Online Appendices for "The MIDAS Touch: Accurate and Scalable Missing-Data Imputation with Deep Learning"

Ranjit Lall* Thomas Robinson†

November 10, 2020

Contents

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

1 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 et al. 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 et al. 2017; Honaker et al. 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 **MIDASpy** 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 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 et al. 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 **MIDASpy** 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_alpha`) — 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.

2 Technical Details on MIDAS Model and Algorithm

2A 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 3), 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 4). Each element i 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 $\theta = \{\boldsymbol{\Omega}, \boldsymbol{\psi}\}$. 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:

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

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.

2B 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 $\mathbf{D}$, $\mathbf{D}_{\text{mis}}$ 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 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 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 $\mathbf{z}$ (described in Equations 5 and 6 in the main text). 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 7). A weight decay regularization term λ is included in the calculation to reduce overfitting:

$$E = L(\mathbf{x}, \mathbf{z}, \mathbf{r}) + \lambda||\mathbb{E}[\mathbf{W}]||^2 \tag{A7}$$

In the backpropagation step, we find the gradient of the loss function with respect to the weights of the network.[1] Since the change in error with respect to the weights

[1]For a more in-depth discussion of the backpropagation procedure, see Goodfellow et al. (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:

$$\frac{\partial E}{\partial \mathbf{W}^{(h)}} = \frac{\partial E}{\partial \mathbf{y}^{(h)}} \cdot \frac{\partial \mathbf{y}^{(h)}}{\partial \mathbf{W}^{(h)}} \tag{A8}$$

$$= \frac{\partial E}{\partial \mathbf{y}^{(h+1)}} \cdot \frac{\partial \mathbf{y}^{(h+1)}}{\partial \mathbf{y}^{(h)}} \cdot \frac{\partial \mathbf{y}^{(h)}}{\partial \mathbf{W}^{(h)}}, \tag{A9}$$

The first term of Equation A9 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:

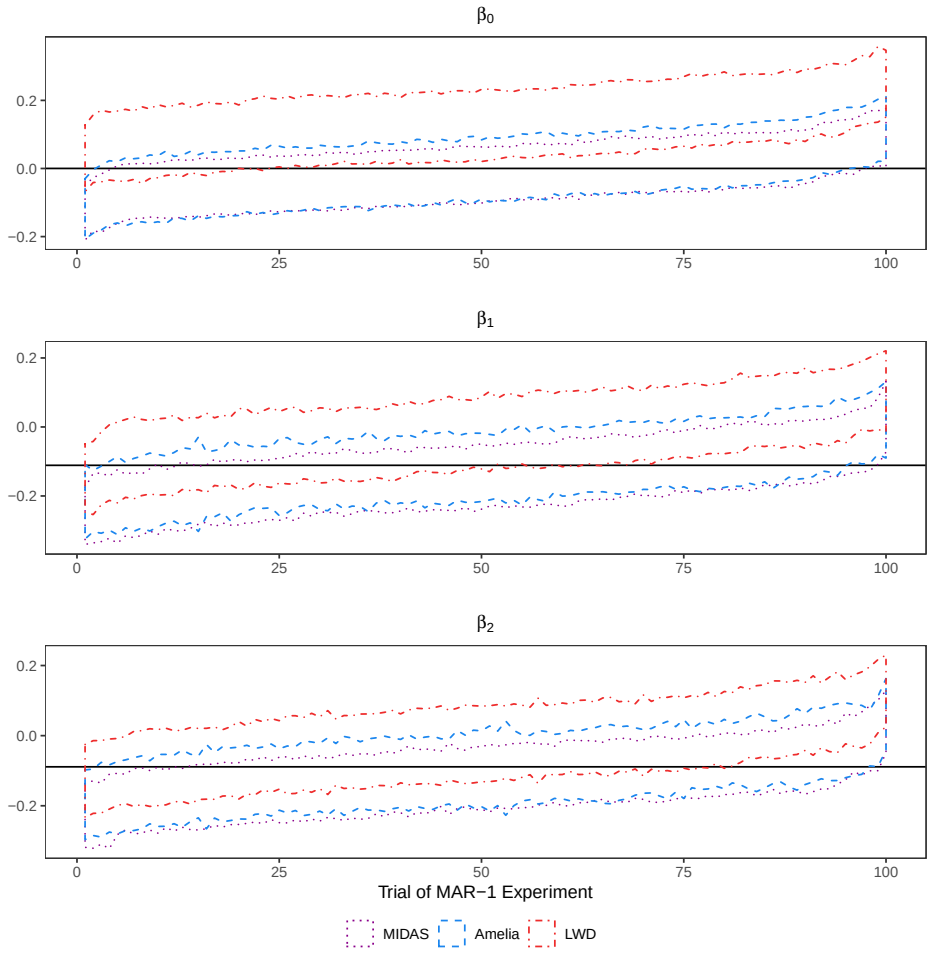
$$\Delta \mathbf{W}^{(h)} = -\gamma \frac{\partial E}{\partial \mathbf{W}^{(h)}} \tag{A10}$$

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

3 Additional Information on Accuracy Tests

3A MAR-1 Experiment

Figure A1. Coverage of Complete-Data Coefficients Across Trials of MAR-1 Simulation Experiment



The solid black lines indicate complete-data ("true") coefficients in the MAR-1 experiment. The dashed lines represent 95% confidence intervals for each method's coefficient estimates across the 100 trials of the experiment (whose densities are plotted in Figure 3 in the main text).

Figure A1 plots the estimated confidence intervals produced by MIDAS, **Amelia**, and list-wise deletion across the 100 trials of the MAR-1 experiment. Similarly to the posterior densities of the estimated coefficients (Figure 3 in the main text), the **Amelia** and MI-

DAS intervals both exhibit good coverage for all three coefficients, encompassing the true estimate with a probability close to the ideal of 0.95.[2] Listwise deletion's coverage is substantially worse in every case, excluding this estimate — and hence failing to appropriately capture uncertainty — in at least 40% of simulations for two of the three coefficients (β_0 and β_1).[3]

3B Applied Test with Adult Dataset

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.

[2] **Amelia**'s coverage rates are marginally closer ($\beta_0 = 0.93$, $\beta_1 = 0.95$, $\beta_2 = 0.94$) to the ideal than MIDAS's ($\beta_0 = 0.93$, $\beta_1 = 0.86$, $\beta_2 = 0.87$), as should be expected under multivariate normal conditions.
[3] The coverage rates are $\beta_0 = 0.22$, $\beta_1 = 0.60$, $\beta_2 = 0.79$.

Table A2. 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. $native\_country$, $occupation$, and $education$ cannot be selected (due to computational issues with **Amelia**).
	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.

4 Applied Accuracy Test on ANES Data

In addition to their multivariate normal simulation exercise (see Section 3.2 of the main text), Kropko et al. (2014a) conduct an applied accuracy test using the 2008 American National Electoral Studies (ANES) dataset. The ANES is, in theory, a good fit for MIDAS: like other electoral surveys, it is a wide and relatively diverse dataset, containing more than 1,000 columns (before any data transformations), many of which are categorical variables with large numbers of classes. Kropko et al., however, focus on a subset of the ANES comprising 11 columns — none of which present difficulties for existing MI algorithms (the categorical variables have few classes) — and the 1,442 complete observations in the dataset.[4] It thus offers another good opportunity to assess MIDAS's relative performance under statistical conditions that are well suited to existing MI algorithms.

In the first step of the test, ordinal variables are transformed into continuous integer-valued variables, binary variables are recoded to 0/1 format, and nominal variables are one-hot encoded. MAR missingness is then simulated in 10 percent of observations, excluding one column per data type, and five completed datasets are generated. The test consists of 20 simulations, across which the two accuracy metrics in Kropko et al.'s multivariate normal simulation are averaged.[5]

We instantiate MIDAS with a three-layer, 512-node network, which we train for 20 epochs. To ensure consistency across missing-data strategies, we make a few minor modifications to the test. First, we supply the one-hot encoded versions of the nominal variables to the marginal draws and **mi**-based strategies, which renders the multinomial logit (labeled "MI:MNL" by Kropko et al.) and renormalized logit ("MI:RNL") variants of the latter indistinguishable. Second, after imputation, we do not convert non-integer predicted prob-

[4]For detailed information on these variables, see Table 1 in Kropko et al. (2014a).
[5]Our description and extension of the test follow the code in Kropko et al.'s replication materials (Kropko et al. 2014b); their article reports a higher number of simulations.

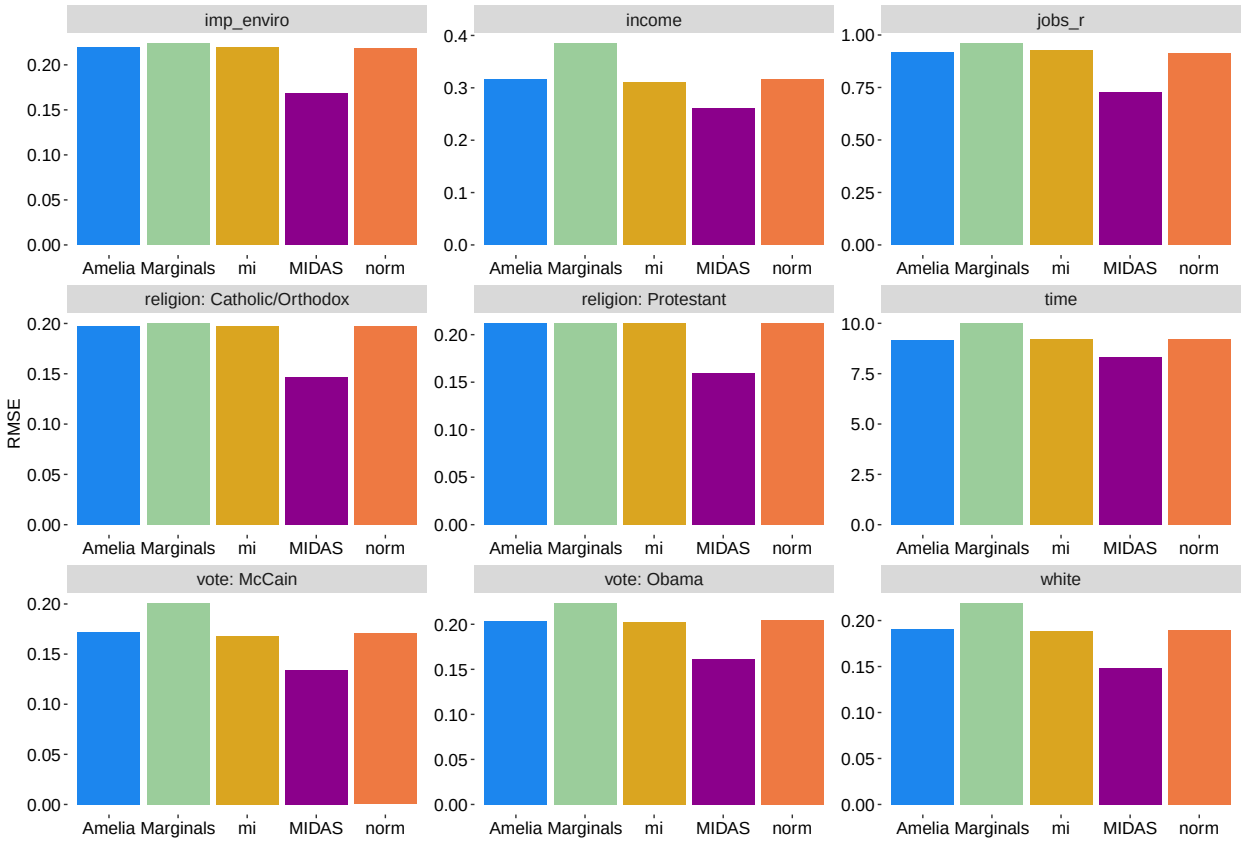
abilities of binary and nominal variables into realized values through draws from further random distributions. This practice can lead to misleading results because the random draws may end up generating realized values with low predicted probabilities, resulting in large imputation error. Consequently, we maintain the raw predicted probabilities when comparing strategies.[6]

As in our applied accuracy test (Section 3.3 of the main text), we do not replicate the model-based component of the test because we do not know the true joint distribution of the data. Furthermore, since missingness in the original ANES is not completely random (according to Little's (1988) standard test), the parameters of a model estimated on the test subset may be nontrivially biased.

Figure A2 plots the average RMSE of imputed values generated by each strategy across the 20 completed datasets simulated in the test. Consistent with the results of our other accuracy tests, MIDAS's imputed values are more accurate than those of the remaining strategies for all 11 variables. This gap is particularly sizable for the nominal *religion* variables, where MIDAS's error is approximately 25% lower than that of every other strategy. It is worth reiterating, moreover, that this test presents favorable conditions for existing MI strategies; were it to be run on a wider subset of the ANES, MIDAS's (relative) performance would likely improve further.

[6]This is not possible in the case of **mi**, which automatically converts imputed values of binary variables to 0 or 1.

Figure A2. Inverse Imputation Accuracy in Kropko et al. Applied ANES Test



Lower RMSE indicates worse average imputation accuracy across the 20 completed datasets simulated in the test. Nominal variables (*vote*, *religion*) are one-hot encoded, with the residual category omitted. Ordinal variables (*income*, *jobs_r*) are transformed to continuous variables prior to imputation and not converted back subsequently. *female*, *age*, *education*, *married* are kept complete across simulations.

5 Additional Information on Scalability Analysis

5A List of Variables in Column-Wise Test

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*

5B List of Variables in Row-Wise 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

6A 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 A3.

6B 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

Table A3. List of CCES Policy Variables Included in Latent Ideology Estimation

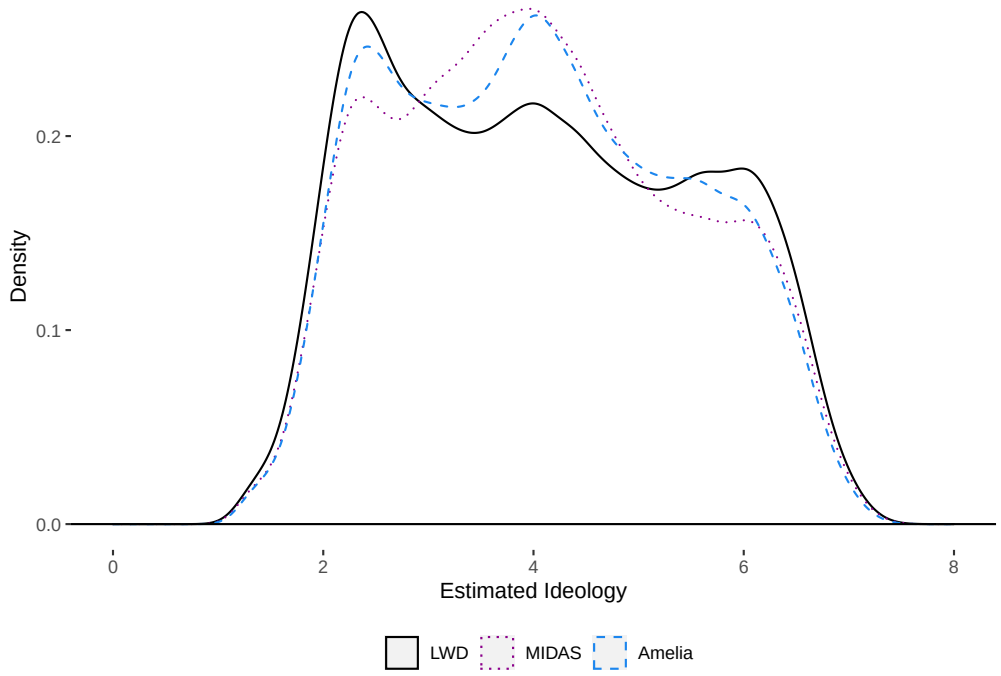
Variable	Policy.Area	Response.Type	Missing
CC18_414A	Minimum Wage	For/Against	8202
CC18_414B	Millionaire's tax	For/Against	8216
CC18_414C	Sales tax	For/Against	8233
CC18_414D	Income tax	For/Against	8230
CC18_414E	Abortion spending	For/Against	8223
CC18_324a	Government Spending	Support/Oppose	8304
CC18_324b	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

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 8).[7] As with MIDAS, we then generate 15 completed datasets and recover latent ideology estimates from the fitted

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

Figure A3. Comparison of Latent Ideology Estimates from Different MI Strategies



values of the regression.

Figure A3 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 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.[8] 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).[9] 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.[10] We select six African countries — Cameroon, Côte D'Ivoire, Congo Republic, Ghana, Niger, and Zambia — over the

[8] 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: $p(X_1, X_2, ..., X_n) \neq p(X_{\pi(1)}, X_{\pi(2)}, ..., X_{\pi(n)})$.

[9] **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.

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

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 each cross-section (31 years $\times$ six countries).[11] 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.

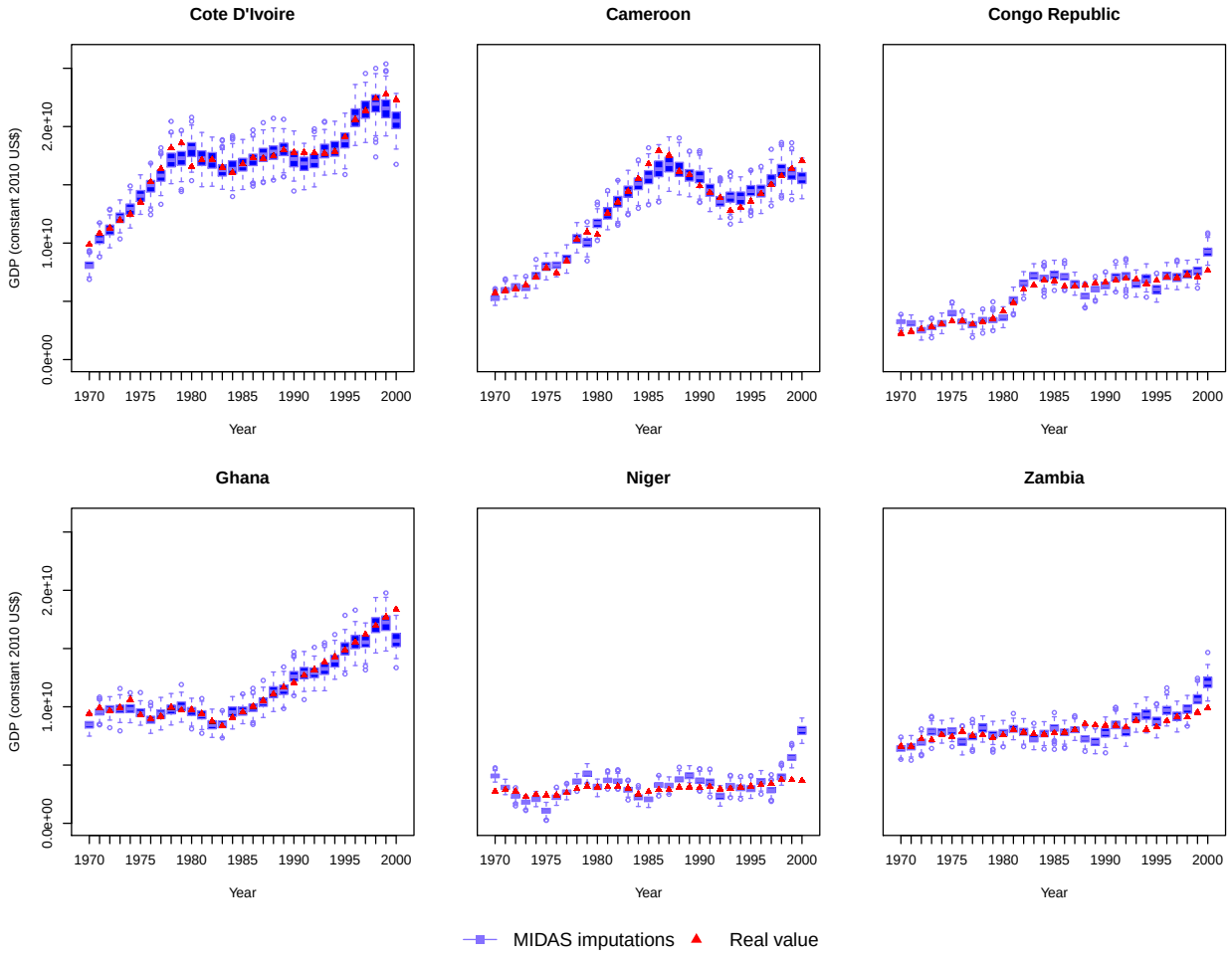
For each sample, we generate lags and leads of all (non-index) variables, since both past and future values of a given variable tend to be correlated with its present value (Honaker et al. 2011, 19). Based on an overimputation analysis (see Section 1), we instantiate MIDAS with two hidden layers of 1024 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.[12] 200 completed datasets are then produced with each model.

Figure A4 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.

[11] 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.

[12] 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 A4. 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.

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 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, M., J. Honaker, and G. King (2017). A unified approach to measurement error and missing data: Overview and applications. *Sociological Methods & Research 46 (3)*, 303–341.

Christiansen, B. (2005). The shortcomings of nonlinear principal component analysis in identifying circulation regimes. *Journal of Climate 18*(22), 4814–4823.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. Cambridge, MA: The MIT Press.

Honaker, J. and G. King (2010). What to do about missing values in time-series cross-section data. *American Journal of Political Science 54 (2)*, 561–581.

Honaker, J., G. King, and M. Blackwell (2011). Amelia II: A program for missing data. *Journal of Statistical Software 45 (7)*, 1–47.

Kingma, D. P. and M. Welling (2013). Auto-encoding variational bayes. *ArXiv e-prints*.

Kropko, J., B. Goodrich, A. Gelman, and J. Hill (2014a). Multiple imputation for continuous and categorical data: Comparing joint multivariate normal and conditional approaches. *Political Analysis 22 (4)*, 497–219.

Kropko, J., B. Goodrich, A. Gelman, and J. Hill (2014b). Replication data for: Multiple Imputation for Continuous and Categorical Data: Comparing Joint Multivariate Normal and Conditional Approaches.

Little, R. J. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American Statistical Association 83 (404)*, 1198–1202.

Probst, P., A.-L. Boulesteix, and B. Bischl (2019). Tunability: Importance of hyperparam-

eters of machine learning algorithms. *Journal of Machine Learning Research 20*(53), 1–32.

Rezende, D. J., S. Mohamed, and D. Wierstra (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286. ACM.

Scholz, M. (2012). Validation of nonlinear pca. *Neural Processing Letters 36 (1)*, 21–30.