Applying the MIDAS Touch: An Accurate and Scalable Approach to Imputing Missing Data*

Ranjit Lall† Thomas Robinson‡

February 11, 2020

Abstract

We propose an accurate, fast, and scalable approach to multiply imputing missing political science data, which we call MIDAS (Multiple Imputation with Denoising Autoencoders). MIDAS employs a class of unsupervised neural networks known as denoising autoencoders, which were recently developed to optimize the task of dimensionality reduction by corrupting a subset of input data and attempting to reconstruct it through a series of nonlinear transformations. We repurpose denoising autoencoders for multiple imputation by treating missing values as a special case of corrupted data and drawing imputed values from a model trained to minimize the reconstruction error on an additional portion of originally observed values. Systematic tests involving real as well as simulated data demonstrate that MIDAS produces highly accurate imputed values and parameter estimates and scales more efficiently to both long and wide datasets than leading existing multiple imputation algorithms. We provide open-source software for implementing MIDAS in the `Python` programming environment.

*The methods described in this paper are implemented through the open-source software **midas**, which is available as a class in the `Python` programming environment (`https://github.com/ranjitlall/MIDAS`). **midas** was developed by Ranjit Lall, Thomas Robinson, and Alex Stenlake.

†London School of Economics and Political Science. Email: r.lall@lse.ac.uk.

‡University of Oxford. Email: thomas.robinson@politics.ox.ac.uk.

Introduction

In recent years, the analysis of political science data with missing values has been characterized by two trends that have yet to be fully reconciled. First, to avoid the problems caused by popular ad-hoc methods such as listwise deletion (discarding rows of the dataset that contain any missing values), scholars are increasingly turning to principled techniques for imputing, or filling in, missing values recommended by the statistics community. The most widely used of these techniques, multiple imputation (MI), involves replacing each missing element with several values that preserve relationships within the observed data while representing uncertainty about the correct imputation. In the words of a prominent scholar of missing-data analysis, "[MI] is now accepted as the best general method to deal with incomplete data in many fields" (van Buuren 2012, 25).

Second, advances in computational power, efficiency, and storage capacity have enabled political scientists to compile unprecedentedly large and complex datasets, ushering in an era of so-called "Big Data" in the discipline. While enormously increasing the amount of information available for analysis, however, this development has not eliminated the problem of missing data. That is, bigger data has not translated into more *complete* data.

The growing scale and complexity of political science data present a major computational challenge for existing MI algorithms, which were generally designed for small or medium-sized applications with relatively simple structures. When applied to larger datasets with features such as high dimensionality, severe nonlinearities, and unconventional functional forms, therefore, these procedures can fail to converge or become prohibitively slow (eventually producing implausible or impossible imputed values). Analysts of such datasets are thus confronted with an unappealing choice: restrict the imputation model to a subset of the dataset, risking bias and reducing statistical efficiency; or employ an ad-hoc method that *can* be applied to the entire dataset, such as listwise deletion or

mean imputation (replacing missing data with observed column averages), creating an even greater risk of bias and guaranteeing inefficiency.

We propose an approach to MI that enables analysts to accurately and quickly impute missing values in large and complex datasets, which we call MIDAS (Multiple Imputation with Denoising Autoencoders). MIDAS employs a class of unsupervised neural networks known as denoising autoencoders (DAs), which were recently developed to optimize the task of dimensionality reduction (or feature selection and extraction in machine learning terminology). DAs corrupt a subset of input data via the injection of stochastic noise and attempt to reconstruct it through a series of nonlinear transformations. The key innovation in MIDAS is to treat missing values as a special case of corrupted data and draw imputations from a DA model trained to minimize the reconstruction error on an additional portion of originally observed values. To reduce the risk of overfitting and further enhance accuracy, we train this imputation-repurposed DA with the regularization technique of dropout, which extends the corruption process deeper into the neural network architecture.

We demonstrate MIDAS's accuracy and scalability through a series of systematic tests involving real as well as simulated data. We begin by conducting two Monte Carlo simulation experiments that assess MIDAS's accuracy under the statistical conditions assumed by the dominant approach to MI: the data follow a joint multivariate normal distribution. The first experiment establishes that MIDAS yields accurate posterior densities for linear regression parameters, while the second shows that the accuracy of MIDAS's imputed values and parameter estimates compares favorably with that of leading existing MI algorithms. We then move to a more realistic setting, introducing varying levels and patterns of missingness into a census-based dataset frequently used for testing machine learning algorithms. We find that MIDAS continues to generate more accurate imputed values than other MI algorithms across almost all missingness proportions and patterns — even performing well

under patterns where MI cannot avoid bias.

We test MIDAS's scalability by sampling increasing numbers of rows and columns from a widely used electoral survey that typifies the kind of large and complex data available to political scientists. MIDAS generates completed datasets in consistently less time than existing MI algorithms, with this speed gap increasing linearly with dataset length and exponentially with dataset width. Even with modestly-sized datasets, MIDAS's efficiency translates into substantial time savings for analysts. For data approaching the dimensions of modern Big Data, where the time taken by existing MI algorithms can be prohibitive, it may make the difference between employing a principled and statistically valid approach to analyzing missing data and resorting to an ad-hoc method likely to result in biased and inefficient inferences.

Finally, we provide an applied illustration of MIDAS's capacity to handle datasets that pose computational problems for existing MI algorithms — that is, to give us access to new substantive knowledge about politics — that involves estimating the latent ideology of respondents to the electoral survey used in the scalability test. We show that substituting MIDAS for listwise deletion, which enables us to recover estimates for more than 10,000 additional respondents, materially alters our understanding of the distribution of latent ideology in the sample and the relationship between this variable and presidential job approval.

In each of these tests, we implement MIDAS with easy-to-use software that we make available as a companion to this study (`https://github.com/ranjitlall/MIDAS`). This software can be freely accessed as a class in the `Python` programming environment (**midas**), and is accompanied by a comprehensive online guide to its functions and arguments, interactive Jupyter Notebook examples of how to apply MIDAS to real data, and a set of diagnostic tools to help users calibrate the technique for their specific application. Further details on the software are provided in the next section and Appendices 1 and 2.

MIDAS: Theory and Implementation

We start by providing a more detailed description of MIDAS and the computational procedure we have developed for implementing it. We follow standard linear algebraic notation throughout this exposition: bold lower-case symbols (e.g., $\mathbf{x}$) denote vectors; bold upper-case symbols (e.g., $\mathbf{W}$) denote matrices; p denotes a probability density or distribution function; subscripts index subsets of a (matrix or vector) set; superscripts index individual elements of a set; and superscripts in parentheses index hidden layers of a neural network.

Multiple Imputation

The first building block of MIDAS, MI, consists of three steps: (1) replacing each missing element in the dataset with M imputed values that preserve relationships among observed elements while reflecting uncertainty about the correct value to impute; (2) analyzing the M completed datasets separately and estimating parameters of interest; and (3) combining the M separate parameter estimates using a simple set of rules (described below).

The theoretical motivation for MI is straightforward. Let $\mathbf{D} = \{\mathbf{D}_{obs}, \mathbf{D}_{mis}\}$ denote a dataset comprising J columns (including all variables to be used in subsequent analyses) in which $\mathbf{D}_{obs}$ is observed and $\mathbf{D}_{mis}$ is missing, and let β denote a parameter of interest. Rubin (1987) demonstrates that the posterior of β is equal to the completed-data posterior marginalized over the posterior of the missing data conditional on the observed data:

$$p(\beta|\mathbf{D}_{obs}) = \int p(\beta|\mathbf{D}_{obs}, \mathbf{D}_{mis}) p(\mathbf{D}_{mis}|\mathbf{D}_{obs}) d\mathbf{D}_{mis} \tag{1}$$

This result implies that the posterior of β can be simulated using M draws from the missing-data posterior $\hat{\mathbf{D}}^1_{mis}, \hat{\mathbf{D}}^2_{mis}, ..., \hat{\mathbf{D}}^M_{mis} \sim p(\mathbf{D}_{mis}|\mathbf{D}_{obs})$. Furthermore, since the expectation of the posterior of β is equal to the marginalized expectation of the M complete-data

posteriors (i.e., $\mathbb{E}(\beta|\mathbf{D}_{obs}) = \mathbb{E}[\mathbb{E}(\beta|\mathbf{D}_{obs}, \mathbf{D}_{mis})|\mathbf{D}_{obs}]$), the overall parameter estimate $\hat{\beta}$ is simply equal to the average estimate across the M completed datasets:

$$\hat{\beta}_M = \frac{1}{M} \sum_{m=1}^{M} \hat{\beta}_m \tag{2}$$

A further implication of Equation (1) is that the posterior variance of β equals the expectation of the M complete-data variances plus the variance of the M complete-data expectations: $var(\beta|\mathbf{D}_{obs}) = \mathbb{E}[var(\beta|\mathbf{D}_{obs}, \mathbf{D}_{mis})|\mathbf{D}_{obs}] + var[\mathbb{E}(\beta|\mathbf{D}_{obs}, \mathbf{D}_{mis})|\mathbf{D}_{obs}]$. It follows that the overall variance of $\hat{\beta}$ is a weighted sum of the estimated variance within (U) and between (B) the M completed datasets:

$$var(\hat{\beta}_M) = U_M + (1 + \frac{1}{M})B_M \tag{3}$$

where $U_M = \frac{1}{M} \sum_{m=1}^{M} var(\hat{\beta}_m)$ and $B_M = \frac{1}{M-1} \sum_{m=1}^{M} (\hat{\beta}_m - \hat{\beta}_M)^2$.

While any joint density $p(\mathbf{D})$ can be used to estimate the missing-data posterior, greater distance from the "true" distribution leads to larger bias and variance in the estimator of β. The dominant approach to MI assumes that the complete data follow a multivariate normal distribution, which implies that each variable is continuous and a linear function of all others (e.g., King et al. 2001; Honaker and King 2010). An alternative approach models each variable's distribution conditionally on all others in an iterative fashion, allowing for a wider class of variable types and distributions (e.g., Kropko et al. 2014). Imputed values, however, need not be drawn from a posterior density. A notable nonparametric approach is predictive mean matching, which involves replacing missing values with observed ones from similar rows (according to a chosen metric) (e.g., Cranmer and Gill 2013).

All approaches to MI share three attractive features. First, they yield unbiased parameter estimates under a fairly wide range of statistical conditions: data are either missing

completely at random (MCAR), i.e., the pattern of missingness is independent of observed and missing data, or *missing at random* (MAR), i.e., this pattern depends on observed data.[1] They cannot avoid bias when data are missing not at random (MNAR), i.e., missingness depends on missing data, though may still perform well if the observed data include strong predictors of missingness (Schafer 1997; Collins, Schafer, and Kam 2001; Mustillo and Kwon 2015; Lall 2016, 2019).[2] Second, they are statistically efficient because they utilize all observed data — including, if available, data that are not part of the subsequent analysis model. Third, from a practical perspective, they are simple to implement because they do not require directly modeling the missingness mechanism and, due to the separation between the imputation and analysis stages, can be combined with standard complete-data methods.

Existing approaches to MI, however, also have a common shortcoming: they often cannot be applied to the kinds of large and complex datasets that are becoming increasingly common in political science. The multivariate normal approach is usually implemented with a variant of either the imputation-posterior algorithm, which draws missing values from their posteriors using Markov chain Monte Carlo (MCMC) methods, or the expectation-maximization algorithm, a similar procedure that substitutes maximum likelihood estimates for posterior draws. For a variety of reasons — including their serial nature, sweep of the entire dataset at each iteration, and simultaneous updating of all parameters — both algorithms "have well-known problems with large data sets...creating unacceptably long run-times or software crashes" (Honaker and King 2010, 564).[3] Moreover, even when they do converge, they often fail to accurately approximate posteriors

[1] Listwise deletion is unbiased only when missingness is completely random or, in the regression context, independent of the error term (Lall 2019).

[2] For formal definitions of these missingness mechanisms, see Little and Rubin (2002, 11-12).

[3] The IP algorithm, like other sequential sampling-based procedures, is also slowed by its need for a large number of "burn-in" iterations before imputation. Assessing the appropriate number of such iterations can be difficult and time-consuming in applications with many parameters, in part because it requires checking many "nuisance parameters" in addition to parameters of interest.

because they become stuck at local maxima or the data diverge significantly from the specified joint distribution. Procedures for implementing the conditional and predictive mean matching approaches also typically employ MCMC methods and hence suffer from similar problems to the imputation-posterior algorithm — problems compounded by their need to model each variable's distribution sequentially, which prevents them from exploiting computational shortcuts enabled by the specification of a joint distribution (van Buuren 2012; Takahashi and Ito 2013). Finally, some approaches seek to improve efficiency by combining one of the above algorithms with bootstrapping (i.e., random sampling with replacement). As each bootstrapped sample is the same size as the original dataset, however, these procedures also tend to slow down substantially or fail to converge when applied to large datasets. We later provide systematic evidence of these scalability problems.

Denoising Autoencoder Neural Networks

To develop a fast and scalable approach to MI, we make use of artificial neural networks, a concept inspired by the structure of the human brain that has been used to enhance the efficiency as well as accuracy of a wide array of computational tasks, from speech recognition to image processing.[4] A neural network consists of a series of nested nonlinear functions usually depicted as a set of interconnected nodes organized in layers. Inputs (i.e., variables) are fed into the network through an *input layer*, processed by one or more *hidden layers*, and returned via an *output layer*. The defining feature of "deep" neural networks is the existence of multiple hidden layers. If $h \ \varepsilon \ \{1, 2, ..., H\}$ index the hidden layers of a network, the model for a "forward pass" — or computation of output values

[4]Foundational works on neural networks include Hebb (1949), McCulloch and Pitts (1943), and Rosenblatt (1958). For an overview of the evolution of neural networks and deep learning, see Goodfellow, Bengio, and Courville (2016, Chapter 1).

given input data — through layer h is:

$$\mathbf{y}^{(h)} = \sigma(\mathbf{W}^{(h)}\mathbf{y}^{(h-1)} + \mathbf{b}^{(h)}) \tag{4}$$

where $\mathbf{y}^{(h)}$ is a vector of outputs from layer h ($\mathbf{y}^{(0)} = \mathbf{x}$ is the input), $\mathbf{W}^{(h)}$ is a matrix of weights connecting the nodes in layer $h-1$ with the nodes in layer h, $\mathbf{b}$ is a vector of biases for layer h, and σ is a nonlinear activation function. The introduction of nonlinearity into the computation process enables neural networks to learn highly complex functional forms with few hidden layers (Hornik, Stinchcombe, and White 1989). This model can be generalized to an arbitrary number of hidden layers H:

$$\mathbf{y} = \Phi(\mathbf{W}^{(H)}[...[\sigma(\mathbf{W}^{(2)}[\sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})] + \mathbf{b}^{(2)})]...] + \mathbf{b}^{(H)}) \tag{5}$$

where $\mathbf{x}$ is a vector of network inputs, $\mathbf{y}$ is a vector of network outputs, and Φ is a link function (i.e., final activation function) that ensures outputs are returned with the appropriate distribution.

The parameters of the network, θ, are weights and biases, which are trained to minimize a loss function $L(\mathbf{y}, \hat{\mathbf{y}})$ that measures the distance between actual and predicted outputs. The training process involves four steps, collectively known as an *epoch*, which are repeated until some convergence criterion is satisfied: (1) performing a forward pass through the full network using current θ; (2) calculating L; (3) using the chain rule to calculate error gradients with respect to weights in each layer, a technique called back-propagation (Rumelhart, Hinton, and Williams 1986; Rumelhart and McLelland 1986); and (4) adjusting weights in the direction of the negative gradient for the next forward pass. Characteristics such as the number of training cycles per epoch, which are determined by the analyst rather than learned in training, are referred to as *hyperparameters*. To mitigate overfitting — a major risk in complex models with large numbers of param-

eters — neural networks are usually trained and tested on separate sets of input-output pairs from the same dataset, with the testing sample used to assess generalization error.

One class of neural networks that is naturally suited to the task of imputing missing data is the DA, an extension of the classical autoencoder — a well-established tool for dimensionality reduction in machine learning — proposed by Vincent et al. (2008, 2010). Classical autoencoders consist of two parts. First, an *encoder* deterministically maps an input vector $\mathbf{x}$ to a lower-dimensional representation $\mathbf{y}$ by compressing it through a series of shrinking hidden layers that culminate in a "bottleneck" layer (indexed by B):

$$\mathbf{y} = f_\theta(\mathbf{x}) = \sigma(\mathbf{W}^{(B)}[...[\sigma(\mathbf{W}^{(2)}[\sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})] + \mathbf{b}^{(2)})]...] + \mathbf{b}^{(B)}) \tag{6}$$

Second, a *decoder* maps $\mathbf{y}$ back to a reconstructed vector $\mathbf{z}$ with the same probability distribution and dimensions as $\mathbf{x}$ by passing it through a parallel series of expanding hidden layers culminating in the output layer:

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = \Phi(\mathbf{W}^{(H)\prime}[...[\sigma(\mathbf{W}^{(B+2)\prime}[\sigma(\mathbf{W}^{(B+1)\prime}\mathbf{y} + \mathbf{b}^{(B+1)\prime})] + \mathbf{b}^{(B+2)\prime})]...] + \mathbf{b}^{(H)\prime}) \tag{7}$$

Weights are adjusted by backpropagation to minimize a loss function $L(\mathbf{x}, \mathbf{z})$. This reconstruction process yields a distributed representation that captures the key axes of variation in $\mathbf{x}$ in a similar manner to principal component analysis. Indeed, an autoencoder with one linear hidden layer and a mean squared error (MSE) loss function performs a transformation essentially equivalent to principal component analysis (Baldi and Hornik 1989).

DAs were developed to prevent autoencoders from learning an identical representation of the input (the identity function) while enabling them to extract more robust features from the data, that is, features that generalize better to new samples from a given data-generating process. They achieve these benefits by partially corrupting inputs through the injection of stochastic noise: $\mathbf{x} \to \tilde{\mathbf{x}} \sim q_D(\mathbf{x}|\tilde{\mathbf{x}})$. The network is then trained to reconstruct

a clean or "denoised" version of the corrupted input:

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = \Phi(\mathbf{W}^{(H)\prime}[...[\sigma(\mathbf{W}^{(2)\prime}[\sigma(\mathbf{W}^{(1)\prime}\tilde{\mathbf{x}} + \mathbf{b}^{(1)\prime})] + \mathbf{b}^{(2)\prime})]...] + \mathbf{b}^{(H)\prime}) \tag{8}$$

Stochastic corruption makes it impossible for the input to be duplicated and forces the network to capture statistical dependencies within the data, with the reconstruction mapping a coordinate system for points lying close to the low-dimensional manifold around which the data naturally concentrate (Vincent et al. 2010, 3380).

The most common corruption process involves forcing a random subset of inputs to 0. In attempting to recover these elements, the DA effectively performs a form of imputation: predicting corrupted (missing) elements based on relationships among uncorrupted (observed) elements. That is, missing values can be seen as a special case of corrupted or noisy input data. Building on this insight, recent studies have successfully adapted DAs to the task of imputing missing medical and traffic data, reporting high levels of accuracy (Beaulieu-Jones and Greene 2016; Beaulieu-Jones and Moore 2017; Duan et al. 2014). These studies, however, neither offer a general model of DA-based imputation nor combine DAs with MI, thereby forgoing the latter's advantages vis-á-vis single imputation in bias reduction, efficiency, and uncertainty representation.

To our knowledge, the only existing attempt to implement MI using DAs comes from Gondara and Wang (2018), who propose a model in which data are provisionally completed using mean imputation (for continuous variables) or mode imputation (for categorical variables) before being stochastically corrupted and passed into the DA.[5] While, Gondara and Wang's full code is not publicly available, preventing us from testing their algorithm, their proposed method suffers from three design issues. First, its loss functions fail to distinguish between originally observed and originally missing values. As a result,

[5]We developed MIDAS independently and without knowledge of Gondara and Wang's research.

reconstruction error is measured against the mean or mode imputations, which fail to preserve relationships among variables and thus tend to produce biased parameter estimates. Second, it injects stochastic noise into inputs once rather than in each training epoch, increasing the risk of overfitting and reducing model robustness.[6] Third, instead of sampling from a single trained network, it trains a different network for each set of imputations. This compromises the algorithm's efficiency — training is by far the longest stage of the MI process, and storing all trained models and imputations in memory is computationally demanding — while yielding no gains in performance. In the remainder of the section, we present an alternative approach to multiply imputing missing data using DAs.

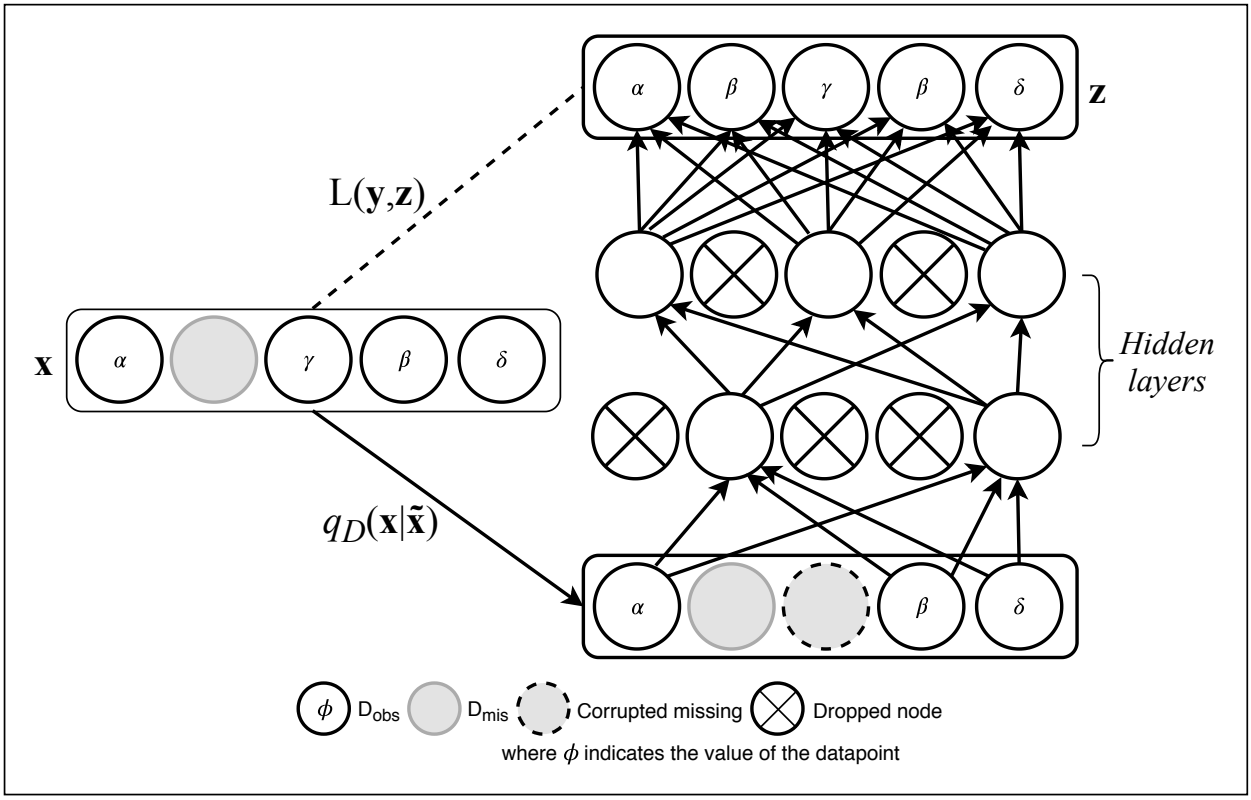
The MIDAS Model

MIDAS modifies the standard DA model in two ways. First, in addition to a random subset of input nodes, it forces all missing values to 0. The task of the DA is thus to predict corrupted values that were both originally missing ($\tilde{\mathbf{x}}_{\text{mis}}$) and originally observed ($\tilde{\mathbf{x}}_{\text{obs}}$) using a loss function that only measures the reconstruction error on the latter.

Second, to further reduce the risk of overfitting, MIDAS regularizes the DA with the complementary technique of dropout. Introduced by Hinton et al. (2012) and elaborated by Srivastava et al. (2014), dropout involves randomly removing (or "dropping") nodes in the hidden layers of a network during training, typically by multiplying outputs from each of these layers by a random Bernoulli vector $\mathbf{v}$ that takes a value of 1 with probability p: $\tilde{\mathbf{y}}^{(h)} = \mathbf{v}^{(h)}\mathbf{y}^{(h)}$, $\mathbf{v}^{(h)} \sim \text{Bernoulli}(N^{(h)}, p)$, where $N^{(h)}$ is the number of nodes in layer h. Dropout is thus a generalization of the idea behind DAs, extending stochastic corruption from the input layer to the hidden layers and thereby enabling the extraction of even more robust features.

[6] We only noticed this problem, which is not apparent from the study's description of the method, after discovering a portion of Gondara and Wang's `Python` code online. See `https://github.com/lgondara/loss-to-followup-DAE/blob/master/programs/Impute_LTF_DAE.py`.

FIGURE 1. MIDAS Neural Network Architecture



Greek letters indicate input/output values; gray nodes indicate missing data, with a solid gray outline denoting originally missing and a dashed black outline denoting corrupted missing. Nodes containing crosses have been stochastically dropped. $\mathbf{x}$ is a vector of inputs, $q_D(\mathbf{x}|\tilde{\mathbf{x}})$ is the corruption process, and $\mathbf{z}$ is the "denoised" version of the corrupted input vector.

Dropout training proceeds by sampling an arbitrary number of "thinned" networks, with a different set of nodes dropped in each iteration. At test time, Hinton et al. and Srivastava et al. propose scaling the weights of a single unthinned network by the probability that their originating nodes were retained during training. To produce *multiple* imputations, MIDAS instead samples M thinned networks. This procedure has recently received a powerful independent justification from Gal (2016) and Gal and Ghahramani (2016), who show through simulation experiments that it results in more accurate parameter estimation with no additional model complexity or training time. Notably, they also prove that dropout training is mathematically equivalent to a Bayesian variational approximation of

a Gaussian process (GP), a common distribution over functions such that, for any finite vector of inputs $\mathbf{x}$, the vector of function values $\mathbf{f} = f(x_1), f(x_2), ..., f(x_n)$ follows a multivariate normal distribution.[7] This result entails that $\mathbf{Wv}$ and $\mathbf{bv}$ can be viewed as random variables $\boldsymbol{\Omega}$ and $\boldsymbol{\psi}$ with expectations $\mathbb{E}[\boldsymbol{\Omega}] = p(\mathbf{Wv})$ and $\mathbb{E}[\boldsymbol{\psi}] = p(\mathbf{bv})$, respectively, and that model uncertainty can be captured by combining outputs from multiple samples of $p(\boldsymbol{\Omega})$ and $p(\boldsymbol{\psi})$. More generally, it implies that MIDAS is a form of *adaptive* joint MI: rather than pre-specifying a joint distribution for the complete data, it discovers such a distribution by placing constraints on the range of data-generating functions that are consistent with the observed data. In doing so, it possesses the flexibility to model a wider class of joint distributions than the dominant approach to MI while depending on less restrictive parametric assumptions.

Viewed from this Bayesian perspective, the feed-forward operation of an imputation-generating DA trained with dropout — a MIDAS network — can be described as:

$$\mathbf{z} = \Phi(g_{\theta'}(\tilde{\mathbf{x}})) = \Phi(\hat{\boldsymbol{\Omega}}^{(H)\prime}[...[\sigma(\hat{\boldsymbol{\Omega}}^{(2)\prime}[\sigma(\hat{\boldsymbol{\Omega}}^{(1)\prime}\tilde{\mathbf{x}} + \boldsymbol{\psi}^{(1)\prime}] + \boldsymbol{\psi}^{(2)\prime})]...] + \boldsymbol{\psi}^{(H)\prime}) \tag{9}$$

where $g \dot{\sim}$ GP, which entails that $p(g(\tilde{\mathbf{x}})|\tilde{\mathbf{x}})) \dot{\sim} N$, and $\mathbf{z}$ represents a fully observed vector containing predicted values of $\tilde{\mathbf{x}}_{\text{obs}}$ and $\tilde{\mathbf{x}}_{\text{mis}}$.[8] To produce a completed dataset, predictions of $\tilde{\mathbf{x}}_{\text{mis}}$ are substituted for $\mathbf{x}_{\text{mis}}$ in $\mathbf{D}$.

[7] Variational approximation is a Bayesian approach to approximating complex posteriors that involves positing a family of potential distributions and finding the member Q that minimizes the Kullback-Leibler divergence to the true posterior P. For introductions to this tool, see Blei, Kucukelbir, and McAuliffe (2017); Grimmer (2010).

[8] The model for imputing individual missing values can be easily derived from Equation 9. Let $\mathbf{D}^{ij}_{\text{mis}}$ denote a missing element in the i^{th} row and j^{th} column of $\mathbf{D}$. Given the dropout distributions and parameter values realized during training, values of $\hat{\mathbf{D}}^{ij}_{\text{mis}}$ are generated by the equation:

$$\hat{\mathbf{D}}^{ij}_{\text{mis}} = \Phi^{j}(\hat{\boldsymbol{\Omega}}^{(H)\prime}([...[\sigma(\hat{\boldsymbol{\Omega}}^{(2)\prime}[\sigma(\hat{\boldsymbol{\Omega}}^{(1)\prime}\tilde{\mathbf{x}} + \boldsymbol{\psi}^{(1)j\prime}] + \boldsymbol{\psi}^{(2)j\prime})]...] + \boldsymbol{\psi}^{(H)j\prime}). \tag{10}$$

The default activation function in MIDAS is an exponential linear unit (ELU) function:

$$\sigma(\alpha, m^{(h)}) = \begin{cases} \alpha(e^{m^{(h)}} - 1) & \text{if } m^{(h)} \leq 0 \\ m^{(h)} & \text{if } m^{(h)} > 0 \end{cases} \tag{11}$$

where $m^{(h)}$ represents $\hat{\boldsymbol{\Omega}}^{(h)}\tilde{\mathbf{y}}^{(h-1)} + \boldsymbol{\psi}^{(h)}$ and α denotes a positive constant initialized as 1. As in a generalized linear regression model, the link function is chosen according to the distribution of $\boldsymbol{D}^j$, that is, the distribution into which $m^{(H)^j}$ must be transformed. Following standard practice, MIDAS assigns identity, logistic, and softmax (i.e., normalized exponential) link functions to continuous, binary, and categorical variables, respectively:

$$\Phi(m^{(H)j}) = \begin{cases} m^{(H)j} & \text{if } \tilde{\mathbf{x}} \text{ is continuous} \\ \frac{e^{m^{(H)j}}}{1+e^{m^{(H)j}}} & \text{if } \tilde{\mathbf{x}} \text{ is binary} \\ \frac{e^{m^{(H)j}}}{\sum_{k=1}^{K} e^{m^{(H)k}}} & \text{if } \tilde{\mathbf{x}} \text{ is categorical} \end{cases} \tag{12}$$

where K is the total number of classes in a categorical variable.

The MIDAS loss functions take the same form as in a regular DA, measuring the distance between $\mathbf{x}$ and $\mathbf{z}$: $L(\mathbf{x}, \mathbf{z})$. As we are only interested in the reconstruction error on predictions of originally observed values, however, these functions are multiplied by a missingness indicator vector $\mathbf{r}$.[9] MIDAS employs root mean squared error (RMSE) and

[9] A vector the same length as $\mathbf{x}$ whose elements are a 1 if the corresponding entry in $\mathbf{x}$ is observed and a 0 if it is missing.

cross-entropy loss functions for continuous and categorical variables, respectively:[10]

$$L(\mathbf{x}, \mathbf{z}, \mathbf{r}) = \begin{cases} [\frac{1}{J} \sum_{j=1}^{J} \mathbf{r}_j (\mathbf{x}_j - \mathbf{z}_j)^2]^{\frac{1}{2}} & \text{if } \mathbf{x} \text{ is continuous} \\ -\frac{1}{J} \sum_{j=1}^{J} \mathbf{r}_j [\mathbf{x}_j \log \mathbf{z}_j + (1 - \mathbf{x}_j) \log(1 - \mathbf{z}_j)] & \text{if } \mathbf{x} \text{ is categorical} \end{cases} \tag{13}$$

Due to the link between dropout and GP approximation, minimizing these functions is equivalent to maximizing the log-likelihood of an approximate GP over the functions supported by the observed data. A more detailed description of the MIDAS model's objective function is provided in Appendix 3A.

Algorithm

The algorithm we have developed to implement MIDAS, which is built using `Python`'s TensorFlow library (enabling parallel processing on supported systems), takes an incomplete dataset $\mathbf{D}$ as its input and returns M completed datasets.[11] The algorithm proceeds in three stages, each comprising a number of smaller steps (see Table 1 for a pseudocode summary).

In the first stage, the input data $\mathbf{D}$ are prepared for training. A missingness indicator matrix $\mathbf{R}$ is constructed for $\mathbf{D}$, allowing us to later distinguish between $\mathbf{D}_{\text{mis}}$ and $\mathbf{D}_{\text{obs}}$, and all elements of $\mathbf{D}_{\text{mis}}$ are set to 0. A DA is then initialized according to the dimensions of $\mathbf{D}$; the default architecture is a five-layer network with 256 nodes per layer. To avoid the common problem of extreme gradients in deep neural network training, we parameterize the network using a variant of Xavier Initialization (Glorot and Bengio 2010) in which

[10] While MSE is more frequently used for continuous variables, we found that RMSE yields more stable training when multiple loss functions are utilized and model residuals are skewed. A small constant is added to the cross-entropy loss function to prevent $\log(0)$ values.

[11] Appendix 1 provides a brief demonstration of the **midas** class in `Python`. Appendix 2 describes two diagnostic tools that can be used to assess the fit of the imputation model and calibrate hyperparameters: the technique of "overimputation" and the generation of new observations using a variational autoencoder component.

TABLE 1. MIDAS Algorithm Pseudocode

Data : Incomplete dataset $\mathbf{D}$
Result : M completed datasets
Parameters : Network weights $\mathbf{\Omega}$
Hyperparameters: Network structure, no. training epochs t, corruption proportion p, mini-batch size s, learning rate γ, weight decay rate λ

begin
  Generate missingness indicator matrix $\mathbf{R}$;
  Set missing values in $\mathbf{D}$ to 0: $\mathbf{D} \rightarrow \mathbf{D}[\mathbf{R} = 0] = 0$;
  Construct DA based on dimensions of $\mathbf{D}$ ($\mathbf{W}$ with variant of Xavier Initialization);
  while epoch $< t$ **do**
    Shuffle $\mathbf{D}$ and $\mathbf{R}$ in same order;
    Slice $\mathbf{D}$ row-wise into n mini-batches $\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n$ of size s;
    Partially corrupt inputs: $\tilde{\mathbf{x}} = [v^{(0,1)}\mathcal{B}_1, ..., v^{(0,n)}\mathcal{B}_n]$, where $\mathbf{v}^{(0)} \sim$ Bernoulli$(p = 0.8)$;
    Partially corrupt outputs of hidden nodes (dropout): $\tilde{\mathbf{y}}^{(h)} = \mathbf{y}^{(h)}\mathbf{v}^{(h)}$, where $\mathbf{v}^{(h)} \sim$ Bernoulli$(p = 0.5)$;
    Perform forward pass through entire network;
    Calculate reconstruction error against $\tilde{\mathbf{x}}_{\mathbf{obs}}$: $E = L(\mathbf{x}, \mathbf{z}, \mathbf{r}) + \lambda||\mathbb{E}[\mathbf{\Omega}_i]||^2$, where $\mathbf{r}$ is missingness indicator vector;
    Backpropagate loss through network to find error gradients: $\frac{\partial E}{\partial \mathbf{w}^{(h)}}$;
    Update weights for next epoch: $\Delta\mathbf{w}^{(h)} = -\gamma\frac{\partial E}{\partial \mathbf{w}^{(h)}}$;
  end
  repeat
    Pass $\mathbf{D}$ into trained DA;
    Construct completed dataset using predictions of $\tilde{\mathbf{x}}_{\text{mis}}$;
  until M completed datasets generated;
end

weights are drawn from a truncated normal distribution:

$$\mathbf{W}^{(h)} \sim (0, \frac{1}{\sqrt{N^{(h)} + N^{(h+1)}}}). \tag{14}$$

In the training stage, five steps are repeated. First, $\mathbf{D}$ and $\mathbf{R}$ are shuffled and sliced

row-wise into paired mini-batches to accelerate convergence.[12] Second, mini-batch inputs are partially corrupted through multiplication by the Bernoulli vector $\mathbf{v}$ (default $p = 0.8$).[13] Third, in line with standard implementations of dropout, outputs from half of the nodes in hidden layers are corrupted using the same procedure. Fourth, a forward pass through the DA is conducted and the reconstruction error on predictions of $\tilde{\mathbf{x}}_{\text{obs}}$ is calculated using the loss functions defined in Equation 13.[14] As a further check against overfitting, we add to these functions a weight decay regularization term λ that penalizes the squared sum of weights. Fifth, loss values are aggregated into a single term and backpropagated through the DA, with the resulting error gradients used to adjust weights for the next epoch. The size of this adjustment is controlled by a learning rate hyperparameter, γ, by which loss function derivatives are multiplied. Mini-batch stochastic gradient descent is implemented with the widely used Adam algorithm (Kingma and Ba 2015). A more formal description of the five steps is provided in Appendix 3B.

Finally, once training is complete, the whole of $\mathbf{D}$ is passed into the DA, which reconstructs the corrupted values according to the data manifold it has learned.[15] A completed dataset is then constructed by replacing $\mathbf{D}_{\text{mis}}$ with predictions of $\tilde{\mathbf{x}}_{\text{mis}}$ from the network's output. This stage is repeated M times.

Accuracy Tests

How does MIDAS perform in practice? The next two sections present tests of the method's accuracy and scalability involving both simulated and real data. We begin with two tough

[12] The default mini-batch size is 16 observations; other common choices are 8, 32, 64, and 128 (powers of 2 enhance memory efficiency). In general, smaller sizes lead to faster convergence at the cost of greater noise and thus less accurate estimates of the error gradient.

[13] Lower values of p increase training time but reduce the degree of overfitting. In our experience, values between 0.7 and 0.95 deliver the best performance.

[14] Categorical variables must be "one-hot" encoded (i.e., converted into separate dummy variables for each of their K classes) before being passed into the DA.

[15] If $\mathbf{D}$ is too large to be fed into the network in its entirety, it can also be sliced into mini-batches.

simulation tests that gauge MIDAS's accuracy under the conditions of joint multivariate normality assumed by the dominant approach to MI. The first is the "MAR-1" Monte Carlo experiment originally conducted by King et al. (2001), which assesses whether MIDAS generates correct posterior densities for linear regression parameters; the second is the continuous component of a more general test conducted by Kropko et al. (2014), which assesses the accuracy of MIDAS's imputed values as well as its parameter estimates. We then test MIDAS's performance on similar metrics in a more realistic context by simulating a variety of missingness patterns in a census-based dataset.

MAR-1 Experiment

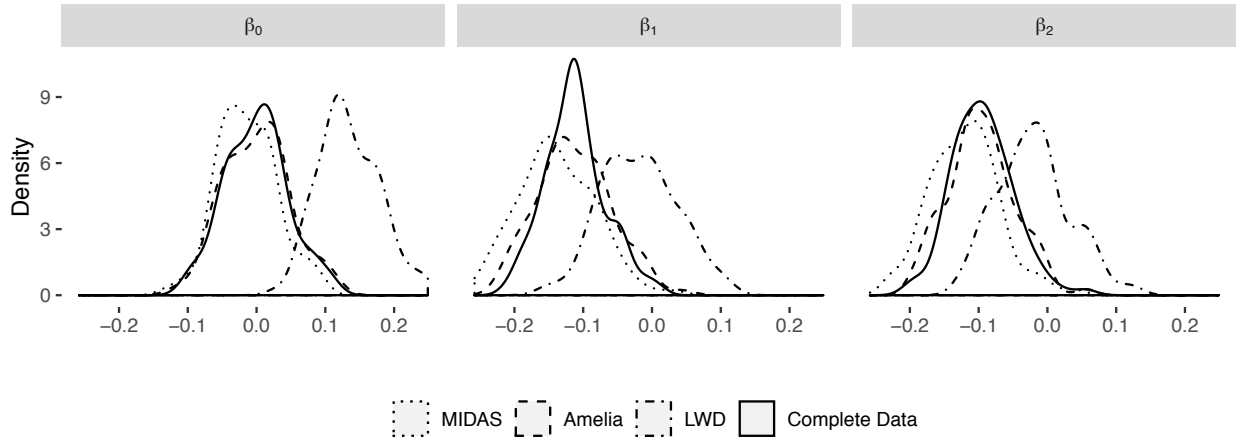
The MAR-1 experiment involves simulating 100 datasets containing 500 rows and five moderately correlated standardized variables $Y, X_1, ..., X_4$ from a multivariate normal distribution. A mixed pattern of missingness is introduced, leaving an average of 72% of rows in each sample fully observed: Y and X_4 are MCAR, while X_1 and X_2 are MAR as a function of X_3.[16] We estimate the linear regression model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2$ using four strategies: (1) MIDAS; (2) multivariate normal MI, implemented with the **Amelia** package in the R statistical software environment (Honaker, King, and Blackwell 2011), which employs an expectation-maximization with bootstrapping algorithm; (3) listwise deletion, and (4) analysis of the complete dataset.[17]

Figure 2 plots the posterior densities of the estimated coefficients on β_0, β_1, and β_2 for each strategy. For all three parameters, MIDAS yields very similar results to **Amelia**. Both sets of estimates are close to the true density, though in the case of β_1 have smaller peaks

[16] The correlation matrix is {1 -0.12 -0.1 0.5 0.1, -0.12 1 0.1 -0.6 0.1, -0.1 0.1 1 -0.5 0.1, 0.5 -0.6 -0.5 1 0.1, 0.1 0.1 0.1 0.1 1}, where commas separate rows. We have slightly modified King et al.'s (2001) reported missingness parameter values. For the two MCAR variables, we draw 500 values from a random uniform distribution $\mathbf{u_{MCAR}}$. Y and X_4 are set as missing if $\mathbf{u_{MCAR}} \geq 0.85$. For X_1 and X_2, we draw separate 500-length vectors from random uniform numbers $\mathbf{u_1}$ and $\mathbf{u_2}$, respectively. X_1 is set as missing if $X_3 < -1$ and $\mathbf{u_1} < 0.9$; X_2 is set as missing if $X_3 < -1$ and $\mathbf{u_2} < 0.9$.

[17] In all performance tests conducted in this and the next section, $M = 10$ for MI algorithms.

FIGURE 2. Estimated Posterior Densities in MAR-1 Simulation Experiment



Coefficient estimates from a linear regression based on Monte Carlo-simulated data with 500 rows, 72% of which are fully observed, and five standardized variables drawn from a multivariate normal distribution.

and larger variances. As Honaker and King (2010, 565) point out, this difference should be expected, given the superior information content of the complete dataset. Listwise deletion estimates, by contrast, are severely biased away the true density of every parameter, mostly possessing the incorrect sign as well as a higher variance than the other densities.

Simulation Test of Imputation and Linear Model Quality

Similarly to the MAR-1 experiment, Kropko et al.'s (2014) simulation-based accuracy test involves generating 1,000 multivariate normal datasets with 1,000 rows and eight standardized variables.[18] MAR missingness is induced in five variables by creating a standardized "latent missingness" indicator for each variable and subtracting it from a uniform random number, with observations registering the lowest q percent of differences set as missing, where $q = \{0.1, 0.1, 0.1, 0.1, 0.25\}$. To assess how the strength of relationships between variables affects MIDAS's performance, we generate two versions of the simulated

[18] Kropko et al. also simulate datasets with binary, ordinal, and unordered categorical variables. We focus on the continuous component of the test because it presents the most favorable conditions for the dominant approach to MI.

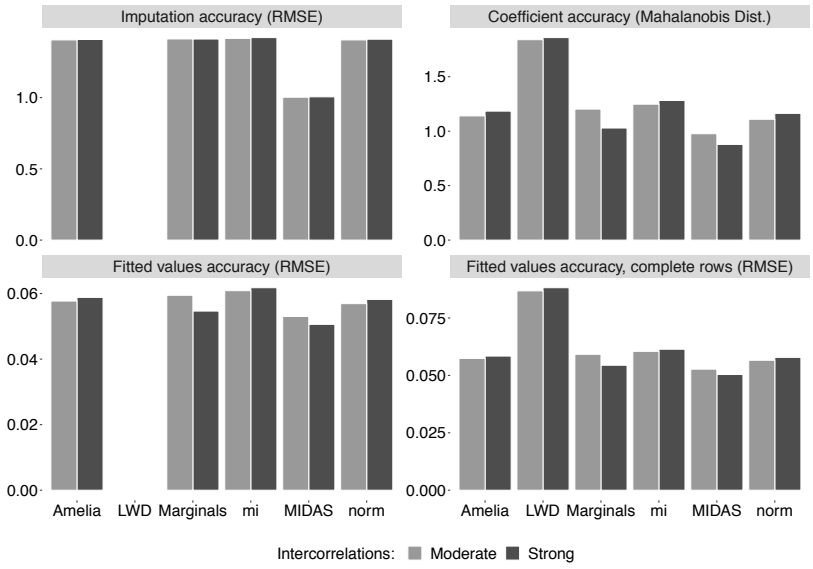
datasets: one in which correlations between variables are moderate and another in which they are strong.[19]

In addition to MIDAS, five missing-data strategies are applied to the incomplete datasets: (1) conditional MI, implemented with the **mi** package in `R` (Su et al. 2011); multivariate normal MI, implemented with (2) **Amelia** and (3) the **norm** package in `R` (which employs a traditional expectation-maximization algorithm) (Schafer and Olsen 1998); (4) listwise deletion; and (5) replacing missing values with draws from each variable's marginal distribution. The six strategies are assessed on two metrics: (1) RMSE relative to true values (averaging imputed values); and (2) the accuracy of coefficient estimates from a regression of one variable on the remaining seven, measured as (i) the Mahalanobis distance between model estimates and complete-data estimates, (ii) the RMSE of model fitted values relative to complete-data fitted values, and (iii) the previous metric excluding incomplete rows (to permit the inclusion of listwise deletion).

The results are displayed in Figure 3. In the moderate-correlation scenario, MIDAS outperforms the other four MI strategies on all four metrics. When we strengthen correlations, this performance gap remains essentially the same in terms of imputation accuracy but becomes even larger in terms of coefficient and fitted-value accuracy (except with respect to marginal draws). MIDAS can thus produce accurate imputed values as well as minimally biased parameter estimates under conditions of joint multivariate normality, with its absolute and relative performance improving with the strength of relationships between variables.

[19] As Kropko et al. use the random data function *rdata.frame* in `R` to simulate the datasets, we model moderate and strong intercorrelations by setting the `eta` argument of this function to 300 and to 1000, respectively. All other test settings are kept the same.

FIGURE 3. Imputation and Linear Model Accuracy in Kropko et al. Simulation Test



The results are based on Monte Carlo-simulated data with 1,000 rows, and eight standardized variables drawn from a multivariate normal distribution. Five variables have MAR missingness imposed, with missingness proportions of $\{0.1, 0.1, 0.1, 0.1, 0.25\}$, respectively.

Applied Test with Adult Dataset

Real data, of course, are rarely jointly multivariate normal. We thus supplement the previous simulation exercises with an applied accuracy test based on the Adult dataset, an extract from the 1994 United States Census that measures 15 characteristics of 48,842 individuals.[20] We select this dataset for two reasons. First, in addition to being frequently used by political scientists, it is a standard benchmarking dataset for machine learning tasks. Second, it is one of the few real social science datasets we were able to find that is almost entirely complete — just 0.009% of values are missing, with 12 of the 15 variables fully observed — which gives us near-complete discretion to manipulate the level and pattern of missingness and mitigates possible concerns about bias from the exclusion of originally missing values. Summary statistics for the dataset are provided in Appendix 4.

[20] Available on the UC Irvine Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/adult.

In contrast to the previous tests, we separately induce varying proportions of MCAR, MAR, and MNAR missingness in the dataset. Specifically, for each missingness pattern, we create four versions of the dataset in which 30%, 50%, 70%, and 90% of columns are randomly selected for corruption.[21] In the MCAR treatment, half of the values in the selected columns are randomly set to missing. In the MAR treatment, a latent missingness indicator L is randomly drawn from the non-selected columns. If L is continuous, a subset of observations at or below its median value are set to missing in the selected columns. If L is categorical, half of its categories are randomly sampled and a subset of corresponding observations in the selected columns are set to missing. The MNAR treatment is essentially the same as the MAR treatment, with the key difference that L is the selected column *itself*.[22] Table 2 describes these three treatments in more detail. To mitigate the effects of sampling variation, we apply each treatment multiple times, generating a total of 60 distinct missingness "masks." Finally, since **Amelia** does not permit the inclusion of categorical variables with more than 10 unique classes, we exclude the $native\_country$, $occupation$, and $education$ variables from the corruption process.

We compare MIDAS against the same five missing-data strategies as the previous test. To facilitate convergence in the MI algorithms, we convert continuous variables to a 0-1 scale.[23] To enable **mi** to converge on the treated datasets in a reasonable time, we modify its settings to generate imputed datasets after either 15 imputation iterations (default

[21] Corruption levels below 30% render the results highly sensitive to column selection; levels above 90% tend to cause numerical stability issues.

[22] Several variables have highly skewed distributions and could thus induce near-complete missingness in the selected columns under the MAR and MNAR treatments. To avoid this problem, we limit the proportion of missingness per column to 50%. In addition, to prevent the omission of entire categorical classes under the MNAR treatment, we set only 95% of the modal class's observations to missing.

[23] The missing-data strategies treat categorical variables differently: listwise deletion, **Amelia**, **mi**, and marginal draws allow them (though **Amelia** cannot handle those with more than 10 unique classes); MIDAS requires one-hot encoded categories; **norm** converts them to their underlying numerical representation. To make the results compatible, we round all imputed values to integers and bound the results between 1 and the number of categories. Values that fall below (above) this range are assigned the first (last) category.

TABLE 2. Missingness Treatments Applied to Adult Dataset

Missingness Pattern	Step	Procedure to Obtain R (Missingness Indicator Vector)
MCAR	1.	Randomly select proportion of columns (0.3, 0.5, 0.7, or 0.9) for missingness treatment.
	2.	$R \sim Bernoulli(p = 0.5)$ for each selected column.
MAR	1.	MCAR step 1.
	2.	L = one column randomly sampled from those *not* selected (latent missingness indicator).
	3a.	If L is continuous, select all rows with values at or below median of L. Sample N/2 rows from this matrix. For each selected column, $R_i = 1$ if row i's value is in this sample.
	3b.	If L is categorical, randomly sample half of all categories. If no. of rows in this matrix > 50% of N, sample N/2 of rows. For each selected column, $R_i = 1$ for all rows in remaining sample.
MNAR	1.	MCAR step 1.
	2.	L = selected column.
	3a.	If L is continuous, MAR step 3a.
	3b.	If L is categorical, select modal category. For each selected column, $R_i = 1$ for all except randomly sampled 5% percent of this sample.

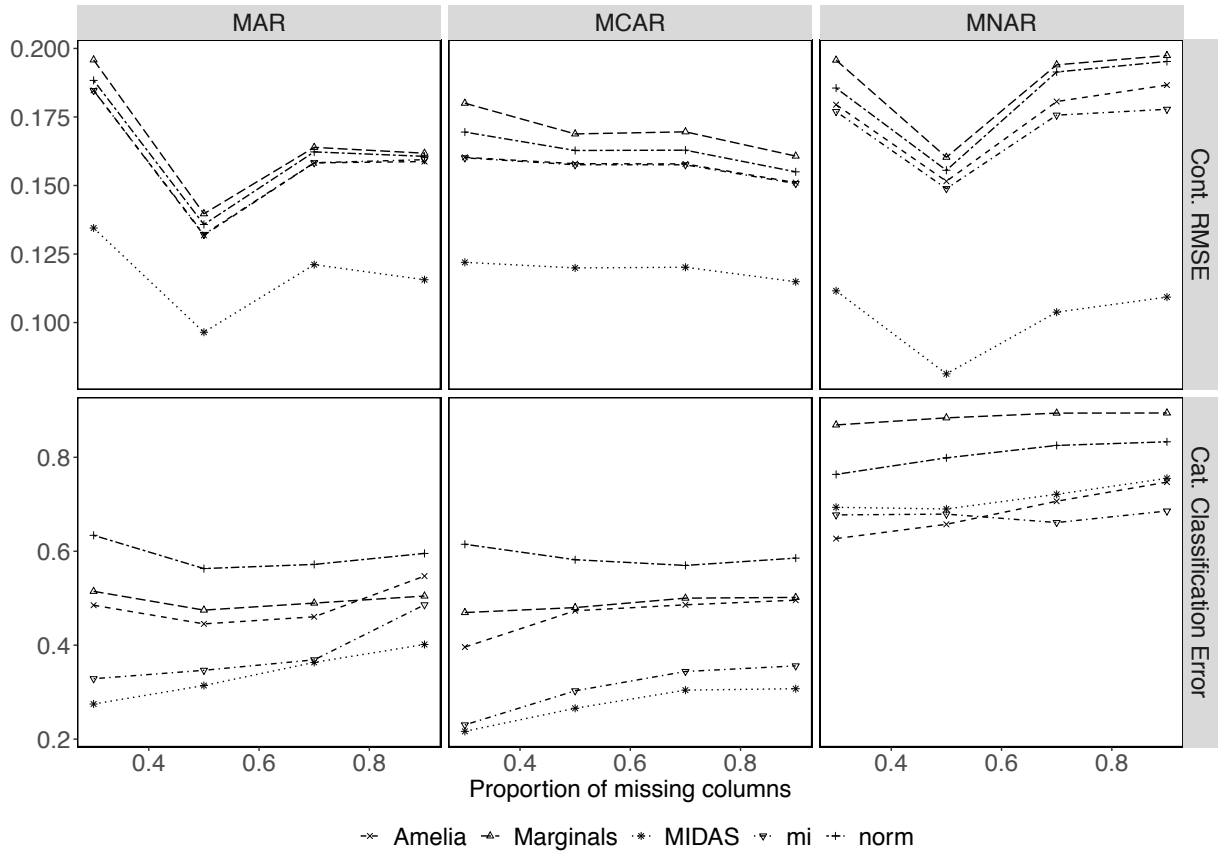
= 30) or the default maximum iteration time of 20 minutes — whichever comes first.[24] **Amelia** only converges with a ridge prior of 1% of the number of rows in the imputation model, a modification that shrinks covariances between variables and thus introduces some degree of bias (Honaker, King, and Blackwell 2011, 19-20). We swap the earlier version of the **norm** package, which is unable to handle the treated datasets, with an updated version based on the same algorithmic logic. We instantiate MIDAS with two hidden layers of 256 nodes, an input corruption proportion of 0.75, and 20 training epochs, leaving all other hyperparameters at their default settings.[25]

We measure imputation accuracy using similar metrics to Kropko et al. (2014): the RMSE of imputed versus actual values for continuous variables; and classification error for categorical variables. In addition, we conduct a model-based accuracy test in which we

[24] Even with these modifications, **mi** took more than 100 hours to complete the test, which constituted 87% of the total runtime. Given the computational demands of the test, we conducted it on an Amazon Web Services Linux m5.xlarge EC2 Instance virtual server (16 GB RAM, 4 vCPUs) running Ubuntu 18.04. A complete guide is included in the replication materials.

[25] The results of the test are robust to a range of alternative hyperparameter choices.

FIGURE 4. Results of Applied Imputation Accuracy Test



MCAR, MAR, and MNAR missingness are separately induced in varying proportions of randomly selected columns in the Adult dataset, with up to 50% of values are set as missing. See Table 2 for details of the missingness treatments.

estimate a logistic regression for each strategy with $class\_labels$ as the outcome variable and the remaining 14 variables in the dataset as (additive) predictors. We then calculate the RMSE of the fitted values compared with those from a model run on (1) the full data and (2) complete rows only. Note, however, that unlike in Kropko et al.'s model-based test, we do not know the true joint distribution of the data. Consequently, we do not wish to infer too much from these results.

Levels of imputation accuracy are summarized in Figure 4. Across almost all corruption levels and missingness patterns, MIDAS's imputed values are more accurate than those of

other MI strategies. This performance advantage is particularly sizable for continuous variables: the mean RMSE of MIDAS imputations is approximately 30% lower than that of the next best algorithm, **mi**.[26] The latter gap is narrower for classification accuracy, though under MAR and MCAR MIDAS and **mi** record significantly lower error than the other three strategies. Under MNAR, **Amelia** and **mi** are the best category classifiers, though MIDAS's performance is comparable. In Appendix 4, we report the results of the model-based tests. On both fitted value metrics, there is substantially less variation in accuracy between strategies, with MIDAS performing slightly above average. As noted earlier, these results should be treated with caution. In sum, as we move to a more realistic setting in which joint multivariate normality does not hold, MIDAS continues to exhibit strong absolute and relative performance on key metrics of accuracy.

Scalability Tests

To facilitate comparison, the previous tests were conducted on small or medium-sized datasets that do not pose (major) computational problems for existing MI algorithms. We now relax this constraint, comparing the algorithms' efficiency in handling progressively larger datasets. We conduct separate tests for increasing numbers of columns and rows, though place greater weight on the former: additional columns are more computationally demanding for MI algorithms than additional rows because they lead to a greater marginal increase in the number and complexity of relationships within the observed data.

[26] Under MAR and MNAR missingness, mean RMSE decreases for all strategies when 50% of columns are treated. This is most likely because the proportion of missing values per corrupted column can fall below 0.5, which reduces mean RMSE if fewer observations are in fact missing. Since the same treated datasets are passed to every method, this issue does not affect the between-method comparison.

Column-Wise Scalability

Rather than scaling up a purely simulated dataset, which is unlikely to capture the variety and richness of real data, we conduct both tests using the 2018 Cooperative Congressional Election Study (CCES), a large-scale electoral survey commonly analyzed by political scientists that encompasses a nationally representative sample of 60,000 respondents in the United States. We focus on the subset of personal profile questions asked to all respondents in addition to a selection of voting- and political activity-related questions (full details are provided in Appendix 5). To generate a baseline sample for the column-wise test, we remove all columns that are perfectly collinear or that contain at least 10,000 missing values (which generally indicates structural missingness associated with survey flow) and all rows that contain responses of "don't know." This leaves a sample of 30,421 rows and 144 variables, the latter consisting of a mixture of nominal, ordered, binary, and continuous types. Once categorical variables are one-hot encoded, there are 443 "effective" columns.

To examine the effect of increasing width on imputation speed, we randomly draw columns without replacement from our formatted CCES sample based on a target number of effective columns. We vary this target from 25 to 400, generating 10 datasets per target. If, after selecting a given variable, the number of effective columns is more than 25% higher than the target, this variable is replaced and a new one is selected. After a given dataset has been generated, we induce 50% MCAR missingness in each column. To ensure that the data do not become too sparse for imputation, we include the fully observed $gender$ and $birthyr$ variables in all samples.

We test the same five MI strategies as before, comparing the time they take to generate 10 completed datasets.[27] For **Amelia**, we follow the advice of Honaker, King, and Blackwell (2011, 13) and declare nominal and ordinal variables to the algorithm. For **norm**, we calculate the time taken both to run the EM algorithm and to complete 10 MCMC simulations

[27]The simulations were run on the same Linux virtual server as the applied accuracy test (see fn. 24).

of the joint posterior. Since each simulation is independent, we parallelize this operation using the **doparallel** and **foreach** packages in R.[28] **mi** and marginal draws require bespoke formatting of the data, which we include in their total runtime. Again, we parallelize the main imputation step for both algorithms. As before, we limit the number of imputation iterations to 15 per chain for **mi**. MIDAS is instantiated with three hidden layers of 256 nodes, a dropout rate of 0.75, and 30 training epochs — a relatively conservative setup, particularly for narrower datasets.[29]

Figure 5 displays the results, including predicted values from a regression of runtime on the effective number of columns (in which we add quadratic terms for **Amelia** and **norm** due to the distribution of their runtimes). The **mi** package and marginal draws are markedly slower than the other strategies. This is apparent at the smallest widths in the case of **mi**: when the target number of columns is 25, it registers a mean runtime of 22 minutes, compared with less than 2 minutes averaged across the other algorithms. By 50 columns, marginal draws are several times slower than the three other algorithms.[30] The latter perform similarly up to this width, with **Amelia** slightly faster than **Norm** and MIDAS but — unlike any other algorithm — failing to converge on several occasions. It is important to reiterate, moreover, that we do not adjust the MIDAS network's size by data width; a three-layer, 256-node network is not necessary for narrow datasets, and a leaner architecture would result in faster computation.

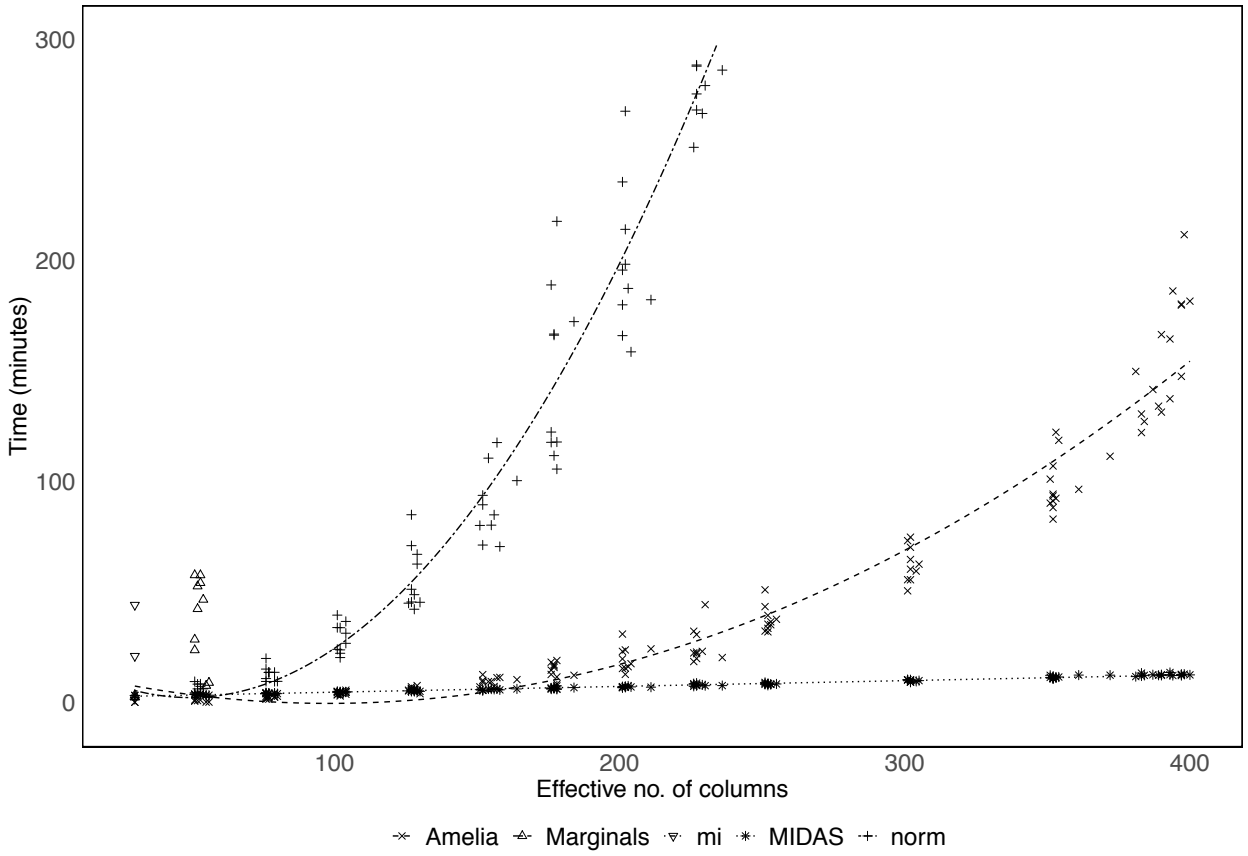
As the number of columns increases, MIDAS's efficiency becomes clear. MIDAS becomes faster than **norm** at a width of around 75 columns and **Amelia** just before 125 columns. By 200 columns, MIDAS is three times quicker than **Amelia** and almost 30 times quicker

[28] Greater degrees of parallelism are supported for **Amelia** but require specialized coding rather than automatic support.

[29] The TensorFlow architecture of the **midas** class supports graphics processing unit (GPU) computation, which would considerably accelerate the neural network training. This requires a system with an Nvidia GPU (with which our server is not equipped).

[30] We therefore drop this strategy and **mi** for higher numbers of columns.

FIGURE 5. Results of Column-Wise Scalability Test



The *y*-axis measures the time taken to generate 10 completed datasets from a sample of the CCES, with 50 percent MCAR missingness; the *x*-axis measures the number of columns in the sample after categorical variables have been one-hot encoded. Individual points indicate observed runtimes; lines indicate predicted values for each strategy from a regression of *y* on *x*. No predictions are made for **mi** and Marginals because they were dropped after 50 columns.

than **norm**.[31] At the maximum number of target columns in the test, 400, **midas** is 12 times faster than **Amelia**. By the standards of modern Big Data, this is still relatively narrow — only around one fifth of the overall width of the CCES (after one-hot encoding). Extrapolating from these results, **Amelia** would take more than 6000 hours to produce 10 completed versions of the full CCES, approximately 100 times longer than MIDAS. As indicated by the slope of the regression lines, MIDAS's efficiency advantage increases exponentially with the effective number of columns: the relationship between computation

[31] Given the slow performance of **norm** at this width, we drop it from the test after 225 columns.

time and data width is linear for MIDAS but quadratic for **Amelia** and the other algorithms. This constitutes a major advantage in the Big Data era, in which datasets often contain thousands or even tens of thousands of variables.

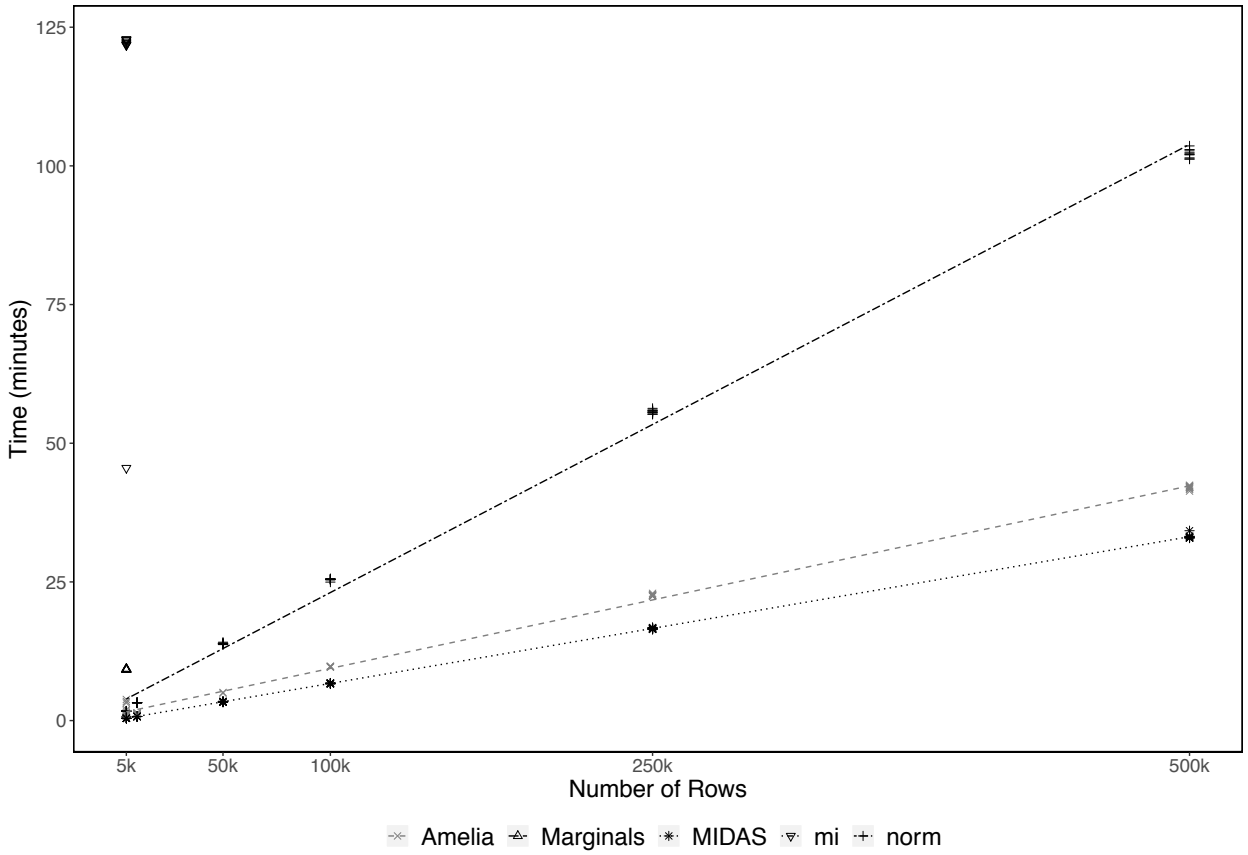
Row-Wise Scalability

We test row-wise scalability by extracting a similar sample from the CCES. To ensure a comparable overall runtime to the column-wise test, we focus on personal profile variables that are continuous, binary, or nominal and have fewer than seven levels. In total, the sample contains 22 variables and 34,441 complete rows. We vary sample length by bootstrapping rows to create datasets with between 5000 and 500,000 rows. We then induce MCAR missingness in 30% of the values in each column. As in the column-wise test, we exclude $birthyr$ and $gender$ from the missingness treatment to prevent excessive sparsity. As the sample is smaller and less complex than before, we shrink the MIDAS network to two layers of 256 nodes and set the number of training epochs to 20.

The results are plotted in Figure 6. Across all dataset lengths, MIDAS is the most efficient strategy. For datasets with 500,000 rows, MIDAS's average runtime is three times quicker than than that of **norm**, the third fastest algorithm, and 25 percent quicker than that of **Amelia**, the second fastest, with these gaps increasing in proportion to sample length. Unlike in the column-wise test, therefore, computation time scales linearly with the number of rows for MIDAS as well as **norm** and **Amelia** — a finding consistent with the less intensive computational demands created by additional rows. Similarly to before, **mi** and marginal draws are the slowest strategies, recording average runtimes of 115 minutes and 9.2 minutes, respectively, at the smallest number of rows (5000). We thus exclude these two strategies for longer datasets.

Note that while the performance gap between MIDAS and **Amelia** is smaller in this test, there are significant caveats to the latter's results. **Amelia** did not converge with

FIGURE 6. Results of Row-Wise Scalability Test



The *y*-axis measures the time taken to generate 10 completed datasets from a sample of the CCES, with 30 percent MCAR missingness; the *x*-axis measures the number of rows in the sample. Individual points indicate observed runtimes; lines indicate predicted values for each strategy (excluding **mi** and Marginals since they were dropped from the test after 5000 rows) from a regression of the effective number of columns on total runtime. **Amelia**'s results are plotted in gray because it only converges with a (bias-inducing) ridge prior in the imputation model.

any dataset without the inclusion of a bias-inducing ridge prior in the imputation model (of 0.005 times the number of rows), most likely due to high correlations among some variables. Even with this modification, it failed to converge in 16 of the 60 iterations of the simulation.[32]

In both tests, therefore, MIDAS exhibits consistently greater scalability than existing MI strategies. The technique's efficiency in handling increasing numbers of rows and,

[32]This occurred at lengths of 50,000 rows (eight times), 100,000 rows (five times), and 250,000 rows (three times).

in particular, columns render it well suited to the emerging era of Big Data, in which the length and width of datasets can cause existing MI algorithms to frequently crash or become prohibitively slow.

Applied Illustration: Estimating Ideology from CCES Data

In this section, we provide a brief illustration of MIDAS's capacity to handle real missing-data situations whose scale presents computational difficulties for existing MI algorithms. We continue to focus on the CCES, whose large number of columns — a feature shared with other major electoral surveys, such as the American National Election Studies, General Social Survey, and British Election Study — often prevents the usage of such algorithms. Specifically, we use MIDAS to shed new light on the distribution of political ideology among respondents, a topic of substantive interest to scholars of electoral politics in the United States and elsewhere.

Respondents to the CCES are asked to report their ideological position on a seven-point scale ranging from 1 for "Very Liberal" to 7 for "Very Conservative." Self-reported ideology, however, is known to be a noisy proxy for underlying beliefs (e.g., Kerlinger 1984; Robinson and Fleishman 1988; Schuman and Presser 1981). Individuals may perceive themselves to be very liberal, for example, but in fact support conservative policies. These discrepancies can arise due to social desirability biases, variation in ideological positions across policy dimensions, and differences in how respondents perceive the survey scale.

A variety of approaches have been proposed to capture respondents' latent ideology, most notably PCA and regression-based estimation. Both of the latter approaches involve estimating ideology using responses to policy questions embedded in a given survey. In the 2018 CCES, individuals are asked their opinion on a series of policy proposals in areas such as the budget, healthcare, and environmental protection. These items have a

markedly higher rate of nonresponse than the CCES in general, with an average of 14% of respondents failing to provide an answer. Does this missingness affect estimates of latent ideology?

Building on recent work by Ramseyer and Rasmussen (2016), we regress respondent i's self-reported ideology on responses to 19 policy questions in the CCES (see Appendix 6 for the list):

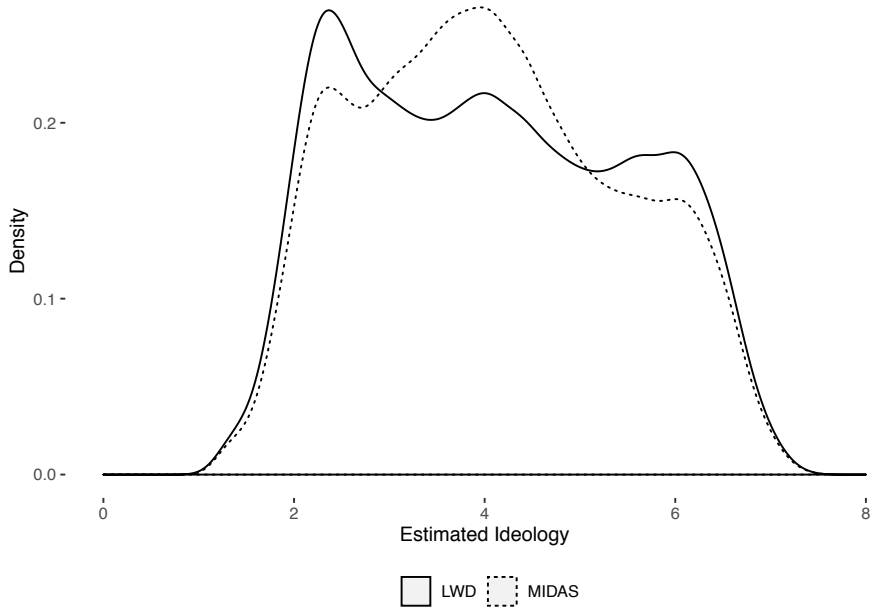
$$\text{Self-Reported Ideology}_i = \alpha + \beta \sum_{j=1}^{19} \text{Policy}_{i,j} + \epsilon_i, \tag{15}$$

where j denotes a given policy question. The fitted values from Equation 15 represent estimates of latent ideology. We compare such estimates under two missing-data strategies: (1) listwise deletion (following Ramseyer and Rasmussen), which risks bias because missingness is plausibly related to the outcome variable; and (2) MIDAS, which we implement using a rich battery of 163 demographic and socioeconomic variables — an imputation model too large for any existing MI algorithm — and a two-layer, 256-node network trained for 200 epochs.[33] MIDAS allows us to produce estimates for more than 10,000 more respondents, almost one-fifth of the full CCES.

Figure 7 plots the kernel densities of the two sets of latent ideology estimates. The two distributions have similar variances but divergent peaks; the null hypothesis that they are samples from the same distribution can be rejected at the 0.1% level under the Kolmogorov-Smirnov test. The distribution of listwise deletion estimates is skewed toward the left (liberal) side of the ideology scale — the modal estimate is 2 — though also contains smaller peaks in the centre and on the right (conservative) side. The MIDAS estimates, in contrast, follow a more normally-shaped distribution, peaking at 4. In the absence of imputation, therefore, there is a danger that analysts could substantially

[33] Some existing MI algorithms, such as **Amelia**, can accommodate subsets of the MIDAS imputation model. These subsets, however, exclude variables that are likely to be strong predictors of missingness in the policy items (such as state of residence in the case of **Amelia**). Consequently, as shown in Appendix 6, the resulting latent ideology estimates are substantially closer to those produced by listwise deletion.

FIGURE 7. Regression-Based Estimates of CCES Respondents' Latent Ideology



overestimate the proportion of strong liberal and strong conservative respondents in the sample.

This finding also has implications for our understanding of the relationship between ideology and other variables of substantive interest in the CCES, for instance, respondents' assessment of President Donald Trump's performance in office. Table 3 summarizes the results of regressing responses to the CCES presidential job approval question, which range from 1 for "strongly approve" to 4 for "strongly disapprove," on (1) self-reported ideology and (2) the MIDAS-based regression estimates of latent ideology:[34]

$$\text{Presidential Job Approval}_i = \alpha + \beta \begin{cases} \text{Self-Reported Ideology}_i \\ \text{Latent Ideology (MIDAS)}_i \end{cases} + \epsilon_i. \qquad (16)$$

In both models, the estimated coefficient on the measure of ideology is negative and

[34]We remove "not sure" responses from the job approval variable for the self-reported model. These values are imputed in the MIDAS model.

TABLE 3. Regression of Presidential Job Approval on Different Measures of Ideology

Measure of Ideology	β	Std. Error	Adj. R^2	N
Self-reported	-0.475	0.002	0.506	48713
Regression-based (MIDAS)	-0.697	0.002	0.616	60000

statistically significant, indicating that respondents classified as more liberal express lower average levels of presidential job approval. As in Ramseyer and Rasmussen's study, however, the regression-based measure is a far better predictor of job approval than the self-reported alternative, possessing a coefficient almost 50% larger (with an identical standard error) and accounting for 20% more model-adjusted variance in the outcome. In substantive terms, shifting from the most liberal to the most conservative respondent raises approval by more than full range of the latter's scale — a sizable increase.

In sum, by leveraging a wealth of contextual information on respondents, MIDAS helps to avoid nonresponse-related biases in the estimation of latent ideology, thereby altering our understanding of this variable's distribution and relationship to presidential job approval. As social science datasets and analytical models become ever larger and more complex — and thus increasingly likely to exceed the capacity of existing MI strategies — we believe that MIDAS could deliver comparable benefits in many other applications.

What Can Go Wrong?

While MIDAS's flexibility render it suitable for a wide range of missing-data problems, there are nevertheless circumstances in which it may perform suboptimally. First, MIDAS cannot, of course, avoid bias when the basic assumptions of MI are violated: data are MNAR and the posited joint density is a poor approximation to the real density. However, as noted earlier — and demonstrated in the applied accuracy test — MIDAS can still perform well under MNAR when there are strong predictors of missingness in the imputation model.

Second, like other approaches to MI, MIDAS is not guaranteed to perform well with certain unconventional data structures, such as non-exchangable data, multilevel data, survival data, and spatially lagged data (Lall 2016). In general, however, we have found MIDAS to be surprisingly effective at learning observed-data relationships within these structures (without the inclusion of any special features in the imputation model). Appendix 7 provides an illustration of this capacity in the context of time-series cross-sectional data — perhaps the most common form of non-exchangable data in applied social science research — adapting an exercise conducted by Honaker and King (2010) to show how MIDAS can impute smooth nonlinear time trends in economic data. This illustration highlights the way in which the richness and adaptability of neural networks often obviate the need for variable selection and feature transformation.

Third, a well-known limitation of variational approximation methods is their tendency to underestimate the variance of the posterior density.[35] In the case of MIDAS, this is most likely to occur when data are concentrated in particular regions of the joint probability space — in other words, when the dataset contains many very similar input-output pairs — which can result in an approximation that only maps a small subset of this space. It should be noted, however, that the inferential consequences of posterior variance underestimation continue to be debated, and some studies find that it may not seriously compromise the accuracy of posterior densities (Blei and Jordan 2006; Braun and McAuliffe 2010; Kucukelbir et al. 2017). Indeed, as illustrated by the MAR-1 experiment, MIDAS can yield approximately correct posteriors even under unfavorable statistical conditions.

Finally, MIDAS inherits the general risks associated with neural network-based methods. These include misspecification of hyperparameters (in particular network structure, activation/link functions, and the number of training epochs), which can result in bias;

[35] This is a consequence of the asymmetric nature of minimizing the Kullback-Leibler divergence between an approximating distribution Q and the true posterior P: $D_{KL}(Q||P) \neq D_{KL}(P||Q)$. This objective penalizes regions of the latent variable space where Q is high and P is low but not vice versa.

overfitting — despite MIDAS's heavy inbuilt regularization — the likelihood of which increases with the size, dimensionality, and sparsity of the dataset; and poor performance on very small datasets. Such risks are compounded by the "black box" nature of neural networks, which makes it difficult for analysts to conduct parameter and posterior checks to identify problems. While MIDAS offer a variety of diagnostic tools to help analysts conduct such checks (discussed in Appendix 2), they do not guarantee detection of all problems.

Concluding Remarks

As the scale and complexity of political science data continue to grow, it is increasingly important that analysts have access to fast and scalable — as well as accurate — methods for dealing with missing values. The approach we have developed in this study, MIDAS, makes it possible to accurately and efficiently multiply impute missing values in large and complex datasets. A battery of tests involving simulated and real data have provided consistent evidence of these advantages. MIDAS not only yields accurate imputed values and parameter estimates under the conditions assumed by the dominant approach to MI — joint multivariate normality — but also outperforms leading MI strategies on imputation accuracy in more realistic scenarios. In addition, it scales more substantially efficiently with real data than these algorithms, exhibiting a speed advantage with that increases with both the number of columns and the number of rows.

To be sure, MIDAS is not a panacea for missing-data problems in the emerging era of Big Data in political science. As discussed earlier, the approach is not guaranteed to perform well with all types of data and may be not be straightforward to optimize for particular applications. Nevertheless, we believe that it constitutes a useful addition to the methodological toolkit of applied researchers and nicely complements the strengths of existing approaches to MI. Indeed, it is precisely the kinds of applications with which these

approaches tend to struggle where MIDAS comes into its own.

In bringing together MI, DAs, and other methods to deliver an efficient solution for missing-data problems, this study contributes to the influential literature on missing data in political science as well as promising emerging research agendas on deep learning, neural networks, Big Data, variational inference, and Bayesian modeling. Furthermore, given the close connection between missing-data problems and the estimation of treatment effects (Ding and Li 2018), we believe that MIDAS may have implications for the study of causal inference — and perhaps even practical uses in this area, for instance, as a tool for mitigating measurement error (Blackwell, Honaker, and King 2017) or imputing counterfactuals for treated units within a potential outcomes framework (Xu 2017). In this respect, it could facilitate ongoing efforts to make Big Data, machine learning, and causal inference "work together" to generate new substantive knowledge about politics (Grimmer 2015).

References

Baldi, Pierre, and Kurt Hornik. 1989. "Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima." *Neural Networks* 2: 53–58.

Beaulieu-Jones, Brett, and Jason H. Moore. 2017. Missing Data Imputation in the Electronic Health Record Using Deeply Learned Autoencoders. In *Pacific Symposium on Biocomputing*, ed. Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Marylyn D. Ritchie, Tiffany A. Murray, and Teri E. Klein. Vol. 22 World Scientific pp. 207–218.

Beaulieu-Jones, Brett K., and Casey Greene. 2016. "Semi-supervised learning of the electronic health record for phenotype stratification." *Journal of Biomedical Informatics* 64 (2): 168–178.

Blackwell, Matthew, James Honaker, and Gary King. 2017. "A Unified Approach to Measurement Error and Missing Data: Overview and Applications." *Sociological Methods & Research* 46 (3): 303–341.

Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe. 2017. "Variational Inference: A Review for Statisticians." *Journal of the American Statistical Association* 112 (518): 859–877.

Blei, David M., and Michael I. Jordan. 2006. "Variational Inference for Dirichlet Process Mixtures." *Bayesian Analysis* 1 (1): 121–143.

Braun, Michael, and Jon McAuliffe. 2010. "Variational Inference for Large-Scale Models of Discrete Choice." *Journal of the American Statistical Association* 105 (489): 324–335.

Collins, Linda M., Joseph L. Schafer, and Chi-Ming Kam. 2001. "A Comparison of Inclusive and Restrictive Strategies in Modern Missing Data Procedures." *Psychological Methods* 6 (4): 330–51.

Cranmer, Skyler J., and Jeff Gill. 2013. "We Have to Be Discrete About This: A Non-Parametric Imputation Technique for Missing Categorical Data." *British Journal of Political Science* 43 (2): 425–449.

Ding, Peng, and Fan Li. 2018. "Causal Inference: A Missing Data Perspective." *Statistical Science* 33(2): 214–237.

Duan, Yanjie, Yisheng Lv, Wenwen Kang, and Yifei Zhao. 2014. A Deep Learning Based Approach for Traffic Data Imputation. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE pp. 912–917.

Gal, Yarin. 2016. Uncertainty in Deep Learning PhD thesis University of Cambridge.

Gal, Yarin, and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning*. ACM pp. 1050–1059.

Glorot, Xavier, and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. ICAIS pp. 249–256.

Gondara, Lovedeep, and Ke Wang. 2018. MIDA: Multiple Imputation Using Denoising Autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining: Advances in Knowledge Discovery and Data Mining*. Springer, Cham pp. 260–272.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Cambridge, MA: The MIT Press.

Grimmer, Justin. 2010. "An introduction to Bayesian inference via variational approximations." *Political Analysis* 19 (1): 32–47.

Grimmer, Justin. 2015. "We Are All Social Scientists Now: How Big Data, Machine Learning, and Causal Inference Work Together." *PS: Political Science & Politics* 48(1): 80–83.

Hebb, Donald Olding. 1949. *The Organization of Behavior: A Neuropsychological Approach*. New York, NY: Wiley.

Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R.

Salakhutdinov. 2012. "Improving neural networks by preventing co-adaptation of feature detectors." *Neural Networks* 2: 1–18.

Honaker, James, and Gary King. 2010. "What to Do about Missing Values in Time-Series Cross-Section Data." *American Journal of Political Science* 54 (2): 561–581.

Honaker, James, Gary King, and Matthew Blackwell. 2011. "Amelia II: A Program for Missing Data." *Journal of Statistical Software* 45 (7): 1–47.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. 1989. "Multilayer Feedforward Networks are Universal Approximators." *Neural Networks* 2 (5): 359–366.

Kerlinger, Fred Nichols. 1984. *Liberalism and Conservatism: The Nature and Structure of Social Attitudes*. Hillsdale: Erblaum.

King, Gary, James Honaker, Anne Joseph, and Kenneth Scheve. 2001. "Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation." *American Political Science Review* 95 (1): 49–69.

Kingma, Diederik P., and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations (ICLR), Conference Track Proceedings*. ICLR.

Kropko, Jonathan, Ben Goodrich, Andrew Gelman, and Jennifer Hill. 2014. "Multiple Imputation for Continuous and Categorical Data: Comparing Joint Multivariate Normal and Conditional Approaches." *Political Analysis* 22 (4): 497–219.

Kucukelbir, Alp, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. 2017. "Automatic Differentiation Variational Inference." *The Journal of Machine Learning Research* 18 (1): 430–474.

Lall, Ranjit. 2016. "How Multiple Imputation Makes a Difference." *Political Analysis* 24 (4): 414–433.

Lall, Ranjit. 2019. "Clarifying the Case for Multiple Imputation: A Rejoinder." London School of Economics.

Little, Roderick J.A., and Donald Rubin. 2002. *Statistical Analysis with Missing Data (Second Edition)*. Hoboken, NJ: Wiley.

McCulloch, Warren S., and Walter Pitts. 1943. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *The Bulletin of Mathematical Biophysics* 5 (4): 114–133.

Mustillo, Sarah, and Soyoung Kwon. 2015. "Auxiliary variables in multiple imputation when data are missing not at random." *The Journal of Mathematical Sociology* 39 (2): 73–91.

Ramseyer, J. Mark, and Eric B. Rasmussen. 2016. "Voter Ideology: Regression Measurement of Position on the Left-Right Spectrum.".

Robinson, John P, and John A. Fleishman. 1988. "A Report: Ideological Identification: Trends and Interpretations of the Liberal-Conservative Balance." *The Public Opinion Quarterly* 52(1): 134–145.

Rosenblatt, Frank. 1958. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review* 65 (6): 386–408.

Rubin, Donald B. 1987. *Multiple Imputation for Nonresponse in Surveys*. New York, NY: John Wiley & Sons.

Rumelhart, David E., and James L. McLelland. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cogniton*. Cambridge, MA: MIT Press.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986. "Learning representations by back-propagating errors." *Nature* 323 (6088): 533–536.

Schafer, Joseph L. 1997. *Analysis of Incomplete Multivariate Data*. London, UK: Chapman and Hall.

Schafer, Joseph L., and Maren K. Olsen. 1998. "Multiple Imputation for Multivariate Missing-Data Problems: A Data Analyst's Perspective." *Multivariate Behavioral Research* 33 (4): 545–571.

Schuman, Howard, and Stanley Presser. 1981. *Questions and Answers: Experiments on Question Form, Wording, and Context in Attitude Surveys*. New York: Academic Press.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *The Journal of Machine Learning Research* 15 (1): 1929–1958.

Su, Yu-Sung, Andrew Gelman, Jennifer Hill, and Masanao Yajima. 2011. "Multiple Imputation with Diagnostics (mi) in R: Opening Windows into the Black Box." *Journal of Statistical Software* 45 (2): 1–31.

Takahashi, Masayoshi, and Takayuki Ito. 2013. Multiple Imputation of Missing Values in Economic Surveys: Comparison of Competing Algorithms. In *Proceedings of the 59th World Statistics Congress of the International Statistical Institute*. ISI pp. 3240–3245.

van Buuren, Stef. 2012. *Flexible Imputation of Missing Data*. Boca Raton: Taylor and Francis.

Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Youshua Bengio, and Pierre-Antoine Manzagol. 2010. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Driterion." *Journal of Machine Learning Research* 11: 3371–3408.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008.

Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*. ACM pp. 1096–1103.

Xu, Yiqing. 2017. "Generalized Synthetic Control method: Causal Inference with Interactive Fixed Effects Models." *Political Analysis* 25(1): 57–76.

Appendices for "Applying the MIDAS Touch: An Accurate and Scalable Approach to Imputing Missing Data"

Ranjit Lall* Thomas Robinson†

February 11, 2020

Contents

*London School of Economics and Political Science. Email: r.lall@lse.ac.uk.
†University of Oxford. Email: thomas.robinson@politics.ox.ac.uk.

1 The MIDAS Algorithm: A Brief Demonstration

This appendix provides a brief demonstration of the **midas** class in the `Python` programming environment, the software we have developed to implement MIDAS. We show how to use the class to multiply impute missing values in the Adult census dataset, the basis for our applied accuracy test.

To access the class, users must have TensorFlow installed as a **pip** package in their `Python` environment. MIDAS is written in TensorFlow 1.X API; users of TensorFlow 2.X can install the correct version via the command line:

```
pip install tensorflow==1.14.0
```

We recommend creating a Conda environment before reinstalling TensorFlow to avoid conflicts with projects using later versions of the package.

Users should then place the files `midas.py` and `adult.csv` in their working `Python` directory. Both files can be obtained from the MIDAS GitHub page (`https://github.com/ranjitlall/MIDAS`). The **midas** class and Adult dataset can then be imported:

```
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import pandas as pd
import tensorflow as tf
from midas import Midas

data_0 = pd.read_csv('adult_data.csv')
data_o.columns.str.strip()
```

As the dataset has a very low proportion of missingness (one of the reasons we selected it for the accuracy test), we randomly set 5000 observed values as missing in each column:[1]

[1]This procedure differs from the more complex missingness treatments in our applied accuracy test (which we implement in `R` because the various MI algorithms require bespoke data formatting).

```
np.random.seed(441)

def spike_in_generation(data):
    spike_in = pd.DataFrame(np.zeros_like(data), columns= data.columns)
    for column in data.columns:
        subset = np.random.choice(data[column].index[data[column].notnull()], 5000,
            ↪ replace= False)
        spike_in.loc[subset, column] = 1
    return spike_in

spike_in = spike_in_generation(data_0)
original_value = data_0.loc[4, 'hours_per_week']
data_0[spike_in == 1] = np.nan
```

Next, we list categorical variables in a vector and one-hot encode them using an inbuilt function in the **pandas** package:

```
categorical = [‘workclass’,‘maritalclass’,‘relationship’,‘race’,‘class_labels’,‘sex
    ↪ ’]

data_1 = data_0[categorical]
data_0.drop(categorical, axis = 1, inplace = True)

constructor_list = [data_0]
columns_list = []

for column in data_1.columns:
    na_temp = data_1[column].isnull()
    temp = pd.get_dummies(data_![column], prefix = column)
```

```
    temp[na_temp] = na.nan
    constructor_list.append(temp)
    columns_list.append(list(temp.columns.values))

data_0 = pd.concat(constructor_list, axis=1§)

na_loc = data_0.isnull()
data_0[na_loc] = np.nan
```

The data are now ready to be fed into the **midas** algorithm, which involves three steps. First, we specify the dimensions, input corruption proportion, and other hyperparameters of the MIDAS neural network. Second, we build a MIDAS model based on the data. The vector of one-hot-encoded column names should be passed to the `softmax_columns` argument, as MIDAS employs a softmax link function for categorical variables. Third, we train the model on the data, setting the number of training epochs as 20 in this example:

```

imputer = Midas(layer_structure = [256,256], vae_layer = False, seed = 89,
    ↪ input_drop = 0.75)
imputer.build_model(data_0, softmax_columns = columns_list)
imputer.train_model(training_epochs = 20)
```

Once training is complete, we can generate any number of imputed datasets using the `generate_samples` function (here we set M as 10). Users can then either write these imputations to separate .csv files or work with them directly in `Python`:

```
# Generate list of imputations
imputations = imputer.generate_samples(m=10).output_list

# Write imputations to separate .csv files
```

```
for i in imputations:
    file_out = ‘‘midas_imp_"␣+␣str(n)␣+␣‘‘.csv"
    i.to_csv(file_out, index=False)
    n += 1
```

2 Summary of Diagnostic Tools

The performance of modern machine learning techniques depends heavily on the length of training — which affects the risk of overfitting — and the choice of model hyperparameters (Probst, Boulesteix, and Bischl 2019). To help users of MIDAS assess the fit of the imputation model and calibrate hyperparameters, we provide two diagnostic tools. The first is the technique of "overimputation" (Blackwell, Honaker, and King 2017; Honaker, King, and Blackwell 2011). This involves sequentially removing observed values from the dataset, generating a large number of imputations for each value, and checking the accuracy of these imputations. Accuracy is measured with (1) the RMSE of imputed values versus true values for continuous variables and (2) classification error for categorical variables. To ensure a good fit, we recommend selecting the number of training epochs that minimizes the average value of these metrics (weighted by the proportion of continuous versus categorical variables). By reducing the risk of overtraining, this "early stopping" rule effectively serves as an extra layer of regularization in a MIDAS network.

In the **midas** class, overimputation can be implemented using the `overimpute` function (described in more detail on the MIDAS GitHub page). This function plots values of the RMSE and classification error metrics for each training epoch. Initially, these values should decline with additional epochs as the MIDAS network learns increasingly accurate approximations of the missing-data posterior. As suggested above, if and when error begins to rise, the number of epochs specified in the `train.model` function (demonstrated in Appendix 1) should be capped before this point. The `plot_all` argument of `overimpute` compares the distribution of overimputed versus original values, allowing users to visually inspect whether the former fall within a reasonable range (implying a good model fit). The default hyperparameter settings for `overimpute` are a corruption proportion (`spikein`) of 0.1 and 100 training epochs (`training_epochs`).

The second diagnostic tool is the generation of entirely new observations using a variational autoencoder component. Variational autoencoders are another extension of the classical autoencoder that encode inputs not to a fixed vector $\mathbf{z}$ but to a *distribution* over the latent space $p(\mathbf{z})$ (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014). The loss function minimized during training includes a regularization term (in addition to the usual reconstruction term) that constrains the latent distribution to approximate normality, reducing the risk of an irregular latent space in which similar data points can become very different after decoding. Samples from the latent distribution $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})$ will thus tend to more closely follow the input density than a regular (deterministic) latent representation $\mathbf{z}$, rendering them better suited to the task of generative modeling.

In the **midas** class, the variational autoencoder component can be activated by setting `vae_layer = True` in the `Midas` function. This inserts a variational autoencoder layer after the denoising portion of a MIDAS network, which probabilistically maps inputs to a latent distribution in the manner described above. After training, samples are drawn from this distribution and decoded to produce new observations. In general, the greater the similarity between these observations and the input data, the better the fit of the imputation model. Default settings for `vae_layer` hyperparameters — which include the number of normal clusters assumed to characterize the input data (`latent_space_size`), the variance of these distributions (`vae_sample_var`), and the strength of our normal prior (`vae_sample_var`) — follow standard conventions in autoencoder applications.

We favor overimputation and data generation over customary train/test split approaches to model validation for two reasons. First, the latter have been found to systematically underestimate error in autoencoders and other unsupervised methods of nonlinear dimensionality reduction where there is no clear target value (Christiansen 2005; Scholz 2012). Second, they prevent us from training the MIDAS network on the full dataset, which impedes accuracy — and could seriously compromise performance at high levels of

missingness.

3 Technical Details on MIDAS Model and Algorithm

3A Objective Function

This section offers additional technical details on the MIDAS model's objective function. Recall from the main text that a traditional autoencoder first maps an input vector $\mathbf{x}$ to a lower-dimensional representation $\mathbf{y}$ via a deterministic series of transformations $\mathbf{y} = f_\theta(\mathbf{x})$, parameterized by $\theta = \{\mathbf{W}, \mathbf{b}\}$ (Equation 6), and then maps this representation back to a reconstructed vector $\mathbf{z}$ via a converse series of transformations $\mathbf{z} = g_{\theta'}(\mathbf{y})$, parameterized by $\theta' = \{\mathbf{W}', \mathbf{b}'\}$ (Equation 7). Each element of the input vector $\mathbf{x}_i$ is thus mapped to a corresponding element of the hidden representation $\mathbf{y}_i$ and the reconstruction $\mathbf{z}_i$. The parameters of this model are trained to minimize the average reconstruction error:

$$\theta^*, \theta'^* = \underset{\theta^*, \theta'^*}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{z}_i) \tag{A1}$$

$$= \underset{\theta^*, \theta'^*}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i, g_{\theta'}(f_\theta(\mathbf{x}_i))) \tag{A2}$$

where L is a loss function (such as a mean squared error function).

In a denoising autoencoder, we again optimize these parameters to minimize the average reconstruction error. Unlike before, however, $\mathbf{z}$ is a deterministic function of $\tilde{\mathbf{x}}$, the corrupted input, instead of $\mathbf{x}$. In a MIDAS model, we only seek to minimize the reconstruction error on corrupted values that were originally observed. That is, we want $\mathbf{z}$ to be as close as possible to $\tilde{x}_{\text{obs}}$ (we do not know the original values of $\tilde{x}_{\text{mis}}$). If $\mathbf{D}$ consists of two random variables X and Y with joint probability distribution $p(X, Y)$, the overall joint distribution can be characterized as:

$$q^0(X, \tilde{X}_{\text{obs}}, \tilde{X}_{\text{mis}}, Y) = q^0(X) q_D(\tilde{X}_{\text{obs}}, \tilde{X}_{\text{mis}} | X) \delta_{f_\theta(\tilde{X}_{\text{obs}}, \tilde{X}_{\text{mis}})}(Y) \tag{A3}$$

where $q^0(X, \tilde{X}, Y)$ is parameterized by $\theta = \{\mathbf{\Omega}, \boldsymbol{\psi}\}$. This implies that Y is a deterministic function of both $\tilde{X}_{\text{obs}}$ and $\tilde{X}_{\text{mis}}$. However, the objective function minimized by stochastic gradient descent only includes the former:

$$\arg\min_{\theta^*, \theta'^*} \mathbb{E}_{q^0}(X, \tilde{X}_{\text{mis}})[L(X, g_{\theta'}, (f_\theta(\tilde{X}_{\text{obs}})))] \tag{A4}$$

The implication of this result is that the MIDAS model minimizes the expected loss over the empirical distribution of not only the observed data but also the subset of corrupted data that were originally observed.

3B Training Steps

As discussed in the main text, a MIDAS network is feedforward: given an initial set of weights and biases, data are propagated forward through the hidden layer of the network and aggregate loss is calculated. Weights and biases are then adjusted via the method of backpropagation. Since the MIDAS network is deep (i.e., it contains more than one hidden layer), this adjustment is made sequentially from the last layer to the first. This section provides a more detailed description of the key training steps in the MIDAS algorithm.

Recall that in the pre-training stage, a missingness indicator matrix $\mathbf{D}$ is generated for the input data $\mathbf{D}$, $\mathbf{D}_{mis}$ is set to 0, and a MIDAS network is parameterized using a variant of Xavier Initialization. In each training epoch, we shuffle and divide $\mathbf{D}$ into B mini-batches $\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_B$ of size s (default $s = 16$); $\mathbf{R}$ is divided into corresponding mini-batches. This step has the advantage of reducing training time — storing all training data in memory and calculating loss for the whole sample are memory-intensive, whereas mini-batches can be processed quickly and in parallel — as well as increasing the frequency of model updates, which ensures more robust convergence (for instance, by avoiding local minima).

In the next step, we partially corrupt the input data by multiplying the B mini-batches

by a random Bernoulli vector $\mathbf{v}$ with $p = 0.8$ (resulting in a corruption rate of 20%):

$$\begin{aligned} \tilde{\mathbf{x}} &= [v^{(0,1)}\mathcal{B}_1, ..., v^{(0,n)}\mathcal{B}_B] \\ \mathbf{v}^{(0)} &\sim \text{Bernoulli}(p = 0.8) \end{aligned} \tag{A5}$$

We then implement dropout regularization by partially corrupting nodes in the hidden layers of the network. This involves multiplying outputs from each layer by another random Bernoulli vector with $p = 0.5$ (a corruption rate of 50%):

$$\begin{aligned} \tilde{\mathbf{y}}^{(h)} &= \mathbf{y}^{(h)}\mathbf{v}^{(h)} \\ \mathbf{v}^{(h)} &\sim \text{Bernoulli}(p = 0.5) \end{aligned} \tag{A6}$$

We then perform a full forward pass through the network — using both the corrupted inputs $\tilde{\mathbf{x}}$ and the corrupted hidden nodes $\tilde{\mathbf{y}}^{(h)}$ — to generate our input reconstruction $\mathbf{z}$:

$$\mathbf{z} = \Phi(\hat{\boldsymbol{\Omega}}^{(H)\prime}[...[\sigma(\hat{\boldsymbol{\Omega}}^{(2)\prime}[\sigma(\hat{\boldsymbol{\Omega}}^{(1)\prime}\tilde{\mathbf{x}} + \boldsymbol{\psi}^{(1)\prime}] + \boldsymbol{\psi}^{(2)\prime})]...] + \boldsymbol{\psi}^{(H)\prime}) \tag{A7}$$

Loss is calculated with respect to the subset of corrupted data that were originally observed ($\tilde{\mathbf{x}}_{\mathbf{obs}}$), which is achieved by multiplying the RMSE and cross-entropy loss functions by a missingness indicator vector $\mathbf{r}$ (see Equation 13 in the main text). A weight decay regularization term λ is included in the calculation to reduce overfitting:

$$E = L(\mathbf{x}, \mathbf{z}, \mathbf{r}) + \lambda||\mathbb{E}[\boldsymbol{\Omega}_i]||^2 \tag{A8}$$

In the backpropagation step, we find the gradient of the loss function with respect to the weights of the network.[2] Since the change in error with respect to the weights

[2]For a more in-depth discussion of the backpropagation procedure, see Goodfellow, Bengio, and Courville (2016, Chapter 6).

in a given layer ($\mathbf{w}^{(h)}$) depends on the weights in the next layer ($\mathbf{w}^{(h+1)}$), this must be calculated sequentially from the output layer to the input layer. Specifically, for each layer, we must derive $\frac{\partial E}{\partial \mathbf{w}^{(h)}}$. Through two applications of the chain rule, this problem becomes more tractable:

$$\frac{\partial E}{\partial \mathbf{w}^{(h)}} = \frac{\partial E}{\partial \mathbf{y}^{(h)}} \cdot \frac{\partial \mathbf{y}^{(h)}}{\partial \mathbf{w}^{(h)}} \tag{A9}$$

$$= \frac{\partial E}{\partial \mathbf{y}^{(h+1)}} \cdot \frac{\partial \mathbf{y}^{(h+1)}}{\partial \mathbf{y}^{(h)}} \cdot \frac{\partial \mathbf{y}^{(h)}}{\partial \mathbf{w}^{(h)}}, \tag{A10}$$

The first term of Equation A10 indicates that the layer-specific partial derivative of the loss function depends on the derivative with respect to outputs from the next layer. The middle term is the partial derivative of the next layer's outputs with respect to the current layer's, which is equivalent to the derivative of the next layer's activation function $\frac{\partial f(\mathbf{y}^{(h+1)}}{\mathbf{y}^{(h+1)}}$. Since $\mathbf{y}^{(h)}$ is the weighted sum of the inputs into layer h, the right term is simply equal to $\mathbf{y}^{(h-1)}$. Note that the latter two terms are straightforward to derive because the functional form of each layer's activation function is known *a priori*.

Once errors have been fully backpropagated through the network, we use the calculated gradients to update the MIDAS network's weights. Each weight is adjusted in the direction of the negative gradient, tempered by some learning rate γ that stabilizes convergence by scaling the step size according to the application at hand:

$$\Delta \mathbf{w}^{(h)} = -\gamma \frac{\partial E}{\partial \mathbf{w}^{(h)}} \tag{A11}$$

Once all weights are updated, the training epoch is complete. This procedure is repeated iteratively until the loss function converges.

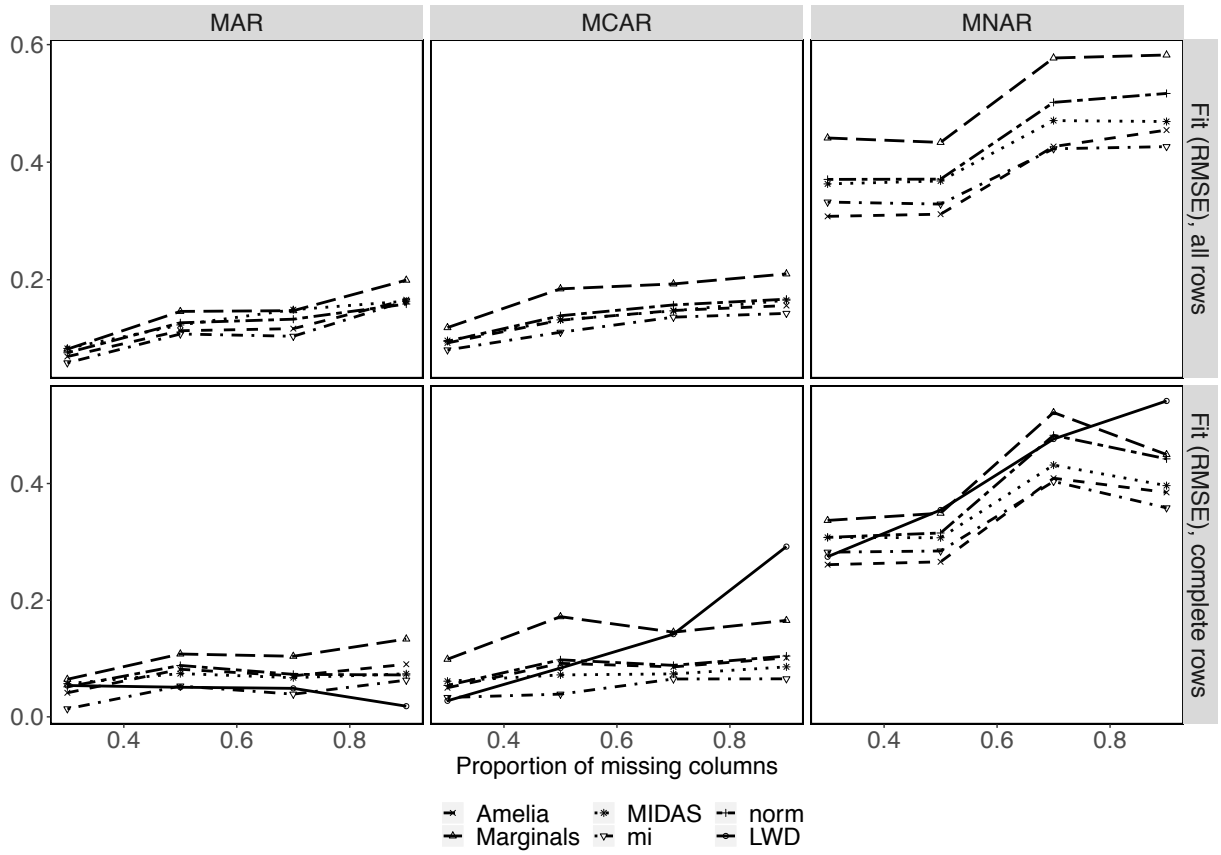
4 Additional Information on Applied Accuracy Test

TABLE A1. Summary Statistics for Adult Dataset

Variable	Type	Missing	Distribution	Description
class_labels (outcome)	Binary	0	>50K: 11,687; ≤50K: 37,155	Annual income
age	Continuous	0	Mean = 38.64; SD = 13.71	Age
workclass	Unordered categorical	2,799	Mode = Private (33,906); 7 other categories	Employment type
fnlwgt	Continuous	0	Mean = 189,664; SD = 105,604	Final weight (expected number in population)
education	Ordinal	0	Mode = HS-grad (15,784); 15 other categories	Highest level of education (categorical)
education_num	Continuous	0	Mean = 10.08; SD = 2.57	Highest level of education (numerical)
marital_status	Unordered categorical	0	Mode = Married-civ-spouse (22,379); 6 other categories	Marital status
occupation	Unordered categorical	2,809	Mode = Prof_speciality (6,172); 13 other categories	Employment sector
relationship	Unordered categorical	0	Mode = Husband (19,716); 5 other categories	Position in family
race	Unordered categorical	0	Mode = White (41,762)	Race
sex	Binary	0	Mode = Male (32,650); 1 other category	Sex
capital_gain	Continuous	0	Mean = 1079; SD = 7,452.019	Capital gains
capital_loss	Continuous	0	Mean = 87.5; SD = 403.00	Capital losses
hours_per_week	Continuous	0	Mean = 40.42; SD = 12.39	Hours worked per week
native_country	Unordered categorical	857	Mode = United-States (43,832); 41 other categories	Country of origin

The dataset has 48,842 rows representing individuals surveyed in the 1994 United States Census.

FIGURE A1. Results of Applied Model-Based Accuracy Test (Adult Dataset)



MCAR, MAR, and MNAR missingness are separately induced in varying proportions (ranging from 0.3 to 0.9) of randomly selected columns in the Adult dataset. In each of these columns, up to 50% of values are set as missing. The first row displays the RMSE of the fitted values for all rows. The second row displays the RMSE of fitted values for only those rows with complete data.

5 Additional Information on Scalability Analysis

List of Variables in Column-Wise Test

We selected 144 variables for inclusion in the column-wise scalability test — a mixture of binary, categorical, ordinal and continuous types. Below is a full list of the variables.

Binary *gender, pew_bornagain, cit1, investor, trans, votereg, edloan, CC18_417a_1, CC18_417a_2, CC18_417a_3, CC18_417a_4, CC18_417a_5, CC18_417a_6, CC18_417a_7, CC18_417a_8, CC18_418a, CC18_414A, CC18_414B, CC18_414C, CC18_414D, CC18_414E, CC18_324a, CC18_324b, CC18_324c, CC18_324d, CC18_415a, CC18_415b, CC18_415c, CC18_415d, CC18_416, CC18_417_a, CC18_417_b, CC18_417_c, CC18_417_d, CC18_417_e, healthins_1, healthins_2, healthins_3, healthins_4, healthins_5, healthins_6, healthins_7, CC18_300_1, CC18_300_2, CC18_300_3, CC18_300_4, CC18_300_5, CC18_300_6, CC18_303_1, CC18_303_2, CC18_303_3, CC18_303_4, CC18_303_5, CC18_303_6, CC18_303_7, CC18_303_8, CC18_303_9, CC18_303_10, CC18_303_11, CC18_320a, CC18_320c, CC18_320d, CC18_321a, CC18_321b, CC18_321c, CC18_321d, CC18_322a, CC18_322b, CC18_322c_new, CC18_322d_new, CC18_322c, CC18_322f, CC18_325a, CC18_325b, CC18_325c, CC18_325d, CC18_325e_new, CC18_325f_new, CC18_326, CC18_327a, CC18_327c, CC18_327d, CC18_327e, CC18_328b, CC18_328d, CC18_328e, CC18_328f, CC18_331a, CC18_331b, CC18_331c, CC18_332a, CC18_332b, CC18_332c, CC18_332e*

Categorical *sexuality, educ, race, employ, internethome, internetwork, marstat, pid3, religpew, ownhome, urbancity, immstat, union_coverage, unionhh, CC18_309a, CC18_309b, CC18_309c, CC18_309d, CC18_316, CC18_318a, CC18_335, CC18_350*

Ordinal *pew_religimp, pid7, ideo5, pew_churatd, pew_prayer, newsint, faminc_new, CC18_421a, CC18_app_dtrmp_post, CC18_422a, CC18_422b, CC18_422c, CC18_422d, CC18_422e, CC18_422f, CC18_422g, CC18_426_1, CC18_426_2, CC18_426_3, CC18_426_4, CC18_426_5, CC18_427_a, CC18_427_b, CC18_427_c, CC18_427_d, CC18_302*

Continuous *birthyr, citylength_1*

List of Variables in Row-Wise Test

The following 22 variables were selected for inclusion in the row-wise scalability test.

Binary *gender, pew_bornagain, cit1, investor, trans, votereg*

Categorical *sexuality, educ, internethome, internetwork, marstat, pid3, ownhome, urbancity, immstat, unionhh*

Ordinal *pew_religimp, pid7, ideo5, pew_churatd, pew_prayer, newsint, faminc_new*

Continuous *birthyr, citylength_1*

6 Additional Information on Latent Ideology Estimation

List of Policy Questions

As discussed in the main text, we estimate CCES respondents' latent ideology by regressing their ideological self-placement on their answers to 19 policy questions in the survey. The former is based on CCES question CC18_334A (*Ideological Placement — Yourself*): "How would you rate each of the following individuals and groups?" Response options range from 1 for "Very Liberal" to 7 for "Very Conservative." The 19 policy variables are listed in Table A2.

TABLE A2. List of CCES Policy Variables Included in Latent Ideology Estimation

Variable	Policy Area	Response Type	Missing
CC18_414A	Minimum Wage	For/Against	8202
CC18_414B	Millionaire's tax	For/Against	8216
CC18_414C	Sales tax	For/Against	8233
CC18_414D	Income tax	For/Against	8230
CC18_414E	Abortion spending	For/Against	8223
CC18_324a	Government Spending	Support/Oppose	8304
CC18_324b	Government Spending	Support/Oppose	8324
CC18_324c	Government Spending	Support/Oppose	8298
CC18_324d	Government Spending	Support/Oppose	8280
CC18_415a	Carbon Dioxide regulation	Support/Oppose	8517
CC18_415b	Fuel efficiency regulation	Support/Oppose	8499
CC18_415c	Renewable energy policy	Support/Oppose	8476
CC18_415d	EPA powers	Support/Oppose	8456
CC18_416	Financial regulation	Support/Oppose	8495
CC18_426_1	State welfare spending	Increase/Decrease (1-5)	8311
CC18_426_2	State healthcare spending	Increase/Decrease (1-5)	8330
CC18_426_3	State education spending	Increase/Decrease (1-5)	8353
CC18_426_4	State law enforcement spending	Increase/Decrease (1-5)	8380
CC18_426_5	State transportation/infrastructure spending	Increase/Decrease (1-5)	8364

Comparison with Amelia

Although existing MI algorithms cannot accommodate the full CCES sample on which we train the MIDAS imputation model, some of them can handle small subsets of this sample that exclude categorical variables with a large number of levels. Importantly, however, some of these omitted variables — such as respondents' state of residence and religion — are likely to be strong predictors of both the policy items and missingness in these variables. When they are excluded from the imputation model, therefore, estimates of latent ideology will tend to be closer to those based on listwise deletion.

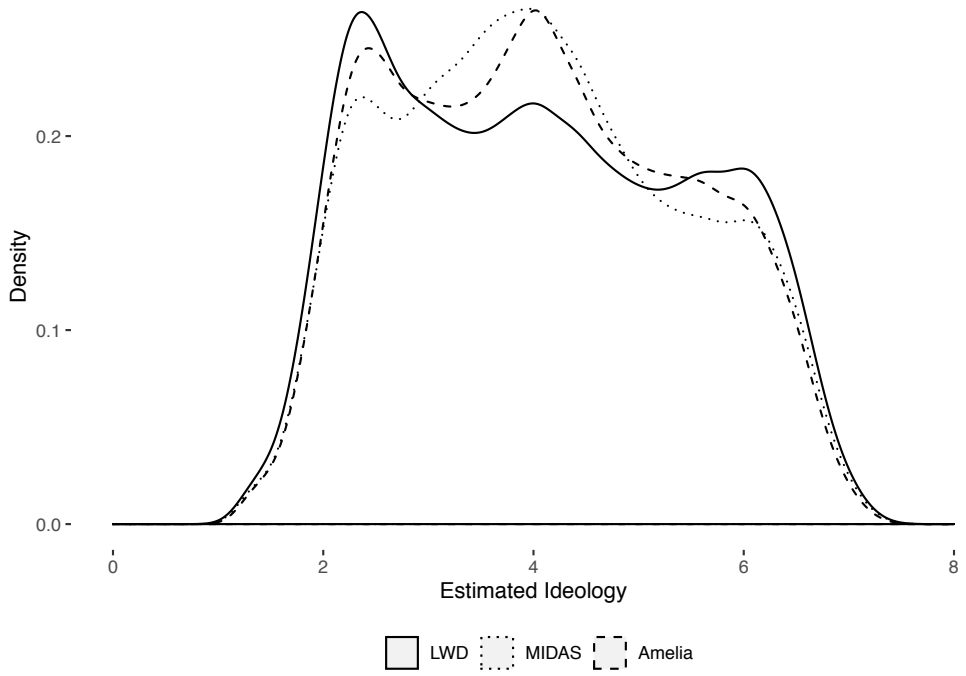
To illustrate this point, we estimate latent ideology using a subset of the CCES data with the **Amelia** package in `R`. Specifically, we include five demographic variables — gender, sexuality, race, sector of employment, and party identification — in addition to the 19 policy variables included in the regression model (Equation 15).[3] As with MIDAS, we then generate 15 completed datasets and recover latent ideology estimates from the fitted values of the regression.

Figure A2 plots the latent ideology estimates from listwise deletion, MIDAS, and **Amelia**. As expected, **Amelia**'s estimates are substantially closer to the listwise deletion estimates than MIDAS's. While the modal category is 4, there is a more pronounced peak on the left (liberal) side of the ideology scale and a flatter tail on the right (conservative) side. Compared to MIDAS, therefore, **Amelia** yields estimates with a clearly more peaked and less normal shape. We can reject the null hypothesis that the three sets of estimates are drawn from the same distribution at the $p < 0.01$ level in Kolmogorov-Smirnov tests.

These inferential differences are also significant from a practical perspective. In real datasets such as the CCES, the pattern and specific determinants of missingness are not known. The best option for users of MI is to leverage as much predictive information about the missingness mechanism and incomplete variables as possible. MIDAS enables

[3] We exclude several demographic variables with a higher number of categories to enable convergence.

FIGURE A2. Comparison of Latent Ideology Estimates from Different MI Strategies



us to utilize considerably more such information than existing MI strategies — with no loss in imputation speed or accuracy — reducing the risk of bias and increasing statistical efficiency.

7 Imputing Time-Series Cross-Sectional Data: An Illustration

Finally, this appendix provides an illustration of MIDAS's ability to handle a particularly common type of non-exchangable data in social science research: time-series cross-sectional data.[4] As Honaker and King (2010) note, the dominant approach to MI tends to perform poorly with such data, yielding imputed values that are implausible based on substantive knowledge or that deviate substantially from previous and subsequent observations in a smoothly varying time series. These problems arise because the approach "assumes that the missing values are linear functions of other variables' observed values, observations are independent conditional on the remaining observed values, and all the observations are exchangable in that the data are not organized in hierarchical structures" (Honaker and King 2010, 565).[5] Although MIDAS — like most MI strategies — does not include any special functionalities for non-exchangable data, we have found that its capacity to learn complex relationships among variables enables it to accurately impute values in time-series cross-sectional settings with only small adjustments to the imputation model.

Building on an experiment conducted by Honaker and King (2010, 565-569) with **Amelia**, we demonstrate this capacity using data from the World Bank's World Development Indicators (WDI), a collection of almost 1,600 time-series indicators of social and economic development covering 217 countries since 1960.[6] We select six African countries — Cameroon, Côte D'Ivoire, Congo Republic, Ghana, Niger, and Zambia — over the period 1970-2000, drop all entirely missing columns, and sequentially remove a single country-year observation of GDP (measured in constant 2010 United States dollars) from

[4] Data are non-exchangeable if observations cannot be reordered without altering their joint distribution. More formally, a sequence of random variables $X_1, X_2, ..., X_n$ is non-exchangeable if its joint distribution is not identical to that of any (finite) permutation of its indices: $p(X_1, X_2, ..., X_n) \neq p(X_{\pi(1)}, X_{\pi(2)}, ..., X_{\pi(n)})$.

[5] **Amelia** seeks to avoid these problems by allowing users to construct a general model of temporal patterns with a sequence of polynomials of the time index. Such a sequence could, of course, be included in a MIDAS model.

[6] http://datatopics.worldbank.org/world-development-indicators/.

each cross-section (31 years $\times$ six countries).[7] This yields 186 different subsets of the WDI, each comprising 186 observations and 1251 variables — samples that are too wide for any existing MI algorithm to process. Note, however, does this setup does not play to MIDAS's strengths either, given that the accuracy of neural networks generally increases with the number of observations.

For each sample, we generate lags and leads of all (non-index) variables, since both past and future values of a given variable tend to be correlated with its present value (Honaker, King, and Blackwell 2011, 19). Based on an overimputation analysis (see Section 2), we instantiate MIDAS with two hidden layers of 1024 and 512 nodes, a learning rate of $3e-5$, a dropout rate of 0.95, and 2000 training epochs. We include country dummies as well as the lags and leads in the imputation models, bringing the total number of variables to 3756.[8] 200 completed datasets are then produced with each model.

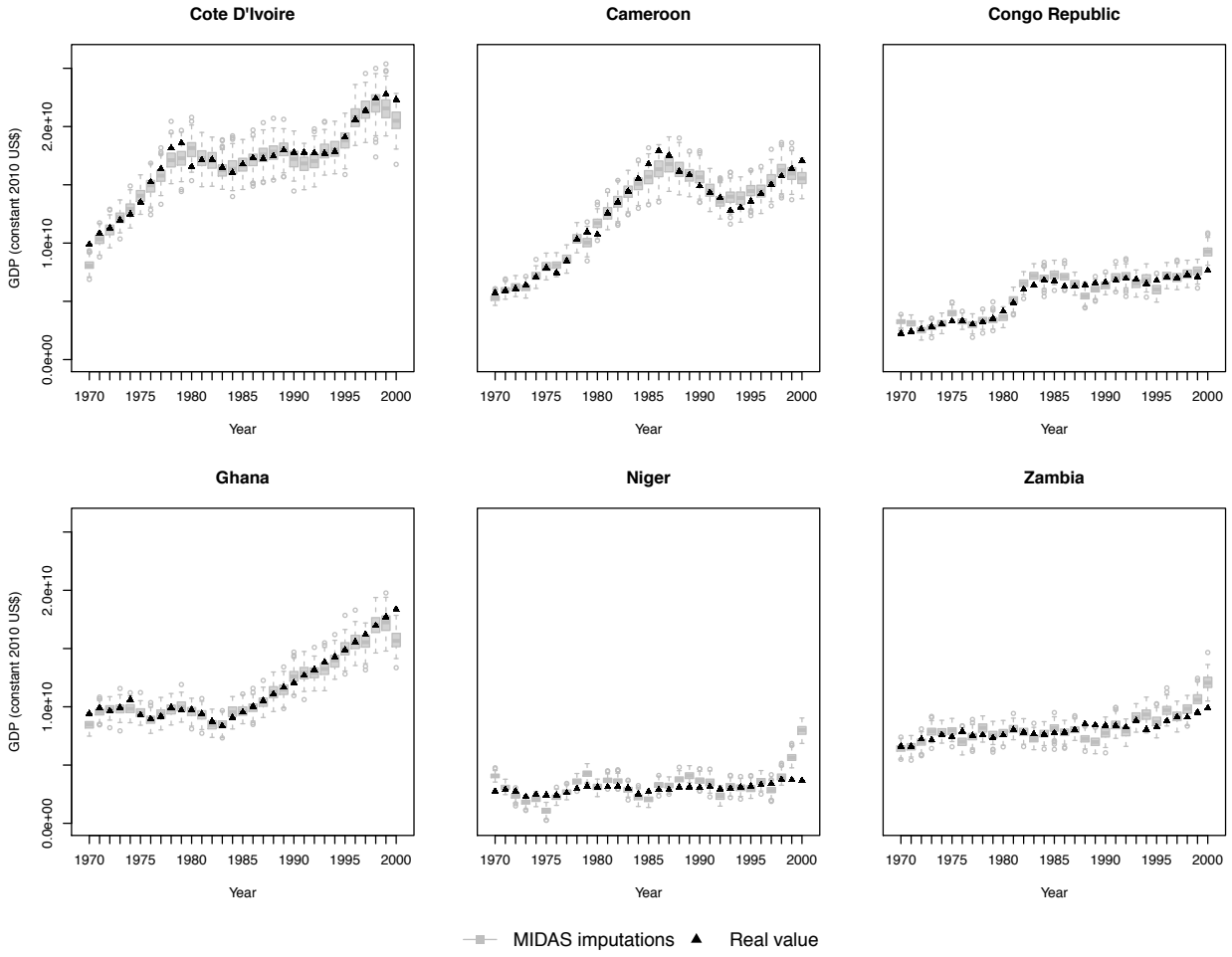
Figure A3 compares real versus MIDAS-imputed values of GDP for the six countries. In general, the latter data track the former remarkably closely through each time series, even capturing trends that were missed by **Amelia**, such as Côte d'Ivoire's cocoa crisis in the late 1970s and Cameroon's strong economic recovery in the mid-1980s (Honaker and King 2010, 569). Only a handful of real values fall outside the interquartile range of MIDAS's imputations, most of which are at the extremities of the time series. This is probably a consequence of the absence both of lags at the beginning of the time series and of leads at the end. Incorporating into the imputation model data from shortly before and after the time period of interest — if available — may help to avoid this problem.

In sum, MIDAS can successfully recover smooth temporal trends in GDP for all six countries. This is particularly notable in light of the absence of explicit features for modeling

[7] We deviate from Honaker and King's selection of countries by substituting Niger for Mozambique, since the latter lacks a complete GDP time series in the WDI.

[8] Given the large number of imputation models in this exercise, we pass all variables other than country, year, and GDP to the `additional_data` argument in MIDAS, which excludes them from the cost function and hence accelerates training.

FIGURE A3. Real Versus MIDAS-Imputed GDP for Six African Countries, 1970-2000



MIDAS imputations are based on variants of the WDI dataset in which country-year observations of GDP are sequentially removed. Each imputation model includes all variables in the WDI that are not entirely missing for the six countries, leads and lags of all non-index variables, and country dummies.

time and the high ratio of variables to observations, which often leads to poor imputation accuracy with existing MI strategies. To be sure, MIDAS would not perform as well in the presence of longer periods of missingness and sharper inflection points in the time series. However, provided that the imputation model contains sufficiently rich information about how observed values are related at different points in time, posterior uncertainty should be low enough to permit valid statistical inference. The inclusion of additional features in

the model, such as polynomials of the time index and flexible basis functions, could further improve MIDAS's performance.

References

Blackwell, Matthew, James Honaker, and Gary King. 2017. "A Unified Approach to Measurement Error and Missing Data: Overview and Applications." *Sociological Methods & Research* 46 (3): 303–341.

Christiansen, Bo. 2005. "The Shortcomings of Nonlinear Principal Component Analysis in Identifying Circulation Regimes." *Journal of Climate* 18(22): 4814–4823.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. Cambridge, MA: The MIT Press.

Honaker, James, and Gary King. 2010. "What to Do about Missing Values in Time-Series Cross-Section Data." *American Journal of Political Science* 54 (2): 561–581.

Honaker, James, Gary King, and Matthew Blackwell. 2011. "Amelia II: A Program for Missing Data." *Journal of Statistical Software* 45 (7): 1–47.

Kingma, Diederik P., and Max Welling. 2013. "Auto-Encoding Variational Bayes." *ArXiv e-prints* .

Probst, Philipp, Anne-Laure Boulesteix, and Bernd Bischl. 2019. "Tunability: Importance of Hyperparameters of Machine Learning Algorithms." *Journal of Machine Learning Research* 20(53): 1–32.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*. ACM pp. 1278–1286.

Scholz, Matthias. 2012. "Validation of Nonlinear PCA." *Neural Processing Letters* 36 (1): 21–30.