, note that it involves solving a least squares system that is only a function of the size of the fixed effects and the *dimension* d_j of the random effects. As this is relatively small (as only g_j, i.e. the number of levels, is often large), it can be done relatively inexpensively.

Table A.1: Comparing the Impact of Acceleration Techniques

(a) Number of Iterations

N	IW	HW	Accelerated HW
250	227.98	1465.17	21.66
500	262.28	1651.98	23.11
1000	183.12	1392.8	22.21
2000	106.9	1134.54	23.09

(b) Estimation Time (seconds)

N	IW	HW	Accelerated HW
250	18.47	274.59	7.77
500	24.03	327.52	8.55
1000	20.18	300.66	8.8
2000	15.1	279.14	10.34

Note: "IW" stands for Inverse-Wishart. "HW" stands for Huang-Wand. "Accelerated HW" uses the two acceleration techniques discussed in the main text. See Appendix B for more details.

Table A.1 illustrates this quantitatively; first, the left panel shows the number of iterations required for convergence across the simulations used in Appendix B. We see that the number of iterations is considerably larger—often by a multiple of 5-to-10—when comparing Inverse-Wishart ("IW") versus the Huang-Wand ("HW") prior. This has a corresponding effect on run-time (the right panel); while the Inverse-Wishart model can be estimated quickly in around 15-25 seconds, the Huang-Wand prior takes considerably longer (275-330 seconds). Yet, after applying acceleration techniques, the number of iterations can be reduced considerably—actually smaller than the original (non-accelerated) Inverse-Wishart model—with a run time that is comparable to the original choice of prior.

B Simulations for Deep Hierarchical Models

This section provides two sets of simulations to illustrate the importance of deep MRP on purely synthetic data. The first focuses on the "prediction" part of MRP and shows the impact of ignoring important interactions or non-linear effects. The second replicates the purely synthetic simulations in Ornstein (2020).

B.1 Simulation 1: Importance of Using Deep MRP

To illustrate the importance of deep MRP, this sub-section provides some simulations on synthetic data; they are designed to show the impact of ignoring important interactions or non-linear effects for the "prediction" part of MRP (i.e., predictive accuracy on held-out data similar to the original survey). Equation A.10 shows the data generating process; each model includes three continuous covariates with (possibly) non-linear functional forms[1] as well as two random effects (one with 5 levels and one with 50 levels) and their interaction. All observations are assigned to random effect groups fully at random.

[1]These are adapted from `mgcv`'s documentation on `gamSim`.

$$x_{i,0} \sim \text{Unif}(0,1);\ x_{i,1} \sim N(0,1);\ \Pr(x_{i,2} = k) = 0.01\ \forall k \in \{0, 1/99, \cdots, 1\} \quad \text{(A.10a)}$$

$$f_0(x_0) = \frac{1}{3} \cdot (\exp(2x_0) - 3.75887)\,; \quad f_1(x_1) = \frac{1}{2}x_1; \quad \text{(A.10b)}$$

$$f_2(x_2) = 1/4 \cdot (0.2 \cdot {x_2}^{11} \cdot (10 \cdot (1 - x_2))^6 + 10 \cdot (10 \cdot x_2)^3 \cdot (1 - x_2)^{10}) - 1$$

$$\alpha_g \sim N(0,1); \quad \alpha_{g'} \sim N(0,1) \quad \text{(A.10c)}$$

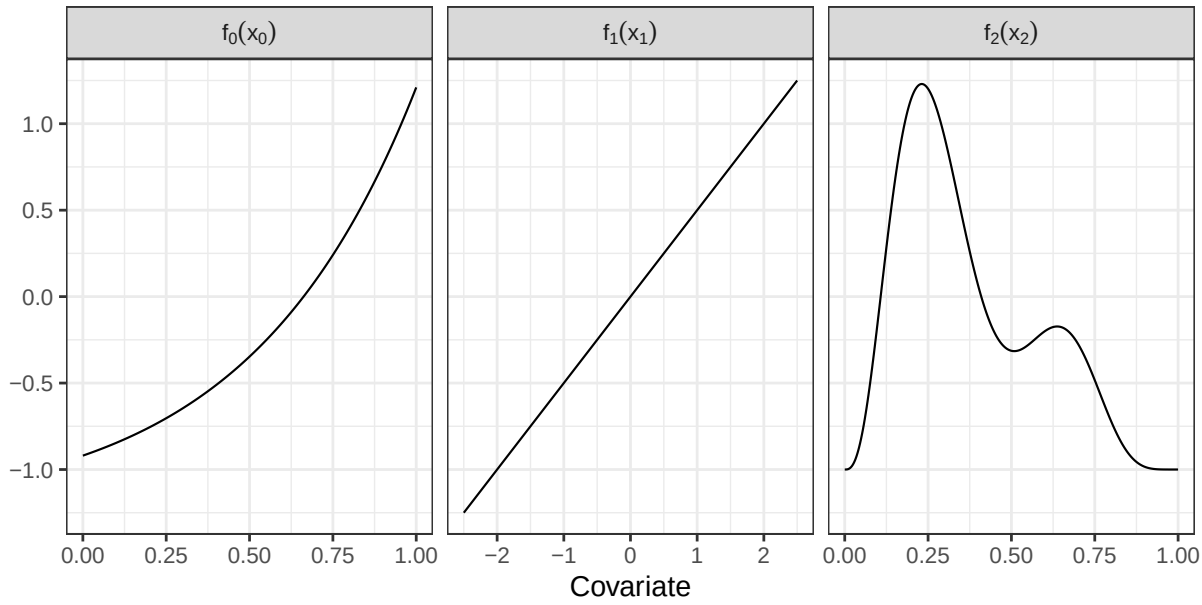
$$\alpha_{g,g'} = 0 \cdot \gamma_{g,g'} + \nu_{g,g'}(1 - \gamma_{g,g'}); \quad \gamma_{g,g'} \sim \text{Bern}(p_{\text{inter}}); \quad \nu_{g,g'} \sim N(0,4) \quad \text{(A.10d)}$$

$$\psi_i = f_0(x_{i,0}) + f_1(x_{i,1}) + f_2(x_{i,2}) + \alpha_{g[i]} + \alpha_{g'[i]} + \alpha_{g[i],g'[i]} \quad \text{(A.10e)}$$

$$y_i \sim \text{Bern}(p_i); \quad p_i = \frac{\exp(\psi_i)}{1 + \exp(\psi_i)} \quad \text{(A.10f)}$$

The effects of three continuous predictors, f_0, f_1, f_2, are shown below across the range of plausible values.

Figure A.2: Effects of Continuous Predictors



The contribution of the interactive random effect, indexed by $\alpha_{g,g'}$ for level g of the first factor and level g' of the second factor, is designed to be large to highlight the important of missing a possibly crucial interactive effect. The one difference from a typical simulation environment is that there is some probability p_{inter} that the random effect $\alpha_{g,g'}$ is zero. This allows to capture scenarios where the interaction is critically important (p_{inter} is small) or whether it is more irrelevant (p_{inter} is large) as only a small proportion of observations have some non-zero effect. In my analysis, I explore two relatively extreme cases where $p_{\text{inter}} = 0.25$ (most interactions matter) and $p_{\text{inter}} = 0.99$ where most interactions are irrelevant.

The prior expectation is that models that are not able to estimate interactions (e.g., simple MRP) should do considerably worse for $p_{\text{inter}} = 0.25$ whereas ignoring the interactions should impose little cost in terms of predictive performance for $p_{\text{inter}} = 0.99$.

This simulation, therefore, is designed to capture the features that deep hierarchical models are designed to address and allows for a comparison where simple hierarchical models are expected to do poorly. It also allows us to compare this against popular alternative machine learning approaches to see how they compare.

I generate datasets of size $N \in \{250, 500, 100, 2000\}$ and a corresponding test dataset of an identical size for the same data generating process. The key quantity the simulations consider is the out-of-sample predictive accuracy on the test data. I consider six models, outlined below.

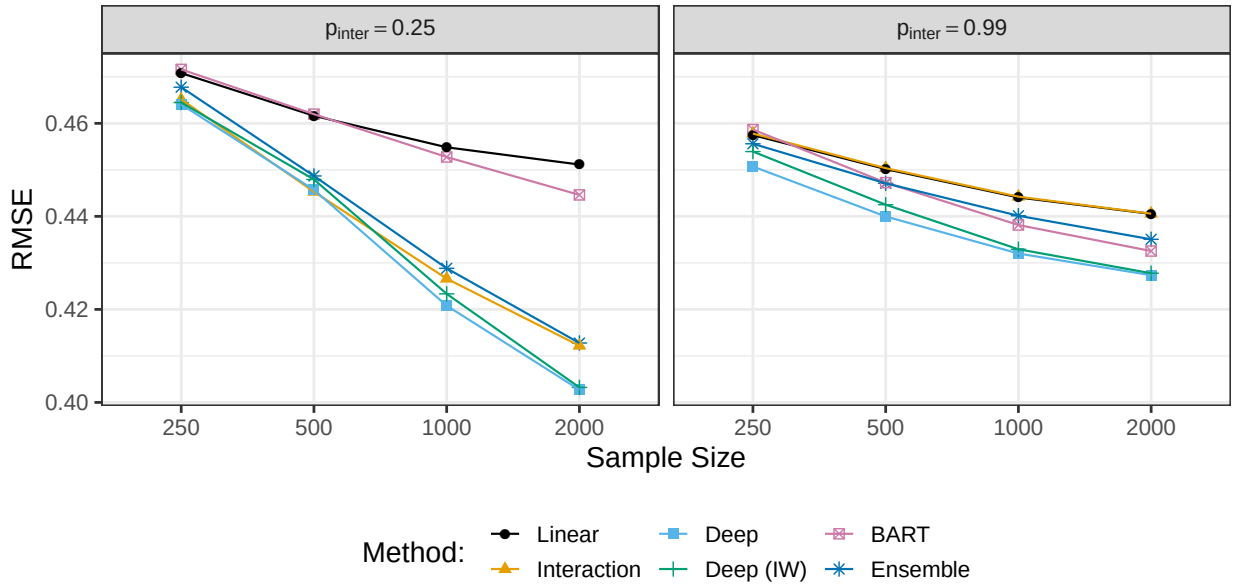
- Linear: This model includes only the two primary random effects, i.e. $\psi_i = \beta_{\text{Int}} + \beta_0 x_{i,0} + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \alpha_{g[i]} + \alpha_{g'[i]}$. It is fit using `lme4`.
- Interaction: This model adds the interactive random effect, i.e. $\alpha_{g[i],g'[i]}$ to the above model. It is fit using `lme4`.
- Deep: This model adds splines for each continuous predictor to the model in addition to $\alpha_{g[i],g'[i]}$. It is fit using the variational approach.
- Deep (IW): This model uses an Inverse-Wishart prior instead of the Huang-Wand prior.
- BART: This model is fit using `bartMachine` on the default settings.
- Ensemble: This includes an ensemble of a random forest, BART, and LASSO (with interactions between the two random effects). It **does not** include any hierarchical models.

Figure A.3 presents the root mean-square error (RMSE) on the out of sample predictions across each level of sparsity and sample size. The results are averaged across 1,000 simulations.

Consider first the panel where interactions are highly important $p_{\text{inter}} = 0.25$. In this case, a simple hierarchical model that includes neither interactions nor splines ("Linear") does very poorly compared to other models. Adding the interactions improves performance considerably, and a model that includes both splines and interactions performs the best. Note that, in this case, BART performs about the same as the simple hierarchical model for small sample sizes, but its advantage grows as it is given more data. However, it is always beaten by considerable margins compared to hierarchical models that include interactions.

Consider now a case where the interactions are mostly irrelevant ($p_{\text{inter}} = 0.99$). In this case, simple hierarchical models and the hierarchical model with interactions do about the same as there is little cost to ignoring the (rare) interactions. By contrast, BART does noticeably better in this setting as it is able to capture the non-linearity of the continuous predictor effectively. At the smallest sample size, however, all three methods perform about the same. It is worth noting that the hierarchical model with interactions and splines again does the best insofar as it can effectively capture the smooth non-linear functional forms using splines even at small sample sizes. Even though the model includes interactions that are irrelevant, this does not materially harm its performance as they are likely aggressively regularized. Indeed, experimentation with a clearly "overfit" hierarchical model (i.e., allowing the splines to vary across the categorical factors—a feature that is *not* present in the data

Figure A.3: Performance of Models on Simulated Data



generating process) does about the same suggesting that, in many cases, the regularization in hierarchical models is reasonably robust against preventing severe overfitting even when the model is simpler than the exact one specified.

Finally, we note that the use of the Inverse-Wishart prior ("Deep (IW)") in Goplerud (2022) results in slightly worse performance across both simulation settings and most sample sizes. This agrees qualitatively with the results in Appendix D that shows the "Deep (IW)" model performs worse for small sample sizes in both performance on held-out data and error for MRP.

I also conducted an analysis of ensembles using this synthetic formulation. I find that of an ensemble that includes LASSO, random forest, BART, and the hierarchical methods, the same story found in the main paper holds—the deep methods are given increasing weight as N increases and are the dominant method for large sample sizes. Yet, as Figure A.3 shows, an ensemble that **excludes** hierarchical models performs considerably worse than the deep hierarchical model on its own.

B.2 Simulation 2: Full Replication of MRP

This second set of simulations uses the data-generating process from the simulations in Ornstein (2020). Unlike most applications of MRP, the data generating process has a linear outcome and is designed to show the power of machine learning that can capture non-linear continuous effects—as they are given meaningful continuous predictors (e.g., "latitude" and "longitude") that are not made available to the MRP model. Thus, it is an interesting question how deep MRP fares in this challenging scenario where the importance of continuous predictors is especially stark.

I thus exactly replicate the analysis in Ornstein (2020), although I run 200 simulations

for each parameter configuration instead of ten. The generative model is shown below. The two key parameters are θ (controlling the amount of non-linearity and interactivity) and ρ (controlling the correlation between the features). Note that $\boldsymbol{z}_i$ are unobserved and the MRP models rely on only the observable $x_{i,1}$, $x_{i,2}$, unit membership $g[i]$, and unit-level continuous predictor $\lambda_g[i]$. Some of the machine learning methods (KNN, GBM, Forest) incorporate lat_g and long_g directly. This exactly follows Ornstein (2020). Thus, this is a difficult test for MRP given that it does not include certain clearly important variables and, rather, only a discretized proxy. The group g is assigned to be equal size such that observations with the smallest value of $z_{i,4}$ are assigned to Unit 1, the next set of observations are assigned to Unit 2. Each unit has N_g observations. There are 200 units.

$$y_i = z_{i,1} + z_{i,2} + z_{i,3} + \theta(D^0_{g[i]} z_{i,1} z_{i,2} - D^1_{g[i]} z_{i,1} z_{i,3}) + \epsilon_i; \quad \epsilon_i \sim N(0, 25) \tag{A.11a}$$

$$[z_{i,1}, z_{i,2}, z_{i,3}, z_{i,4}]^T \sim N(\mathbf{0}, \boldsymbol{\Sigma}); \quad \boldsymbol{\Sigma}_{i,j} = I(i = j) + \rho I(i \neq j) \tag{A.11b}$$

$$x_{i,j} = f(z_{i,j}) \quad j \in \{1, 2\}; \quad f(z) = 1 + I(z \geq 0) + I(z \geq 1) + I(z \geq 1) \tag{A.11c}$$

$$D^0_g = \sqrt{2} - \sqrt{\text{lat}_g^2 + \text{long}_g^2}; \quad D^1_g = \frac{\sqrt{\text{lat}_g^2 + \text{long}_g^2}}{2} \tag{A.11d}$$

$$\text{lat}_g, \text{long}_g \sim \text{Unif}(0, 1); \quad \lambda_g = \frac{1}{N_g} \sum_{i:g[i]=g} f(z_{i,3}) \tag{A.11e}$$

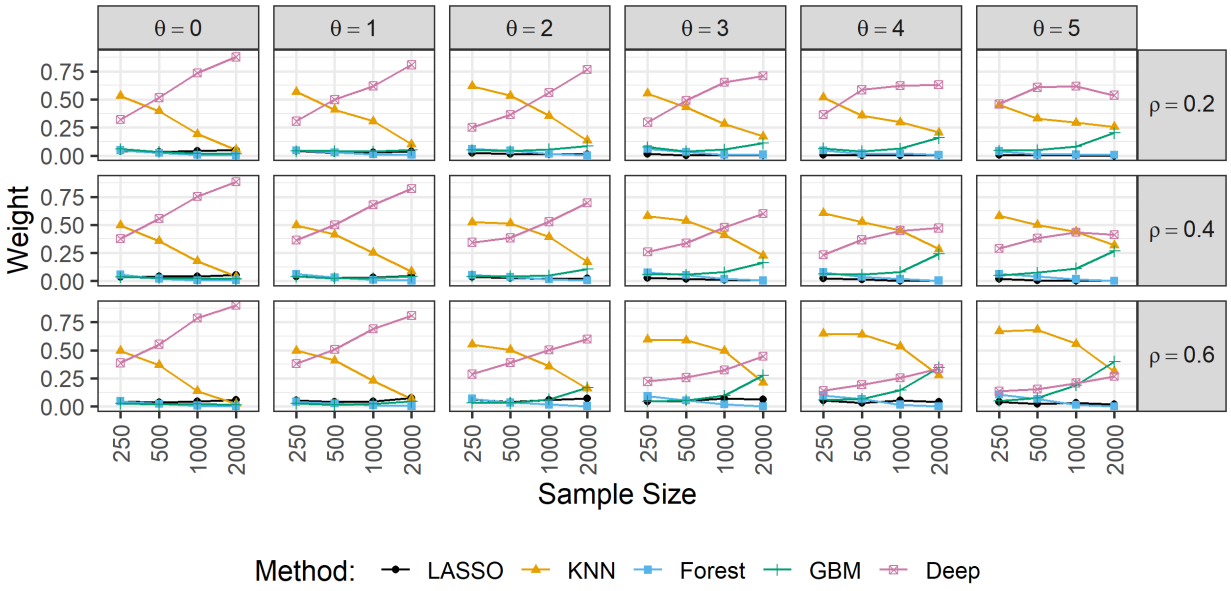
The simulation proceeds by drawing a "population" of 15,000 observations for each of the 200 units. Then, a sample of some size (e.g., 1,500) is taken at random for the "survey". A model is fit to that sample, and then the results are post-stratified with the true population average as the ground truth. Please see Ornstein (2020) for further details.

For MRP, I include random effects for $x_{i,1}$, $x_{i,2}$ and unit g. I also include the unit-level continuous predictor λ_g linearly. Deep MRP adds the "two-way" interactive model in the main text, i.e. interactions between $x_{i,1}$ and $x_{i,2}$, $x_{i,1}$ and unit g, $x_{i,2}$ and unit g.

Figure A.4 shows the weights from the corresponding ensemble of five methods. We see, as expected, that deep MRP receives increasing weight as the sample size increases. It is almost always the highest or second highest-weighted method, although it is less important when both ρ (correlation between covariates) and θ (interactivity) are extremely high. This is again likely because, when examining the cross-validated error used when fitting the ensemble, Deep MRP performs well at all parameter configurations and sample sizes. In many cases, it has the lowest RMSE and, even in case where it is given low weight in the ensemble, it is highly competitive and is never more than five percentage points worse than the main high-performing competitor of KNN.

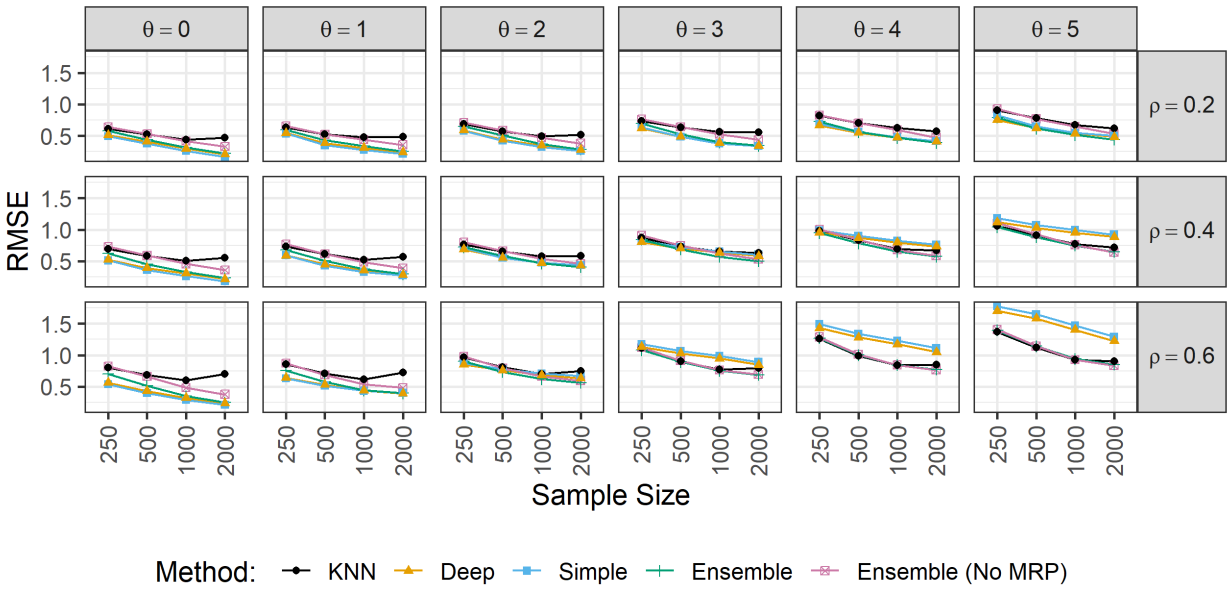
Next, I examine the performance of Simple MRP, Deep MRP, the top-performing machine learning method (KNN), and the ensemble in terms of average RMSE for the post-stratified estimates. I also show an ensemble that *does not* include any hierarchical models. We see that there is little difference between simple MRP and deep MRP for small-to-moderate amounts of heterogeneity and that both perform noticeably better than the ensemble that does not include any hierarchical models—even at a sample size of 2,000. Thus, for those scenarios, (deep) MRP appears to be a critical component of the ensemble's performance.

Figure A.4: Average Weight in Ensemble



In the cases where the hierarchical models perform less well (i.e., large heterogeneity θ and correlation ρ), we see that while performing less well than KNN, deep MRP improves upon simple MRP.

Figure A.5: Average RMSE for Post-Stratified Estimates



Overall, even in a challenging case where the simulations are designed to be very difficult for MRP given the importance of continuous predictors, MRP still performs well.

C Estimation Time

This section provides a direct comparison of the estimation times of the algorithms considered across the various simulations in this paper. Table A.2 reports the estimation time for three settings (purely synthetic simulations [Appendix B.1], ensembles [main paper], and BART [main paper]). It shows that, in all cases, deep MRP estimated with variational inference is highly competitive in terms of time need for estimation. All of the models are estimated using a single core and 8 GB of memory. All times are the average time in seconds, averaged across all models estimated for a particular sample size.

Across a variety of sample sizes and different model complexities, we see that deep MRP (i.e., including many interactions) is highly competitive in terms of estimation time with other ML methods. When comparing against BART (Panel C), we see that even the most complex variational model is within 15 seconds of BART on average. For the largest data sizes (e.g., 10,000), we see that the variational methods are actually faster than BART.[2]

Look beyond BART, the ensemble analysis (Panel B) shows that the variational methods are considerably faster than other standard machine learning techniques (e.g., random forests, K-nearest neighbors, etc.). The reason for this speed gain is that those models require tuning of the hyper-parameters (Ornstein 2020) such as the number of variables included in each tree for a random forest. By contrast, hierarchical models do not require explicit tuning of the amount of regularization as it is estimated internally in the model. This provides considerable savings in terms of cost of estimation.

In the synthetic examples, I also compare the variational model against the same model (i.e., three splines and interaction random effects) estimated using `mgcv` (via `gamm4`). It is considerably faster, especially as the sample size grows.

D Additional Empirical Results: Ensemble

This section outlines additional empirical results for the ensemble analysis replicating Ornstein (2020). The only substantial modification is adjusting the replication archive where a linear hierarchical model is used in the ensemble instead of a logistic hierarchical model.

I briefly describe how the ensemble was created using the standard "stacking" procedure—see Ornstein (2020) for a detailed discussion. I split the data into $K = 5$ folds. For each fold k, the model is estimated using the other folds and an out-of-sample prediction is obtained for each observation in fold k. Thus, out-of-sample predictions are constructed for the entire training dataset and each model. Those out of sample predictions are then used to construct the ensemble weights. The constituent models are then fit on the entire training dataset and a weighted average of their predictions is constructed. This is then post-stratified.

The power of the ensemble versus its constituent methods is illustrated in Figure A.6. It shows the percentage gap in mean absolute error (MAE) versus the ensemble of five methods reported in Figure 2. For model k, the table reports $(\text{MAE}_k - \text{MAE}_{\text{Ens}})/\text{MAE}_{\text{Ens}} \cdot 100$ where MAE_{Ens} reports the MAE for the five-model ensemble in the main text. A positive number

[2]Note that the time needed for the variational methods for this panel include the time for predicting on the census data.

Table A.2: Timing of Methods

(a) Synthetic (Appendix B.1)

N	Simple	Inter	Deep	BART	GAMM4
250	0.26	0.62	7.77	2.25	10.14
500	0.3	0.9	8.55	2.66	19.61
1000	0.47	1.38	8.8	4.94	34.86
2000	0.79	2.2	10.34	9.85	59.35

(b) Ensemble (Main Text)

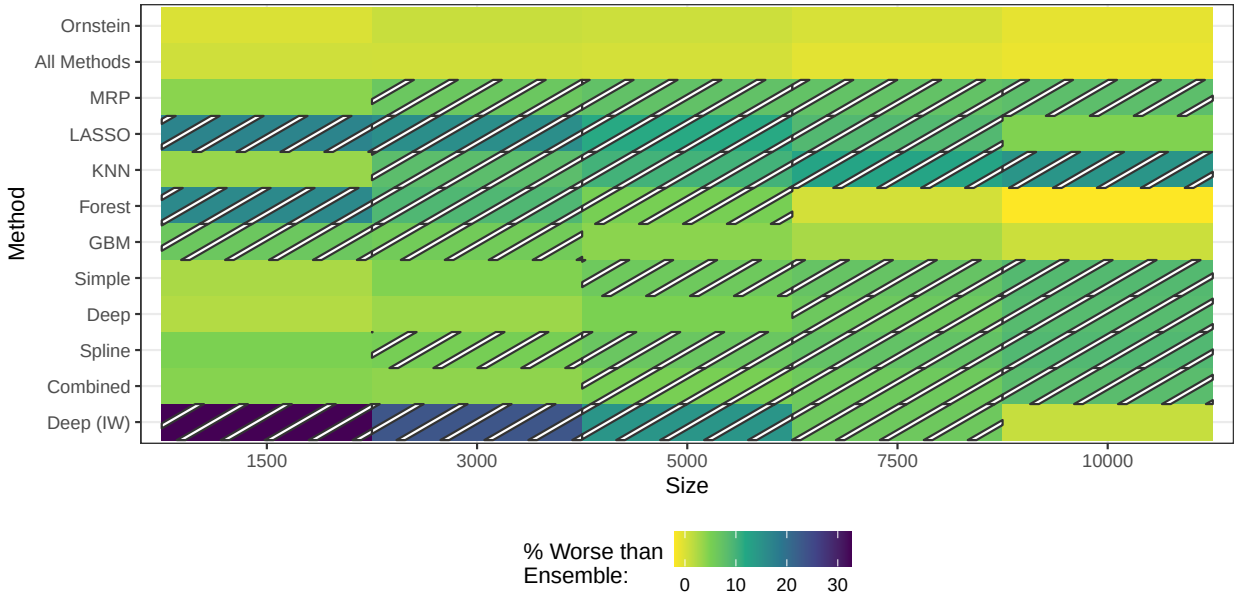
N	Methods in Ornstein (2020)					Variational Methods for MRP				
	MRP	LASSO	KNN	Forest	GBM	Simple	Deep	Spline	Combined	Deep (IW)
1500	2.33	0.94	160.88	94.63	1232.07	3.51	15.93	7.32	22.91	10.55
3000	3.52	1.47	201.14	202.22	1384.28	3.58	17.73	7.94	25.4	11.26
5000	5.13	2.15	235.51	252.15	1484.34	3.79	20.48	8.74	29.05	11.9
7500	7.31	3.04	296.54	322.59	1618.11	4.12	23.84	9.8	33.42	13.27
10000	9.85	3.84	362.99	396.81	1769.78	4.54	27.19	10.98	37.64	15.08

(c) BART (Main Text)

N	Simple	Deep	Spline	Combined	BART
1500	4.1	18.94	8.39	26.04	13.2
3000	4.31	22.64	9.3	30.73	21.05
4500	4.55	26.87	10.3	35.64	30.01
6000	4.99	31.92	11.71	41.72	41.04
7500	5.15	34.97	12.44	45.97	50.87
10000	5.53	41.26	13.88	52.82	68.76

Note: All times are in seconds and averaged across all surveys and/or simulations. In Panel (a), "Simple" and "Inter" refer to models estimated using `lme4` with no interactions or interactions, respectively. "Deep" refers to a model fit using the variational methods in this paper that include the random effect for interaction as well as splines for each continuous covariate. "GAMM4" represents the model with interaction random effects and splines fit with `gamm4`. In Panel (b), the methods correspond to the four variational models estimated in the main text; see Table 1 in the main document for details. Appendix D provides more details on the other methods. In Panel (c), the methods correspond to the models described in the main text and below (Appendix E).

Figure A.6: Relative Mean Absolute Error of Ensembles and Constituent Methods



Note: This plot reports the percentage point gap in mean absolute error between an ensemble of five methods and alternative specifications. Figure 2 describes the abbreviations. Diagonal stripes indicate that the ensemble in the main text out-performs this competitor by at least 5%.

(blue) indicate the model performs worse than the ensemble; a negative indicates that it beats the ensemble.

I also show the performance of the original ensemble reported in Ornstein ("Ornstein") and an ensemble that includes ten methods (the five in Ornstein; five variational MRP specifications - "All Methods"). Recall that "MRP" refers to a simple MRP fit with the Laplace approximation and flat prior using `glmer`.

A key implication of Figure A.6 shows that individual models usually perform noticeably worse than the ensemble in almost all circumstances and by often non-trivial margins (e.g., more than 5%). Put another way, every constituent method performs more than 5% worse than the ensemble at some sample size examined.

I next show the ensemble weights that come from the 'All Models" ensemble that puts together many different MRP specifications into a single ensemble. The six MRP models (MRP with Laplace approximation and flat prior ["MRP"] and the five variational models) are given, collectively, around 40-50% of the weight.

The table also shows an expected but important trade-off between simple and deep MRP; as sample size increases, the deep MRP models (e.g., "Deep" and "Combined") are given increasing weight whereas the traditional MRP models (e.g., "MRP" and "Simple") are given decreasing weight. This is reasonable as the ensemble upweights more complex methods as the amount of data increases. The relatively weak performance of the spline-based methods (e.g., "Spline" and "Combined") may be due to the limited variation in the continuous variables as they are measured at the state-level.

Table A.3: Weights Given to Models in Ensemble

Sample Size	Methods in Ornstein (2020)					Variational Methods for MRP				
	MRP	LASSO	KNN	Forest	GBM	Simple	Deep	Spline	Comb.	Deep (IW)
1500	0.119	0.071	0.207	0.042	0.285	0.107	0.061	0.040	0.014	0.053
3000	0.125	0.049	0.176	0.050	0.220	0.139	0.095	0.056	0.029	0.060
5000	0.094	0.047	0.168	0.045	0.210	0.140	0.114	0.073	0.032	0.078
7500	0.065	0.039	0.147	0.032	0.239	0.111	0.167	0.061	0.050	0.088
10000	0.050	0.041	0.138	0.038	0.247	0.102	0.167	0.057	0.060	0.098

Note: This table shows the ensemble weights averaged across all simulations. The first five columns include a hierarchical model fit with a flat prior and Laplace approximation (MRP), LASSO, k-Nearest Neighbors (KNN), a random forest (Forest), and a gradient boosting machine (GBM). The next four columns report hierarchical models estimated with variational inference and a Huang-Wand prior; "Comb." stand for the "Combined" model. The final column ("Deep [IW]") reports the results of the deep model using the Inverse-Wishart prior discussed in the main text.

E Additional Empirical Results: BART

This section outlines additional empirical results for the analyses comparing MRP against BART. I begin by explaining the error in the replication data for Bisbee (2019) and then provide additional analyses re-examining the main analyses in the paper.

E.1 Explanation of the Prediction Error

I began by examining the replication code on the paper's Dataverse.[3] The problem arises in the predict stage on the lines colored in red and blue.

```
mrp.form <- as.formula("y ~ pvote + religcon + (1|age) + (1|educ) + (1|gXr)
   + (1|stateid) + (1|region)")

...

temp<-catch.warning(glmer(mrp.form,
family=binomial(link="logit"),
data=sample.data))
model<-temp$value
mrp.warn<-ifelse(is.null(temp$warning)==T, 0, 1)

ranvars <- names(ranef(model))
full.ranefs <- rep(list(NA),length(ranvars))

i = 1
```

[3]See doi:10.7910/DVN/LMW871, Version 1.1

```
for(rv in ranvars) {
full.ranefs[[i]]<- data.frame(effect = rep(NA,length(unique(census.data[,rv
   ])))){}
rownames(full.ranefs[[i]]) <- as.character(c(unique(census.data[,rv])))
for(j in as.character(unique(census.data[,rv])))\{
full.ranefs[[i]][j,1] <- ranef(model)[rv][[1]][j,1]}
}
full.ranefs[[i]][,1][is.na(full.ranefs[[i]][,1])] <- 0
i = i+1
}

names(full.ranefs) <- ranvars

if(length(fixef(model)) > 1) {
fevars <- names(fixef(model))[2:length(fixef(model))]
feres <- apply(sapply(fevars, function(fe) fixef(model)[fe]*census.data[,fe
   ]),1,sum)
} else {
feres <- 0
}

# Create a prediction for each cell in Census data
res <- 0
for(rv in ranvars) {
temp <- full.ranefs[[rv]][census.data[,rv],1]
res <- res+temp
}
mrp.p <- invlogit(fixef(model)["(Intercept)"]
+ res
+ feres)
```

The problem is that when `unique` is called in R, it does not sort the values. Thus, `unique(census.data[,rv])` depends on the order of the rows in the data. For example, if the first values of some column x were, "1, 1, 3, 2, 4, 5,...", then unique would return "1, 3, 2, 4, 5". This creates a discrepancy between the code in red and blue when the predictions are assigned to post-stratification cells.

This causes a critical problem: Arbitrarily reshuffling the row order on the test data will give different predictions. The effect is to randomly inject noise by sometimes arbitrarily giving certain post-stratification cells the wrong random effect depending solely on the order of the rows in the test data (census data). A safe method for predicting would be to use the inbuilt `predict` function from the `lme4` package. In my replication analysis, I call this second function on the identical fitted model from `lme4`.

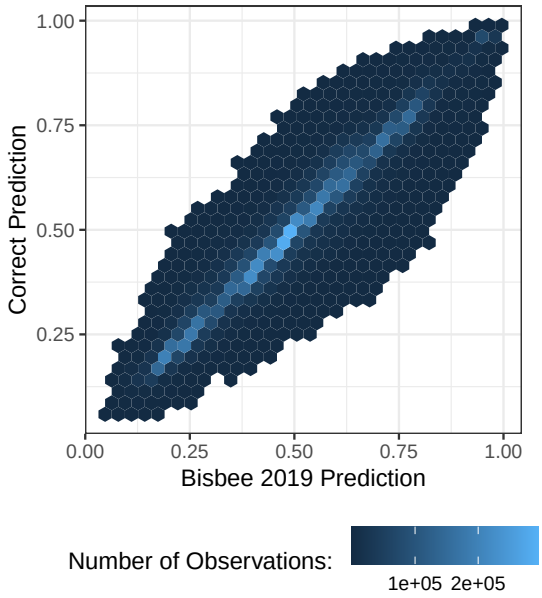
```
mrp.p.correct <- predict(model, newdata = census.data, allow.new.levels =
   TRUE, type = 'response')
```

I have also found that modifying the red line to include a sort appears to solve the problem, although I have not tested this extensively.

```
rownames(full.ranefs[[i]]) <- as.character(
c(sort(unique(census.data[,rv])))
)
```

In terms of raw impact, across all sample sizes, simulations, surveys and states, the mean absolute error of the custom prediction method is 0.043; the mean absolute error using the inbuilt `predict` function is 0.029—nearly 50% smaller versus the incorrect method. As another test, if the custom prediction function in the replication archive was correct, these predictions should be perfectly correlated. Figure A.7 shows the relationship between the state-level predictions between the two models. They are not identical; the correlation is 0.97. While this is high, note the visual inspection shows considerable variability.

Figure A.7: Comparing Prediction Methods

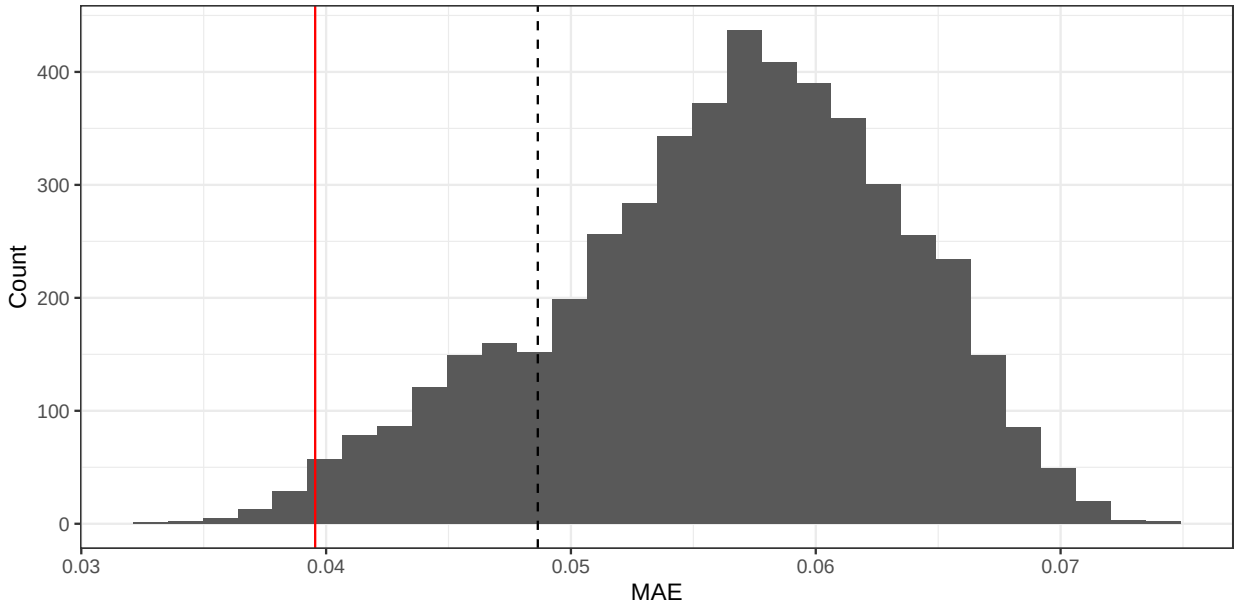


This raises the question of why are the predictions so highly correlated given the possibility of this error to inject considerable noise. The explanation is as follows: The structure of the Buttice and Highton (2013) data (i.e., the provided order of the rows) means that, for most random effects, unique usually returns a sensible result. Specifically, if one calls the relevant `unique` on each factor, it returns an order that often is the correct numerical order. The factor `region` is never in the correct order, the factor `gXr` (gender and race) is out of order for about 50% of the surveys, and the factor `educ` is out of order in a few instances. In those cases, if `lme4` estimates the corresponding σ_j^2 to be degenerate and equal to zero, the mis-aligned factors are estimated to have zero effect and the problem does not arise.[4]

[4]If one uses a package such as `bglmer` that puts a prior on the coefficients, then the error should occur every time.

I conjectured that the problem can be made much worse by shuffling the rows of the test data (census data) to make factors mis-aligned with a much higher probability. To illustrate the problem, I drew a random sample of 1,500 observations from Survey 1. I fit a single simple MRP model using `lme4`. I then perform predictions and post-stratification as follows: I first randomly shuffled the rows of the census data (test data) and then used both prediction methods. As expected, identical results are obtained using `predict` across all permutations. Using the code provided above, however, the results are quite variable. The distribution of the post-stratification error is shown in Figure A.8 where the correct prediction error is shown with a vertical bar in red. A dashed vertical bar shows the prediction error from the unpermuted data using the custom prediction method. It demonstrates that, if the test data came in with some random order, the problem would likely have been even more severe.

Figure A.8: Random Permutations of Census Data



E.2 Additional Results for BART

In this section, I replicate other results in Bisbee (2019): (i) presenting fuller results on the mean absolute error, (ii) comparing the correlation between methods and the truth, (iii) showing the sensitivity to omitting state-level predictors, and (iv) sensitivity to sample size.

First, I replicate Table 2 to include all four variational methods. As above, it shows that the spline-based methods perform slightly worse than those without splines. There is also some interesting comparisons between the two simple formulations fit with either the Laplace approximation and flat prior ("MRP") and with the Huang-Wand prior and variational inference ("Simple"). For small sample sizes, the variational method does better by about 1.5 percentage points, although it does slightly worse as sample size becomes very large. This suggests a more nuanced role for the prior and its interaction with sample size that future work into MRP should explore.

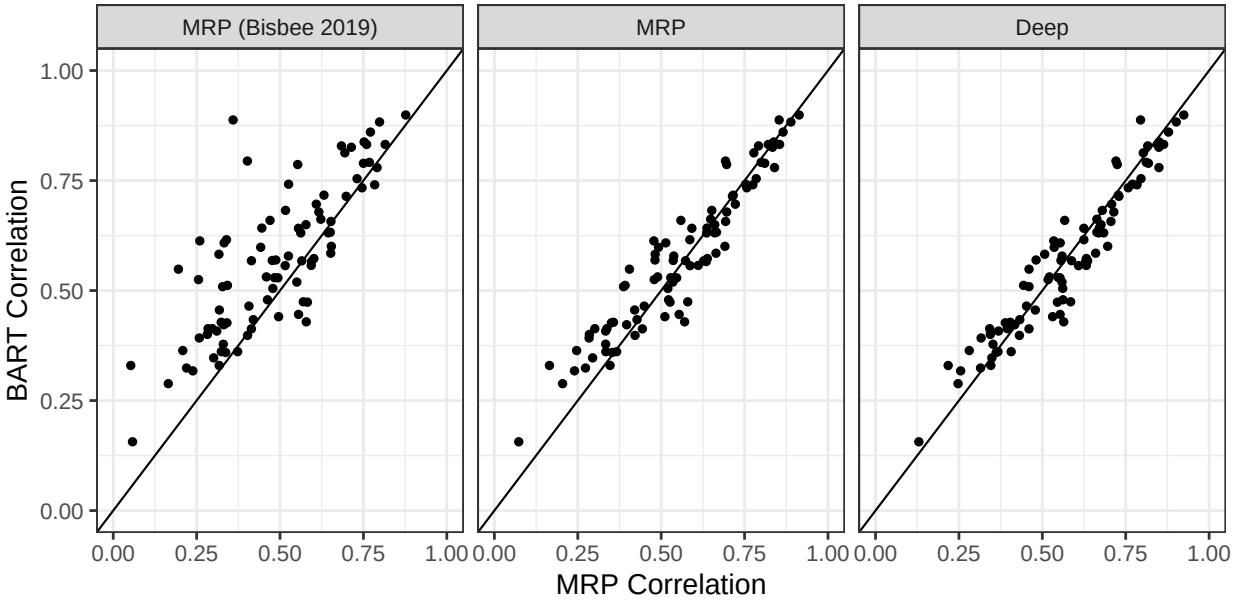
Table A.4: Relative Mean Absolute Error versus BART

	Sample Size					
Method	1500	3000	4500	6000	7500	10000
MRP (Bisbee 2019)	22.55	26.29	30.95	35.18	38.78	44.25
MRP	4.52	1.66	1.22	1.07	1.07	1.08
Simple	3.08	0.12	0.21	0.61	1.15	1.84
Deep	2.61	-1.04	-1.05	-0.59	0.14	1.11
Spline	5.57	1.51	1.09	1.43	2.04	2.78
Combined	5.04	0.28	-0.32	-0.01	0.66	1.49

Note: This table reports percentage gap in mean absolute error between BART and the alternative methods; positive numbers indicate that BART out-performs its competitor. Figures 2 and 3 describe the methods.

Following Bisbee (2019), I show the relationship between the correlation of each method's estimates against the truth. As before, the corrected simple MRP and deep MRP show quite similar correlations to BART.

Figure A.9: Visualizing Performance (Correlation): MRP versus BART



Note: Each plot compares the correlation from an MRP method (on the horizontal axis) to the MAE from BART (on the vertical axis). The 45-degree line is shown. Points above the line indicate that MRP performs worse. The three methods shown are the simple MRP with flat prior, Laplace approximation, and prediction method in Bisbee (2019) ("MRP (Bisbee 2019)"), the simple MRP with the same specifications but a correct prediction ("MRP"), and a deep MRP fit with a Huang-Wand prior and variational inference ("Deep").

Table A.5 show the percentage gap in correlation versus BART averaged across the 89 surveys and six sample sizes: $(\rho_k - \rho_{\text{BART}})/\rho_{\text{BART}} \cdot 100$ where ρ_{BART} indicates the correlation of the estimates from the BART model and the true state-level values, averaged across 200

simulations. A **negative** number indicates that BART out-performs the other method. As expected from the main text, BART slightly out-performs the MRP models, although at the smallest sample size of 1,500, deep MRP has a slight edge. In general, however, the differences are rather small. For most sample sizes, the difference in correlation between BART and MRP is between about 0-4%.

Table A.5: Relative Correlation versus BART

Method	1500	3000	4500	6000	7500	10000
MRP (Bisbee 2019)	-15.32	-16.96	-17.55	-17.84	-17.59	-17.49
MRP	-4.48	-4.22	-4.24	-4.13	-3.88	-3.63
Simple	-1.09	-2.12	-3.14	-3.68	-3.99	-4.28
Deep	0.38	-0.47	-1.66	-2.31	-2.91	-3.44
Spline	-3.29	-3.40	-3.96	-4.36	-4.63	-4.88
Combined	-1.61	-1.59	-2.25	-2.76	-3.25	-3.66

Note: This table reports the table reports the relative gap (percentage points) in correlation against the truth between BART and the alternative methods named in each row: $(\rho_k - \rho_{\text{BART}})/\rho_{\text{BART}} \cdot 100$. Negative numbers indicate that BART out-performs its competitor. The method names are as described in Figures 2 and 3.

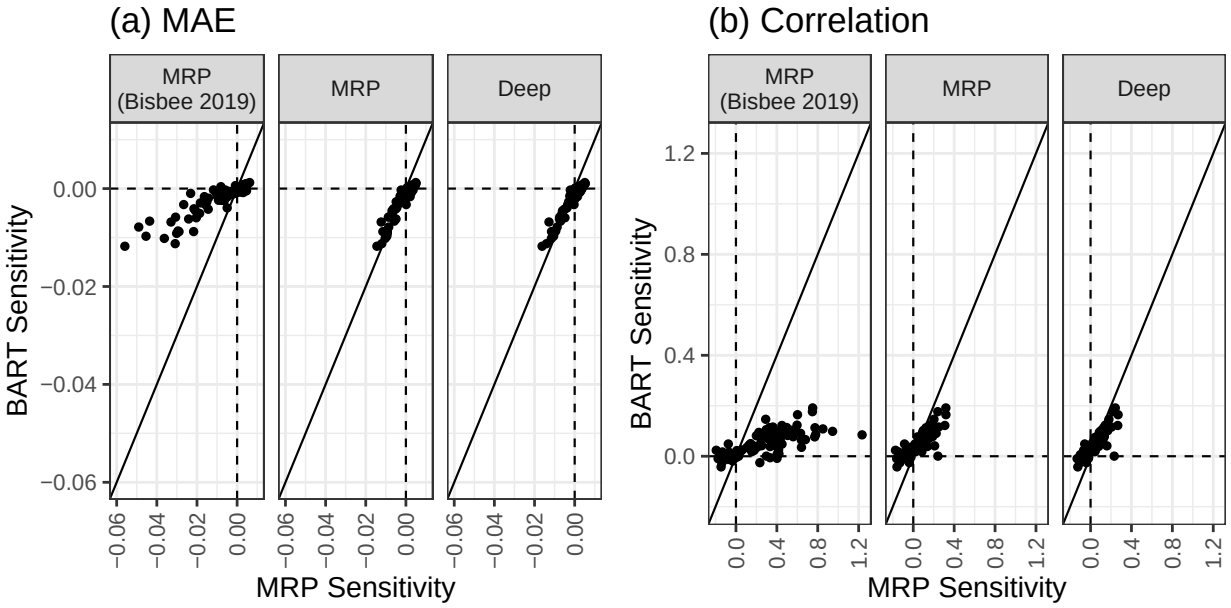
The next result in Bisbee (2019) is about the robustness of BART to mis-specification. The test is as follows: Remove the two contextual predictors (presidential vote share and religiosity) and see how the mean absolute error changes (increases) and correlation changes (decreases). The claim presented in the paper is that there are relatively minor changes to BART in this mis-specified model whereas it dramatically impacts the performance of MRP. Figure A.10 shows this is not the case. Both the corrected simple MRP and deep MRP lie near the 45-degree line and seem to have similar performance to BART (i.e., similar expected degradations in performance).

An additional claim about BART is that it can "do more with less data" (Bisbee, 2019, p. 1063). This is tested by examining how the mean absolute error at the state-level changes with the number of observations for that state. Bisbee (2019) does this by comparing the regression coefficients on (scaled) number of observations and absolute error between BART and MRP.

While a reasonable strategy, it has the limitation of conflating two points: As the above results suggest, (traditional) MRP often has a slight disadvantage in performance versus BART. The separate method-by-method regression has the disadvantage of comparing the *slope* with respect to the number of samples but ignoring the *intercept*. Indeed, one might expect that since MRP performs less well at baseline, it might be reasonably expected to have a steeper slope. This is not necessarily evidence that it performs worse with less data, but rather that it can "catch up" to BART quickly as the state-level sample size increases. To illustrate this, Figure A.11 proceeds as follows: for each simulation-state estimate, it takes the number of observations in each state and uses this to predict the state absolute error. It uses a smooth curve to plot the relationship by the four methods under consideration in this Appendix.[5] I present results for 1,500 for simplicity. The two dashed lines indicate the

[5]Specifically, given the skewed distribution of the sample size per state, I use `geom_smooth(..., method`

Figure A.10: Sensitivity to Mis-Specification



Note: This figure reports the change in MAE or correlation between the main specification and one that removes the state-level continuous predictors. Negative values for MAE indicates that doing so hurts performance; positive values for correlation indicates that doing so hurts performance.

25-75th percentile range of observed state-level sample sizes.

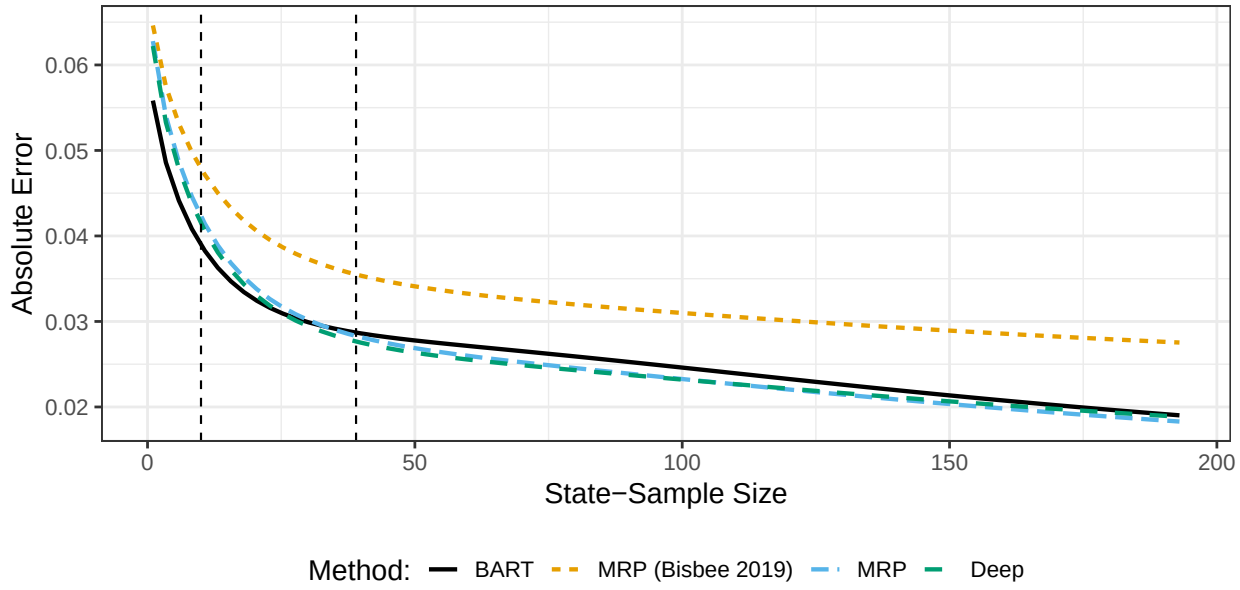
It shows limited evidence in favor of BART. While BART does perform slightly better for the smallest sample sizes, when we are in the range of most observations (around 10-39 observations per state), the differences are very minor in the curves. An interesting point to note is also that the curves seem to decline more quickly (i.e., improve faster as sample size for a state increases) for the MRP models than with BART. In total, however, the differences between BART and all correctly specified MRP models are rather small in substantive magnitude.

Finally, I replicate one figure from the supporting information in Bisbee (2019). Specifically, a claim made is that "one possible explanation for BARP's superior performance across sample sizes is the method's insulation from particularly challenging surveys". This is demonstrated by the inability of MRP to well-predict certain of the eighty-nine surveys even with large amounts of data. Figure A.12 replicates the plot in the supplemental information by plotting the mean absolute error by sample size as separate lines for each of the eighty-nine surveys. It shows that corrected simple MRP and deep MRP both show visually similar patterns to BART, i.e. declining error as sample size increases.

Overall, therefore, this suggests that while BART has a slight edge under certain circumstances (e.g., for very small numbers of observations per state), there is limited evidence of it being systematically better than MRP. It does not appear to be substantially more robust to mis-specification nor have materially better performance in the simulations considered in Bisbee (2019).

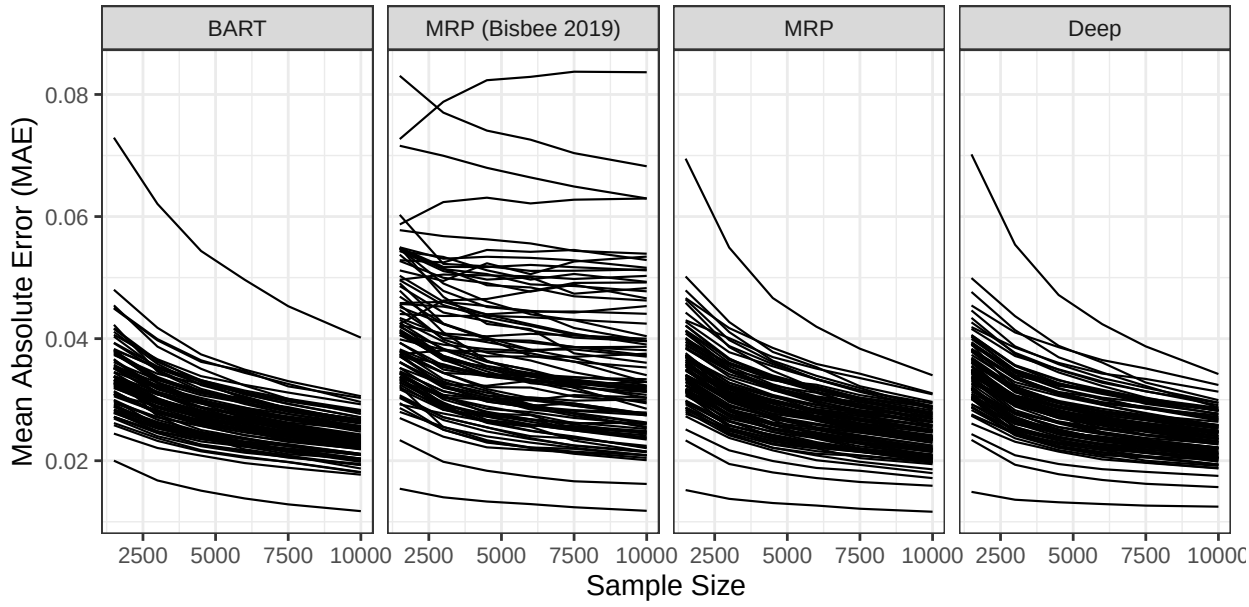
= 'gam', formula = y ∼ s(log(x), k =5)).

Figure A.11: State-Level Sample Size and Absolute Error



Note: This figure plots a smooth regression between the number of state-level observations and the observed absolute error in the state prediction. The curve for each method is shown in the corresponding color.

Figure A.12: Performance of Methods by Survey



Note: The figure plots the mean absolute error (MAE) for each survey across the six sample sizes. Results are averaged across 200 simulations for each survey and sample size.

F Software

This appendix provides a brief demonstration of how to use the accompanying software available at `https://cran.r-project.org/package=vglmer` or `https://github.com/mgoplerud/vglmer`; it can be installed directly into R using, e.g., `install.packages("vglmer")` for the CRAN version or `devtools::install_github("mgoplerud/vglmer")` for the most up-to-date version. Broadly speaking, the package has been designed to be as similar as possible to `lme4` in terms of how random effects are specified. A similar syntax to `mgcv` is used for estimating splines. The default options use a Huang-Wand prior as well as both acceleration techniques; these can be modified as shown below. Please note this may change in future versions of the package.

For simplicity, I rely on a dataset from the `pscl` package on the votes of members of Congress for the Iraq War. The first model illustrates the flexibility of the accompanying software from this paper (`vglmer`) as it includes multiple random effects, random intercepts and random slopes, as well as splines. The second example illustrates how default options could be modified if desired.

```
# Fitting Ensembles
library(SuperLearner)
# Fitting variational hierarchical models ("variational + glmer")
library(vglmer)

# Load data from pscl on the Iraq War Vote
iraq_data <- pscl::iraqVote
iraq_data$region <- state.region[match(iraq_data$state.abb, state.abb)]
head(iraq_data)
#>   y state.abb             name  rep state.name gorevote region
#> 1 1        AL  SESSIONS (R AL) TRUE    Alabama    41.59  South
#> 2 1        AL    SHELBY (R AL) TRUE    Alabama    41.59  South

# "v_s(gorevote)" estimates a non-linear effect using penalized splines.
fit_vglmer <- vglmer(y ~ v_s(gorevote) + (1 | region) +
(1 + gorevote | rep), data = iraq_data, family = 'binomial')

fit_vglmer <- vglmer(y ~ v_s(gorevote) + (1 | region) +
(1 + gorevote | rep), data = iraq_data,
# Manually specify  a Huang-Wand prior
control = vglmer_control(prior_variance = 'hw'),
family = 'binomial')
```

From this model, output can be extracted in a highly similar way to `lme4`. Predictions can also be done in a straightforward fashion.

```
# Terms can be extracted in a similar fashion to lme4
ranef(fit_vglmer)
```

```
#> $region
#>               id (Intercept)
#> 1     Northeast   0.2501737
#> 2         South   0.2967595
#> 3 North Central  -0.3242042
#> 4          West  -0.2227290
#>
.... [Other Random Effects Not Shown for Space]

fixef(fit_vglmer)
#> (Intercept)    gorevote
#>   8.7964438  -0.1423059
vcov(fit_vglmer)
#>             (Intercept)      gorevote
#> (Intercept)  2.18534247 -0.0449548442
#> gorevote    -0.04495484  0.0009501862

# Predict and turn into probability scale
plogis(
predict(fit_vglmer, newdata = iraq_data, allow_missing_levels = TRUE)
)
```

Finally, a key implication of the paper is that hierarchical models perform well in ensembles. The accompanying package extends the popular `SuperLearner` package in R to accommodate hierarchical models that can take a formula as an argument. Thus, one can easily specify an ensemble that includes both simple and deep MRP, and use a data-driven procedure to select the optimal combination.

```
# The formula must be added manually like any other tuning parameter for
# SuperLearner (e.g., using "create.Learner" or manually as below)
SL_v1 <- function(...){
SL.vglmer(formula = y ~ v_s(gorevote) + (1 | region) +
(1 + gorevote | rep), ...)
}

fit_SL <- SuperLearner(Y = iraq_data$y,
X = iraq_data[,c('gorevote', 'region', 'rep')],
family = binomial(), verbose = TRUE,
SL.library = c('SL.ranger', 'SL.glmnet', 'SL.bartMachine', 'SL_v1'))
```

References

Bates, Douglas, Martin Mächler, Ben Bolker and Steve Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* 67(1):1–48.

Bisbee, James. 2019. "BARP: Improving Mister P Using Bayesian Additive Regression Trees." *American Political Science Review* 113(4):1060–1065.

Buttice, Matthew K. and Benjamin Highton. 2013. "How Does Multilevel Regression and Poststratification Perform with Conventional National Surveys?" *Political Analysis* 21(4):449–467.

Chung, Yeojin, Andrew Gelman, Sophia Rabe-Hesketh, Jingchen Liu and Vincent Dorie. 2015. "Weakly Informative Prior for Point Estimation of Covariance Matrices in Hierarchical Models." *Journal of Educational and Behavioral Statistics* 40(2):136–157.

Eilers, Paul H.C. and Brian D. Marx. 1996. "Flexible smoothing with B-splines and penalties." *Statistical Science* 11(2):89–121.

Gelman, Andrew. 2006. "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis* 1(3):515–533.

Gelman, Andrew and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Goplerud, Max. 2022. "Fast and Accurate Estimation of Non-Nested Binomial Hierarchical Models Using Variational Inference." *Bayesian Analysis* 17(2):623–650.

Huang, Alan and Matt P. Wand. 2013. "Simple Marginally Noninformative Prior Distributions for Covariance Matrices." *Bayesian Analysis* 8(2):439–452.

Jaakkola, Tommi S. and Yuan Qi. 2007. Parameter Expanded Variational Bayesian Methods. In *Neural Information Processing Systems 2007*.

Ornstein, Joseph T. 2020. "Stacked Regression and Poststratification." *Political Analysis* 28(2):293–301.

Polson, Nicholas G., James G. Scott and Jesse Windle. 2013. "Bayesian Inference for Logistic Models Using Pólya–Gamma Latent Variables." *Journal of the American Statistical Association* 108(504):1339–1349.

Ruppert, David, Matt P. Wand and Raymond J. Carroll. 2003. *Semiparametric Regression*. Cambridge University Press.

Van Dyk, David A. and Ruoxi Tang. 2003. "The One-Step-Late PXEM Algorithm." *Statistics and Computing* 13(2):137–152.

Varadhan, Ravi and Christophe Roland. 2008. "Simple and Globally Convergent Methods for Accelerating the Convergence of Any EM Algorithm." *Scandinavian Journal of Statistics* 35(2):335–353.

Zhao, Yihua, John Staudenmayer, Brent A. Coull and Matt P. Wand. 2006. "General Design Bayesian Generalized Linear Mixed Models." *Statistical Science* 21(1):35–51.