MIBEN: Robust Multiple Imputation with the Bayesian Elastic Net

By

Kyle M. Lang

Submitted to the Department of Psychology and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Committee members

Wei Wu, Chairperson

Pascal Deboeck

Carol Woods

Paul Johnson

William Skorupski

Date defended: May 8, 2015

The Dissertation Committee for Kyle M. Lang certifies
that this is the approved version of the following dissertation :

MIBEN: Robust Multiple Imputation with the Bayesian Elastic Net

Wei Wu, Chairperson

Date approved: May 8, 2015

Abstract

Correctly specifying the imputation model when conducting multiple imputation remains one of the most significant challenges in missing data analysis. This dissertation introduces a robust multiple imputation technique, *Multiple Imputation with the Bayesian Elastic Net* (MIBEN), as a remedy for this difficulty. A Monte Carlo simulation study was conducted to assess the performance of the MIBEN technique and compare it to several state-of-the-art multiple imputation methods.

Acknowledgements

I would first like to thank my Ph.D. Advisor, Dr. Wei Wu, who has been a steadfast source of support, mentorship, and sage advice throughout my graduate training. I would also like to thank the other members of my dissertation committee, Drs. Carol Woods, Pascal Deboeck, Billy Skorupski, Paul Johnson, and Vince Staggs, for their very helpful suggestions during the development of this project. I wish to thank Dr. Vince Staggs, in particular, for the gracious accommodations that he made to facilitate the defense of this dissertation's proposal. I would like to thank my beloved wife, Dr. Eriko Fukuda, whose unyielding love and support has made this dissertation possible. Without you by my side, Eriko, it is very unlikely that I would have had the fortitude to see my graduate training to its end. I thank my parents, Scott and Lori Lang, my grandmother, Beverly Bareiss, and my siblings, Anthony, Dalton, and Maggie Lang, for continually acting as my immutable champions—regardless of the outcome of any academic pursuit. I must thank Anthony, additionally, for his patient and thoughtful programming advice which helped me immensely while writing the Gibbs sampler underlying the MIBEN method. Finally, I wish to acknowledge the continual contribution of all of my colleagues in the University of Kansas Quantitative Psychology Program. Working in such an intellectually stimulating environment has undoubtedly improved the quality of this project. In particular, I wish to highlight the contributions of Jared Harpole, Terry Jorgenson, and Mauricio Garnier-Villarreal who have each had a very direct impact of the outcome of this project through our many stimulating discussions of Bayesian statistics, missing data analysis, and computational methods.

Contents

Figures

Tables

Algorithms

Code Listings

Chapter 1

Introduction & Literature Review

Any data analytic enterprise will inevitably suffer from missing data, and psychological research is certainly no exception. The sensitive nature of the topics that psychologists study virtually ensures that the participants will fail to respond to some questions when data are collected via self-reported measures. Data can also go missing due to random errors like equipment failures that affect an experimental apparatus or a computer malfunction corrupting some portion of data stored electronically. No matter the reason for the missing data, the result is a data set that contains "holes" where some of the cells with true, population-level values are empty. Such *incomplete* data cannot be analyzed via traditional statistical tools (which require data to be fully observed), so something must be done to address the nonresponse.

This is the purpose of *missing data analysis*: a set of tools that are (usually) applied during data pre-processing to edit the data in such a way that they can be analyzed with standard complete data techniques. The methodological literature on missing data analysis is certainly extensive, and many powerful tools have been developed to address the issues caused by nonresponse (e.g., multiple imputation [MI] – Rubin, 1978, 1987; the expectation maximization [EM] algorithm – Dempster, Laird, & Rubin, 1977; full information maximum likelihood [FIML] – Anderson, 1957; and sequential regression imputation [SRI]/multiple imputation with chained equations [MICE] – Raghunathan, Lepkowski, Van Hoewyk, & Solenberger, 2001; van Buuren,

Brand, Groothuis-Oudshoorn, & Rubin, 2006). The most powerful of these approaches, such as MI and FIML, are known as *principled* missing data treatments, because they address the nonresponse by modeling the underlying distribution of the missing data. With an appropriate model for the missingness, these methods can either simulate plausible replacements for the missing data with random draws from the posterior predictive distribution of the nonresponse (in the case of MI) or partition the missing information out of the likelihood during model estimation (in the case of FIML).

In the interest of brevity, I will not give an extensive overview of missing data theory, but interested readers are encouraged to explore the wealth of work available on modern missing data analysis. Readers interested in accessible treatments with less of the mathematical details should consider Little, Jorgensen, Lang, and Moore (2013), Little, Lang, Wu, and Rhemtulla (in press), and Schafer and Graham (2002) for papers on the subject or Enders (2010), Graham (2012), and van Buuren (2012) for book length treatments. Those who desire a more technical discussion and a thorough explanation of the underlying mathematics should consider Little and Rubin (2002), Rubin (1987), Schafer (1997), and Carpenter and Kenward (2012) which are all excellent book-length treatments (the final reference being the most approachable for non-mathematicians).

While modern, principled missing data treatments can solve the majority of missing data problems, a number of practical difficulties still remain when implementing missing data analyses. Because principled missing data treatments require a model of the nonresponse, their performance can be adversely affected by misspecification of this model. When using FIML to treat nonresponse, ensuring an adequate model for the missingness is relatively simple. Because FIML partitions the missingness out of the likelihood during model estimation, using the *saturated correlates approach* (Graham, 2003) to include any important predictors of the nonresponse mechanism will usually suffice. However, FIML cannot be used in all circumstances. In psychological research, there are two very common situations where FIML is inapplicable. The first such situation occurs when the raw data must be collapsed into composite scores (e.g., scale scores or parcels) before analysis (Enders, 2010). The second situation arises when the data

analyst employs a modeling scheme that does not allow ML estimation methods (e.g., ordinary least squares regression, decision tree modeling, back-propagated neural networks). In these situations, as well as any time that the data analyst simply desires a "completed" data set, MI is the method of choice. MI also supports sensitivity analyses more easily than FIML does, so MI will likely be preferred to FIML when the tenability of the *missing at random* (MAR) assumption is questionable and must be explored via sensitivity analysis (Carpenter & Kenward, 2012).

In most versions of MI, the missing data are described by a discriminative linear model (usually a Bayesian generalized linear model [GLM]). Thus, correctly specifying a model for the missing data (i.e., the *imputation model*) is analogous to specifying any GLM and requires parameterizing three components. (1) A systematic component: the conditional mean of the missing data which is usually taken to be a linear combination of some set of exogenous predictors. (2) A random component: the residual probability distribution of the missing data after partialing out the conditional mean. (3) A linking function to map the systematic component to the random component. To minimize the possibility of misspecifying the imputation model, the imputation scheme must satisfy four requirements. First, the distributional form assumed for the missing data (i.e., the random component, from the GLM perspective) must be a "close" approximation to the true distribution of the missing data (Schafer, 1997). Second, all important predictors of the missing data and the nonresponse mechanism must be included as predictors in the imputation model (Collins, Schafer, & Kam, 2001; Rubin, 1996). Third, all important nonlinearities (i.e., interaction and polynomial terms) must be included in the imputation model (Graham, 2009; Von Hippel, 2009). Fourth, the imputation model should not be *over-specified*. That is, the systematic component should not contain extraneous predictors that do not contribute explanatory power to the imputation process (Graham, 2012; van Buuren, 2012).

In practice, the first point is often of little concern since the multivariate normal distribution is a close enough approximation for many missing data problems (Honaker & King, 2010; Schafer, 1997; Wu, Jia, & Enders, 2014), and the MICE framework makes it easy to swap in other distributions on a variable-by-variable basis (van Buuren, 2012; van Buuren & Groothuis-

Oudshoorn, 2011). The last three points, however, cannot be side-stepped so easily. If the imputation model fails to reflect important characteristics of the relationship between the missing data and the rest of the data set, the final inferences can be severely compromised (Barcena & Tusell, 2004; Drechsler, 2010; Honaker & King, 2010; Von Hippel, 2009). Including useless predictors is also problematic as they cannot improve the quality of the imputations but will decrease the precision of the imputed values by adding noise to the fitted imputation model (Von Hippel, 2007).

Correctly parameterizing the imputation model (i.e., satisfying the four requirements describe above) remains one of the most challenging issues facing missing data analysts. This difficulty is necessarily exacerbated in situations where the number of variables exceeds the number of observations—the so-called $P > N$ problem. Such problems imply a system of equations with more unknown variables than independent equations and are said to have *deficient rank*. Readers with some exposure to linear algebra will recall that such systems do not have a unique solution. Traditionally, such *underdetermined* systems were not common in psychological research, but they are becoming more prevalent with the increasing availability of *big-data* sources. Such problems commonly arise when conducting secondary analyses of publicly available databases, for example, particularly in medical and health-outcomes research where a very large number of attributes are often tracked for a comparatively small number of patients. The push towards interdisciplinary research will also expose more psychologists to disciplines where $P > N$ problems are common (such situations are the rule, rather than the exception, in genomics, for example). In the case of missing data analysis, there is another mechanism that can tip otherwise well behaved problems into the $P > N$ situation. If the incomplete data set contains nearly as many variables as observations, the process of including all of the interaction and polynomial terms necessary to correctly model the nonresponse may push the number of predictors in the imputation model higher than the number of observations. Until quite recently, missing data analysts faced with such degenerate cases had very few, principled, tools to apply. However, new developments in regularized regression modeling offer tantalizing possibilities for

robust solutions to this persistent issue.

This dissertation will introduce one such solution: *Multiple Imputation with the Bayesian Elastic Net* (MIBEN). The MIBEN algorithm is a robust multiple imputation scheme that augments the Bayesian elastic net due to Li and Lin (2010) and employs it as the elementary imputation method underlying a novel implementation of *multiple sequential regression imputation* (MSRI). The MIBEN algorithm has been developed specifically to address the difficulties inherent in parameterizing good imputation models. By incorporating both automatic variable selection to pare down large pools of auxiliary variables and model regularization to stabilize the estimation and reduce spurious variability in the imputations, the MIBEN approach has been designed as a very stable imputation platform.

1.1 Notational & Typographical Conventions

This paper contains many references to statistical software packages, and it relies heavily on mathematical notation to clarify the exposition. Therefore, before continuing with the substantive discussion, I will define the notational conventions that will be employed for subsequent mathematical copy as well as the typographic conventions that I will use when discussing computer software.

Scalar-valued realizations of random variables will be represented with lower-case Roman letters (e.g., x, y). Vector-value realizations of random variables will be represented by bold-faced, lower-case Roman letters (e.g., $\mathbf{x}$, $\mathbf{y}$). These vectors are assumed to be column-vectors unless specifically denoted as row-vectors in context. Matrices containing multiple observations of vector-valued random variables will be represented by bold-faced, upper-case Roman letters (e.g., $\mathbf{X}$, $\mathbf{Y}$). Unobserved, population-level spaces from which these random variables are realized will be represented by upper-case Roman letters in Gothic script (e.g., $\mathfrak{X}$, $\mathfrak{Y}$). Unknown model parameters will be represented by lower-case Greek letters (e.g., μ, β, θ, ψ), while vectors of such parameters will be represented by bold-faced lower-case Greek letters (e.g., $\boldsymbol{\mu}$, $\boldsymbol{\beta}$).

Where convenient, matrices of unknown model parameters will be represented by capital Greek letters (e.g., Θ, Ψ). Estimated model parameters will be given a "hat" (e.g., $\hat{\mu}$, $\hat{\beta}$). Unless otherwise specified, all data sets will be represented as $N \times P$ rectangular matrices in *Observations* $\times$ *Variables* format with $n = 1, 2, \ldots, N$ indexing observations and $p = 1, 2, \ldots, P$ indexing variables. For the remainder of this paper the terms *observation*, *subject*, and *participant* will be used interchangeably as will the terms *variable* and *attribute*.

Several probability density functions (PDFs) will be employed repeatedly in the following derivations, so it is convenient to describe their notation here. $\text{N}(\mu, \sigma^2)$ represents the univariate normal (Gaussian) distribution with mean μ and variance σ^2, $\text{MVN}(\boldsymbol{\mu}, \Sigma)$ represents the multivariate normal (Gaussian) distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ, $\text{Unif}(a, b)$ represents the the uniform distribution on the closed interval $[a, b]$, and $\Gamma(k, \theta)$ represents the gamma distribution with shape k and scale θ. All other mathematical notation (or modifications to the conventions specified above) will be defined in context.

When discussing computer software, references to entire programming languages will be denoted by the use of san-serif font (e.g., R, C++). References to software packages or libraries will be denoted by the use of bold-faced font (e.g., **mice**, **Eigen**). Finally, inline listings of program syntax and references to individual functions will be denoted with the use of typewriter font (e.g., `foo <- 3.14`, `quickpred`).

1.2 Regularized Regression Models

A very important concept in statistical modeling is the idea of model regularization or penalized estimation. Data analysts are always naïve to the *true* model and are often faced with a large pool of potential explanatory variables and little a priori guidance as to which are most "important." In these situations, it is critical that the model (or the model search algorithm) be able to balance the *bias-variance trade-off* and keep the estimator from wandering into the realm of high variance solutions that overfit the observed data at the expense of replicability and validity.

One of the most common methods for achieving this aim in statistical modeling is to include a *penalty term* into the objective function being optimized. The purpose of this penalty term is to bias the fitted solution towards simpler models by increasing the value of the objective function (in the case of minimization) by a number that is proportional to the model complexity (usually a function of the number of estimated parameters or their magnitude). Although many (if not most) statistical modeling methods can be viewed as entailing some form of regularization, there are three particularly germane extensions of ordinary least-squares (OLS) regression that make the regularization especially salient, namely, ridge regression, LASSO, and the elastic net.

1.2.1 Ridge Regression

Consider linear models of the form $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ where $\mathbf{y}$ is a column vector containing N observations of a scalar-valued dependent variable, $\mathbf{X}$ is a $N \times P$ matrix of independent variables, $\boldsymbol{\beta}$ is a column vector containing P regression coefficients, and $\boldsymbol{\varepsilon} \sim \mathrm{N}(0,\ \sigma^2)$ is a column vector of N normally distributed error terms. *Ridge Regression* (which is also known as ℓ_2-penalized regression due to the form of its penalty term) is a widely implemented form of regularized regression that can be applied to models of this form. It was originally proposed by Hoerl and Kennard (1970) who were seeking a method to mitigate the effects of multicollinearity in multiple linear regression models. To do so, they developed a penalized estimator that decreased the variance of the fitted solutions (i.e., mitigated the "bouncing beta weights" problem), but did so at the expense of no longer producing a best linear unbiased estimator, which is a well known, and highly desirable, property of traditional OLS regression. Thus, ridge regression is a classic example of manipulating the bias-variance trade-off. Incorporating the ridge penalty produces a biased solution (to a degree that the analyst can control), but doing so often yields considerably better real-world results in terms of prediction accuracy and out-of-sample validity (Hastie, Tibshirani, & Friedman, 2009).

The implementation of ridge regression is a straight-forward extension of traditional OLS

regression. To illustrate, recall the residual sum of squares loss function used in OLS regression:

$$RSS_{ols} = \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^T \hat{\boldsymbol{\beta}} \right)^2, \tag{1.1}$$

where N is the total sample size, y_n is the (centered) outcome for the nth observation, $\mathbf{x}_n = \{x_{n1}, x_{n2}, \ldots, x_{nP}\}^T$ is a P-vector of (standardized) predictor values for the nth observation, and $\hat{\boldsymbol{\beta}} = \{\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_P\}^T$ is a P-vector of fitted regression coefficients. By minimizing Equation 1.1, OLS regression produces fitted coefficients of the form:

$$\hat{\boldsymbol{\beta}}_{ols} = \left(\mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{y}, \tag{1.2}$$

where $\hat{\boldsymbol{\beta}}_{ols}$ is a P-vector of OLS regression coefficients. To implement ridge regression, Equation 1.1 is extended by adding the squared ℓ_2-norm of the regression coefficients as a penalty term:

$$RSS_{\ell_2} = \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^T \hat{\boldsymbol{\beta}} \right)^2 + \lambda \sum_{p=1}^{P} \hat{\beta}_p^2, \tag{1.3}$$

where λ is a tuning parameter that dictates how strongly the solution is biased towards simpler models, $\hat{\beta}_p$ is the pth fitted regression coefficient, and the last term in the equation is the squared ℓ_2-norm of the regression coefficients (i.e., $\| \hat{\boldsymbol{\beta}} \|_2^2 = \sum_{p=1}^{P} (\hat{\beta}_p - \bar{\beta})^2 = \sum_{p=1}^{P} \hat{\beta}_p^2$). By minimizing Equation 1.3, ridge regression produces regularized coefficients of the form:

$$\hat{\boldsymbol{\beta}}_{\ell_2} = \left(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_P \right)^{-1} \mathbf{X}^T \mathbf{y}, \tag{1.4}$$

where $\mathbf{I}_P$ is the $P \times P$ identity matrix. Examination of Equation 1.4 clarifies how the ridge penalty addresses multicollinearity. The ridge penalty has the effect of adding a small constant value λ to each diagonal element of the cross-products matrix of the predictors $\mathbf{X}^T \mathbf{X}$. So, in situations with severe multicollinearity, when the determinant of $\mathbf{X}^T \mathbf{X}$ equals zero (within computer precision), the ridge penalty "tricks" the fitting function into thinking that this determinant is nonzero. The

cross-products matrix can then be inverted, and the estimation becomes tractable once more.

1.2.2 The LASSO

Ridge regression is a very effective and powerful regularization technique that performs particularly well when the most salient problem is multicollinearity (Dempster, Schatzoff, & Wermuth, 1977). However, ridge regression does little to address another common goal of regularized modeling: variable selection. Thus, ridge regression may perform poorly when the number of predictors is large relative to the number of observations, especially when the true solution is *sparse*. That is, when many of the predictors have no association with the outcome. In such circumstances, ridge regression shrinks the coefficients of unimportant predictors towards zero and produces a tractable estimation problem, but it must still allot some nonzero value to each coefficient. Thus, useless predictors remain in the model (Hastie et al., 2009).

In an attempt to improve the performance of regularized regression with sparse models, Tibshirani (1996) developed the *Least Absolute Shrinkage and Selection Operator* (LASSO). Implementing the LASSO technique is very similar to implementing ridge regression in that a penalty term is simply added to the usual OLS loss function. However, the LASSO employs the ℓ_1-norm of the regression coefficients as its penalty term (thus, it is also known as ℓ_1-penalized regression). While this difference may seem like a small distinction, it brings a considerable advantage: the LASSO will force the coefficients of unimportant predictors to *exactly* equal zero. Thus, the LASSO can perform an automatic variable selection and intuitively address model sparsity.

As with ridge regression, the LASSO is implemented via a simple modification of Equation 1.1 that results in the following penalized loss function:

$$RSS_{\ell_1} = \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^T \hat{\boldsymbol{\beta}}\right)^2 + \lambda \sum_{p=1}^{P} \left|\hat{\beta}_p\right|, \tag{1.5}$$

where the last term now represents the ℓ_1-norm of the regression coefficients (i.e., $\|\hat{\boldsymbol{\beta}}\|_1 = \sum_{p=1}^{P} |\hat{\beta}_p - \bar{\beta}| = \sum_{p=1}^{P} |\hat{\beta}_p|$). An unfortunate consequence of replacing the squared ℓ_2-norm with

the ℓ_1-norm is that there is no longer a closed-form solution for $\hat{\boldsymbol{\beta}}_{\ell_1}$. Thus, LASSO models must be estimated by minimizing Equation 1.5 iteratively via quadratic programming procedures.

1.2.3 The Elastic Net

The LASSO demonstrates certain advantages over traditional ridge regression, but it entails its own set of limitations. For researchers facing $P > N$ scenarios, one of the LASSO's biggest limitations is that it cannot select more nonzero coefficients than observations, thus there is an artificially impose upper bound on the number of "important" predictors that can be included in the model (Tibshirani, 1996). While this limitation is usually trivial, in certain circumstances this cap on the allowable number of predictors may lead the fitted model to poorly represent the data. One answer to this limitation (and the inability of ridge regression to produce sparse solutions) is the *Elastic Net*. The elastic net was introduced by Zou and Hastie (2005) as a compromise between the ridge and LASSO options. The elastic net incorporates both an ℓ_1 and a squared ℓ_2 penalty term. By doing so, the elastic net produces sparse solutions, but it also addresses multicollinearity in a more reasonable manner by tending to select groups of highly correlated variables to be included in or excluded from the model simultaneously. The elastic net can also produce solutions with more non-zero coefficients than observations.

The elastic net is also implemented by modifying Equation 1.1. In this case, by incorporating both the ℓ_1 and squared ℓ_2 penalties to produce the following loss function:

$$RSS_{enet} = \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^T \hat{\boldsymbol{\beta}}\right)^2 + \lambda_2 \sum_{p=1}^{P} \hat{\beta}_p^2 + \lambda_1 \sum_{p=1}^{P} \left|\hat{\beta}_p\right|, \tag{1.6}$$

where λ_2 corresponds to the *ridge* penalty parameter and λ_1 corresponds to the *LASSO* penalty parameter. In the original implementation by Zou and Hastie (2005), λ_1 and λ_2 were chosen sequentially with a grid-based cross-validation procedure. Their method entailed choosing a range of values for one of the parameters and finding the conditionally optimal value for the other parameter by K-fold cross-validation. Choosing the penalty parameters with this method,

and minimizing Equation 1.6 after conditioning on the optimal values of λ_1 and λ_2, produces the so-called *naïve elastic net*. Empirical evidence suggests that the naïve elastic net over-shrinks the fitted regression coefficients due to the sequential method by which the penalty parameters are chosen. So, Zou and Hastie (2005) suggested a correction factor for the naïve estimates. These corrected estimates then represent the genuine elastic net. The suggested correction is given by:

$$\boldsymbol{\beta}_{enet} = (1 + \lambda_2)\,\hat{\boldsymbol{\beta}}_{naïve}, \tag{1.7}$$

where $\boldsymbol{\beta}_{enet}$ and $\hat{\boldsymbol{\beta}}_{naïve}$ are P-vectors of fitted regression coefficients for the elastic net and naïve elastic net, respectively.

1.3 Bayesian Model Regularization

As discussed above, Frequentist model regularization operates by including a penalty term into the loss function to add a "cost" for increasing the model's complexity. There is a direct analog to this concept in Bayesian modeling. From the Bayesian perspective, model regularization implies a prior belief that the true model's parameters are somehow bounded or that some take trivial values (i.e., the true model is sparse). The Bayesian can impose a preference for simpler models (both in terms of sparsity and coefficient regularization) by giving the regression coefficients informative priors. The scales of these prior distributions play an analogous role to the penalty parameters λ_1 and λ_2 in the Frequentist models. Through carefully tailored priors, Bayesian analogs of ridge regression, LASSO, and the elastic net have all been developed.

1.3.1 Bayesian Ridge & LASSO

Ridge regression is actually a somewhat trivial case of model regularization from the Bayesian perspective. This triviality arises from the fact that a ridge-like penalty can be incorporated simply by giving the regression coefficients informative, zero-mean Gaussian prior distributions (Goldstein, 1976). The smaller the variance of these prior distributions, the larger the ridge-type

penalty on the posterior solution. The Bayesian analog to a LASSO penalty is achieved by giving each regression coefficient a zero-mean Laplacian (i.e., double exponential) prior distribution (Gelman et al., 2013). However, Park and Casella (2008) showed that naïvely incorporating such a Laplacian prior can induce a multi-modal posterior distribution. They went on to develop an alternative formulation of the Bayesian LASSO that incorporated a conditional prior for the regression coefficients that depended on the noise variance. They proved that this formulation will produce uni-modal posteriors in typical situations (see Park & Casella, 2008, Appendix A).

1.3.2 Bayesian Elastic Net

Bayesian formulations of the elastic net rely on prior distributions that combine characteristics of both the Gaussian and Laplacian distributions (just as the Frequentist elastic net employs both ℓ_2- and ℓ_1-penalties). Several authors have developed flavors of such a prior. Some of these formulations are relatively complicated like the one employed in the *multiple Bayesian elastic net* (MBEN; Yang, Dunson, & Banks, 2011). The MBEN incorporates a Dirichlet process into the prior for the regression coefficients to group and shrink them towards multiple values. Other authors have developed elastic net priors tailored to specific applications, such as vector auto-regressive modeling (Gefang, 2014) and signal compression (Cheng, Mao, Tan, & Zhan, 2011).

1.3.2.1 Implementation of the Bayesian Elastic Net

While more complicated formulations of the elastic net prior can have certain advantages (e.g., shrinkage towards multiple values rather than just zero), the method introduced here will employ the prior developed by Li and Lin (2010). Theirs was one of the earliest formulations of a Bayesian elastic net (BEN), and it represents a relatively straight-forward extension of the Park and Casella (2008) Bayesian LASSO. The Li and Lin (2010) BEN was motivated by the observation, made by Zou and Hastie (2005), that the original elastic net solution is equivalent to finding the marginal

posterior mode $\boldsymbol{\beta}|\mathbf{y}$ when the regression coefficients are given the following prior:

$$\pi(\boldsymbol{\beta}) \propto \exp\left\{-\lambda_1\|\boldsymbol{\beta}\|_1 - \lambda_2\|\boldsymbol{\beta}\|_2^2\right\}, \tag{1.8}$$

where $\|\boldsymbol{\beta}\|_1$ and $\|\boldsymbol{\beta}\|_2^2$ represent the ℓ_1- and squared ℓ_2-norm of the regression coefficients, respectively. Combining this intuition with an uninformative prior for the noise variance and an extension of the Park and Casella (2008) conditional prior for the regression coefficients, Li and Lin (2010) began their development with the following hierarchical representation of the BEN:

$$\mathbf{y} \mid \boldsymbol{\beta}, \sigma^2 \sim \mathrm{N}\left(\mathbf{X}\boldsymbol{\beta},\ \sigma^2\mathbf{I}_N\right), \tag{1.9}$$

$$\boldsymbol{\beta} \mid \sigma^2 \sim \exp\left\{-\frac{1}{2\sigma^2}\left(\lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\boldsymbol{\beta}\|_2^2\right)\right\}, \tag{1.10}$$

$$\sigma^2 \sim \frac{1}{\sigma^2}, \tag{1.11}$$

where N represents the number of observations, $\mathbf{I}_N$ represents the $N \times N$ identity matrix, and Equation 1.11 denotes an improper prior distribution for the noise variance. Although this formulation is conceptually appealing due to its direct correspondence to the original elastic net, Li and Lin (2010) noted that the absolute values in Equation 1.10 lead to unfamiliar posterior distributions. So, to facilitate Gibbs sampling from the fully conditional posteriors, they introduced an auxiliary parameter $\boldsymbol{\tau}$ which leads to an alternative parameterization of the model given above:

$$\mathbf{y} \mid \boldsymbol{\beta}, \sigma^2 \sim \mathrm{N}\left(\mathbf{X}\boldsymbol{\beta},\ \sigma^2\mathbf{I}_n\right), \tag{1.12}$$

$$\boldsymbol{\beta} \mid \boldsymbol{\tau}, \sigma^2 \sim \prod_{p=1}^{P} \mathrm{N}\left(0,\ \left(\frac{\lambda_2}{\sigma^2} \cdot \frac{\tau_p}{\tau_p - 1}\right)^{-1}\right), \tag{1.13}$$

$$\boldsymbol{\tau} \mid \sigma^2 \sim \prod_{p=1}^{P} \text{Trunc-}\Gamma\left(\frac{1}{2},\ \frac{8\lambda_2\sigma^2}{\lambda_1^2},\ (1, \infty)\right), \tag{1.14}$$

$$\sigma^2 \sim \frac{1}{\sigma^2}, \tag{1.15}$$

where P represents the number of predictors in the model and Equation 1.14 represents the truncated gamma distribution with support on the open interval $(1, \infty)$. Introducing $\boldsymbol{\tau}$ simplifies the computations by removing the need to explicitly incorporate the ℓ_1-norm into any of the priors. The fully conditional posterior distributions of the BEN's parameters are then given by:

$$\begin{aligned} \boldsymbol{\beta} \mid \mathbf{y}, \sigma^2, \boldsymbol{\tau} &\sim \text{MVN}\left(\mathbf{A}^{-1}\mathbf{X}^T\mathbf{y},\ \sigma^2\mathbf{A}^{-1}\right), \\ &\text{with } \mathbf{A} = \mathbf{X}^T\mathbf{X} + \lambda_2 \cdot \text{diag}\left(\frac{\tau_1}{\tau_1 - 1},\ \ldots,\ \frac{\tau_P}{\tau_P - 1}\right), \end{aligned} \tag{1.16}$$

$$\frac{1}{\left(\tau_p - 1\right)} \mid \mathbf{y}, \sigma^2, \boldsymbol{\beta} \sim \text{IG}\left(\mu = \frac{\sqrt{\lambda_1}}{2\lambda_2|\beta_p|},\ \lambda = \frac{\lambda_1}{4\lambda_2\sigma^2},\right),\ \ p = 1,\ 2,\ \ldots,\ P, \tag{1.17}$$

$$\begin{aligned} \sigma^2 \mid \mathbf{y}, \boldsymbol{\beta}, \boldsymbol{\tau} &\sim \left(\frac{1}{\sigma^2}\right)^{\frac{N}{2}+P+1} \left\{\Gamma_U\left(\frac{1}{2},\ \frac{\lambda_1^2}{8\sigma^2\lambda_2}\right)\right\}^{-P} \exp\left(-\frac{1}{2\sigma^2} \cdot \xi\right), \\ &\text{with } \xi = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_2 \sum_{p=1}^{P} \frac{\tau_p}{\tau_p - 1}\beta_p^2 + \frac{\lambda_1^2}{4\lambda_2} \sum_{p=1}^{P} \tau_p, \end{aligned} \tag{1.18}$$

where $\Gamma_U(\alpha, x) = \int_x^\infty t^{\alpha-1}e^{-t}dt$ represents the upper incomplete gamma function and Equation 1.17 represents the inverse Gaussian distribution with a PDF as given by Chhikara and Folks (1988). Clearly, the conditional posterior distribution of σ^2 does not follow any familiar functional form, but it can be sampled via a relatively simple rejection sampling scheme. Li and Lin (2010) noted that the expression on the right hand side of Equation 1.18 is bounded above by:

$$\Gamma\left(\frac{1}{2}\right)^{-P} \left(\frac{1}{\sigma^2}\right)^{a+1} \exp\left\{-\frac{1}{\sigma^2}b\right\}, \tag{1.19}$$

$$\text{with } a = \frac{N}{2} + P,\ \ b = \frac{1}{2}\left[\left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^T\left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right) + \lambda_2 \sum_{p=1}^{P} \frac{\tau_p}{\tau_p - 1}\beta_p^2 + \frac{\lambda_1^2}{4\lambda_2} \sum_{p=1}^{P} \tau_p\right].$$

Leveraging this relationship, they suggested the procedure described by Algorithm 1 to draw variates from Equation 1.18. Given the preceding specification, the joint posterior distribution

Algorithm 1 Rejection Sampling of σ^2

loop
  Draw: A candidate variate Z
    $Z \sim \text{Inv-}\Gamma(a, b)$ with a and b as in Equation 1.19
  Draw: A threshold variate U
    $U \sim \text{Unif}(0, 1)$
  if $\ln(U) \leq P \cdot \ln\left(\Gamma\left(\frac{1}{2}\right)\right) - P \cdot \ln\left\{\Gamma_U\left(\frac{1}{2}, \frac{\lambda_1^2}{8Z\lambda_2}\right)\right\}$ **then**
    $\sigma^2 \leftarrow Z$
    break
  else
    goto 2
  end if
end loop

of the BEN can be estimated by incorporating the sampling statements represented by Equations 1.16–1.18 into a Gibbs sampling scheme with block updating of $\boldsymbol{\beta}$ and $\boldsymbol{\tau}$.

Choosing the Penalty Parameters. While it is possible to choose values for λ_1 and λ_2 via cross-validation (as with the Frequentist elastic net), the Bayesian paradigm offers at least two, superior, alternatives. First, the penalty parameters can be added into the model hierarchy as hyper-parameters and given their own hyper-priors. Li and Lin (2010) suggested $\lambda_1^2 \sim \Gamma(a,\ b)$ and $\lambda_2 \sim \text{GIG}(\lambda = 1,\ \psi = c,\ \chi = d)$, where $\text{GIG}(\lambda,\ \psi,\ \chi)$ is the generalized inverse Gaussian distribution as given by Jørgensen (1982). Since these priors maintain conjugacy, λ_1 and λ_2 can then be directly incorporated into the Gibbs sampler. However, Li and Lin (2010) noted that the posterior solutions can be highly sensitive to the choice of (a, b) and (c, d). Therefore, the approach that they actually employ for the BEN (as well as the method recommended by Park & Casella, 2008, for the Bayesian LASSO) is the empirical Bayes Gibbs sampling method described by Casella (2001). This empirical Bayes method estimates the penalty parameters with Monte Carlo EM (MCEM) marginal maximum likelihood in which the expectations needed to specify the conditional log-likelihood are approximated by the averages of the stationary Gibbs samples. Both Li and Lin (2010) and Park and Casella (2008) suggested that this approach, while more computationally expensive, produces equivalent results to the augmented Gibbs sampling

approach. For the Li and Lin (2010) formulation of the BEN, the appropriate conditional log-likelihood (ignoring terms that are constant with respect to λ_1 and λ_2) is given by:

$$\begin{aligned} Q\left(\Lambda \mid \Lambda^{(i-1)}\right) &= P \cdot \ln(\lambda_1) - P \cdot E\left[\ln\left\{\Gamma_U\left(\frac{1}{2}, \frac{\lambda_1^2}{8\sigma^2\lambda_2}\right)\right\} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right] \\ &\quad - \frac{\lambda_2}{2}\sum_{p=1}^{P} E\left[\frac{\tau_p}{\tau_p - 1}\cdot\frac{\beta_p^2}{\sigma^2} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right] - \frac{\lambda_1^2}{8\lambda_2}\sum_{p=1}^{P} E\left[\frac{\tau_p}{\sigma^2} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right] \\ &\quad + constant, \\ &= R\left(\Lambda \mid \Lambda^{(i-1)}\right) + constant, \end{aligned} \tag{1.20}$$

and the gradient is given by:

$$\begin{aligned} \frac{\delta R}{\delta \lambda_1} &= \frac{P}{\lambda_1} + \frac{P\lambda_1}{4\lambda_2} E\left[\left\{\Gamma_U\left(\frac{1}{2}, \frac{\lambda_1^2}{8\sigma^2\lambda_2}\right)\right\}^{-1} \varphi\left(\frac{\lambda_1^2}{8\sigma^2\lambda_2}\right)\frac{1}{\sigma^2} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right] \\ &\quad - \frac{\lambda_1}{4\lambda_2}\sum_{p=1}^{P} E\left[\frac{\tau_p}{\sigma^2} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right], \end{aligned} \tag{1.21}$$

$$\begin{aligned} \frac{\delta R}{\delta \lambda_2} &= -\frac{P\lambda_1^2}{8\lambda_2^2} E\left[\left\{\Gamma_U\left(\frac{1}{2}, \frac{\lambda_1^2}{8\sigma^2\lambda_2}\right)\right\}^{-1} \varphi\left(\frac{\lambda_1^2}{8\sigma^2\lambda_2}\right)\frac{1}{\sigma^2} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right] \\ &\quad - \frac{1}{2}\sum_{p=1}^{P} E\left[\frac{\tau_p}{\tau_p - 1}\cdot\frac{\beta_p^2}{\sigma^2} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right] + \frac{\lambda_1^2}{8\lambda_2^2}\sum_{p=1}^{p} E\left[\frac{\tau_p}{\sigma^2} \,\middle|\, \Lambda^{(i-1)}, \mathbf{Y}\right], \end{aligned} \tag{1.22}$$

where $\varphi(t) = t^{-\frac{1}{2}}e^{-t}$, i indexes iteration of the MCEM algorithm, $\Lambda = \{\lambda_1, \lambda_2\}$, $\mathbf{Y} = \{\mathbf{y}, \mathbf{X}\}$, and *constant* represents a collection of terms that do not involve λ_1 or λ_2.

1.3.2.2 Performance of the Bayesian Elastic Net

Li and Lin (2010) used a Monte Carlo simulation study to compare their BEN to the Park and Casella (2008) Bayesian LASSO, as well as the original, Frequentist elastic net and LASSO. They found that the two Bayesian approaches consistently outperformed the Frequentist approaches in terms of prediction accuracy, although the Bayesian versions were much more computation-

ally demanding than their Frequentest analogs were. The BEN and Bayesian LASSO performed similarly in many conditions, but the BEN was the superior method for small sample sizes and when the true model was not especially sparse.

1.4 Model Regularization for Missing Data Analysis

Although many data analysts may not realize it, regularized regression models are ubiquitous in missing data analysis. This ubiquity stems from the fact that nearly all of the normal-theory regression models used by current imputation software are actually ridge regression models. Since well specified imputation models will likely contain many predictors (Howard, Rhemtulla, & Little, in press; Rubin, 1996; Von Hippel, 2009), multicollinearity can become a serious issue, and the increased indeterminacy introduced by nonresponse only exacerbates the problem. Thus, most software packages for imputation offer the option to include a "ridge prior" to regularize the cross-products matrix of the predictors in the imputation model. When the analyst invokes the `prior = ridge(`λ`)` option in SAS **PROC MI**, the `empri =` λ option in the R package **Amelia II**, or the `ridge =` λ option in the R package **mice** the value chosen for λ is proportional to the ridge parameter in Equations 1.3 and 1.4. In my experience, including such a prior is often necessary to ensure stable convergence when creating multiple imputations—especially when using an imputation method that employs the joint modeling paradigm.

The LASSO has not been applied to missing data analyses nearly as widely as ridge regression has, but a few authors have considered it as an imputation engine. The R package **imputeR** (Feng, Nowak, Welsh, & O'Neill, 2014) includes the capability to conduct a single deterministic imputation using the Frequentist LASSO. This method is significantly limited, though, since it does not model uncertainty in the imputed data or the imputation model. A much more principled implementation, based on the Bayesian LASSO, was developed by Zhao and Long (2013). They developed an MI scheme in which the Bayesian LASSO model was first trained on the observed portion of the data and the missing values were then replaced by M random draws

from the posterior predictive distribution of the outcome. In this way, they achieved a fully principled MI method in which uncertainty in both the missing data and the imputation model were accounted for via Bayesian simulation. They also described how to use their method to treat general missing data patterns via data augmentation and sequential regression imputation.

Zhao and Long (2013) compared their Bayesian LASSO-based MI scheme to several methods based on frequentist regularized regression models, namely, the LASSO, the adaptive LASSO, and the elastic net. They incorporated these Frequentist methods into MI schemes in which the uncertainty was modeled by first creating M bootstrap resamples of the incomplete data, and then fitting the regularized model to the observed portion of the (resampled) data. Once the model moments had been estimated, they filled-in the missingness (in the original, un-resampled data) with the corresponding elements of the M sets of model implied outcomes (this general framework also underlies the *bootstrapped EM* algorithm employed by the R package **Amelia II**). They also implemented a strategy in which the regularized models were simply used for variable selection and the imputation was subsequently performed with standard, normal-theory MI. They found that the Bayesian LASSO-based MI method generally performed better than the other imputation schemes in terms of recovering the regression coefficients of models fit to the imputed data. Interestingly, they also found that their Bayesian LASSO-based method could outperform a normal-theory MI that used the true imputation model (which is unknowable in practice), when the number of predictors in the true model was relatively large.

At the time of this writing, it appears that the elastic net has received almost no attention as a missing data tool. Other than the comparison conditions employed by Zhao and Long (2013), I was unable to find any published research using the elastic net for imputation. Furthermore, I was unable to find any papers, at all, that employ a fully Bayesian construction of the elastic net for this purpose. This dissertation addresses this gap in the literature by introducing a flexible and robust MI method. In the following, I describe a novel MI algorithm that is based on the Li and Lin (2010) BEN and examine its performance via a Monte Carlo simulation study. The method I introduce extends the previous work done in this area in several important ways. First,

the basis of the algorithm is a very flexible and powerful model: the elastic net. Second, the implementation is fully Bayesian, thereby allowing for fully principled multiple imputations. Third, the algorithmic framework surrounding this model is more general and fully featured than that of other regularized regression-based imputation methods.

1.5 Multiple Imputation with the Bayesian Elastic Net

This dissertation introduces a novel MI scheme. The method, *Multiple Imputation with the Bayesian Elastic Net* (MIBEN), is a principled MI algorithm based on the Li and Lin (2010) BEN. MIBEN is a very flexible imputation tool that uses MSRI and data augmentation to treat general patterns of nonresponse under the assumption of a MAR nonresponse mechanism. MIBEN can treat an arbitrary number of incomplete variables and easily incorporates auxiliary variables (which can also contain missing data) into the imputation model. The MIBEN algorithm is designed to leverage the excellent prediction performance of the BEN to create optimal imputations without requiring the missing data analyst to manually select which variables to include in the imputation model. Thus, MIBEN is particularly well suited to situations where the data imputer is presented with a large pool of possible auxiliary variables but has little a priori guidance as to which may be important predictors of the missing data or the nonresponse mechanism. Because the BEN is optimized for $P > N$ problems, the MIBEN algorithm is also expected to perform better than currently available alternatives when employed with underdetermined systems.

In addition to the powerful imputation model underlying the MIBEN algorithm, there are several key features of the supporting framework considerably improve MIBEN's capabilities. Most importantly, the imputations are created through iterative data augmentation (Tanner & Wong, 1987). By including the missing data as another parameter in the Gibbs sampler, the imputations and the parameters of the imputation model are iteratively refined in tandem. This approach has three major advantages over other methods. First, it simplifies the treatment of general missing data patterns. Second, it ensures that the posterior predictive distribution of the nonresponse ac-

curately models all important sources of uncertainty (since uncertainty in the imputation model is conditioned on uncertainty in the imputed data and visa versa; Rubin, 1987). Finally, the iterative nature of the data augmentation process mitigates any spurious order-related effects that may be introduced by the sequential aspect of the MSRI algorithm. The MIBEN algorithm also uses a multi-stage MCEM algorithm that employs several of the computational tricks described by Casella (2001) and a robust two-step optimization of the BEN's penalty parameters. As shown below, this multi-stage MCEM method produces very good convergence properties.

1.5.1 Assumptions of the MIBEN Algorithm

The MIBEN method has been designed as a robust and flexible MI tool. However, MIBEN still places certain assumptions on the imputation model, so this flexibility does have limits. The implementation described here requires the following key assumptions:

1. Each imputed variable's residuals (i.e., conditioning on the fitted imputation model) are independent and identically normally distributed.

2. The imputation model is linear in the coefficients.

3. The missing data follow a MAR mechanism.

The conditional normality of the incomplete variables is not an inherent requirement of the MIBEN method since extending the BEN to accommodate categorical outcomes is a relatively straight-forward process. Chen et al. (2009) described a method of incorporating binary outcomes into the BEN via a probit transformation of the the raw model-implied outcomes. Although this method can be directly incorporated into the structure presented here, I have currently implemented only the normal-theory version.

MIBEN is a parametric technique and, therefore, is inherently less flexible than fully nonparametric imputation approaches (e.g., those based on K-nearest neighbors or decision trees), but it does allow the missing data analyst to relax certain key assumptions. Most notably, MIBEN is robust to over-specification of the imputation model. The performance of traditional MI methods

will deteriorate when useless, noise variables are included in the imputation model (van Buuren, 2012). MIBEN, on the other hand, will remain unaffected because the algorithm will simply eliminate any useless variables from the fitted imputation model. So long as the MAR assumption holds (thereby ensuring that all important predictors of the missing data are included on the data set), MIBEN should not be vulnerable to under-specification of the imputation model either. The automatic variable selection of the underlying BEN should automatically include all important auxiliary variables into the imputation model. Thus, MIBEN is expected to correctly specifying the predictor set of the imputation model without any overt input from the data imputer.

MIBEN also places very few restrictions on the matrix of auxiliary variables. Because the BEN is a discriminative model, there are no requirements placed on the distribution of the auxiliary variables (because their distribution is never modeled). Furthermore, the auxiliary variables only enter the imputation model through matrix multiplication, so all of their observed information can be easily incorporated by zero-imputing any of their missing data (i.e., to compute their crossproduct matrix with pairwise-available observations). Thus, accommodating nonresponse on the auxiliary variables requires only a trivial complication of the MIBEN algorithm. Most importantly, the powerful regularization of the BEN prior allows the imputation model's predictor matrix (and, by extension, the matrix of auxiliary variables) to have deficient rank (i.e., $P > N$) while still maintaining estimability and low-variance imputations.

1.6 Specification of the MIBEN Algorithm

The MIBEN algorithm directly employs the fully conditional posteriors given by Equations 1.16–1.18. However, to facilitate the imputation task, two additional sampling statements must be incorporated into the Gibbs sampler. First, the intercept is reintroduced to the model with an uninformative Gaussian prior. This leads to the following fully conditional posterior:

$$\alpha \sim \mathrm{N}\left(\bar{\mathbf{y}},\ \sqrt{\frac{\sigma^2}{N}}\right), \tag{1.23}$$

where $\bar{\mathbf{y}}$ represents the arithmetic mean of the variable being imputed. The original Bayesian elastic net omitted an intercept term because the data were centered before model estimation (thereby making an estimated intercept unnecessary). I have reintroduced the intercept here, however, to allow for the possibility of imputing values with a different conditional mean than that of the observed part of the data. Second, the imputations must be updated at each iteration of the Gibbs sampler. These updates are accomplished by replacing the missing values with random draws from their posterior predictive distribution according to the following rule:

$$\begin{aligned} \mathbf{y}_{imp}^{(i)} &= \tilde{\alpha}^{(i)}\mathbf{1}_P + \mathbf{X}\tilde{\boldsymbol{\beta}}^{(i)} + \boldsymbol{\varepsilon}, \\ \text{with } \boldsymbol{\varepsilon} &\sim \mathrm{N}\left(0,\ \widetilde{\sigma^2}^{(i)}\right), \end{aligned} \tag{1.24}$$

where $\mathbf{1}_P$ represents a P-vector of ones, $\boldsymbol{\varepsilon}$ is an N-vector of residual errors, the tildes designate their associated parameters as draws from the appropriate posterior distributions, and the (i) superscript indexes iteration of the Gibbs sampler. Incorporating these two additional sampling statements into the original hierarchy given by Li and Lin (2010) fleshes out all of the components needed for the MIBEN Gibbs sampler.

The overall MIBEN algorithm can be broken into three qualitatively distinct modules: (1) an initial data pre-processing module and (2) a Gibbs sampling module that is nested within (3) an MCEM module. The data pre-processing module takes an arbitrarily scaled, incomplete, rectangular data matrix and creates K *target objects* corresponding to the K target variables being imputed. For each target object, nuisance variables (e.g., ID variables) are removed, the focal target variable is mean-centered, and the remaining (predictor) variables are standardized.

Let $\mathbf{Y}_{inc}$ be an $N \times P$ rectangular data matrix that is subject to an arbitrary pattern of nonresponse. Without loss of generality, assume that the first K columns of $\mathbf{Y}_{inc}$ contain the variables to be imputed while the remaining $V = P - K$ columns contain auxiliary variables. For simplicity, assume that all nuisance variables have already been excluded from the $\mathbf{Y}_{inc}$. The pseudocode given in Algorithm 2 provides the conceptual details of the MIBEN data pre-processing module.

Algorithm 2 MIBEN Data Pre-Processing Module

Input: An incomplete data set $\mathbf{Y}_{inc}$
Output: K target objects $\left\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(K)}\right\}$
Define: $dvArray[K]$:= An empty array of vectors
Define: $predArray[K]$:= An empty array of matrices
Define: $\mathbf{Y}_{targets}$:= The first K columns of $\mathbf{Y}_{inc}$
Draw: $\mathbf{Y}_{imp,init} \sim \text{MVN}\left(\overline{\mathbf{Y}}_{targets},\ \text{cov}\left(\mathbf{Y}_{targets}\right)\right)$
for $n = 1$ to N **do**
  for $k = 1$ to K **do**
    if $\mathbf{Y}_{inc}[n,\ k] == MISSING$ **then**
      $\mathbf{Y}_{inc}[n,\ k] \leftarrow \mathbf{Y}_{imp,init}[n,\ k]$
    end if
  end for
  for $p = (K + 1)$ to P **do**
    if $\mathbf{Y}_{inc}[n,\ p] == MISSING$ **then**
      $\mathbf{Y}_{inc}[n,\ p] \leftarrow 0$
    end if
  end for
end for
for $k = 1$ to K **do**
  $dvArray[k] \leftarrow MeanCenter(\mathbf{Y}_{inc}[\ ,\ k])$
  $predArray[k] \leftarrow Standardize(\mathbf{Y}_{inc}[\ , \neg k])$
  $\mathcal{T}^{(k)} \leftarrow \{dvArray[k],\ predArray[k]\}$
end for
return K initialized target objects $\left\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(K)}\right\}$

After execution of Algorithm 2, each $\mathcal{T}^{(k)}$ contains data structures formatted for treatment via sequential regression imputation (i.e., each $\mathcal{T}^{(k)}$ contains the outcome variable and predictor set for a single conditional imputation equation). Given a set of K target objects constructed as above and taking L to be the number of MCEM iterations and J to be the number of Gibbs sampling iterations to employ within a particular iteration of the MCEM algorithm, the pseudocode given by Algorithm 3 shows the conceptual details of the MIBEN Gibbs sampler.

Algorithm 3 MIBEN Gibbs Sampling Module

Input: K initialized target objects $\left\{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \ldots, \mathcal{T}^{(K)}\right\}$
Output: Updated posterior estimates of all imputation model parameters and imputations
if $l == 1$ **then**
    Initialize: All parameters with draws from their respective prior distributions.
else
    Initialize: All parameters with their posterior expectations from MCEM iteration $l - 1$
end if
for $j = 1$ to J **do**
    for $k = 1$ to K **do**
        Set: $\mathbf{y} \leftarrow \mathcal{T}^{(k)}[[1]]$
        Set: $\mathbf{X} \leftarrow \mathcal{T}^{(k)}[[2]]$
        Update: $\boldsymbol{\tau}$ according to Equation 1.17
        Update: α according to Equation 1.23
        Update: $\boldsymbol{\beta}$ according to Equation 1.16
        Update: σ^2 by applying Algorithm 1
        Update: $\mathbf{y}_{imp}$ according to Equation 1.24
    end for
end for
Pre-Optimize: λ_1 and λ_2 by numerically maximizing Equation 1.20 with a derivative-free optimization method
Optimize: λ_1 and λ_2 by refining their pre-optimized estimate with a gradient-based optimization method employing the analytic gradient given by Equations 1.21 and 1.22
return Updated posterior estimates of $\boldsymbol{\tau}$, α, $\boldsymbol{\beta}$, σ, $\mathbf{y}_{imp}$, λ_1, and λ_2

As noted on Line 4 of Algorithm 3, initial starting values for all parameters are draws from their respective prior distributions. For parameters with informative priors (i.e., $\boldsymbol{\tau}$ and $\boldsymbol{\beta}$), these draws are taken from the appropriate components of the model hierarchy given by Equations 1.12–1.15. For parameters with uninformative priors (i.e., α and σ), however, the following data-dependent starting values were employed:

$$\alpha_{init}^{(k)} \sim \text{Unif}\left(-\sqrt{\frac{\text{Var}\left(\mathbf{y}^{(k)}\right)}{N_k}},\ \sqrt{\frac{\text{Var}\left(\mathbf{y}^{(k)}\right)}{N_k}}\right), \tag{1.25}$$

$$\sigma_{init}^{(k)} = \sqrt{\text{Var}\left(\mathbf{y}^{(k)}\right)}, \tag{1.26}$$

where $\mathbf{y}^{(k)}$ is the kth target variable, N_k is the number of non-missing observations of $\mathbf{y}^{(k)}$, and the Var$(\cdot)$ operator returns the variance of its argument. Starting values for the penalty parameters λ_1 and λ_2 are user-supplied. For the study reported below, I employed $\lambda_{1,init} = 0.5$ and $\lambda_{2,init} = P/10$, but empirical evidence suggests that the starting values of λ_1 and λ_2 have little effect on the estimation process unless these starting values are very different from their ML estimates. Such poorly chosen starting values will not corrupt MIBEN's estimates but will slow its convergence.

The final computational component of the MIBEN method is the MCEM module within which the Gibbs sampler described above is nested. The MCEM algorithm employed by MIBEN requires the expectations in Equations 1.20–1.22 to be approximated by the posterior means of the appropriate Gibbs samples. This substitution requires that the process described by Algorithm 3 be fully executed within each iteration of the MCEM algorithm (which can require several hundred iterations for difficult problems). Naturally, this leads to a very high computational burden, but Casella (2001) suggested several short-cuts that can considerably mitigate this computational demand.

Most importantly, Casella (2001) noted that, until the final few Gibbs samples, accurate estimates of Λ are not necessary. Thus, the speed of the MCEM algorithm can be dramatically increased by running a large number of "approximation" iterations in which very small Gibbs samples are simulated (e.g., with as few as 20 retained draws). Although these approximation

iterations are very noisy, they will rapidly bring the estimate of Λ into the neighborhood of its ML estimate. Once the estimate of Λ is within this neighborhood, a small number of "tuning" MCEM iterations can be run to "dial-in" this estimate using larger Gibbs samples.

This multi-stage approach is implemented in MIBEN. A large number of MCEM approximation iterations are run with very small Gibbs samples followed by a few tuning iterations with larger Gibbs samples and finally a single large Gibbs sample is drawn to represent the stationary posterior distribution of the imputation model parameters. So long as the estimates of the penalty parameters stabilize during the approximation and tuning phases of the MCEM algorithm, the Gibbs samples themselves only need to converge for the final iteration. Convergence of the MCEM estimates of Λ can be judged graphically by scrutinizing the trace plots of the Λ estimates. Upon convergence, these plots will randomly oscillate around an equilibrium level. Systematic linear or curvilinear trends in these trace plots indicate that the system has not yet converged on the optimal estimates of Λ. Convergence of the final Gibbs samples can be judged by a number of criteria; for the study reported here, I used the *potential scale reduction factor* ($\hat{R}$).

Take M to be the number of imputations to create, L_1 to be the number of MCEM approximation iterations and L_2 the number of MCEM tuning iterations, J_1 to be the number of Gibbs sampling iterations employed within each of the MCEM approximation iterations, J_2 to be the number of Gibbs sampling iterations used during the tuning phase of the MCEM algorithm, and J_3 be the number of Gibbs sampling iterations used to represent the stationary posterior distribution of the imputation model parameters. Then, the pseudocode presented in Algorithm 4 incorporates all of the MIBEN modules into the overall MIBEN algorithm.

Algorithm 4 MIBEN Algorithm

Input: An incomplete data set $\mathbf{Y}_{inc}$
Output: M completed data sets $\left\{\mathbf{Y}_{comp}^{(1)}, \mathbf{Y}_{comp}^{(2)}, \ldots, \mathbf{Y}_{comp}^{(M)}\right\}$
Execute: Algorithm 2
for $l = 1$ to L_1 **do** {MCEM Burn-In Iterations}
    Set: $J \leftarrow J_1$
    Execute: Algorithm 3
end for
for $l = 1$ to L_2 **do** {MCEM Tuning Iterations}
    Set: $J \leftarrow J_2$
    Execute: Algorithm 3
end for
Set: $j \leftarrow J_3$
Execute: Algorithm 3 {Approximate the stationary posterior}
Draw: M replicates of the missing data, $\left\{\mathbf{Y}_{imp}^{(1)}, \mathbf{Y}_{imp}^{(2)}, \ldots, \mathbf{Y}_{imp}^{(M)}\right\}$, from the stationary posterior predictive distribution of $\mathbf{Y}_{targets}$
Transform: $\left\{\mathbf{Y}_{imp}^{(1)}, \ldots, \mathbf{Y}_{imp}^{(M)}\right\} \rightarrow \left\{\mathbf{Y}_{imp,offset}^{(1)}, \ldots, \mathbf{Y}_{imp,offset}^{(M)}\right\}$ by adding back each target variable's mean {*un-center* the imputations}
return M completed data sets with missing elements of $\mathbf{Y}_{inc}$ replaced by corresponding elements of $\left\{\mathbf{Y}_{imp,offset}^{(1)}, \ldots, \mathbf{Y}_{imp,offset}^{(M)}\right\}$.

1.7 Hypotheses

The MIBEN method is expected to outperform current state-of-the-art techniques in terms of recovering the analysis model's true parameters. Several key characteristics of the BEN can inform the likely performance of the MIBEN method. First, the accuracy of the BEN's predictions is very good. So, the imputations produced by MIBEN should be unbiased, and, at worst, should be no more biased than imputations created from less-regularized methods. Second, the BEN is specifically designed to reduce the variance of its predictions. Thus, the imputations created by MIBEN should lead to more precise standard errors and narrower confidence intervals than those produced by less-regularized imputation methods. Third, because the penalty parameters λ_1 and λ_2 are estimated directly from the data, the BEN underlying MIBEN should be able to retain the meaningful variance in the predicted values but exclude most of the extraneous noise. Thus, MIBEN should produce confidence intervals with coverage rates that are approximately

nominal, and, at worst, it should produce coverage rates that are at least as close to nominal as the coverage rates produced by less-regularized methods. Fourth, because the elastic net is optimized for underdetermined systems, MIBEN's superior performance should be ever more salient when applied to $P > N$ and $P \approx N$ problems. Finally, because the Bayesian LASSO can perform poorly when the true model is not sparse, MIBEN should outperform Bayesian LASSO-based MI in conditions with dense imputation models. Appealing to these justifications, the following hypotheses are posed:

1. In all conditions, MIBEN will lead to parameter estimates that are negligibly biased and have confidence intervals with approximately nominal coverage rates.

2. In all conditions, the CIs derived from MIBEN will be narrower than those derived from Normal-theory MICE: (a) including all possible predictors, (b) including predictors chosen via best subset selection, and (c) employing the true imputation model

3. When the system is overdetermined, MIBEN will lead to parameter estimates that are no more biased than those estimated under Normal-theory MICE: (a) including all possible predictors, (b) including predictors chosen via best subset selection, and (c) employing the true imputation model

4. When the system is overdetermined, the coverage rates for CIs derived from MIBEN will be at least as close to nominal as those derived from Normal-theory MICE: (a) including all possible predictors, (b) including predictors chosen via best subset selection, and (c) employing the true imputation model

5. When the system is underdetermined, MIBEN will lead to parameter estimates that are less biased than those estimated under Normal-theory MICE: (a) employing the true imputation model and (b) including predictors chosen via best subset selection

6. When the system is underdetermined, the coverage rates for CIs derived from MIBEN will be closer to nominal than those derived from Normal-theory MICE: (a) employing the true

imputation model and (b) including predictors chosen via best subset selection

7. When the true imputation model is sparse and the system is overdetermined, MIBEN and Bayesian LASSO-based MI will lead to approximately equivalent parameter estimates.

8. When the true imputation model is sparse and the system is underdetermined, MIBEN will lead to parameter estimates that are superior to those estimated under Bayesian LASSO-based MI in terms of: (a) Bias and (b) Confidence Interval Coverage

9. When the true imputation model is not sparse, MIBEN will lead to parameter estimates that are superior to those estimated under Bayesian LASSO-based MI in terms of: (a) Bias and (b) Confidence Interval Coverage

Chapter 2

Methods

The performance of the MIBEN algorithm was assessed via a Monte Carlo simulation study. This study scrutinized the MIBEN algorithm by comparing it to several alternative MI methods in terms of its ability to recover the true, population-level, coefficients of a multiple linear regression model.

2.1 Experimental Design

The Monte Carlo simulation was broken into two experiments that each targeted distinct areas of the problem space. Experiment 1 was designed to give a detailed image of the MIBEN algorithm's performance in well-conditioned $P \ll N$ situations. Experiment 2, on the other hand, was designed to explore the performance of MIBEN in ill-conditioned $P \approx N$ and $P > N$ situations.

2.1.1 Simulation Parameters

Four design parameters were varied in the simulation: total sample size (N), proportion of missing data on the analysis variables (PM), number of potential auxiliary variables (V), and degree of sparsity in the true imputation model (DS). In both Experiments 1 and 2, two sparsity conditions were included: a sparse condition and a dense condition. In the dense condition all regression

coefficients took non-zero values in the population, while in the sparse condition, half of the potential auxiliary variables had no association with the analysis variables.

The variable-selection/dimension-reduction capabilities of the various MI methods were not of interest during Experiment 1. Therefore, I fixed $V = 12$ and varied the remaining three design parameters across the following levels: $PM = \{.1, .2, .3\}$, $N = \{200, 400\}$, $DS = \{Sparse, Dense\}$. Experiment 1 followed a factorial design and contained $1(V) \times 3(PM) \times 2(N) \times 2(DS) = 12$ fully crossed conditions. Likewise, the effect of sample size was not interesting in Experiment 2. So, I fixed $N = 200$ and varied the remaining design parameters across the following levels: $PM = \{.1, .2, .3\}$, $V = \{150, 250\}$, $DS = \{Sparse, Dense\}$. Experiment 2 also conformed to a factorial design and contained $1(N) \times 3(PM) \times 2(V) \times 2(DS) = 12$ fully crossed conditions.

2.1.2 Comparison Conditions

To judge the relative performance of the MIBEN algorithm, the parameters produced by analysis models fit to data treated with the MIBEN method were compared to analogous parameters estimated under four alternative missing data treatments. Three of these comparison conditions were applications of normal-theory MICE as implemented in the R package **mice** (van Buuren & Groothuis-Oudshoorn, 2011). Each of these MICE-based conditions employed Bayesian linear regression as the elementary imputation method, but they differed in which predictors entered their respective imputation models.

The first comparison condition employed the true imputation model (i.e., the model containing only the variables of the analysis model and those potential auxiliary variables that were actually used to impose the MAR missingness). The second condition employed a method of best subset selection to choose which variable to include in the imputation model. This best subset of predictors included all of the variables in the analysis model and a subset of auxiliary variables selected with the `quickpred` function which is included as a convenience function in the **mice** package. The `quickpred` function selects predictors for the imputation model according to the strengths of their correlations with (1) the incomplete variable being imputed and (2) the nonre-

sponse indicator for the variable being imputed. For the current study, a threshold of $r = .5$ was used to choose predictors. This value was chosen according to guidance given by Graham (2012) who noted that auxiliaries that are correlated with the incomplete variables at lower than $r = .5$ will tend to have a minimal impact on the imputation performance. Experiment 1 included another MICE-based condition in which the imputation model naïvely included all of the potential auxiliary variables and all of the variables in the analysis model. This method was not included in Experiment 2 because the naïve imputation models were intractable when $P > N$.

To ground MIBEN in the most recent literature, a variation of the Zhao and Long (2013) Bayesian LASSO-based MI was also included. This method was very similar to the sequential regression-based extension reported by Zhao and Long (2013) except that the underlying Bayesian LASSO model employed the Park and Casella (2008) prior, whereas the original Zhao and Long (2013) implementation used a slightly more complex parameterization of the LASSO prior. The method employed here, which I will refer to as *multiple imputation with the Bayesian LASSO* (MIBL), estimates the LASSO penalty parameter via the same multi-stage MCEM framework used by MIBEN. Numerical optimization was not necessary, however, because Park and Casella (2008) provided a deterministic update rule for the EM estimator. Thus, the two-step optimization of the penalty parameters described on Lines 19 and 20 of Algorithm 3 was replaced by a single deterministic update calculation.

2.1.3 Outcome Measures

A heuristic image of each method's performance was built up by comparing several outcome measures. Bias introduced by the imputations was quantified with two measures: *percentage relative bias* (PRB) and *standardized bias* (SB).

$$PRB = 100 \cdot \left(\frac{\bar{\hat{\theta}} - \theta}{\theta} \right), \quad SB = \frac{\bar{\hat{\theta}} - \theta}{SD_{\hat{\theta}}},$$

where θ represents the focal parameter's true value, $\bar{\hat{\theta}}$ represents the mean of the estimated parameters, and $SD_{\hat{\theta}}$ represents the empirical standard deviation of the focal parameter. Following the recommendations of Muthén, Kaplan, and Hollis (1987) and Collins et al. (2001), respectively, $|PRB| > 10$ and $|SB| > 0.40$ were considered indicative of problematic bias. Variation induced strictly by the Monte Carlo simulation was quantified by the *Monte Carlo Standard Deviation* of the focal parameters.

$$SD_{MC} = \sqrt{R^{-1} \sum_{r=1}^{R} \left(\hat{\theta}_r - \bar{\hat{\theta}} \right)^2},$$

where R is the number of Monte Carlo replicates and $\hat{\theta}_r$ is the estimated parameter from the rth Monte Carlo replication. To assess the integrity of hypothesis tests conducted under the various imputation techniques, the confidence interval coverage rates and average confidence interval width were also computed.

$$CI_{cover} = R^{-1} \sum_{r=1}^{R} \mathrm{I}\left(\theta \in \widehat{CI}_r \right), \quad CI_{width} = R^{-1} \sum_{r=1}^{R} \left[\widehat{CI}_{r,upper} - \widehat{CI}_{r,lower} \right],$$

where $\widehat{CI}_r$ is the estimated confidence interval from the rth replication, $\widehat{CI}_{r,upper}$ and $\widehat{CI}_{r,lower}$ represent the upper and lower bounds, respectively, of the estimated confidence interval for the rth replication, and $\mathrm{I}(\cdot)$ is the indicator function that returns 1 when its argument is true and 0 otherwise. Following the recommendation of Burton, Altman, Royston, and Holder (2006), CI coverage rates that were greater than two standard errors of the nominal coverage probability p above or below the nominal coverage rate were considered problematic. The standard error of the nominal coverage rate was defined as: $SE(p) = \sqrt{p(1-p)/R}$.

2.2 Data Generation

2.2.1 Population Data

The data were generated in two stages. First, a $N \times 3$ matrix $\mathbf{X}$ containing the independent variables of the analysis model was simulated according to the following model:

$$\begin{aligned} \mathbf{X} &= \mathbf{Z}\boldsymbol{\zeta} + \Theta, \qquad (2.1) \\ \text{with } \mathbf{Z} &\sim \text{MVN}(\mathbf{0}_V, \mathbf{I}_V), \\ \text{and } \Theta &\sim \text{MVN}(\mathbf{0}_3, \Omega_X), \\ \text{where } \Omega_X &= \left(\frac{1}{2}\right) \boldsymbol{\zeta}^T \text{Cov}(\mathbf{Z}) \boldsymbol{\zeta} \circ \mathbf{I}_3. \qquad (2.2) \end{aligned}$$

In the preceding, $\mathbf{0}_V$ represents a V-vector of zeros, $\mathbf{I}_V$ represents the $V \times V$ identity matrix, the Cov$(\cdot)$ operator returns the covariance matrix of its argument, and the $\circ$ operator represents the element-wise matrix product (i.e., the Hadamard product). The matrix $\mathbf{Z}$ in Equation 2.1 contains the exogenous auxiliary variables. These auxiliaries were related to the analysis variables via the coefficient matrix $\boldsymbol{\zeta}$ which took different forms according to the sparsity of the focal condition:

$$\boldsymbol{\zeta}_{dense} = \begin{bmatrix} \zeta_{1,1} & \zeta_{1,2} & \zeta_{1,3} \\ \zeta_{2,1} & \zeta_{2,2} & \zeta_{2,3} \\ \vdots & \vdots & \vdots \\ \zeta_{V,1} & \zeta_{V,2} & \zeta_{V,3} \end{bmatrix}, \quad \boldsymbol{\zeta}_{sparse} = \begin{bmatrix} \zeta_{1,1} & \zeta_{1,2} & \zeta_{1,3} \\ \zeta_{2,1} & \zeta_{2,2} & \zeta_{2,3} \\ \vdots & \vdots & \vdots \\ \zeta_{\frac{V}{2},1} & \zeta_{\frac{V}{2},2} & \zeta_{\frac{V}{2},3} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix},$$

where each $\zeta_{v,j} \sim \text{Unif}(.25,\ .5)$.

Once the predictors in the analysis model were simulated as above, the dependent variable $\mathbf{y}$

was created as a function of the variables in $\mathbf{X}$:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{2.3}$$

$$\text{with } \boldsymbol{\beta} = \{.2,\ .4,\ .6\}^T,$$

$$\text{and } \boldsymbol{\varepsilon} \sim \mathrm{N}\left(0, \omega_y^2\right),$$

$$\text{where } \omega_y^2 = \left(\frac{1}{5}\right)\boldsymbol{\beta}^T \mathrm{Cov}(\mathbf{X})\boldsymbol{\beta}. \tag{2.4}$$

The terms $\boldsymbol{\beta}^T \mathrm{Cov}(\mathbf{X})\boldsymbol{\beta}$ (in Equation 2.4) and $\boldsymbol{\zeta}^T \mathrm{Cov}(\mathbf{Z})\boldsymbol{\zeta}$ (in Equation 2.2) quantify the reliable variance/covariance of $\mathbf{y}$ and $\mathbf{X}$, respectively. Thus, by simulating the data as described, $\mathbf{y}$ and $\mathbf{X}$ maintain a constant signal-to-noise ratio of 5:1 and 2:1, respectively. After simulating $\mathbf{y}$, $\mathbf{X}$, and $\mathbf{Z}$ as above, these three data matrices were merged into a single $N \times (4 + V)$ data frame $\mathbf{Y}_{full}$ that represented the fully observed population realization of $\mathfrak{Y}$ for the rth Monte Carlo replication.

2.2.2 Missing Data Imposition

Item nonresponse was imposed on all variables in $\mathbf{Y}_{full}$. A 10% nonresponse rate was imposed on the auxiliary variables via a simple *missing completely at random* (MCAR) mechanism. The probability of a cell $z_{n,v} \in \mathbf{Z}$ being unobserved was modeled as a Bernoulli trial with a 10% chance of "success." For the analysis variables $\mathbf{Y}$, the missingness was imposed via a MAR mechanism in which the propensity to respond was given as a linear function of a randomly selected subset of the potential auxiliary variables. Thus, the nonresponse mechanism affecting the analysis variables was modeled as a straight-forward probit regression:

$$P(\mathbf{y}_k = \mathit{MISSING} \mid \mathbf{Z}) = \Phi\left(\tilde{\mathbf{Z}}\boldsymbol{\xi}\right), \tag{2.5}$$

where $\boldsymbol{\xi} = \{.25,\ .5\}^T$ in Experiment 1 and $\boldsymbol{\xi} = \{.25,\ .25,\ .25,\ .25,\ .25,\ .5,\ .5,\ .5,\ .5,\ .5\}^T$ in Experiment 2, $\tilde{\mathbf{Z}}$ was a $N \times 2$ matrix in Experiment 1 and a $N \times 10$ matrix in Experiment 2 whose columns contained, respectively, 2 or 10 randomly selected columns of $\mathbf{Z}$ that were associated

with nontrivial regression coefficients, and $\Phi(\cdot)$ represents the standard normal cumulative distribution function. By drawing $\tilde{\mathbf{Z}}$ from only the columns of $\mathbf{Z}$ associated with nontrivial regression coefficients all of the true auxiliary variables remain in the *active set* of $\{\mathbf{X}, \mathbf{Z}\}$. This parameterization minimized the possibility of true auxiliaries being selected out of the imputation model and producing a set of auxiliary data that was predictive of the incomplete variables $\mathbf{Y}$ but unrelated to the nonresponse propensity. For each variable $\mathbf{y}_k \in \{\mathbf{y}, \mathbf{X}\}$ $k = 1, 2, 3, 4$ missingness was imposed according to the process described by Algorithm 5.

Algorithm 5 Impose MAR Missingness

Input: A complete data set $\mathbf{Y}_{full}$
Output: An incomplete data set $\mathbf{Y}_{inc}$
Define: $PM \coloneqq$ The percentage of missing data to impose
Define: $J \coloneqq$ The number of true auxiliary variables
Set: $\mathbf{Y} \leftarrow \{\mathbf{y},\ \mathbf{X}\}$
for $k = 1$ to 4 **do**
  Set: $\tilde{\mathbf{Z}} \leftarrow J$ randomly selected columns of $\mathbf{Z}$ $s.t.$ $\beta_{z_j} \neq 0$
  Compute: $P(\mathbf{y}_k = MISSING \mid \mathbf{Z})$ according to Equation 2.5
  for $n = 1$ to N **do**
    if $P(\mathbf{y}_k = MISSING \mid \mathbf{Z}) \geq 1 - PM$ **then**
      $\mathbf{Y}[n,\ k] \leftarrow MISSING$
    end if
  end for
end for
Set: $\mathbf{Y}_{inc} \leftarrow Merge(\mathbf{Y},\ \mathbf{Z})$
return $\mathbf{Y}_{inc}$

2.3 Procedure

2.3.1 Computational Details

To ease the computational burden of this study, the Monte Carlo simulation was conducted using parallel processing methods. This parallel computing was implemented via the R package **parallel** (R Core Team, 2014). To ensure replicability of the experiment, and to ensure the independence of the Monte Carlo replications, the pseudo-random numbers were generated accord-

ing to the L'ecuyer, Simard, Chen, and Kelton (2002) method as implemented in the R package **rlecuyer** (Sevcikova & Rossini, 2012).

The code to run the simulation was written in the R statistical programming language (R Core Team, 2014). All of the MICE-based comparison conditions were run using the R package **mice** (van Buuren & Groothuis-Oudshoorn, 2011). MIBEN and MIBL were implemented with a new R package, **mibrr**[1](i.e., *Multiple Imputation with Bayesian Regularized Regression*), that I developed for this project. The **mibrr** package employs the multi-stage MCEM algorithm and Gibbs sampler described in Section 1.6 to fit the MIBEN and MIBL imputation models. **mibrr** only uses R for data pre- and post-processing; all Gibbs sampling and marginal MCEM optimization required to fit the MIBEN and MIBL imputation models is done in C++ and linked back to the R layer via the **Rcpp** package (Eddelbuettel & François, 2011). The numerical (pre-)optimization of MIBEN's penalty parameters is accomplished through a robust, redundant procedure. Both pre-optimization and final optimization of the penalty parameters is done with the C++ package **nlopt** (Johnson, 2014). At each stage, the maximization is initially attempted via a preferred optimization routine. If this initial attempt fails, a series of three additional optimization routines are sequentially attempted until either the parameters are successfully (pre-)optimized, or the final candidate optimization routine fails. In the latter case, the program exits with an error. Table 2.1 gives information on the various optimization routines employed in the **mibrr** package.

Experiment 1 was run on a personal computer with an Intel Core i7 3610QM processor, 8GB RAM, and a 750GB mechanical hard disk running Debian GNU/Linux 7.8. The computations were run in parallel across the 8 virtual cores of the 3610QM processor. Experiment 2 was run on an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) c4.8xlarge cluster computing instance running Ubuntu Server 14.04 LTS. This instance employed 36 virtual processor cores physically located on several 2.9 GHz Intel Xeon E5-2666 v3 processors, 60GB of RAM, and a 100GB SSD with 3000 provisioned IOPS. The computations were run in parallel across the 36 virtual cores of the instance.

[1]This package is freely available for testing purposes. A copy can be accessed by request to the author.

Algorithm Precedence	Pre-Optimization Routines			Final Optimization Routines		
	Algorithm	*NLopt Key*	*Citation*	*Algorithm*	*NLopt Key*	*Citation*
1	Bound-Constrained Optimization by Quadratic Approximation	BOBYQA	Powell (2009)	Method of Moving Asymptotes	MMA	Svanberg (2002)
2	Constrained Optimization by Linear Approximation	COBYLA	Powell (1994)	Shifted Limited-Memory Variable-Metric (Rank-2 Method)	VAR2	Vlček and Lukšan (2006)
3	Subplex	SBPLX	Rowan (1990)	Shifted Limited-Memory Variable-Metric (Rank-1 Method)	VAR1	Vlček and Lukšan (2006)
4	Principle Axis	PRAXIS	Brent (1973)	Low-Storage BFGS	LBFGS	Liu and Nocedal (1989); Nocedal (1980)

Table 2.1: Optimization routines employed by the **mibrr** package

2.3.2 Choosing the Number of Monte Carlo Replications

In the current project, two classes of parameter were of principle interest: the regression coefficients of the analysis model and their associated standard errors. Thus, a power analysis was conducted to compute how many replications were needed to capture these effects to an acceptable degree of accuracy.

Given a focal parameter θ, Burton et al. (2006) gave the following formula to determine the number of Monte Carlo replications R needed to ensure a $1 - (\alpha/2)$ probability of measuring $\hat{\theta}$ to an accuracy of δ:

$$R = \left(\frac{Z_{1-(\alpha/2)}\sigma}{\delta} \right)^2, \tag{2.6}$$

where $Z_{1-(\alpha/2)}$ represents the $1 - (\alpha/2)$ quantile of the standard normal distribution and σ is the known standard deviation of θ.

2.3.2.1 Estimating σ

There were two difficulties with implementing Equation 2.6 for the current project. First, the Monte Carlo sampling variances of $\boldsymbol{\beta}$ and SE_β were not known before running the simulation. Second, it was not immediately obvious how the introduction of missing data would affect the Monte Carlo sampling variances of these parameters. The first issue was addressed by running an initial pilot simulation. For each sample size $N = \{200,\ 400\}$ in Experiment 1 and each number of potential auxiliary variables $V = \{150,\ 250\}$ in Experiment 2, 50,000 replicates of $\mathbf{Y}_{full}$ were simulated and used to fit the analysis model given by Equation 2.3. These 50,000 model fits were then used to compute the empirical Monte Carlo standard deviations of $\boldsymbol{\beta}$ and SE_β.

The second issue, however, required more careful consideration. The ubiquitous *fraction of missing information* (FMI) is the key quantity to consider when assess the effect of missing data on a given model. The FMI quantifies the amount of a parameter's information that has been lost to nonresponse. Because information is inversely proportional to variance, the FMI also quantifies the increase in a parameters sampling variability that is strictly due to nonresponse (Rubin, 1987). Clearly, the final Monte Carlo sampling variability of $\boldsymbol{\beta}$ and SE_β will be some combination of the quantity described in the previous paragraph and the FMI. However, the FMI can only be computed once the missing data analysis is complete because its value will be relatively larger or smaller depending on the quality of the missing data treatment.

Though the exact FMI cannot be computed before the missing data analysis is run, a plausible interval for the expected FMI can be inferred. The FMI can be somewhat larger than the proportion of missing data (Savalei & Rhemtulla, 2011), but, in practice, it is often reasonable to expect that the FMI is approximately equal to or somewhat smaller than the PM (Enders, 2010)—especially when the data follow a MAR mechanism and the imputation model is well-parameterized. Therefore, the projected influence of the missing data was included into the current power analysis by specifying three values for FMI to encompass a plausible range: FMI = $\{PM/2,\ PM,\ 2PM\}$. For each of these values of FMI, the projected Monte Carlo standard

deviation was taken to be:

$$SD_{MC,Proj} = \widehat{SD}_{MC,Pilot}\left(1 + \sqrt{\frac{\text{FMI}}{1 - \text{FMI}}}\right), \tag{2.7}$$

where $\widehat{SD}_{MC,Pilot}$ is the complete-data Monte Carlo SD estimated from the pilot simulation described above. The fractional term under the radical in Equation 2.7 is the *relative increase in variance*, which gives the proportional increase in the focal parameters sampling variability due to nonresponse. Thus, the weighting term inside the parentheses in Equation 2.7 represents a scaling factor that adjusts the parameters' sampling variances for the expected impact of nonresponse. The required replicates R were then computed by substituting $SD_{MC,Proj}$ in for σ in Equation 2.6.

2.3.2.2 Power Analysis Results

For the current power analysis, the target accuracy (i.e., δ in the denominator of Equation 2.6) was specified as a proportion of the true parameter's magnitude. Because the standard error of $\boldsymbol{\beta}$ does not have a true population-level value, its "true value" SE_β was taken to be the the average of the 50,000 replicates from the pilot study (i.e., $SE_\beta \coloneqq 2 \times 10^{-5} \sum \widehat{SE}_{\beta,MC}$). The target accuracies were defined to be, at least, 5 percent of these true values: $\delta_\beta = .05 \cdot \boldsymbol{\beta}$ and $\delta_{SE_\beta} = .05 \cdot SE_\beta$. Thus, the final power analysis entailed computing the number of replications required to achieve these target accuracies given sample sizes of $N = \{200,\ 400\}$ (for Experiment 1), number of potential auxiliaries $V = \{150,\ 250\}$ (for Experiment 2), proportions missing of $PM = \{.1,\ .2,\ .3\}$, and FMI Levels of FMI $= \{PM/2,\ PM,\ 2PM\}$. Figures 2.1, 2.2, and 2.3 summarize the findings.

Based on the findings of the power analysis, $R = 500$ replications were chosen for the current simulation. Figures 2.2 and 2.1 show that, in plausible circumstances, 500 replications are sufficient to ensure a 95% chance of measuring $\hat{\boldsymbol{\beta}}_{MC}$ in Experiment 1 and $\widehat{SE}_{\beta,MC}$ in both experiments to within 2.5% of their true values. Unfortunately, in Experiment 2, ensuring a 95% chance of measuring small values of $\hat{\boldsymbol{\beta}}_{MC}$ to within 5% of their true values would require a pro-

hibitively large number of replications (as seen in the first two columns of Figure 2.3). Because computational demands were already a paramount limitation of the current project, and because the moderate and large effects are adequately captured with $R = 500$ replications, this number was deemed sufficient for Experiment 2, as well—with the acknowledgment that the precision of estimates of the small effects may suffer. As seen in the rightmost two columns of Figure 2.3, $R = 500$ is sufficient to ensure a 95% chance of measuring the small effects to an accuracy of 10% of their true values.

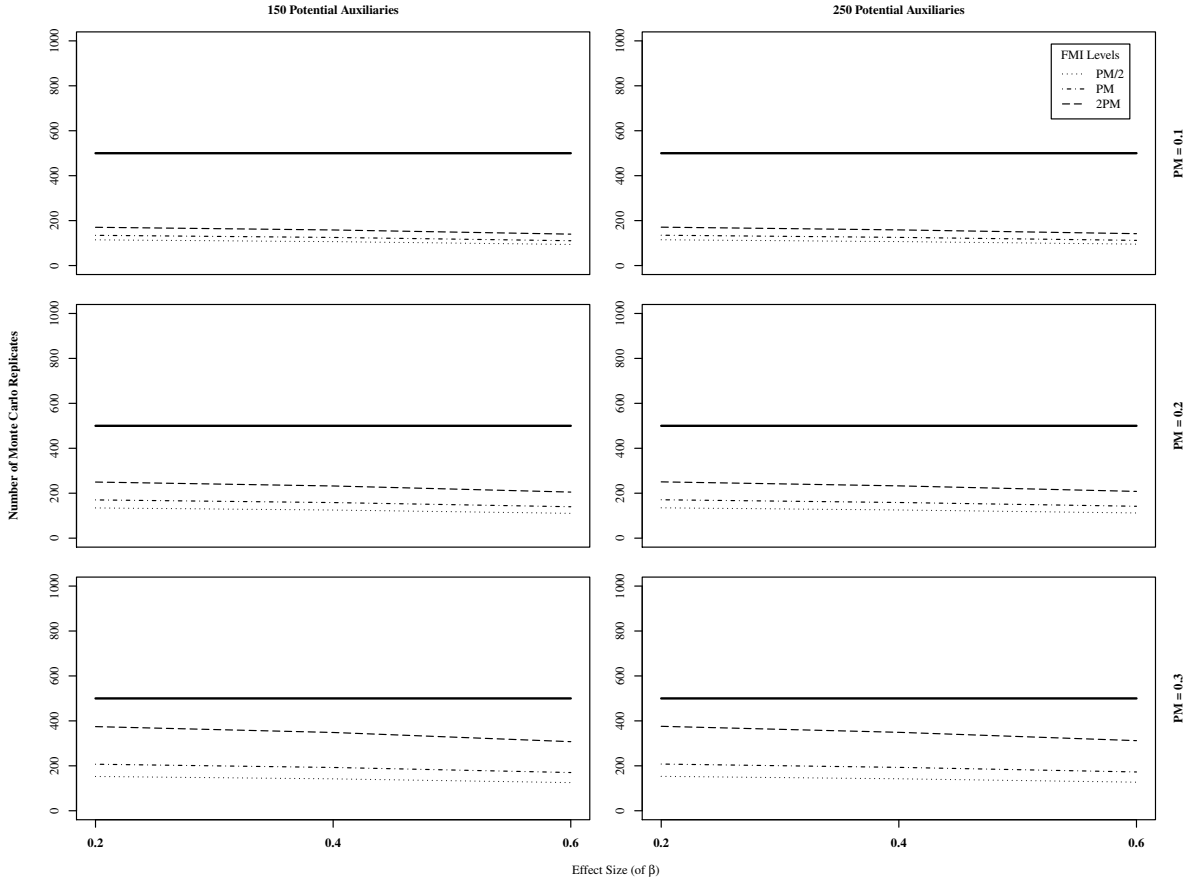

Figure 2.1: Monte Carlo replications required to capture $\widehat{SE}_{\beta,MC}$ to an accuracy of 2.5% of its true value in Experiment 2

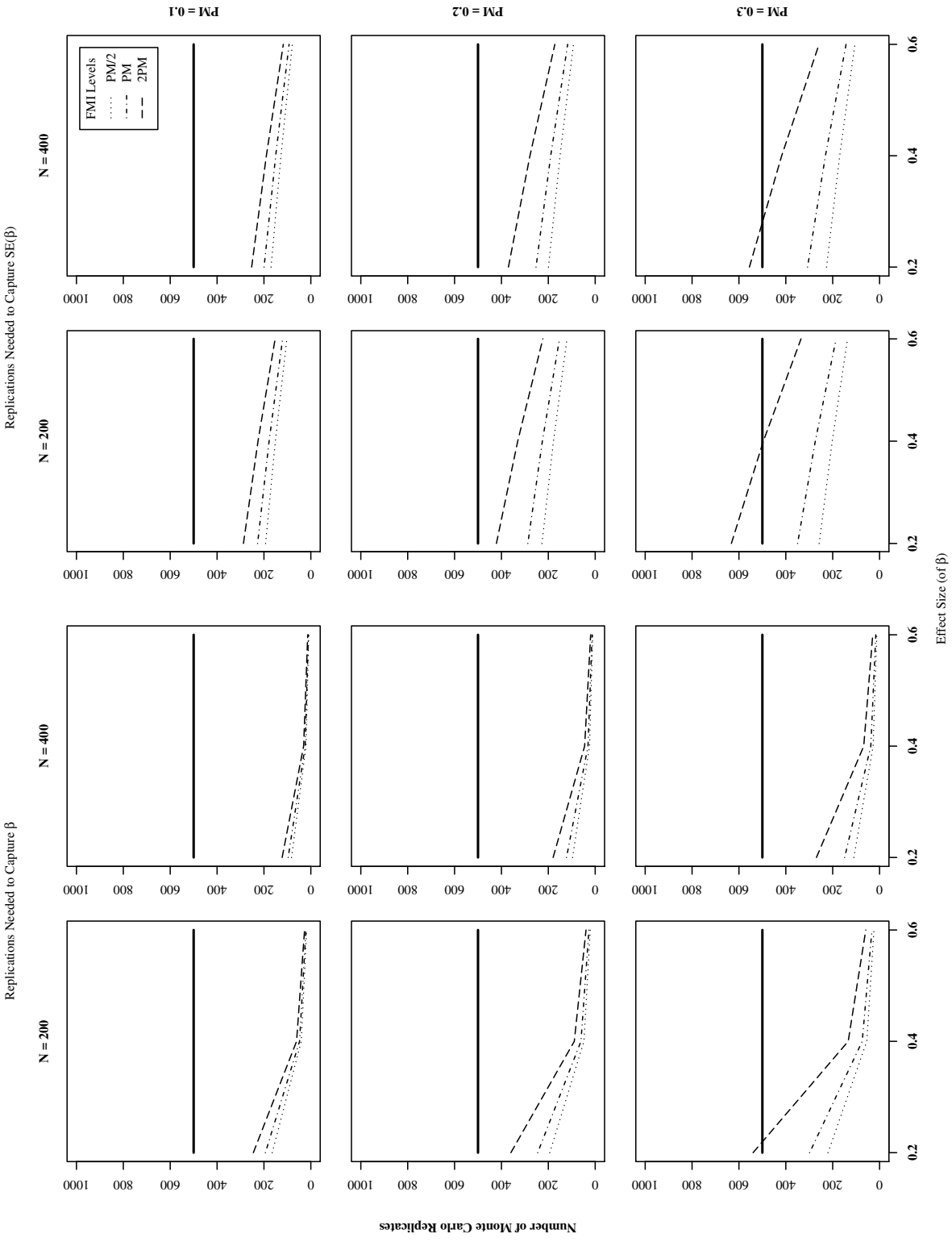

Figure 2.2: Monte Carlo replications required to capture $\hat{\boldsymbol{\beta}}_{MC}$ and $\widehat{SE}_{\beta,MC}$ to accuracies of 2.5% of their respective true values in Experiment 1

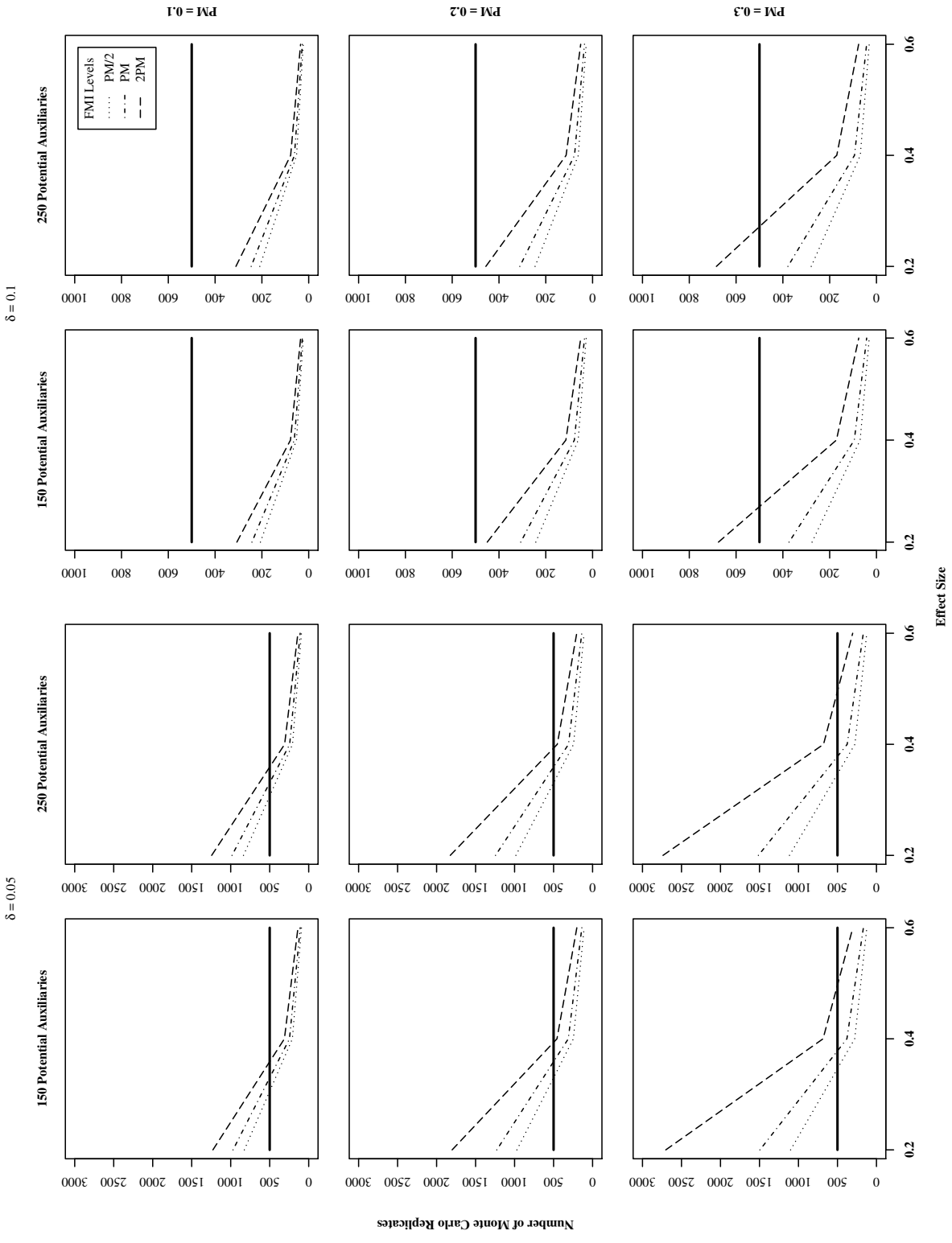

Figure 2.3: Monte Carlo replications required to capture $\hat{\boldsymbol{\beta}}_{MC}$ to an accuracy of 5% (left two columns) and 10% (right two columns) of its true value in Experiment 2

2.3.3 Parameterizing the MIBEN & MIBL Gibbs Samplers

To implement MIBEN and MIBL several key parameters must be specified: (1) the number of MCEM approximation iterations, (2) the number of MCEM tuning iterations, (3) the size of the Gibbs samples drawn during the MCEM approximation phase (and the associated number of burn-in Gibbs iterations to discard), (4) the size of the Gibbs samples drawn during the MCEM tuning phase (and the associated number of burn-in Gibbs iterations to discard), and (5) the size of the final posterior Gibbs sample to draw (and the associated number of burn-in Gibbs iterations to discard). For the current study, these values were each chosen by running a small set of exploratory replications in which different candidate values were auditioned and convergence was judged as in the full simulation study. This approach led to choosing the set of values contained in Table 2.2 to parameterize the MIBEN and MIBL Gibbs samplers.

		Proportion Missing	Sample Size	Potential Auxiliaries	MIBEN MCEM Approximation Iterations	MIBL MCEM Approximation Iterations	MCEM Tuning Iterations	MCEM Approx: Gibbs Burn-In	MCEM Approx: Gibbs Sample Size	MCEM Tune: Gibbs Burn-In	MCEM Tune: Gibbs Sample Size	Posterior Burn-In	Posterior Gibbs Sample Size
Experiment 1	Sparse	0.1	400	12	50	75	10	25	25	100	200	250	500
		0.2	400	12	50	75	10	25	25	100	200	250	500
		0.3	400	12	50	75	10	25	25	100	200	250	500
		0.1	200	12	50	75	10	25	25	100	200	250	500
		0.2	200	12	50	75	10	25	25	100	200	250	500
		0.3	200	12	50	75	10	25	25	100	200	250	500
	Dense	0.1	400	12	50	75	10	25	25	100	200	250	500
		0.2	400	12	50	75	10	25	25	100	200	250	500
		0.3	400	12	50	75	10	25	25	100	200	250	500
		0.1	200	12	50	75	10	25	25	100	200	250	500
		0.2	200	12	50	75	10	25	25	100	200	250	500
		0.3	200	12	50	75	10	25	25	100	200	250	500
Experiment 2	Sparse	0.1	200	250	200	200	20	25	25	200	300	500	1000
		0.2	200	250	200	300	20	25	25	200	300	500	1000
		0.3	200	250	200	400	20	25	25	200	300	500	1000
		0.1	200	150	150	200	15	25	25	200	300	500	1000
		0.2	200	150	150	200	15	25	25	200	300	500	1000
		0.3	200	150	150	200	15	25	25	200	300	500	1000
	Dense	0.1	200	250	200	200	20	25	25	200	300	500	1000
		0.2	200	250	300	300	20	25	25	200	300	500	1000
		0.3	200	250	400	400	20	25	25	200	300	500	1000
		0.1	200	150	150	200	15	25	25	200	300	500	1000
		0.2	200	150	150	200	15	25	25	200	300	500	1000
		0.3	200	150	150	200	15	25	25	200	300	500	1000

Table 2.2: Iterations of the MIBEN and MIBL Gibbs samplers & MCEM algorithms

2.3.4 Simulation Workflow

For each replication, a single population realization $\mathbf{Y}_{full} \in \mathfrak{Y}$ of the full data was simulated according to Equations 2.1 and 2.3. The appropriate degree of nonresponse was then imposed according to the procedures describe in Section 2.2.2. These missing data were then imputed 100 times by the MIBEN algorithm as well as each of the MI methods described in Section 2.1.2. Finally, for each of these sets of imputed data, 100 replicates of the analysis model given by Equation 2.3 were estimated, and their parameter estimates were pooled via *Rubin's Rules* (Rubin, 1987). After running all 500 replications, the performance of each of the MI methods was quantified by computing the suite of outcome measures described in Section 2.1.3.

Chapter 3

Results

3.1 Convergence Rates

Convergence rates of all imputation and analysis models were very high. Only two imputation models failed to converge. Both failures occurred with MIBEN, in Experiment 1, with sparse imputation models, $PM = 0.1$, and $N = 400$. These failures occurred when the MCEM algorithm failed to locate a non-zero value for the ridge penalty parameter because the ℓ_2-regularization was unnecessary. All other imputation and analysis models converged. For MIBEN and MIBL, convergence of the imputation model parameters' final Gibbs samples was assessed via the *potential scale reduction factor* ($\hat{R}$). Using the criterion $\hat{R} \leq 1.1$ to indicate stable convergence, all final Gibbs samples converged to their respective stationary posterior distributions. Experiment 1 took approximately 16.7 hours to run, and Experiment 2 ran for approximately 92.7 hours.

Although I had no hypothesis regarding convergence properties, this area is one where MIBEN clearly outstrips MIBL. The deterministic update rule used to estimate the Bayesian LASSO's penalty parameter (see Park & Casella, 2008, p. 683) is robust and computationally efficient (within iteration), but it takes small steps in the parameter space. Compared to this deterministic update, the two-step, numerical optimization employed to estimate the BEN's penalty parameters in MIBEN is much more computationally expensive within a single iteration, but it

takes much larger steps in the parameter space. Thus, MIBEN's MCEM algorithm converges in far fewer iterations than MIBL's version does. MIBEN's MCEM iterates also tend to produce an unambiguous "elbow" pattern in the penalty parameters' trace plots that eases the task of assessing convergence while MIBL's version tends to produce much smoother trace plots that are more difficult to interpret. Figure 3.1 shows trace plots of ten randomly selected replications from Experiment 2 with $PM = 0.3$, 250 potential auxiliaries, and sparse imputation models.

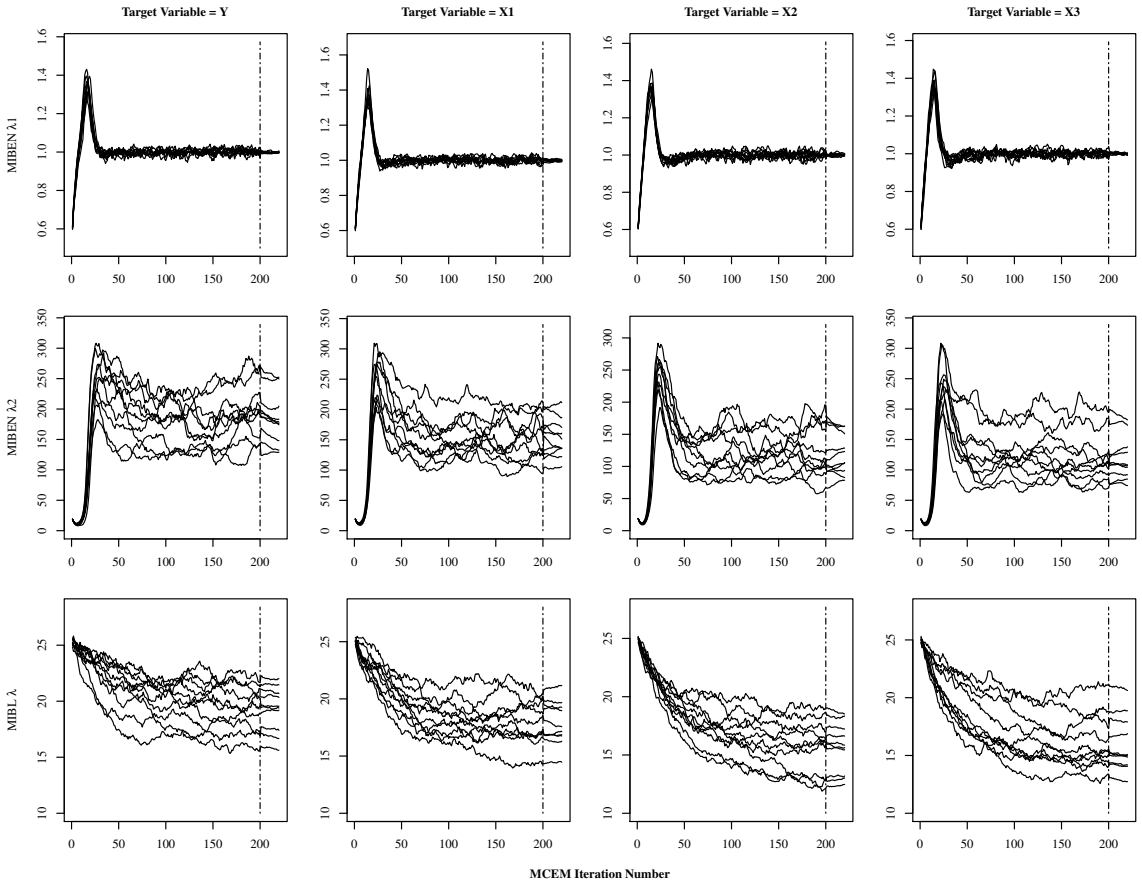

Figure 3.1: Trace plots of MIBEN and MIBL penalty parameters for 10 randomly selected replications of Experiment 2 with $PM = 0.3$, $V = 250$, and sparse imputation models
Note: Dashed horizontal lines indicate the beginning of the MCEM tuning phase

3.2 Overdetermined Models

The results of Experiment 1 showed very strong performance for both MIBEN and MIBL when the imputation models were highly overdetermined. Both MIBEN and MIBL produced unbiased estimates with nearly optimal confidence interval coverage rates (although there was a tendency for all imputation methods to induce overcoverage of the true regression slopes with sparse models). The three MICE-based approaches also did very well except when estimating intercepts, where they tended to produce positively biased estimates with confidence intervals that considerably undercovered the true parameter values. Thus, Hypotheses 3 and 4 are fully supported since MIBEN performed as well as, or better than, the MICE-based approaches for overdetermined imputation models. Hypothesis 7 was also supported since MIBEN and MIBL produced nearly identical results for the overdetermined models. Figures 3.2 and 3.3 contain plots of each method's PRB for sparse and dense imputation models, respectively. Likewise, Figures 3.4 and 3.5 show each method's SB for sparse and dense models, respectively, and Figures 3.6 and 3.7 show analogous plots of the CI coverage rates. The dashed lines in the plots contained in Figures 3.6 and 3.7 represent two *SE*s of the nominal coverage probability above and below the nominal coverage rate (i.e., $.95 \pm 2 \times SE(p)$).

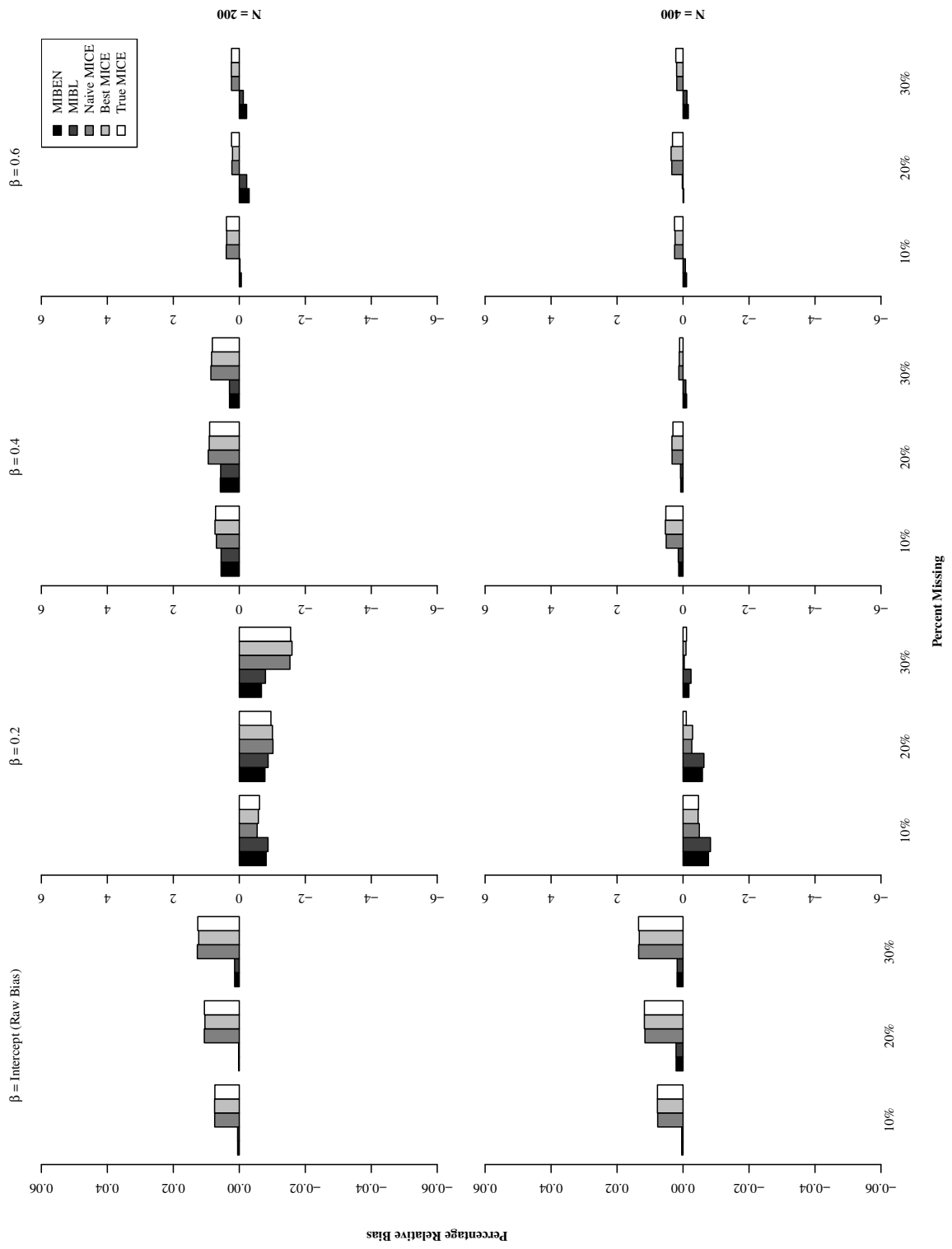

Figure 3.2: Percentage relative bias for Experiment 1 sparse imputation models
Note: Raw bias is reported for the intercepts because their true values were $\beta_0 = 0$.

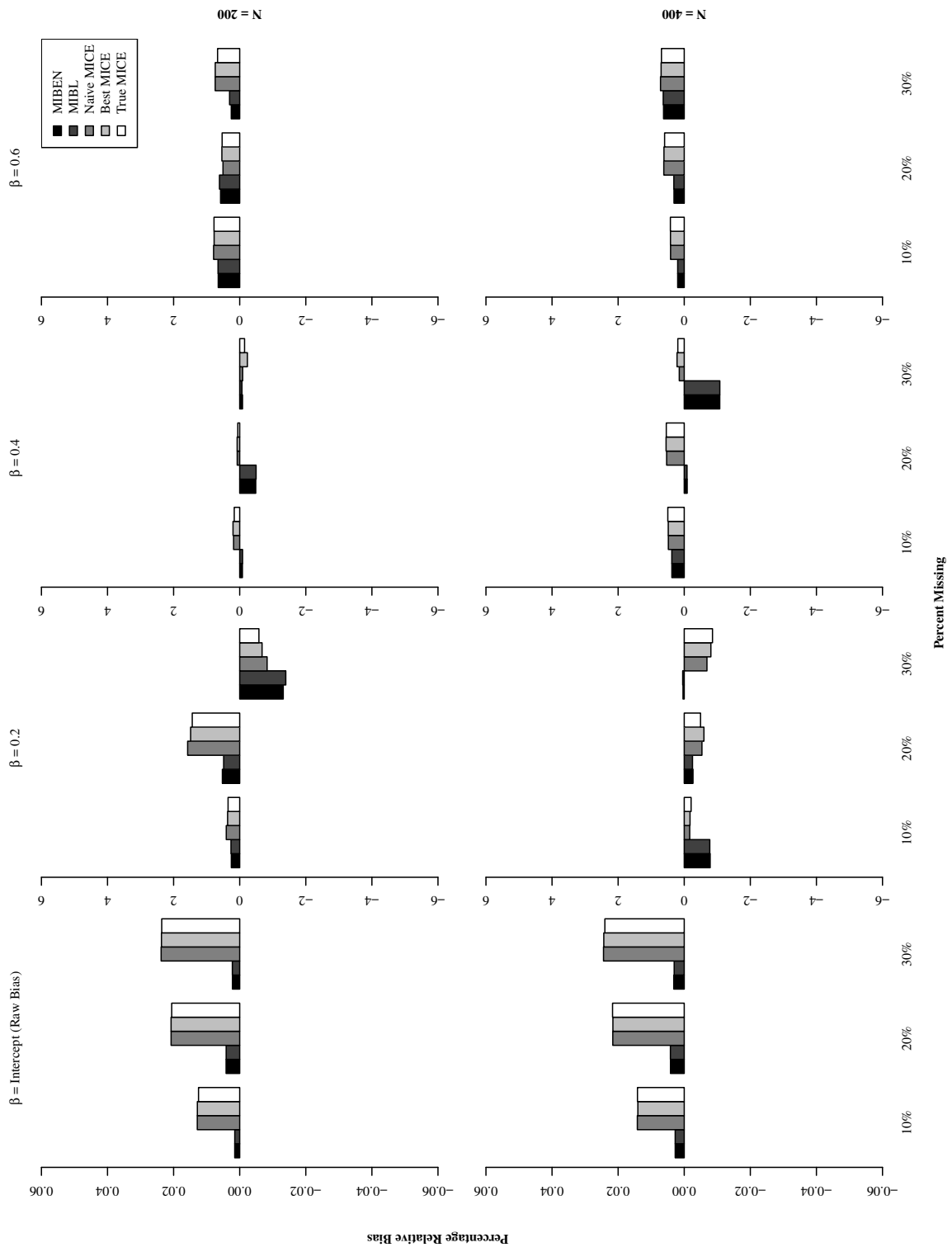

Figure 3.3: Percentage relative bias for Experiment 1 dense imputation models
Note: Raw bias is reported for the intercepts because their true values were $\beta_0 = 0$.

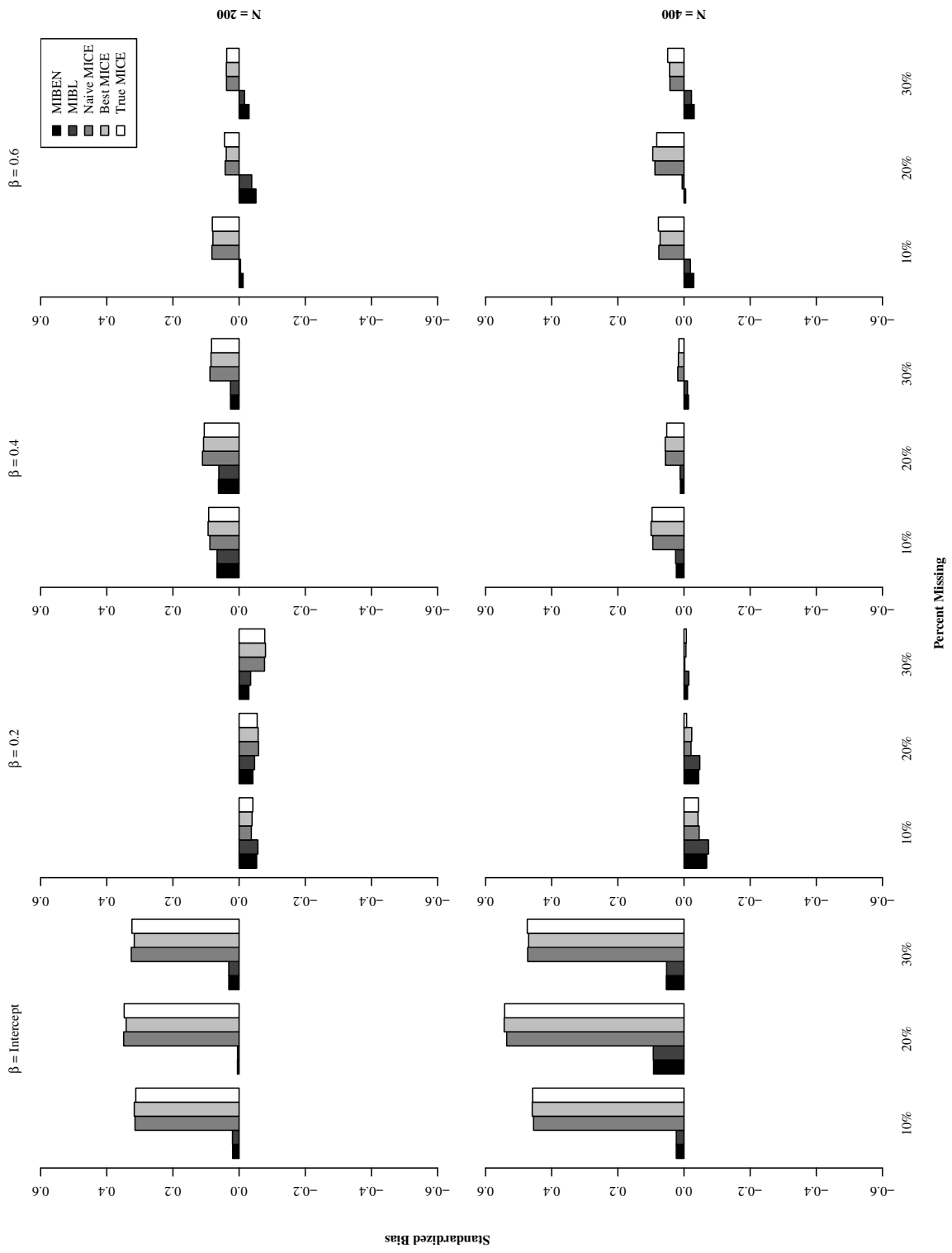

Figure 3.4: Standardized bias for Experiment 1 sparse imputation models

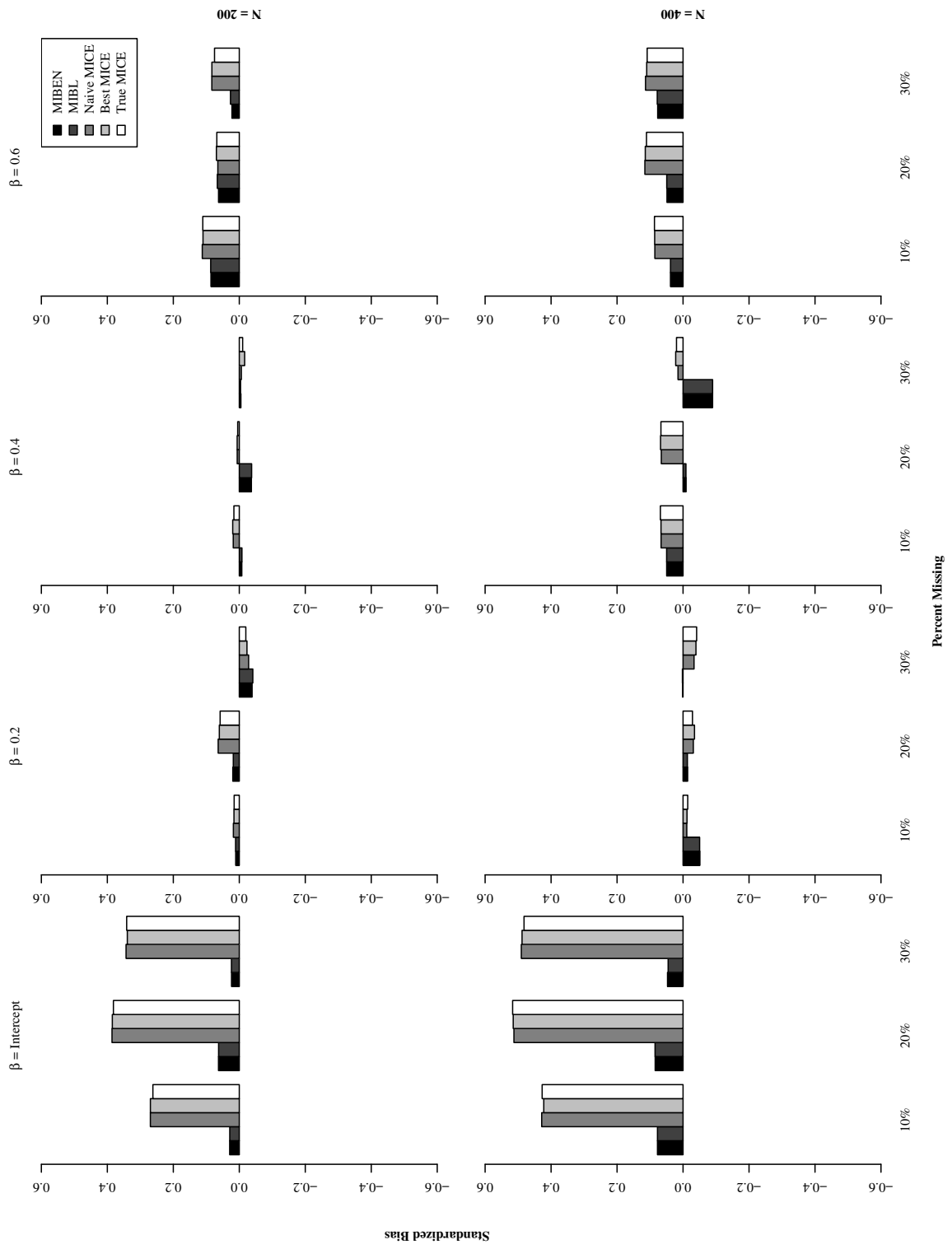

Figure 3.5: Standardized bias for Experiment 1 dense imputation models

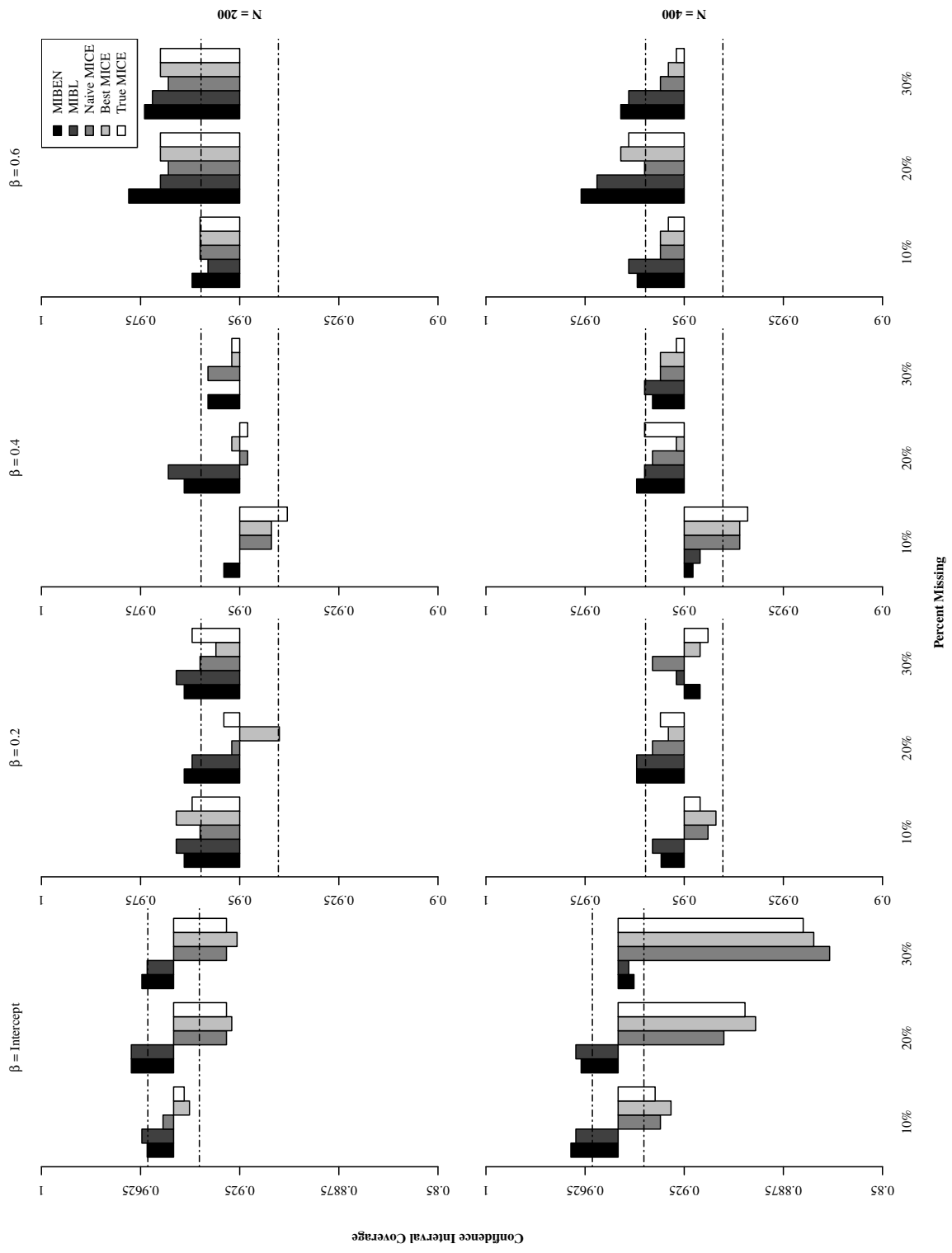

Figure 3.6: Confidence interval coverage for Experiment 1 sparse imputation models
Note: Dashed lines represent $.95 \pm 2 \times SE(p)$

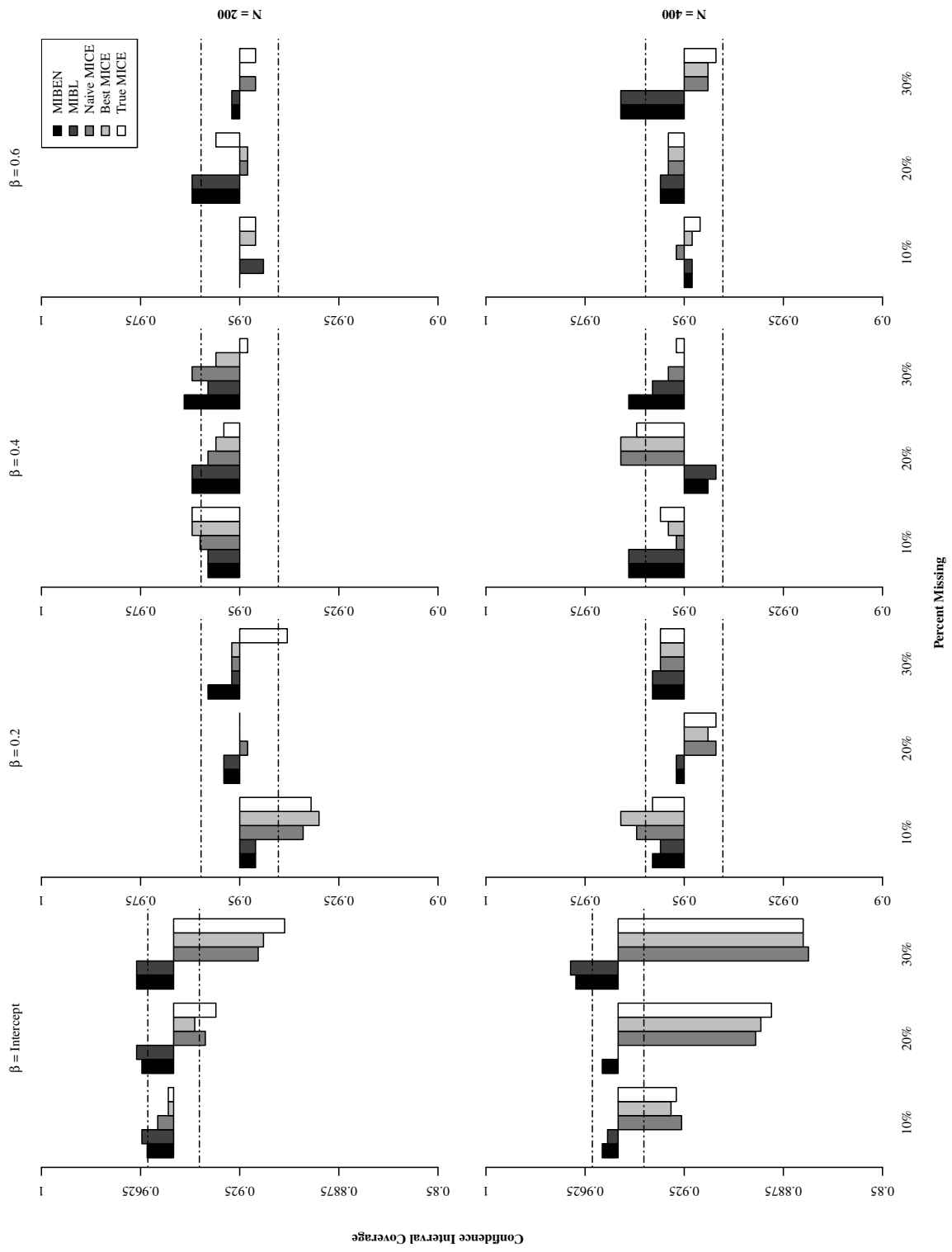

Figure 3.7: Confidence interval coverage for Experiment 1 dense imputation models
Note: Dashed lines represent .95 ± 2 × $SE(p)$

3.3 Underdetermined Models

When $P > N$ or $P \approx N$, there was not a clearly strongest method in terms of bias in the analysis model parameters. Figures 3.8 and 3.9 show the PRB of each method for sparse and dense imputation models, respectively. Figures 3.10 and 3.11 contain analogous plots of the SB. As seen in these figures, neither metric reflected much of a performance difference in estimating the moderate ($\beta = 0.4$) and large ($\beta = 0.6$) effects, but there was some differentiation when estimating small effects ($\beta = 0.2$) and intercepts, particularly in terms of PRB. The MICE-based methods tended to overestimate the intercepts for sparse models while MIBEN and MIBL provided unbiased estimates of the intercepts in all conditions. On the other hand, MIBEN and MIBL tended to underestimate the small effects to a greater extent in the sparse models while all tested methods tended to underestimate the small effects for dense models. Specifically in terms of SB, MIBEN and MIBL produced unbiased estimates across the board, while a small degree of positive SB in the intercepts remained for the MICE-based methods. A possible explanation for this pattern is discussed in more detail below, but these findings indicate little to no support for Hypothesis 5 since there is no evidence that MIBEN systematically produced lower parameter bias than the MICE-based methods did, except when estimating intercepts.

The patterns of CI coverage rates are also somewhat ambiguous. Figures 3.12 and 3.13 contain plots of the CI coverage rates induced by each method for sparse and dense models, respectively. All methods clearly demonstrated occasionally problematic departures from the nominal coverage rate, but one general pattern emerged. When coverage was problematic, MIBEN and MIBL tended to lead to *overcoverage* while the MICE-based approaches tended to induce *undercoverage*. This difference suggests that MIBEN and MIBL will tend to induce higher Type II error rates while the MICE-based approaches will tend to induce inflated Type I error rates. It may be that Type II errors are the lesser of two evils, but these findings do not support Hypothesis 6 because the CI coverage was not systematically closer to nominal for MIBEN than it was for the MICE-based methods.

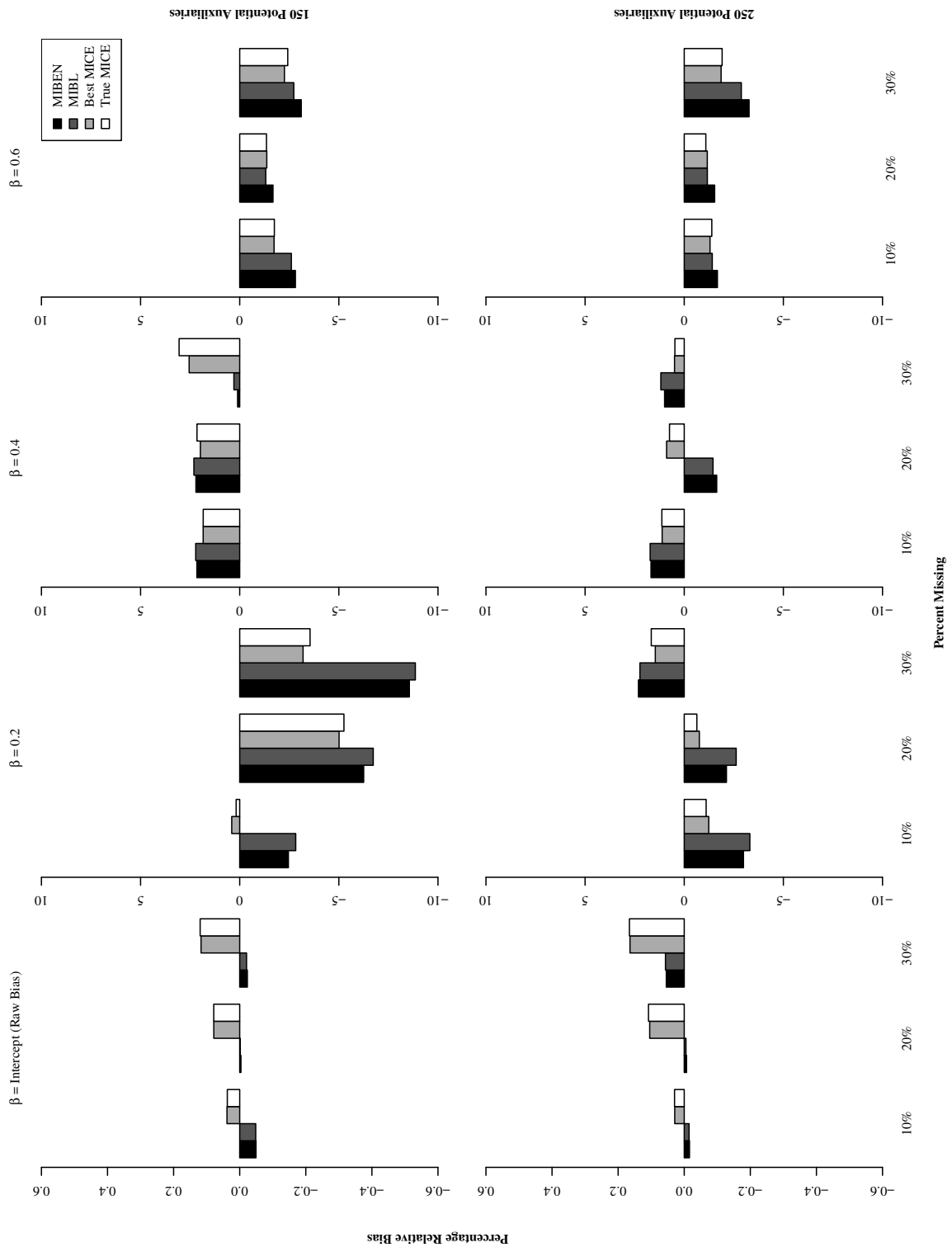

Figure 3.8: Percentage relative bias for Experiment 2 sparse imputation models
Note: Raw bias is reported for the intercepts because their true values were $\beta_0 = 0$.

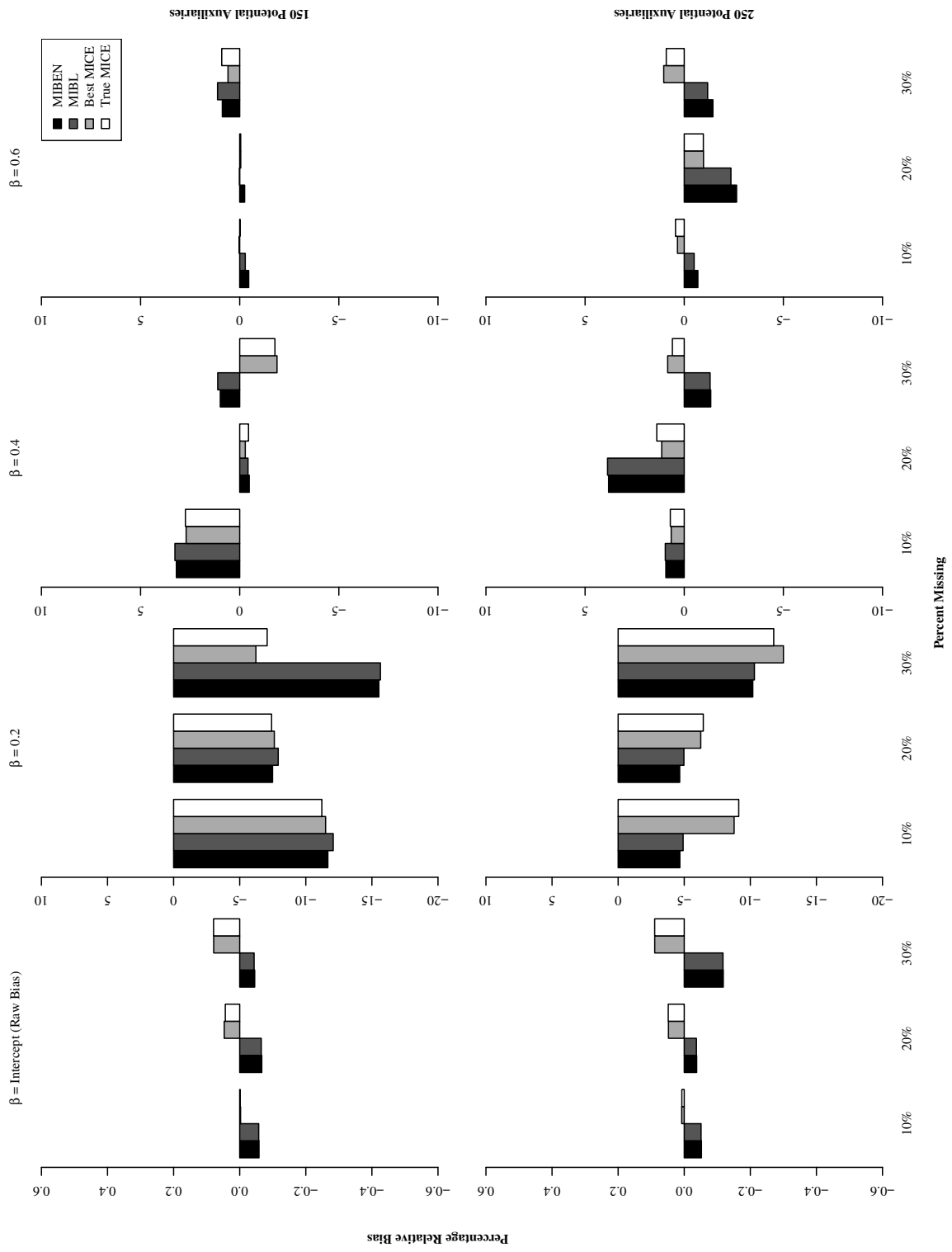

Figure 3.9: Percentage relative bias for Experiment 2 dense imputation models
Note: Raw bias is reported for the intercepts because their true values were $\beta_0 = 0$.

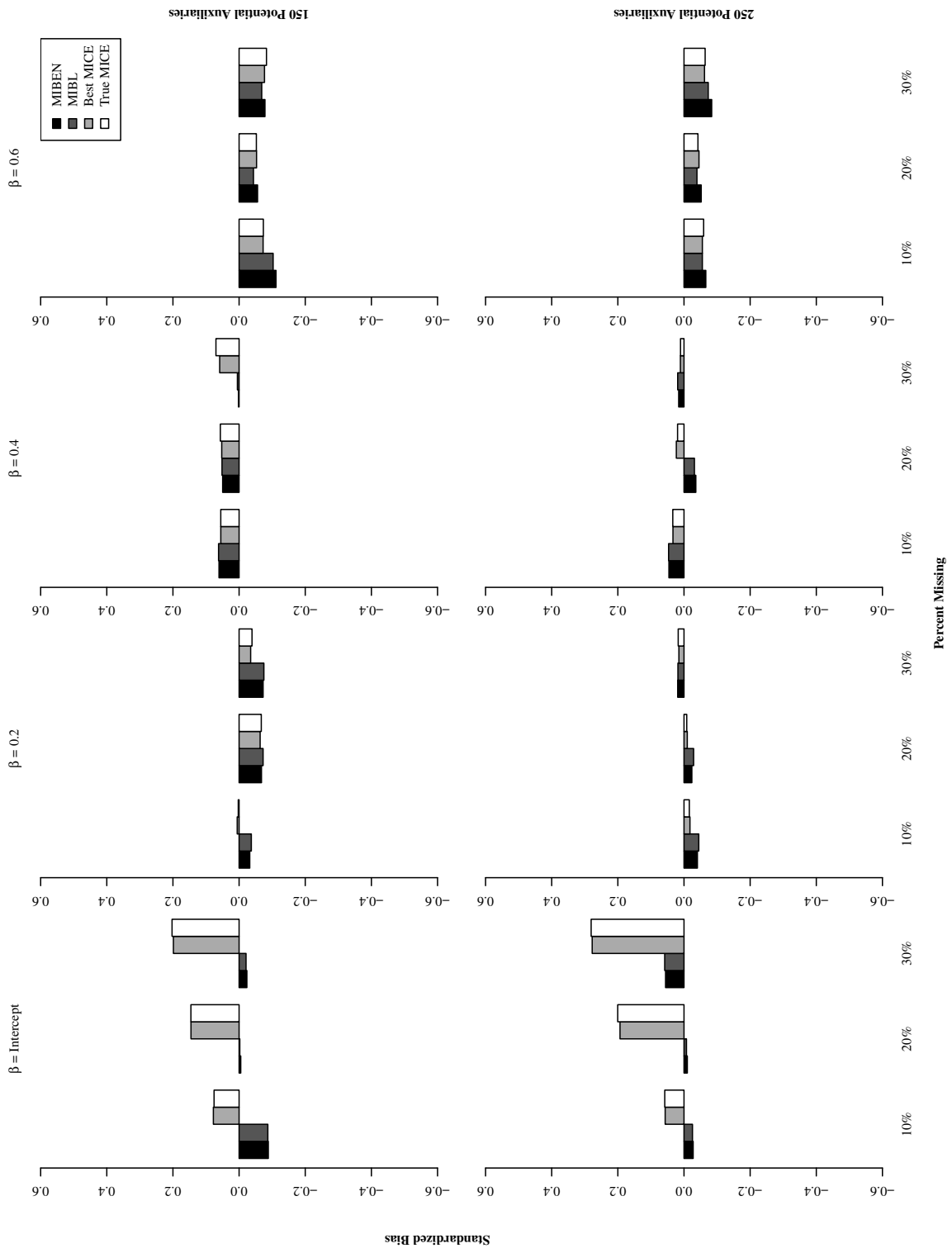

Figure 3.10: Standardized bias for Experiment 2 sparse imputation models

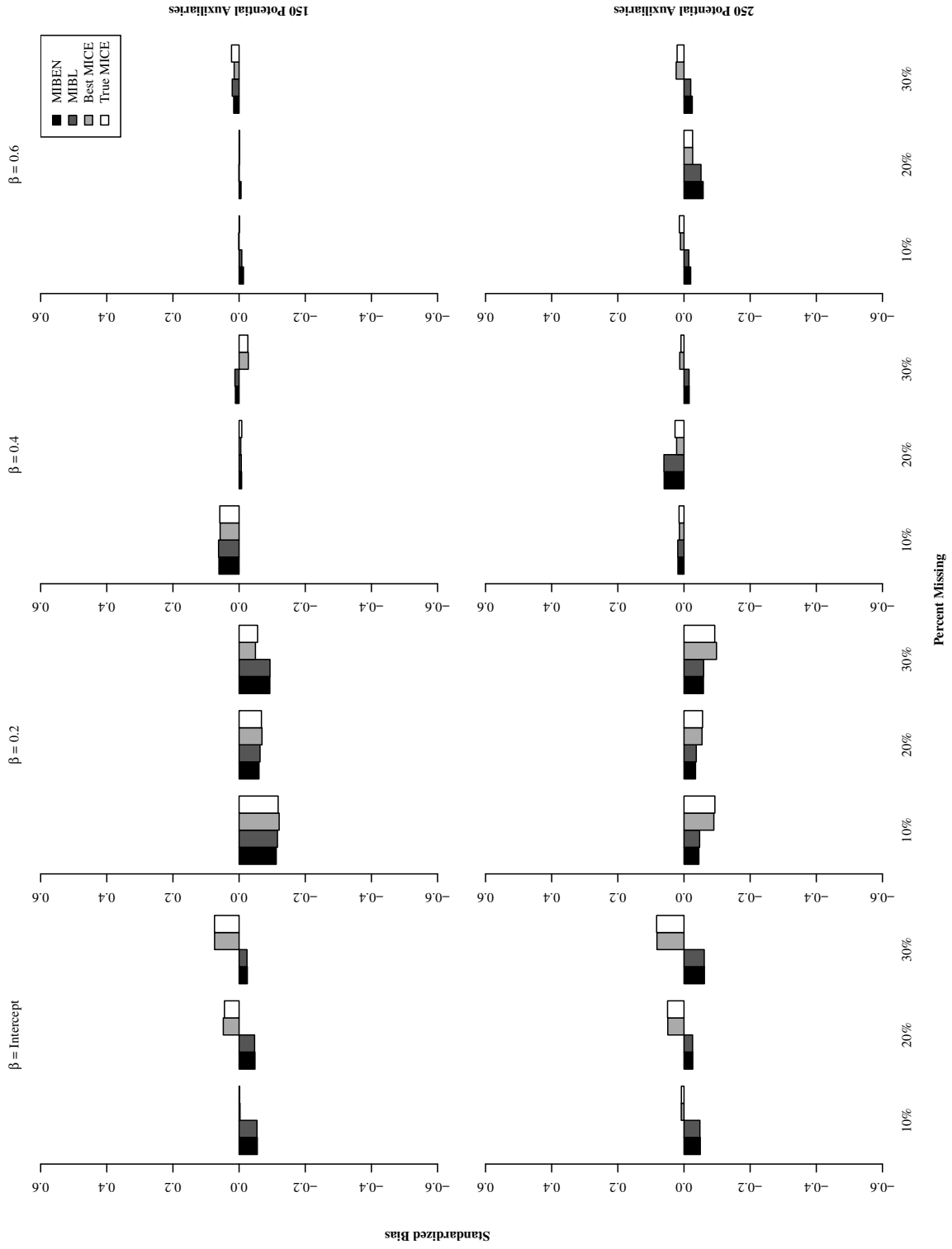

Figure 3.11: Standardized bias for Experiment 2 dense imputation models

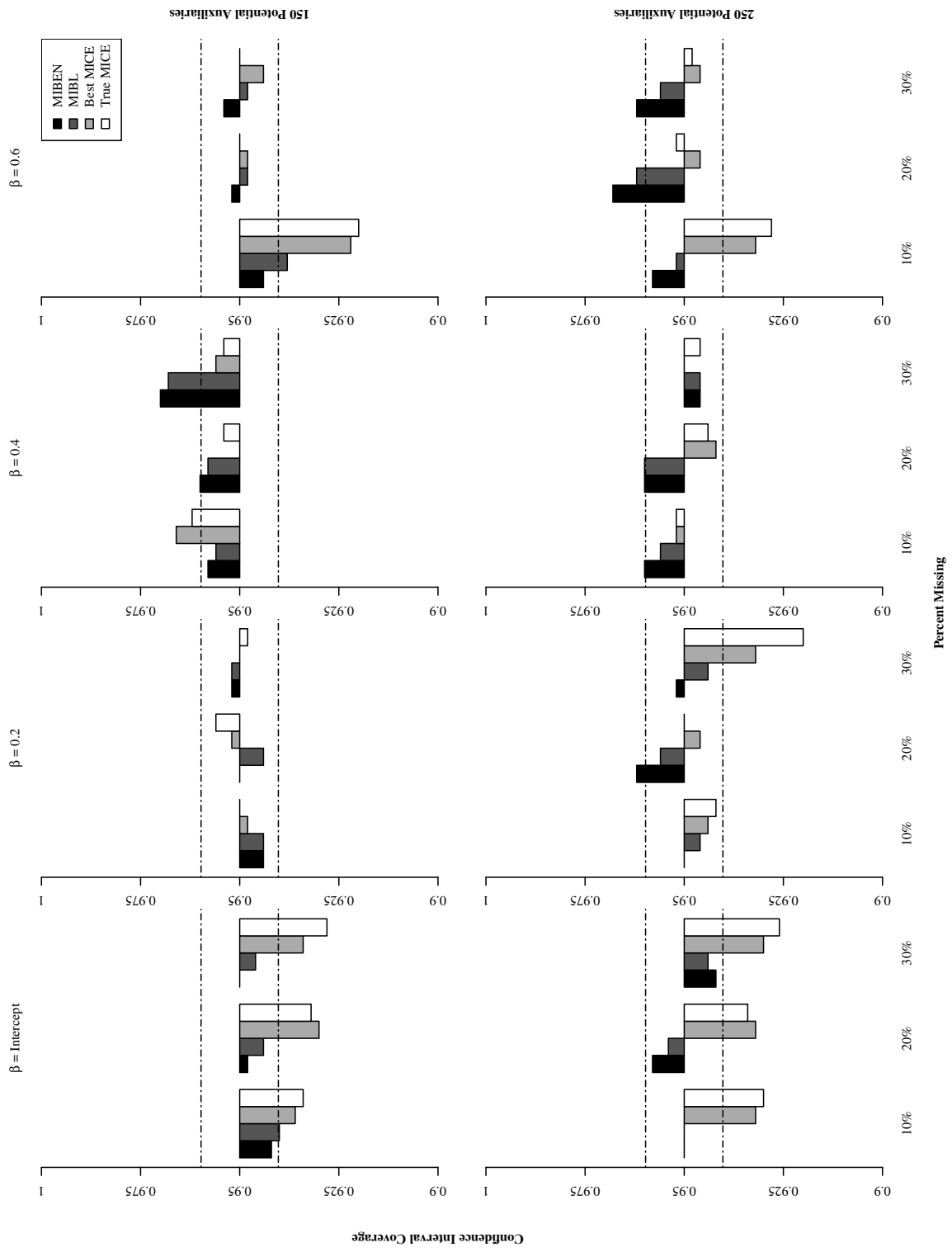

Figure 3.12: Confidence interval coverage for Experiment 2 sparse imputation models
Note: Dashed lines represent $.95 \pm 2 \times SE(p)$

Figure 3.13: Confidence interval coverage for Experiment 2 dense imputation models
Note: Dashed lines represent $.95 \pm 2 \times SE(p)$

3.4 General Findings

In Experiment 1, Hypothesis 1 was mostly supported as MIBEN led to universally unbiased parameter estimates but the coverage rates for some of those parameters were somewhat too high. This problematic CI coverage was most pronounced for the large effects with sparse imputation models. In Experiment 2, Hypothesis 1 was also partially supported. MIBEN produced better CI coverage rates in Experiment 2 than it did in Experiment 1, but it led to problematic degrees of PRB for the small effects ($\beta = 0.2$) with dense imputation models. In terms of SB, however, MIBEN led to unbiased parameter estimates for all conditions. Graham (2012) noted that PRB can exaggerate the absolute bias in small effects. He suggested favoring SB over PRB when judging the practical importance of bias in small effects, if there is disagreement between the two metrics. Therefore, it could be argued that MIBEN led to universally unbiased parameter estimates, at least from the perspective of practical importance. As discussed below, the inconsistency between PRB and SB also reflects sensible Bayesian behavior on the part of MIBEN.

Two overall patterns permeated all of the experimental conditions: (1) MIBEN and MIBL produced nearly identical results, and (2) MIBEN and MIBL produced more variable results than the MICE-based approaches produced. The first pattern means that Hypotheses 8 and 9 are definitively rejected since MIBEN did not systematically outperform MIBL in any condition tested for this project. Finding that MIBEN and MIBL produced more variable solutions that the MICE-based methods was somewhat surprising and clearly rejects Hypothesis 2 since the additional variability made MIBEN's CIs universally wider than those derived from MICE-based methods. Figures 3.14 and 3.15 contain plots of each method's average CI width for Experiment 1's sparse and dense models, respectively, and Figures 3.16 and 3.17 contain analogous plots from Experiment 2. As a consequence of producing more variable analysis model fits than the MICE-based methods, MIBEN and MIBL also had higher Monte Carlo variability than the MICE-based methods did. This higher simulation variability can be seen in Figures 3.18, 3.19, 3.20, and 3.21 which show plots of the SD_{MC} for each method in Experiment 1 sparse models, Experiment 1 dense models, Experiment 2 sparse models, and Experiment 2 dense models, respectively.

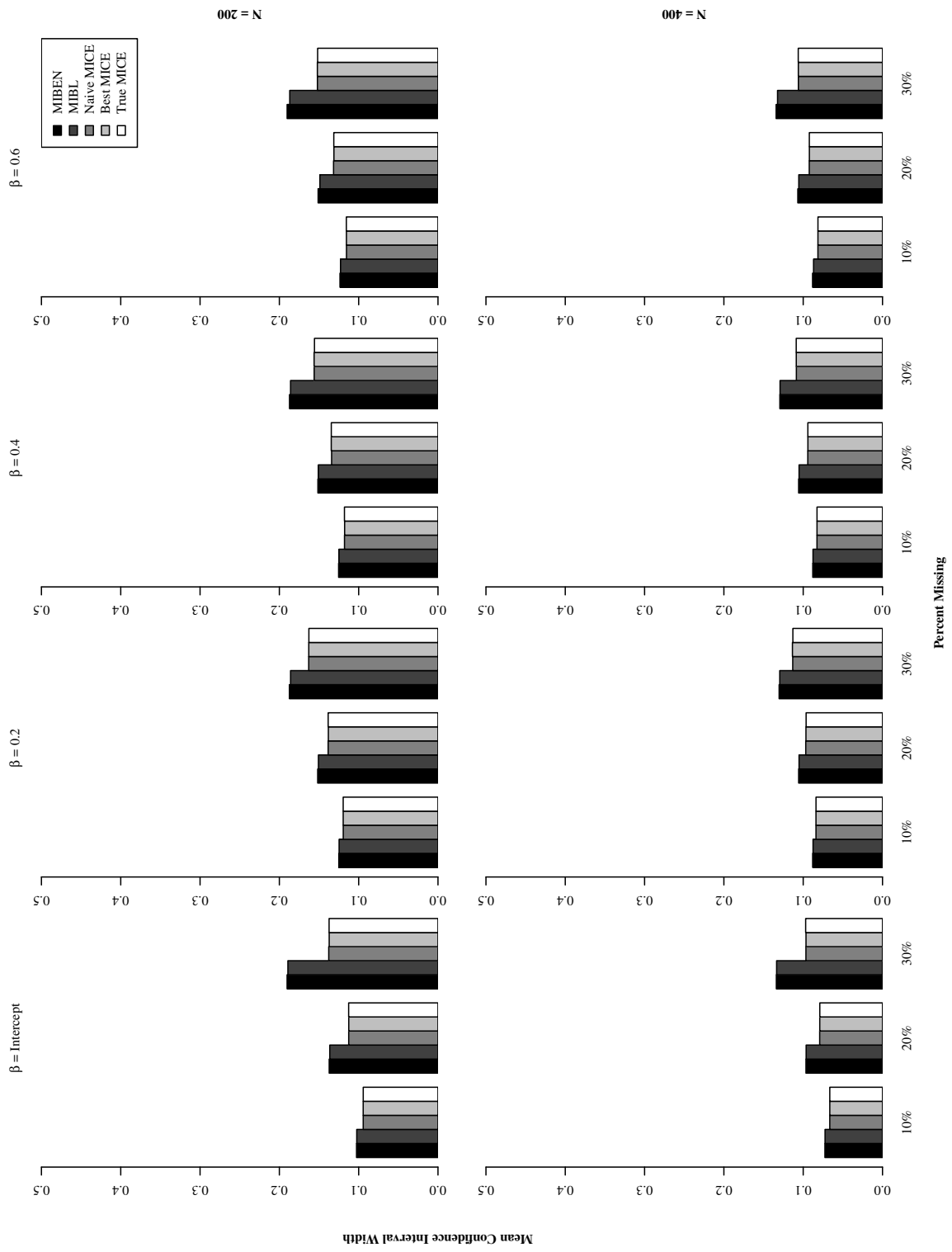

Figure 3.14: Mean confidence interval width for Experiment 1 sparse imputation models

Figure 3.15: Mean confidence interval width for Experiment 1 dense imputation models

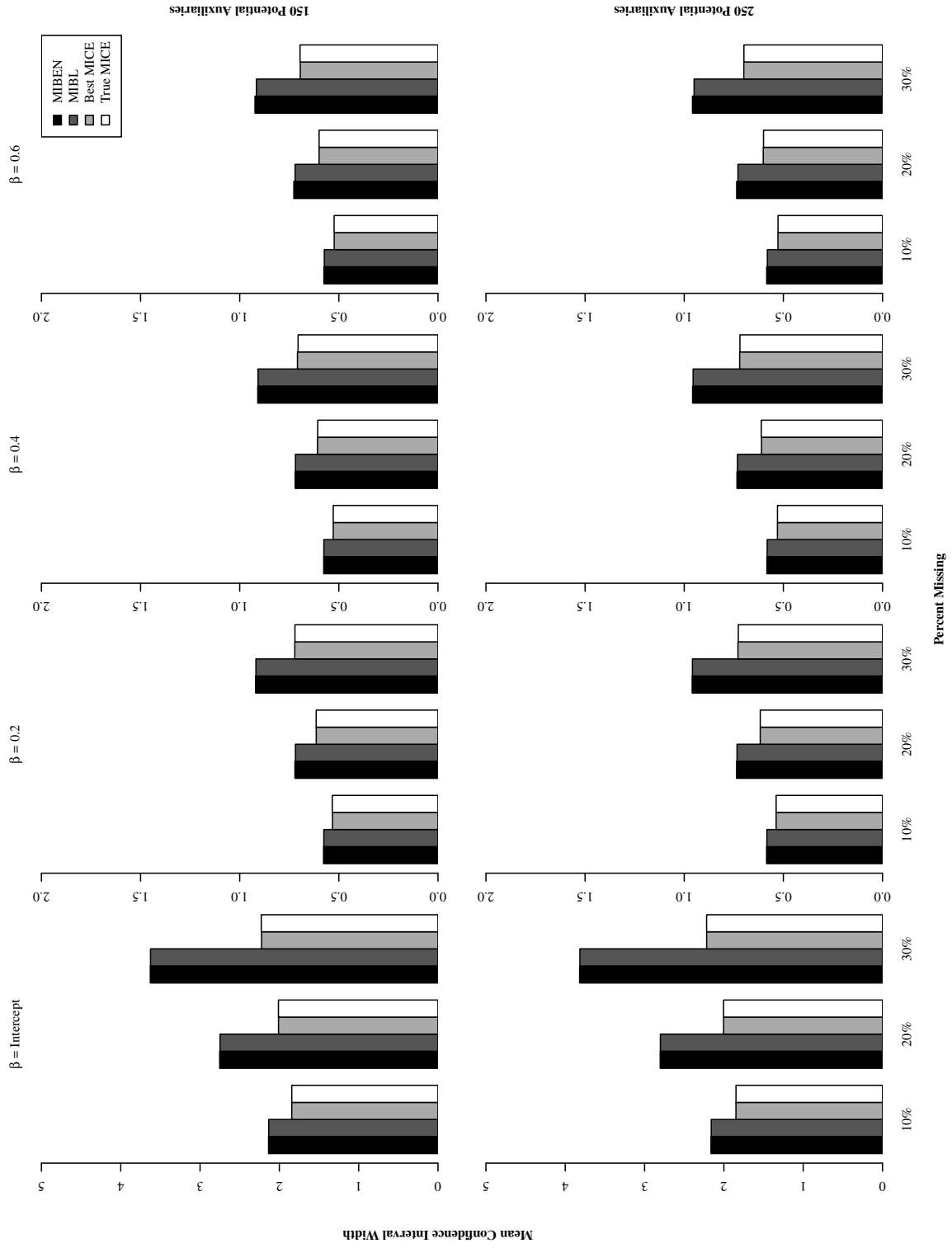

Figure 3.16: Mean confidence interval width for Experiment 2 sparse imputation models

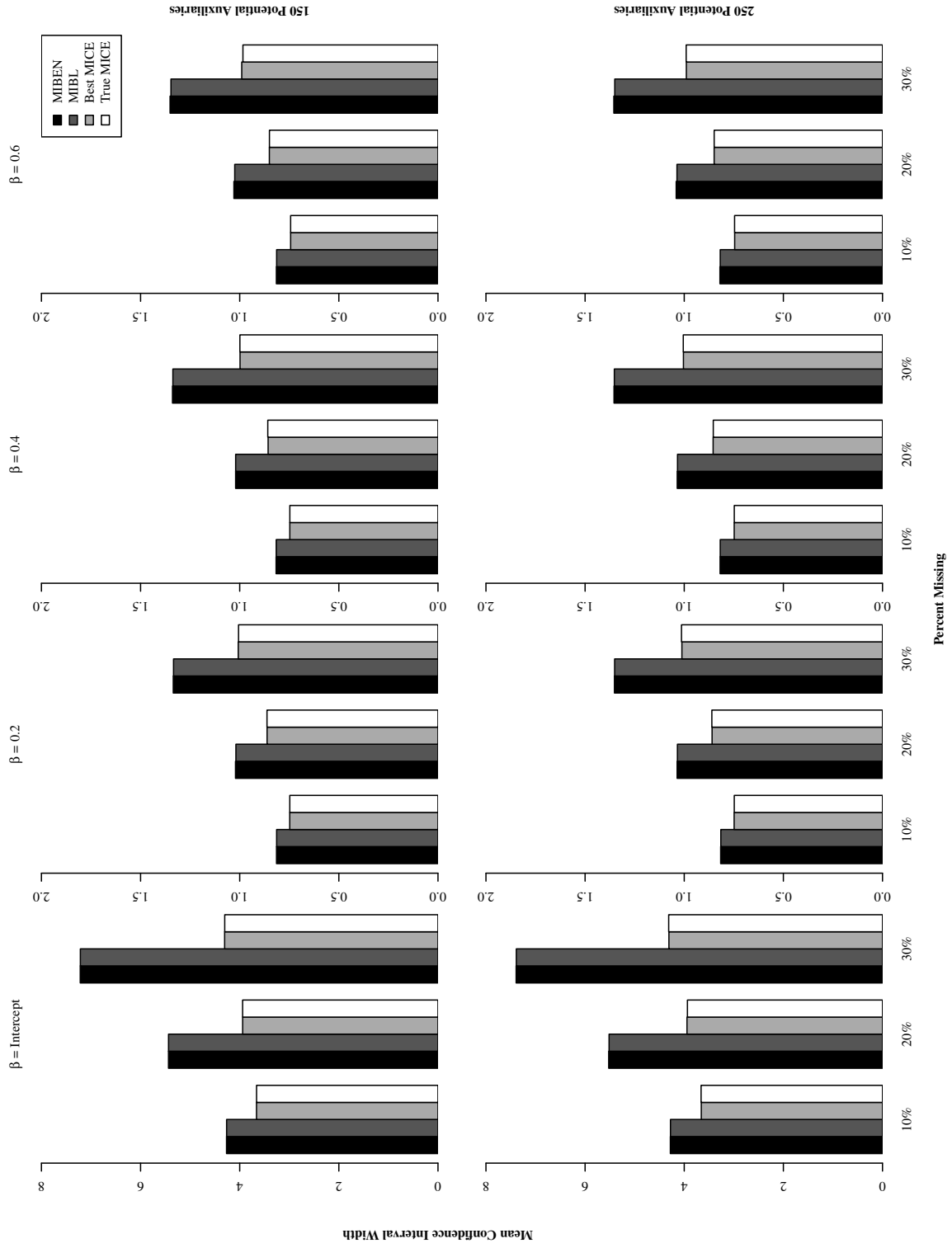

Figure 3.17: Mean confidence interval width for Experiment 2 dense imputation models

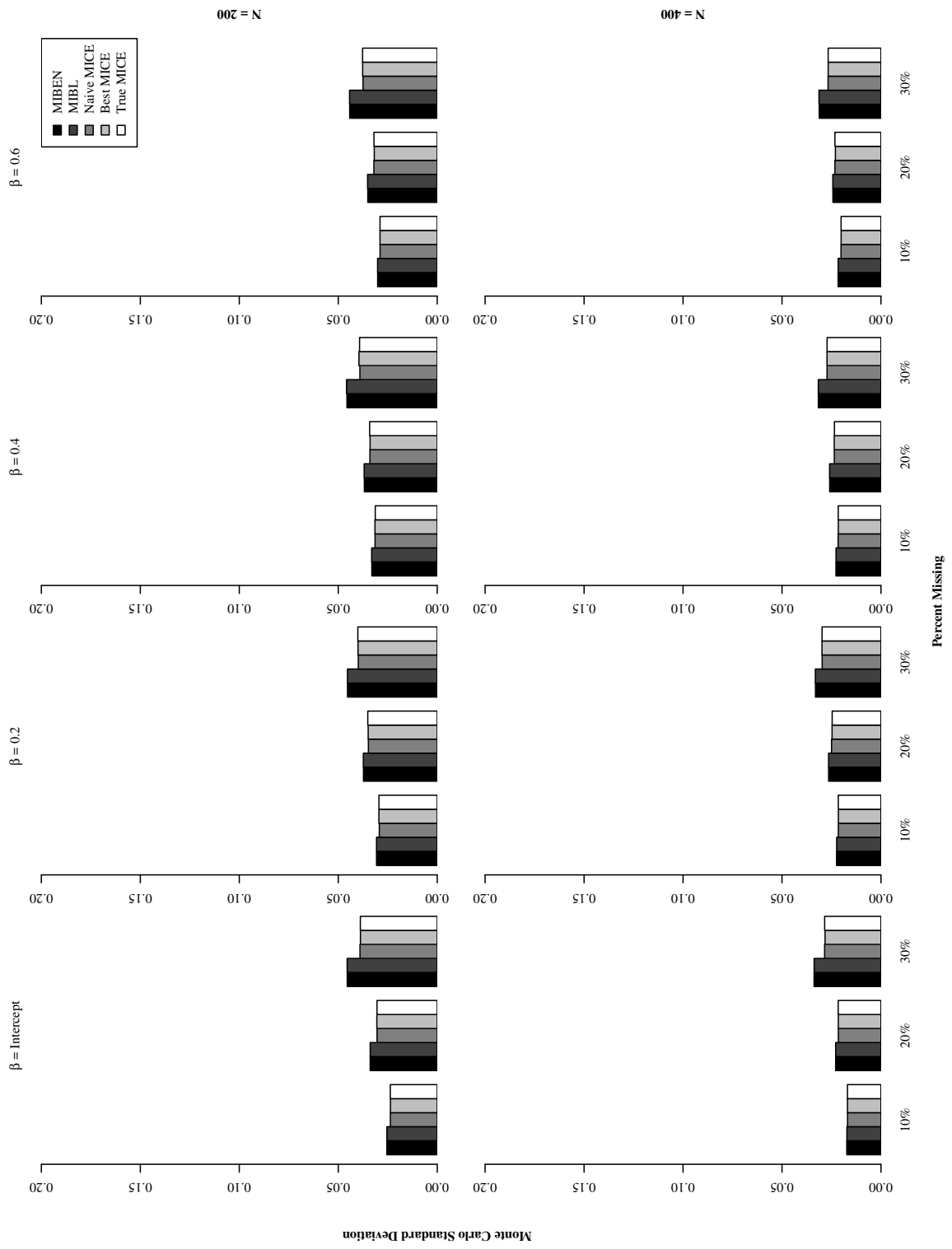

Figure 3.18: Monte Carlo SD for Experiment 1 sparse imputation models

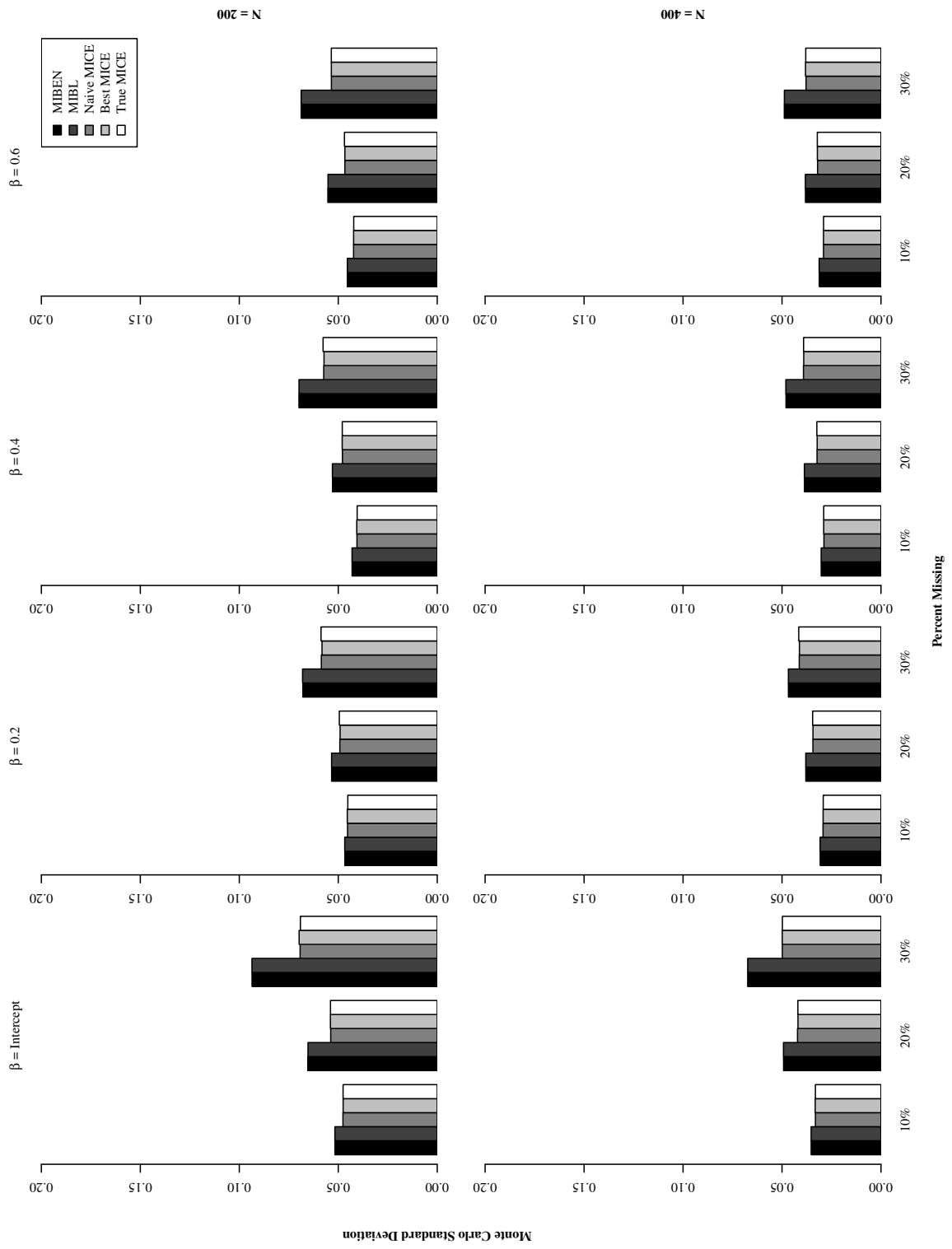

Figure 3.19: Monte Carlo SD for Experiment 1 dense imputation models

Figure 3.20: Monte Carlo SD for Experiment 2 sparse imputation models

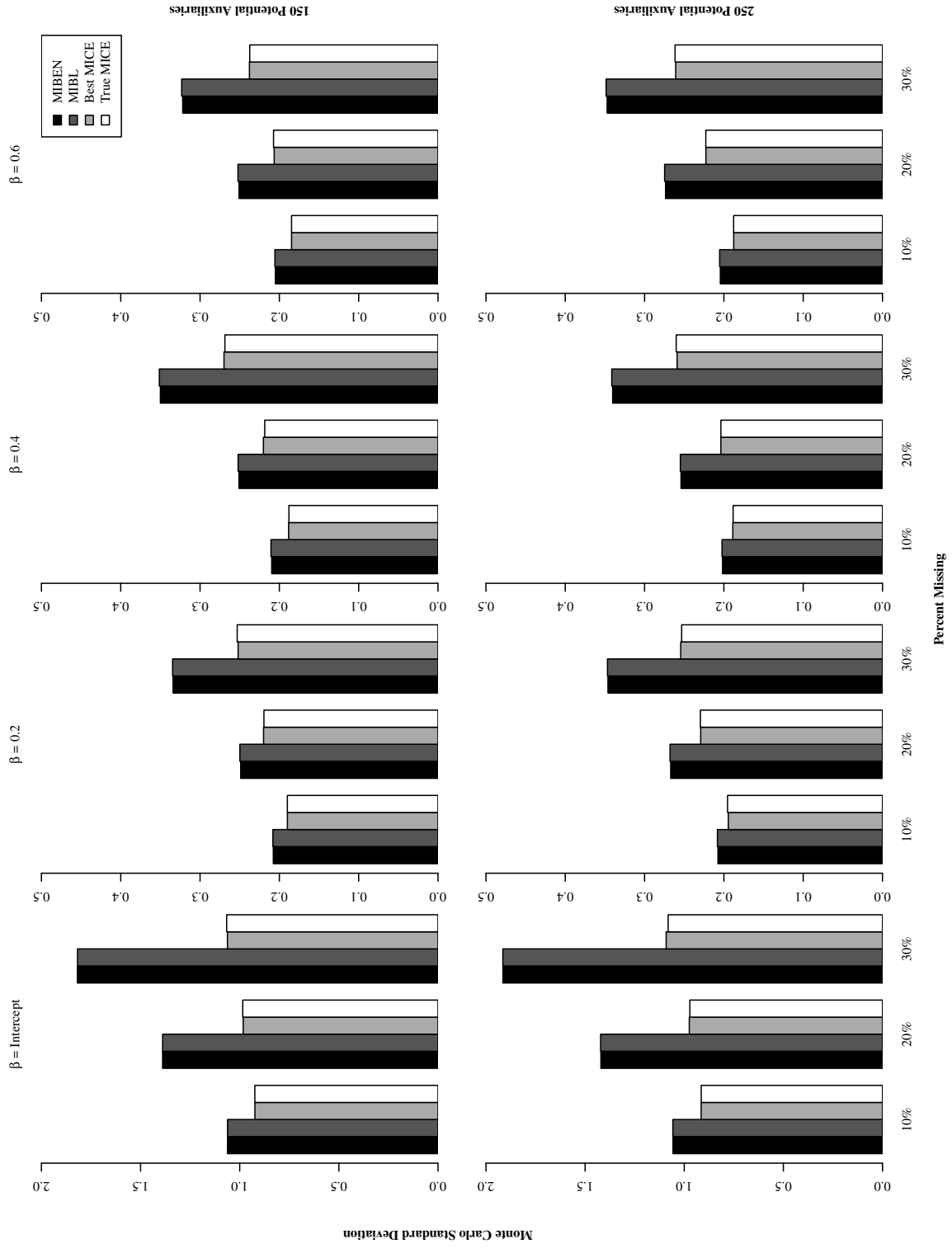

Figure 3.21: Monte Carlo SD for Experiment 2 dense imputation models

Chapter 4

Discussion

I have introduced Multiple Imputation with the Bayesian Elastic Net (MIBEN), and examined its performance with a Monte Carlo simulation study in which MIBEN was compared to an adaptation of the Zhao and Long (2013) Bayesian LASSO-based MI method (MIBL) and MICE with predictors chosen through three different model selection schemes. Each method was judged based on its ability to produce multiple imputations that facilitated accurate estimation of the parameters of a linear regression model and produced well-calibrated confidence intervals for those estimates.

In terms of the conditions explored for this study, MIBEN performed well and led to satisfactory results, overall. MIBEN and MIBL produced nearly identical results, but the MCEM algorithm underlying MIBEN converged much faster, and more decisively, than the version employed by MIBL. The importance of this superior convergence behavior should not be underestimated as both of these methods entail a considerable computational expense. In terms of parameter bias, there was no clearly superior method except when estimating the intercepts with sparse imputation models where MIBEN and MIBL performed considerably better than the MICE-based methods did. On the other hand, MIBEN and MIBL did tend to induce somewhat larger relative biases in the estimates of the small effects than the MICE-based methods did.

The two systematic patterns in the parameter bias (i.e., poorly estimated intercepts with

MICE-based methods and poorly estimated small effects with MIBEN/MIBL), can both be explained by the influence of the underlying model regularization. The simulated data were very noisy, and the outcome data had the lowest signal to noise ratio of all target variables because of the two-stage data generation process (see Section 2.2.1). The true value for the intercept was also $\beta_0 = 0$. Taking both of these aspects into consideration it becomes clear that accurately estimating the intercepts was actually a very challenging task. The MICE-based imputations appear to have been systematically "off the mark" in that they consistently implied a non-zero conditional mean for the outcome variable. On the other hand, the very strong regularization inherent in MIBEN and MIBL is ideally suited to overcoming the high rates of noise in the outcome variable and creating imputations that correctly implied a conditional mean of zero for the outcome variable. This behavior is not an artifact of this simulation's data generation process, either. MIBEN's strong regularization should always facilitate accurate imputations of the outcome's conditional mean because MIBEN's data pre-processing module centers all target variables before imputation. Thus, the correct conditional mean for each of MIBEN's elementary imputation equations will be $\beta_0 = 0$, regardless of the value in the original incomplete data. When MIBL is implemented with these same data pre-processing steps (as was done for this study), the preceding rationale suggests that it should also impute correct estimates for the target variables' conditional means.

This high degree of regularization can be a double-edged sword, however, and it also explains the poorer performance of MIBEN and MIBL in facilitating estimation of small effects. The effects in question were $\beta = 0.2$ in the population, which will border on nonsignificance when the observed data are very noisy. In these cases MIBEN and MIBL seem to be somewhat over-regularizing the fitted imputation models and creating a larger than desired proportion of imputations that imply a zero effect when the true $\beta = 0.2$.

In terms of confidence interval coverage rates, MIBEN and MIBL tended to produce more desirable results than the MICE-based methods did. Although all methods induced sporadic coverage problems, the violation of MIBEN and MIBL were always conservative (i.e., overcoverage)

while those induced by MICE tended to be liberal (i.e., undercoverage). Though it is up to the end-user of these methods to judge whether Type I or Type II errors are more problematic for their particular application, I suggest that Type II errors are the lesser of two evils in most missing data problems.

Additionally, the coverage rates for the intercepts were much too low for the MICE-based approaches, while they were nearly nominal for MIBEN and MIBL. Thus, it appears that the MICE-based imputations implied a non-zero conditional mean for the outcome variable with a high degree of certainty, whereas MIBEN and MIBL not only produced imputations that accurately reflected the outcome's conditional mean, but also incorporate the uncertainty in that estimate in a sensible way. Although intercepts are often ignored in social science research, there are many situations in which accurately estimating the conditional mean of a given outcome is critically important. Any forecasting problem, for example, is absolutely dependent on being able to accurately infer the conditional mean of the forecasting quantity. In such applications, MIBEN or MIBL would be expected to perform much better than the MICE-based methods examined here.

MIBEN and MIBL produced more variable estimates than the MICE-based methods did. This increased variability led to wider confidence intervals and higher Monte Carlo SDs for MIBEN and MIBL than for the MICE-based methods. While this additional variability may seem undesirable, I suggest that it is actually indicative of sensible behavior for the circumstances of the current study. MI is a fundamentally Bayesian procedure, regardless of how the imputations themselves are created, and one of the most important aspects of any Bayesian data analysis is quantifying uncertainty in the posterior estimates (Gelman et al., 2013). A well-calibrated imputation model should capture the additional uncertainty introduced by nonresponse (or sampling variability) by producing posterior distributions for the imputation model parameters that are more diffuse than those that would have been estimated for completely observed (or low-noise) data. These widened posteriors, by extension, should induce posterior predictive distribution of the missing data that have higher variability than the completely observed data's posterior

distribution would have exhibited (Rubin, 1987). In practice, these wider posteriors will increase the variability of the imputations and induce higher levels of between imputation variance, relative to situations with greater certainty about the imputation model's parameter estimates. This reasonable behavior is what MIBEN and MIBL seem to be demonstrating in this study.

While neither MIBEN and MIBL nor the MICE-based methods tended to induce much bias in the analysis model parameters, when such bias was present the MICE-based methods also tended to attribute too much certainty to the parameter estimate whereas MIBEN and MIBL "self-corrected" by inducing additional variance into the imputations to accurately reflect the uncertainty in any problematic estimates. This discrepancy is made clear by contrasting the percentage relative bias and standardized bias, particularly as shown in Figures 3.8 and 3.10, respectively. As Figure 3.8 shows, the MICE-based methods induced bias in the intercept estimates while MIBEN and MIBL tended to induce bias in the estimated small effects. When comparing this pattern to Figure 3.10, however, it becomes clear that MIBEN and MIBL have increased the variance in the imputations to account for their inherent limitations in estimating such small effects in the presence of noisy data—which is the appropriate response when faced with high levels of uncertainty in the estimated parameters. Thus, there are no appreciable differences in the small effects' standardized bias between methods. The MICE-based methods, on the other hand, appear to have metaphorically "doubled down" on there biased estimates of the intercepts by failing to increase the between imputation variance to account for the parameter bias. Thus, the standardized bias in the intercept estimates reported in Figure 3.10 is even more severe than the raw bias reported in Figure 3.8.

This same pattern carries over to the CI coverage rates. While neither MIBEN and MIBL nor the MICE-based methods produce universally nominal coverage rates, the MICE-based methods tend to undercover the true parameters (suggesting too much certainty in the parameter estimates), while the MIBEN and MIBL methods tend to overcover the true parameters (suggesting low certainty in the parameter estimates). Again, with such noisy data and additional missing data compounding the uncertainty, I suggest that the behavior of MIBEN and MIBL is more

sensible under these circumstances.

The similar performance of MIBEN and MIBL was somewhat surprising but is sensible, in hindsight. Although the data generating models in experiment 2 were, potentially, highly underdetermined, their parametric structure was actually very simple. These models were straightforward regression models in which all potential auxiliary variables entered the model linearly and were linearly related to a normally distributed outcome variable. In such cases, it seems that the additional regularization provided by the Bayesian elastic net does not hold any appreciable advantage over the simpler regularization of the Bayesian LASSO. The fact that MIBEN and MIBL performed equivalently here validates MIBEN's baseline performance. Future work will explore more complex data generating models for which MIBEN is more uniquely well suited.

4.1 Limitations & Future Directions

The study presented here was necessarily limited in its scope. Future research will explore more complex data generating models to elucidate the circumstances in which MIBEN can outperform MIBL. An important first step will be to introduce correlations among the auxiliary variables. If the auxiliaries are subject to multicollinearity, the ridge-type penalization of the BEN may become more crucial, and MIBEN could considerably outperform MIBL. Another key complication will be to include nonlinear functions of the auxiliaries (e.g., interactions and polynomial terms) into the systematic component of the data generating model. Simulating data with a "grouped" item structure (e.g., by using a factor analytic model to generate the data) may also bring out some of the unique strengths of MIBEN relative to MIBL.

The MIBEN method and the **mibrr** R package are also in active development, and their capabilities are currently being extended. A key improvement will be incorporating the ability to impute incomplete categorical variables. The performance of MICE-based methods is not nearly as clear-cut when dealing with incomplete categorical data (Lang, 2014; Wu et al., 2014), so MIBEN may dramatically outperform currently available methods in this domain. The way that

missingness on the auxiliary variables is treated must also be considered. Although the method employed here (e.g., pairwise deletion through zero-imputation) worked well for the current study, it may induce bias when the auxiliary variables are subject to MAR missingness, rather than the MCAR missingness simulated here. Future work will test this possibility and consider alternative strategies for addressing missing data on the auxiliary variables. Finally, methods for *online* convergence monitoring will be considered. Actively monitoring the convergence of both the Gibbs samples and the penalty parameters' MCEM chains could dramatically improve the computational efficiency of MIBEN by avoiding the use of wastefully long Markov Chains.

4.2 Conclusion

I have introduced MIBEN, a novel multiple imputation scheme that leverages the strengths of the Bayesian elastic net to create principled multiple imputations of normally distributed missing data under the assumption of MAR missingness. The MIBEN method was found to perform as well as or better than current state-of-the-art techniques based on normal-theory MI within a MICE framework and MI based on the Bayesian LASSO. Although MIBEN is not yet ready for production-level deployment, the results of this study show that the MIBEN method has great promise, and future research will work to maximize this potential and extend MIBEN into a very powerful and highly featured missing data tool.

References

Anderson, T. W. (1957). Maximum likelihood estimates for a multivariate normal distribution when some observations are missing. *Journal of the American Statistical Association*, *52*(278), 200–203.

Barcena, M. J., & Tusell, F. (2004). Fitting multivariate responses using scalar trees. *Statistics & Probability Letters*, *69*(3), 253–259. doi: 10.1016/j.spl.2004.06.012

Brent, R. P. (1973). *Algorithms for minimizatin without derivatives*. Englewood Cliffs, NJ: Prentice-Hall. (Reprinted by Dover, 2002)

Burton, A., Altman, D. G., Royston, P., & Holder, R. L. (2006). The design of simulation studies in medical statistics. *Statistics in Medicine*, *25*(24), 4279–4292. doi: 10.1002/sim.2673

Carpenter, J. R., & Kenward, M. G. (2012). *Multiple imputation and its applications*. John Wiley & Sons.

Casella, G. (2001). Empirical Bayes Gibbs sampling. *Biostatistics*, *2*(4), 485–500.

Chen, M., Carlson, D., Zaas, A., Woods, C., Ginsburg, G. S., Hero, I., Alfred, ... Carin, L. (2009). *The Bayesian elastic net: Classifying multi-task gene-expression data* (Tech. Rep.). Duke University. (Retrieved from http://people.ee.duke.edu/ lcarin/BayesianEnet.pdf)

Cheng, K.-Y., Mao, Q.-R., Tan, X.-Y., & Zhan, Y.-Z. (2011). A novel sparse learning method: Compressible Bayesian elastic net model. *Journal of Information and Computing Science*, *6*(4), 295–302.

Chhikara, R. S., & Folks, J. L. (1988). *The inverse Gaussian distribution: Theory, methodology, and applications*. Boca Raton, FL: CRC Press.

Collins, L. M., Schafer, J. L., & Kam, C. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods*, *6*(4), 330–351. doi: 10.1037//1082-989X.6.4.330

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1–38. doi: 10.2307/2984875

Dempster, A. P., Schatzoff, M., & Wermuth, N. (1977). A simulation study of alternatives to ordinary least squares. *Journal of the American Statistical Association*, *72*(357), 77–91.

Drechsler, J. (2010). Using support vector machines for generating synthetic datasets. In *Proceedings of the 2010 international conference on privacy in statistical databases* (pp. 148–161).

Eddelbuettel, D., & François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*(8), 1–18. Retrieved from `http://www.jstatsoft.org/v40/i08/`

Enders, C. K. (2010). *Applied missing data analysis*. New York, NY: The Guilford Press.

Feng, L., Nowak, G., Welsh, A. H., & O'Neill, T. J. (2014, July). A general imputation framework in R [Computer software manual]. Retrieved from `http://cran.r-project.org/web/packages/imputeR/index.html` (R package version 1.0.0)

Gefang, D. (2014). Bayesian doubly adaptive elastic-net lasso for VAR shrinkage. *International Journal of Forecasting*, *30*(1), 1–11.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis* (3rd ed.). Boca Raton, FL: CRC press.

Goldstein, M. (1976). Bayesian analysis of regression problems. *Biometrika*, *63*(1), 51–58.

Graham, J. W. (2003). Adding missing-data-relevant variables to fiml-based structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *10*(1), 80–100. doi: 10.1207/S15328007SEM1001_4

Graham, J. W. (2009). Missing data analysis: Making it work in the real world. *Annual review of psychology*, *60*, 549–576.

Graham, J. W. (2012). *Missing data: Analysis and design*. New York, NY: Springer. doi: 10.1007/

978-1-4614-4018-5

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning* (2nd ed.). New York, NY: Springer. doi: 10.1007/978-0-387-84858-7

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Honaker, J., & King, G. (2010). What to do about missing values in time-series cross-section data. *American Journal of Political Science*, *54*(2), 561–581. doi: 10.1111/j.1540-5907.2010.00447.x

Howard, W., Rhemtulla, M., & Little, T. D. (in press). Using principal components as auxiliary variables in missing data estimation. *Multivariate Behavioral Research*.

Johnson, S. G. (2014, May). *The NLopt nonlinear-optimization package*. Retrieved from `http://ab-initio.mit.edu/nlopt` (C++ package version 2.4.2)

Jørgensen, B. (1982). *Statistical properties of the generalized inverse Gaussian distribution*. New York, NY: Springer. doi: 10.1007/978-1-4612-5698-4

Lang, K. M. (2014). What to do with incomplete nominal variables? A Monte Carlo simulation study comparing methods for creating multiple imputations of unordered factors. In K. M. Lang (Chair). *Practical Issues in Missing Data Analysis*. Symposium conducted at the Modern Modeling Methods (M^3) Conference, Storrs, CT.

L'ecuyer, P., Simard, R., Chen, E. J., & Kelton, W. D. (2002). An object-oriented random-number package with many long streams and substreams. *Operations Research*, 1073–1075. doi: 10.1287/opre.50.6.1073.358

Li, Q., & Lin, N. (2010). The Bayesian elastic net. *Bayesian Analysis*, *5*(1), 151–170.

Little, R. J. A., & Rubin, D. B. (2002). *Statistical analysis with missing data* (2nd ed.). Hoboken, NJ: Wiley-Interscience.

Little, T. D., Jorgensen, T. D., Lang, K. M., & Moore, E. W. G. (2013). On the joys of missing data. *Journal of Pediatric Psychology*, 1–12. doi: 10.1093/jpepsy/jsto48

Little, T. D., Lang, K. M., Wu, W., & Rhemtulla, M. (in press). Missing data. In D. Cicchetti (Ed.), *Developmental psychopathology* (3rd ed., pp. 000–000). New York, NY: Wiley.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, *45*(1-3), 503–528.

Muthén, B., Kaplan, D., & Hollis, M. (1987). On structural equation modeling with data that are not missing completely at random. *Psychometrika*, *52*(3), 431–462.

Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, *35*(151), 773–782.

Park, T., & Casella, G. (2008). The Bayesian LASSO. *Journal of the American Statistical Association*, *103*(482), 681–686.

Powell, M. J. D. (1994). A direct search optimization method that models the objective and constraint functions by linear interpolation. In S. Gomez & J.-P. Hennart (Eds.), *Advances in optimization and numerical analysis* (Vol. 275, pp. 51–67). Boston: Kluwer Academic Publishers.

Powell, M. J. D. (2009). *The BOBYQA algorithm for bound constrained optimization without derivatives* (Tech. Rep. No. NA2009/06). Department of Applied Mathematics and Theoretical Physics, Cambridge England.

R Core Team. (2014). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `http://www.R-project.org/`

Raghunathan, T. E., Lepkowski, J. M., Van Hoewyk, J., & Solenberger, P. (2001). A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, *27*(1), 85–96.

Rowan, T. (1990). *Functional stability analysis of numerical algorithms*. Ph.d. thesis, University of Texas at Austin.

Rubin, D. B. (1978). Multiple imputations in sample surveys – A phenomenological Bayesian approach to nonresponse. In *Proceedings of the section on survey research methods* (pp. 20–28).

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys* (Vol. 519). New York, NY: John Wiley & Sons.

Rubin, D. B. (1996). Multiple imputation after 18+ years. *Journal of the American Statistical Association*, *91*(434), 473–489.

Savalei, V., & Rhemtulla, M. (2011). *Some explorations of the local and global measures of missing information* Paper presented at the 76th annual and the 17th international meeting of the Psychometric Society, Hong Kong.

Schafer, J. L. (1997). *Analysis of incomplete multivariate data* (Vol. 72). Boca Raton, FL: Chapman & Hall/CRC.

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of state of the art. *Psychological Methods*, *7*(2), 147–177. doi: 10.1037//1082-989X.7.2.147

Sevcikova, H., & Rossini, T. (2012). rlecuyer: R interface to RNG with multiple streams [Computer software manual]. Retrieved from `http://CRAN.R-project.org/package=rlecuyer` (R package version 0.3-3)

Svanberg, K. (2002). A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, *12*(2), 555–573.

Tanner, M. A., & Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, *82*(398), 528–540.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

van Buuren, S. (2012). *Flexible imputation of missing data.* Boca Raton, FL: CRC Press. doi: 10.1201/b11826

van Buuren, S., Brand, J. P. L., Groothuis-Oudshoorn, C. G. M., & Rubin, D. B. (2006). Fully conditional specification in multivarite imputation. *Journal of Statistical Computation and Simulation*, *76*(12), 1049–1064.

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, *45*(3), 1–67.

Vlček, J., & Lukšan, L. (2006). Shifted limited-memory variable metric methods for large-scale unconstrained optimization. *Journal of Computational and Applied Mathematics*, *186*(2),

365–390.

Von Hippel, P. T. (2007). Regression with missing Ys: An improved strategy for analyzing multiply imputed data. *Sociological Methodology*, *37*(1), 83–117.

Von Hippel, P. T. (2009). How to impute interactions, squares, and other transformed variables. *Sociological Methodology*, *39*(1), 265–291.

Wu, W., Jia, F., & Enders, C. K. (2014). A comparison of imputation strategies to ordinal categorical data. *Multivariate Behavioral Research*. (Manuscript under review)

Yang, H., Dunson, D., & Banks, D. (2011). *The multiple Bayesian elastic net* (Tech. Rep.). Duke University. Retrieved from `http://stat.duke.edu/~hy35/MBEN.pdf`

Zhao, Y., & Long, Q. (2013). Multiple imputation in the presence of high-dimensional data. *Statistical Methods in Medical Research*, XXX–XXX. doi: 10.1177/0962280213511027

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320.

Appendix A

R Code for Key Functions

Code Listing A.1: Function to simulate complete data

```
simData <- function(control, parms)
{
    require(mvtnorm)
    nObs <- parms$maxObs
    nTargets <- length(parms$targetNames)
    nAux <- parms$maxAux
    zColin <- parms$auxColin
    sparse <- control$sparse
    xSNR <- parms$xSNR
    ySNR <- parms$ySNR

    zSigma <- matrix(zColin, nAux, nAux)
    diag(zSigma) <- 1.0
    Z <- rmvnorm(nObs, rep(0, nAux), zSigma)

    if(sparse) {
        Zeta <- rbind(
            matrix(runif((nTargets - 1) * (nAux / 2), 0.25, 0.5),
                   ncol = nTargets - 1),
            matrix(rep(0, (nTargets - 1) * (nAux / 2) ),
                   ncol = nTargets - 1)
            )
    } else {
        Zeta <- matrix(runif((nTargets - 1) * nAux, 0.25, 0.5),
                       ncol = nTargets - 1)
    }
```

```
    noiseVar <- (1 / xSNR) * (
        t(Zeta) %*% cov(Z) %*% Zeta * diag(nTargets - 1)
        )
    X <- Z %*% Zeta +
        rmvnorm(nObs, rep(0, nTargets - 1), noiseVar)

    beta <- c(0.2, 0.4, 0.6)
    noiseVar <- (1 / ySNR) * (t(beta) %*% cov(X) %*% beta)
    y <- X %*% beta + rnorm(nObs, 0, noiseVar)
    Y <- cbind(y, X)
    outData <- data.frame(Y, Z)
    colnames(outData) <-
        c( parms$targetNames, paste0( "z", c(1 : nAux) ) )

    list(data = outData, trueBeta = beta)
}# END simData()
```

Code Listing A.2: Function to impose missing data

```
imposeMissing <- function(inData, control, parms)
{
    targetNames <- parms$targetNames
    auxNames <- colnames(inData)[!colnames(inData) %in% targetNames]
    nObs <- nrow(inData)
    nTrueAux <- control$nTrueAux
    pmAux <- parms$pmAux
    sparse <- control$sparse
    pmTarget <- control$pm

    trueAux <- list()
    for(i in 1 : length(targetNames)) {
        if(sparse) {
            trueAux[[i]] <-
                sample(auxNames[1 : (length(auxNames) / 2)], nTrueAux)
        } else {
            trueAux[[i]] <- sample(auxNames, nTrueAux)
        }
    }

    if(parms$missingAux) {
        ## Impose MCAR missingness on the auxiliaries:
        imposeMCAR <- function(x, pm)
```

```
        {
            missFlag <- rbinom(length(x), 1, pm)
            x[as.logical(missFlag)] <- NA
            x
        }# END imposeMCAR()

        incAuxData <-
            apply(inData[ , auxNames], 2, FUN = imposeMCAR, pm = pmAux)
        colnames(incAuxData) <- auxNames
    } else {
        incAuxData <- inData[ , auxNames]
    }

    ## Impose MAR missingness on the target variables:
    imposeMAR <- function(x, inData, trueAux, pm, auxNames)
    {
        useAux <- trueAux[[x]]
        ksi <- c( rep(0.25, length(useAux) / 2),
                  rep(0.5, length(useAux) / 2) )
        linPred <- as.matrix(inData[ , useAux]) %*% ksi
        missPropensity <- pnorm(linPred, mean(linPred), sd(linPred))
        missPattern <- missPropensity >= 1 - pm
        inData[ , x][missPattern] <- NA
        inData[ , x]
    }# END imposeMAR()

    incTargetData <- sapply(c( 1 : length(targetNames) ),
                            FUN = imposeMAR,
                            inData = inData,
                            trueAux = trueAux,
                            pm = pmTarget,
                            auxNames = auxNames)
    colnames(incTargetData) <- targetNames

    list(data = data.frame(incTargetData, incAuxData),
         trueAux = trueAux)
}# END imposeMissing()
```

Code Listing A.3: Wrapper function to fit the analysis models

```
fitModels <- function(impData)
{
    require(mitools)
```

```
    fitMods <- function(x)
        {
            lm(y ~ x1 + x2 + x3, data = as.data.frame(x))
        }
    miFits <- lapply(impData, FUN = fitMods)
    MIcombine(miFits)
}# END fitModels()
```

Code Listing A.4: Function to compute the number of Monte Carlo replicates

```
calcReps <- function(betaMat, seMat, pm, acc, betaTrue)
{
    betaMCSD <- apply(betaMat, 2, sd)

    ## Calculate a range of plausible MCSD values:
    midMissBetaSD <-
        betaMCSD * ( 1 + sqrt( pm / (1 - pm) ) )
    bestMissBetaSD <-
        betaMCSD * ( 1 + sqrt( (0.5 * pm) / (1 - (0.5 * pm) ) ) )
    worstMissBetaSD <-
        betaMCSD * ( 1 + sqrt( (2 * pm) / (1 - (2 * pm) ) ) )

    ## Calculate the required number of Monte Carlo replications:
    betaReps <- rbind(
        ( (1.96 * bestMissBetaSD[-1]) / (acc * betaTrue) )^2,
        ( (1.96 * midMissBetaSD[-1]) / (acc * betaTrue) )^2,
        ( (1.96 * worstMissBetaSD[-1]) / (acc * betaTrue) )^2
    )

    colnames(betaReps) <- c("beta1", "beta2", "beta3")
    rownames(betaReps) <- c("best", "middle", "worst")

    seMCSD <- apply(seMat, 2, sd)
    seTrue <- colMeans(seMat)

    ## Calculate a range of plausible MCSD values:
    midMissSESD <-
        seMCSD * ( 1 + sqrt( pm / (1 - pm) ) )
    bestMissSESD <-
        seMCSD * ( 1 + sqrt( (0.5 * pm) / (1 - (0.5 * pm) ) ) )
    worstMissSESD <-
        seMCSD * ( 1 + sqrt( (2 * pm) / (1 - (2 * pm) ) ) )
```

```
    ## Calculate the required number of Monte Carlo replications:
    seReps <- rbind(
        ( (1.96 * bestMissSESD) / (acc * seTrue) )^2,
        ( (1.96 * midMissSESD) / (acc * seTrue) )^2,
        ( (1.96 * worstMissSESD) / (acc * seTrue) )^2
        )

    colnames(seReps) <- c("int", "beta1", "beta2", "beta3")
    rownames(seReps) <- c("best", "middle", "worst")

    list(beta = betaReps, se = seReps)
}# END calcReps()
```

Code Listing A.5: Function to run a single simulation condition

```
runCondition <- function(x, compData, control, parms, rp)
{
    control$pm <- pm <- as.numeric(x["pm"])
    control$nObs <- nObs <- as.numeric(x["nObs"])
    control$nAux <- nAux <- as.numeric(x["nAux"])
    control$nTrueAux <- as.numeric(x["nTrueAux"])

    saveParams <- parms$saveImpModParams

    nTargets <- length(parms$targetNames)
    nVars <- ncol(compData)

    ## Subset the complete data according to
    ## the parameterization of the current condition:
    if(control$expNum == 1) {
        completeData <- compData[1 : nObs, ]
    } else {
        if(nAux == (nVars - nTargets)) {
            completeData <- compData
        } else {
            completeData <- data.frame(
                compData[ , parms$targetNames],
                compData[ , (nTargets + 1) : (nTargets + (nAux / 2))],
                compData[ , (nVars - (nAux / 2) + 1) : nVars]
            )
        }
    }
```

```
    ## Impose missingness on the complete data:
    missData <- imposeMissing(inData = completeData,
                              control = control,
                              parms = parms)

##### IMPUTE THE MISSINGNESS #####

    nImps <- parms$nImps

    ## Specify a character vector of methods for use by mice():
    methVec <- c( rep("norm", nTargets), rep("", nAux) )

    if(control$expNum == 1) {# Naive MICE is only run in Experiment 1
        ## Run vanilla MICE:
        naiveMiceOut <- try(
            mice(missData$data,
                 m = nImps,
                 method = methVec,
                 maxit = control$miceIters,
                 printFlag = parms$verbose)
        )

        if(class(naiveMiceOut) != "try-error") {
            naiveMiceImps <- list()
            for(i in 1 : nImps) {
                naiveMiceImps[[i]] <- complete(naiveMiceOut, i)
            }
        }
    }

    ## Run MICE using best subset selection:
    bestPreds <- quickpred(missData$data,
                           mincor = 0.5,
                           include = parms$targetNames)
    bestPreds[(nTargets + 1) : nrow(bestPreds), ] <- 1

    bestMiceOut <- try(
        mice(missData$data,
             m = nImps,
             method = methVec,
             predictorMatrix = bestPreds,
             maxit = control$miceIters,
             printFlag = parms$verbose)
    )
```

```
if(class(bestMiceOut) != "try-error") {
    bestMiceImps <- list()
    for(i in 1 : nImps) {
        bestMiceImps[[i]] <- complete(bestMiceOut, i)
    }
}

## Run MICE using the true imputation model:
truePreds <- matrix(0, nAux + nTargets, nAux + nTargets)

for(i in 1 : nTargets) {
    truePreds[i , ] <-
        as.numeric(colnames(missData$data) %in%
                   missData$trueAux[[i]])
}

truePreds[1 : nTargets, 1 : nTargets] <- 1
diag(truePreds) <- 0
truePreds[(nTargets + 1) : nrow(truePreds), ] <- 1

trueMiceOut <- try(
    mice(missData$data,
         m = nImps,
         method = methVec,
         predictorMatrix = truePreds,
         maxit = control$miceIters,
         printFlag = parms$verbose)
)

if(class(trueMiceOut) != "try-error") {
    trueMiceImps <- list()
    for(i in 1 : nImps) {
        trueMiceImps[[i]] <- complete(trueMiceOut, i)
    }
}

currentStreamName <- paste0(parms$streamStem, rp)
myThin <- as.numeric(x["finalGibbs"] / nImps)

## Run MIBEN:
mibenOut <- try(
    miben(rawData = missData$data,
          nImps = parms$nImps,
          targetVariables = parms$targetNames,
          nEmApproxIters = as.numeric(x["nMibenEmApprox"]),
```

```
            nEmTuneIters = as.numeric(x["nEmTune"]),
            nEmApproxBurn = as.numeric(x["emApproxBurn"]),
            nEmApproxGibbs = as.numeric(x["emApproxGibbs"]),
            nEmTuneBurn = as.numeric(x["emTuneBurn"]),
            nEmTuneGibbs = as.numeric(x["emTuneGibbs"]),
            nPosteriorBurn = as.numeric(x["finalBurn"]),
            nPosteriorThin = myThin,
            returnModelParams = saveParams,
            verboseIters = parms$verbose,
            gibbsControl =
                list(createRngStream = FALSE,
                     streamName = currentStreamName),
            optControl =
                list(smoothingWindow =
                         as.numeric(x["smoothWindow"]))
            )
    )

    ## Run MIBL:
    miblOut <- try(
        mibl(rawData = missData$data,
             nImps = parms$nImps,
             targetVariables = parms$targetNames,
             nEmApproxIters = as.numeric(x["nMibenEmApprox"]),
             nEmTuneIters = as.numeric(x["nEmTune"]),
             nEmApproxBurn = as.numeric(x["emApproxBurn"]),
             nEmApproxGibbs = as.numeric(x["emApproxGibbs"]),
             nEmTuneBurn = as.numeric(x["emTuneBurn"]),
             nEmTuneGibbs = as.numeric(x["emTuneGibbs"]),
             nPosteriorBurn = as.numeric(x["finalBurn"]),
             nPosteriorThin = myThin,
             returnModelParams = saveParams,
             verboseIters = parms$verbose,
             gibbsControl =
                 list(createRngStream = FALSE,
                      streamName = currentStreamName),
              optControl =
                  list(smoothingWindow =
                           as.numeric(x["smoothWindow"]))
             )
    )

##### FIT ANALYSIS MODELS AND CHECK CONVERGENCE #####

    ## Store MIBEN & MIBL auxiliary output:
```

```
rHatList <- lambdaList <- paramList <- list()

if(class(mibenOut) != "try-error") {
    rHatList$miben <- mibenOut$rHats
    lambdaList$miben <- mibenOut$lambdaHistory
    if(saveParams) paramList$miben <- mibenOut$params
} else {
    rHatList$miben <- lambdaList$miben <-
        paramList$miben <- vcovList <- "ERROR_42"
}

if(class(miblOut) != "try-error") {
    rHatList$mibl <- miblOut$rHats
    lambdaList$mibl <- miblOut$lambdaHistory
    if(saveParams) paramList$mibl <- miblOut$params
} else {
    rHatList$mibl <- lambdaList$mibl <- paramList$mibl <- "ERROR_42"
}

ds <- ifelse(control$sparse, "sparse", "dense")
saveRDS(rHatList,
        file = paste0(parms$outDir,
            "rHats/dissSimRHats",
            "_n", control$nObs,
            "_v", control$nAux,
            "_", ds,
            "_pm", 100 * pm,
            "_rep", rp,
            ".rds")
        )

saveRDS(lambdaList,
        file = paste0(parms$outDir,
            "lambdas/dissSimLambdas",
            "_n", control$nObs,
            "_v", control$nAux,
            "_", ds,
            "_pm", 100 * pm,
            "_rep", rp,
            ".rds")
        )

if(saveParams) {
    saveRDS(paramList,
            file = paste0(parms$outDir,
```

```
                "params/dissSimParams",
                "_n", control$nObs,
                "_v", control$nAux,
                "_", ds,
                "_pm", 100 * pm,
                "_rep", rp,
                ".rds")
            )
}

## Fit the analysis models and track imputation model convergence:
impConvList <- list()

if(control$expNum == 1) {
    if(class(naiveMiceOut) != "try-error") {
        naiveMicePooled <- try( fitModels(naiveMiceImps) )
        impConvList$naiveMice <- TRUE
    } else {
        naiveMicePooled <- "ERROR_42"
        impConvList$naiveMice <- FALSE
    }
}

if(class(bestMiceOut) != "try-error") {
    bestMicePooled <- try( fitModels(bestMiceImps) )
    impConvList$bestMice <- TRUE
} else {
    bestMicePooled <- "ERROR_42"
    impConvList$bestMice <- FALSE
}

if(class(trueMiceOut) != "try-error") {
    trueMicePooled <- try( fitModels(trueMiceImps) )
    impConvList$trueMice <- TRUE
} else {
    trueMicePooled <- "ERROR_42"
    impConvList$trueMice <- FALSE
}

if(class(mibenOut) != "try-error") {
    mibenPooled <- try( fitModels(mibenOut$imps) )
    impConvList$miben <- TRUE
} else {
    mibenPooled <- "ERROR_42"
    impConvList$miben <- FALSE
```

```
}

if(class(miblOut) != "try-error") {
    miblPooled <- try( fitModels(miblOut$imps) )
    impConvList$mibl <- TRUE
} else {
    miblPooled <- "ERROR_42"
    impConvList$mibl <- FALSE
}

## Check convergence of analysis models:
modConvList <- list()

if(control$expNum == 1) {
    fullConverge <- class(naiveMicePooled) != "try-error" &&
        naiveMicePooled != "ERROR_42"
    if(fullConverge) {
        modConvList$naiveMice <- TRUE
    } else {
        if(class(naiveMicePooled) == "try-error") {
            modConvList$naiveMice <- FALSE
        } else {
            modConvList$naiveMice <- NA
        }
    }
}

fullConverge <- class(bestMicePooled) != "try-error" &&
    bestMicePooled != "ERROR_42"
if(fullConverge) {
    modConvList$bestMice <- TRUE
} else {
    if(class(bestMicePooled) == "try-error") {
        modConvList$bestMice <- FALSE
    } else {
        modConvList$bestMice <- NA
    }
}

fullConverge <- class(trueMicePooled) != "try-error" &&
    trueMicePooled != "ERROR_42"
if(fullConverge) {
    modConvList$trueMice <- TRUE
} else {
    if(class(trueMicePooled) == "try-error") {
```

```
        modConvList$trueMice <- FALSE
    } else {
        modConvList$trueMice <- NA
    }
}

fullConverge <- class(mibenPooled) != "try-error" &&
    mibenPooled != "ERROR_42"
if(fullConverge) {
    modConvList$miben <- TRUE
} else {
    if(class(mibenPooled) == "try-error") {
        modConvList$miben <- FALSE
    } else {
        modConvList$miben <- NA
    }
}

fullConverge <- class(miblPooled) != "try-error" &&
    miblPooled != "ERROR_42"
if(fullConverge) {
    modConvList$mibl <- TRUE
} else {
    if(class(miblPooled) == "try-error") {
        modConvList$mibl <- FALSE
    } else {
        modConvList$mibl <- NA
    }
}

## Pack all convergence checks into a single list:
convList <- list(imps = impConvList,
                 mods = modConvList)

saveRDS(convList,
        file = paste0(parms$outDir,
            "convChecks/dissSimConvergence",
            "_n", control$nObs,
            "_v", control$nAux,
            "_", ds,
            "_pm", 100 * pm,
            "_rep", rp,
            ".rds")
        )
```

```
    ## Pack all results into a single list:
    resultsList <- list()
    resultsList$miben = mibenPooled
    resultsList$mibl = miblPooled
    resultsList$bestMice = bestMicePooled
    resultsList$trueMice = trueMicePooled
    if(control$expNum == 1) {
        resultsList$naiveMice = naiveMicePooled
    }

    saveRDS(resultsList,
            file = paste0(parms$outDir,
                "results/dissSimResults",
                "_n", control$nObs,
                "_v", control$nAux,
                "_", ds,
                "_pm", 100 * pm,
                "_rep", rp,
                ".rds")
            )

    list() # Return nothing
}# END runCondition()
```