Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

When to Impute? Imputation before and during cross-validation

Byron C. Jaeger*[1] | Nicholas J. Tierney[2] | Noah R. Simon[3]

[1]Department of Biostatistics, University of Alabama at Birmingham, Birmingham, Alabama
[2]Department of Econometrics and Business Statistics, Monash University, Melbourne, Victoria, Australia
[3]Department of Biostatistics, University of Washington, Seattle, Washington

Correspondence
*Byron C. Jaeger. Email: bcjaeger@uab.edu

Present Address
327M Ryals Public Health Building 1665 University Blvd Birmingham, Alabama 35294-0022

Cross-validation (CV) is a common sample splitting technique to estimate generalization error for prediction models. For pipeline modeling algorithms (*i.e.,* modeling procedures with multiple steps), it has been recommended that the *entire* sequence of steps be carried out during each replicate of CV to mimic the application of the entire pipeline to an external testing set. There is one exception: unsupervised variable selection (*i.e.,* ignoring the outcome) can be applied before conducting CV without incurring bias. However, it is unclear whether an unsupervised operation that modifies the training data (*i.e.,* imputation) rather than removing columns from the training data (*i.e.,* variable selection) can be applied prior to CV without causing model error estimates to become overly optimistic. We conducted empirical experiments to assess whether conducting unsupervised imputation prior to CV would bias estimates of generalization error. Results from simulation and resampling studies show that despite a slight optimistic bias, the reduced variance of imputation before CV compared to imputation during each replicate of CV lead to a lower overall root mean squared error for external R^2. In conclusion, unsupervised imputation before CV appears valid in certain settings and may be a helpful strategy that enables analysts to use more flexible imputation techniques without incurring high computational costs.

KEYWORDS:
Class file; LaTeX; Statist. Med.; Rmarkdown;

1 | INTRODUCTION

In evaluating the performance of predictive modeling algorithm, it is understood that so-called training error (the predictive error measured on observations used to fit the model) is a poor proxy for generalization error (the performance of the model on future, as-yet-unseen, observations). The training error of a model will often be overly optimistic for the generalization error. As such, it is standard practice to use sample-splitting methods to estimate generalization error. These methods train and test

models using separate datasets. Cross-validation (CV) is a common sample-splitting method that partitions a dataset into v non-overlapping subsets (*i.e.,* folds). Each fold is then used as an internal testing set for a modeling algorithm that is developed using data from the $k - 1$ remaining folds. Aggregating errors from all k replications of this procedure provides an estimate of the modeling algorithm's generalization error, making k-fold CV an ideal technique to 'tune' modeling algorithms (*i.e.,* select optimal values for parameters that govern the algorithm's fitting procedure).

Machine learning analyses often involve 'pipeline' modeling algorithms, which are sequences of operations that may include data pre-processing, predictor variable selection, model fitting, and ensembling. [1] For example, a pipeline may begin by centering and scaling predictor values, then filter out redundant correlated predictors, and finally fit a regression model to the remaining data. To estimate the generalization error of pipeline modeling algorithms using CV, it is recommended that the entire sequence of steps be carried out during each replicate of CV to mimic the application of the entire pipeline to an external testing set. However, it has been stated that unsupervised variable selection steps (*i.e.,* steps that ignore the outcome variable) can be applied before conducting CV without incurring bias. [2] Since unsupervised predictor variable selection does not involve outcome variables, it does not give the selected predictors an unfair advantage during CV.

Missing data (MD) occur frequently in machine learning analyses, and several learning algorithms (e.g., regression) are incompatible with MD. Imputation is a technique that replaces MD with estimated values, and is often among the most computationally expensive operations in pipeline modeling algorithms. For example, the `missForest` imputation algorithm may fit one random forest model for each column that contains MD. Computational expense of applying `missForest` or other complex imputation strategies during each replicate of CV may lead analysts to prefer more convenient but less effective strategies to handle MD. A more computationally efficient approach would be to implement 'unsupervised imputation' (*i.e.,* imputing MD without accessing outcome information) *before* conducting CV. However, it is unclear whether an unsupervised operation that modifies the training data (*i.e.,* imputation) rather than removing columns from the training data (*i.e.,* variable selection) can be applied prior to CV without causing model error estimates to become overly optimistic. In addition, it is worth investigating whether unsupervised operations of this type can result in poorly selected tuning parameters and thus degrade the external performance of downstream models.

In this manuscript, we conduct empirical studies assessing whether unsupervised imputation implemented to the training data before CV (a strategy we will refer to as `I`$\rightarrow$`CV`) can yield error estimates for the entire pipeline modeling algorithm that consistently estimate its generalization error. We compare estimated pipeline error according to `I`$\rightarrow$`CV` with estimated pipeline error when unsupervised imputation is applied *during each replicate* of CV (a strategy we will refer to as `CV` $\circlearrowleft$ `I`). Both simulated and real data are leveraged to draw these comparisons, and all scripts used to generate results are publically available on the first author's GitHub. Our analysis also introduces and applies the `ipa` R package (**i**mputation for **p**redictive **a**nalytics), which provides functions to create single or multiple imputed training and testing sets for prediction modeling.

The rest of this manuscript is organized as follows. In Section 2, we discuss MD mechanisms and prevailing MD strategies for statistical inference and machine learning. In Section 3, we explicitly map the order of operations for CV ↻ I and I → CV. In section 4, we conduct a simulation study to assess empirical differences between CV ↻ I and I → CV. The two procedures are compared using real data in Section 5. Last, in Section 6, we organize the data from preceding sections to form recommendations for practitioners.

2 | MISSING DATA

Missing data mechanisms

MD mechanisms were first formalized by Rubin,[3] who developed a framework to analyze MD that supposes each data point has some probability of being missing. If the probability of missingness is unrelated to the data (*i.e.,* all data are equally likely to be missing), then the data are missing completely at random (MCAR). When the probability of missingness is related to observed variables in the data (*i.e.,* all data within observed groups are equally likely to be missing), the data are missing at random (MAR). If the probability of missingness is determined by reasons that are unknown or unobserved, the data are missing not at random (MNAR). To illustrate, if a doctor did not run labs for a patient because the clinic was too crowded at the time, the patient's data are MCAR. If instead the doctor chose not to measure the patient's labs because the patient was too young, the patient's data are MAR. If the patient missed the appointment because the patient was too sick, the patient's data are MNAR. In the context of statistical learning, previous findings have shown that when data are MNAR, imputation alone is often less effective than incorporating features that characterize missing patterns (*e.g.,* missingness incorporated as an attribute).[4,5,6] Since the primary aim of the current study is to assess the differences between two implementations of imputation (*i.e.,* I → CV and CV ↻ I), we focus analyses on cases where data are MAR or MCAR.

Missing data strategies for statistical inference

The primary objective for statistical inference in the presence of MD is to obtain valid test statistics for statistical hypotheses. Imputation to the mean and, more broadly, MD strategies that create a single imputed value, have been shown to increase type I errors for inferential statistics by artificially reducing the variance of observed data and ignoring the uncertainty attributed to MD, respectively. Multiple imputation, a widely recommended strategy to handle MD for statistical inference, is capable of producing valid test statistics when data are MCAR or MAR because it can simultaneously address the two shortcomings listed in the previous sentence. It is notable that the 'accuracy' of imputed values is not critical for the success of multiple imputation, given sufficient estimates of conditional distributions[7]. Instead, it is the consistency of the estimated covariance matrix for regression coefficients that makes this strategy ideal for statistical inference.

Missing data strategies for statistical prediction

The primary objective for statistical prediction in the presence of MD is to develop a prediction function that accurately generalizes to external data that contain missing values. In contrast to statistical inference, single imputation is often used for prediction models. Moreover, imputation strategies with greater accuracy often lead to better performance of downstream models (*i.e.,* models fitted to the imputed data). For example, Jerez et al. found that single imputation using machine learning models provided superior downstream model prognostic accuracy compared to multiple imputation based on regression and expectation maximization.[8] The results of this analysis exemplify a perspective that will be taken throughout the current study. Namely, the authors treated imputation strategies as components of the modeling pipeline, with parameters that can be 'tuned' in the same manner as a prediction model.

3 | ORDER OF OPERATIONS

In the context of statistical prediction, analysts usually work with a training set and and an external testing set. A pipeline modeling algorithm developed with data from the training set can be externally validated using data from the testing set. Workflows to develop and validate a pipeline model often include three steps: (1) selection of pipeline parameter values (*i.e.,* parameters relevant to any operation in the pipeline), (2) training and application of data pre-processing operations for the training and testing sets, separately, (3) development of a prediction model using data from the training set, and (4) validation of the prediction function using data from the testing set. (**Figure** 1). Pipeline parameter values may be set apriori or determined empirically using resampling (*e.g., k*-fold CV). We refer to the $k-1$ folds and k remaining fold used to internally train and test a modeling algorithm as analysis and assessment sets, respectively, to avoid abuse of notation.

If CV is applied to facilitate selection of pipeline parameter values, it is critical that analysis data are separated from assessment data before any 'learning' is done. The entire *supervised* pipeline must be run using only the assessment data. This applies both to supervised data pre-processing steps (*e.g.,* selecting all variables with high correlation to the outcome) as well as supervised modeling procedures (*e.g.,* regression). 'Data leakage' can occur when outcome information from the assessment set is leveraged to modify the analysis set, *e.g.,* supervised variable selection is performed on a stacked set comprising analysis and assessment data, rather than just the assessment data (CITE). There are a number of examples showing wildly optimistic estimates of generalization error because of data leakage. In scenarios with a larger number of features, even simple methodologies such as selecting those features with high individual correlation to the outcome can induce substantial bias (CITE).

To remove any possibility of data leakage, all steps of the pipeline may be performed in analysis and assessment sets, separately, within each replicate of CV. For example, consider centering and scaling predictor variables such that they have zero mean and unit variance. As these operations do not involve the outcome, they are entirely unsupervised. Nevertheless, centering

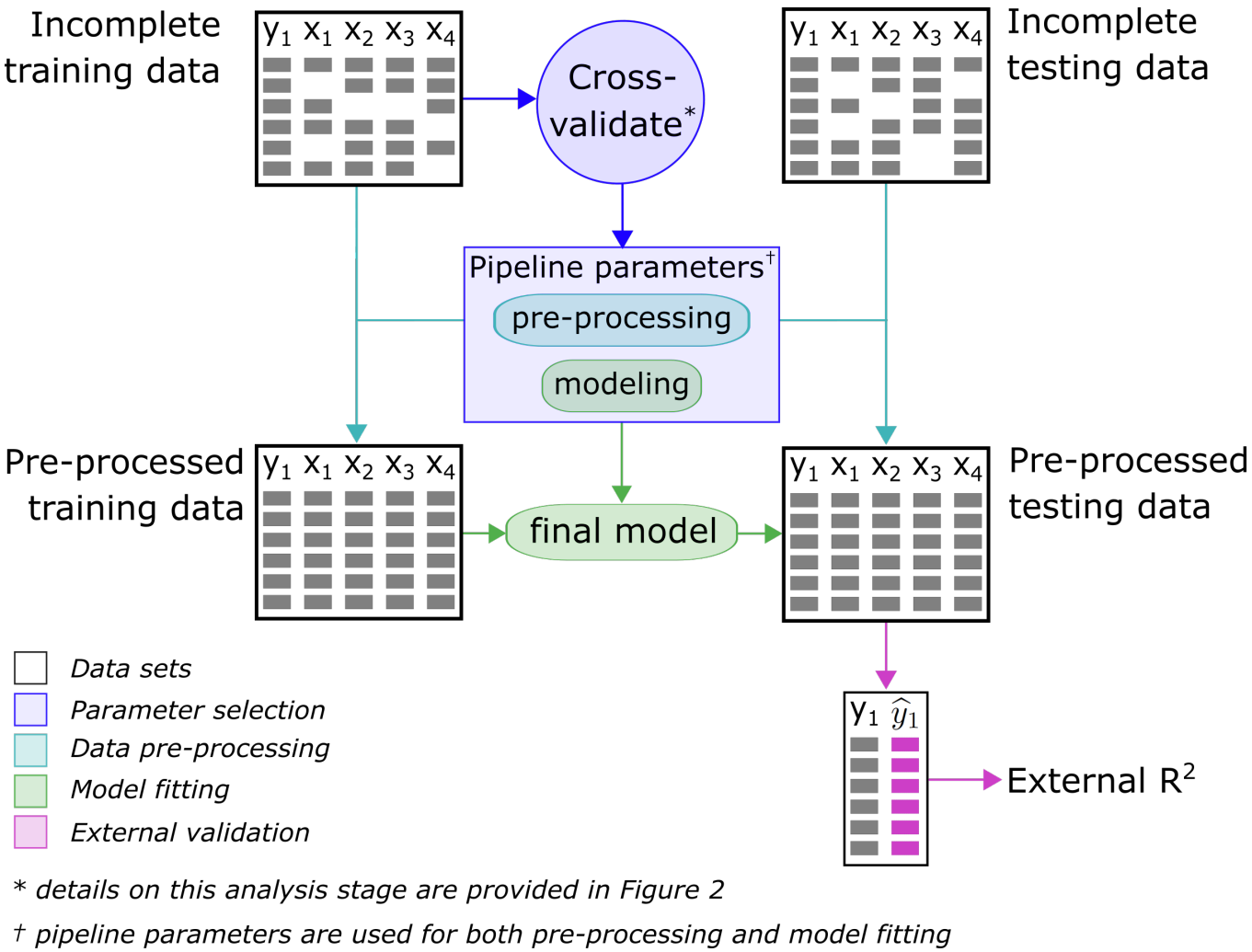


FIGURE 1 A modeling pipeline for machine learning analysis.

and scaling operations are usually completed in analysis and assessment sets, separately, during each replicate of CV. Specifically, the means and standard deviations are computed using the analysis data and then those values are applied to center and scale predictors in both the analysis and assessment sets (CITE). We refer to this traditional implementation of CV as `CV` ↻ `I` (**Figure** 2) and refer to our experimental implementation of CV (*i.e.,* one where unsupervised imputation occurs before CV begins) as `I`→ `CV` (**Figure** 3). Regardless of which implementation is applied, the output of CV is a set of pipeline parameter values and an estimate of generalization error. The pipeline parameter values are subsequently used to develop and validate a final prediction model using the full training set and testing set, respectively. Notably, intermediate estimates of generalization error are often used by CV learners [DEFINE CV LEARNER IN INTRO OR OTHER] (*e.g.,* `glmnet.cv`) to select tuning parameter values. Differences in the values selected by competing strategies (*i.e.,* `CV` ↻ `I` or `I`→ `CV`) may have measurable impact on the generalization error of their downstream models.

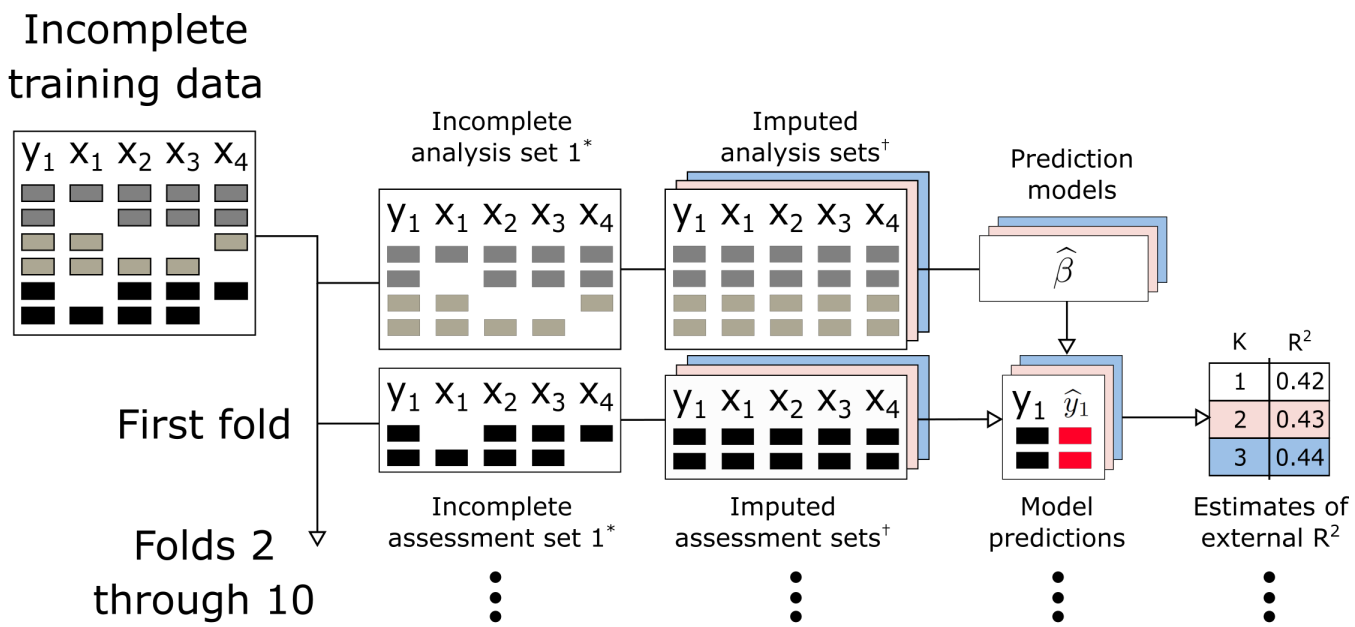


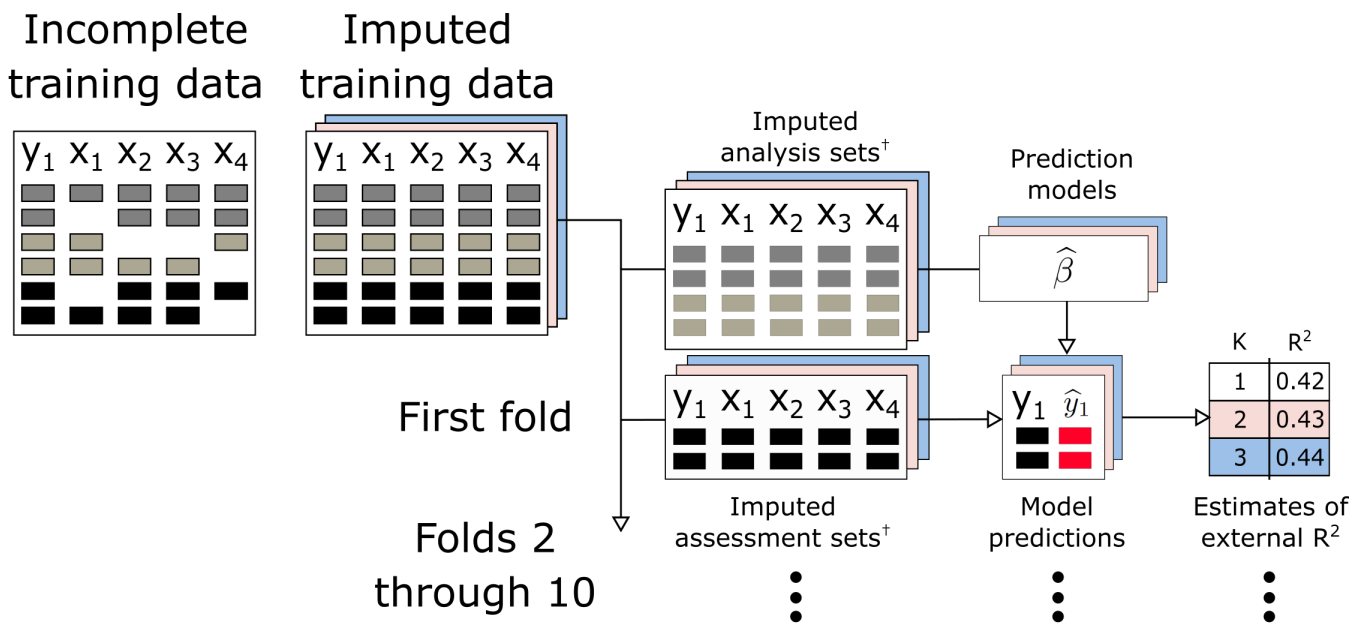
FIGURE 2 A traditional workflow for cross-validation that applies k-nearest neighbor unsupervised imputation within each replicate (*i.e.,* CV ↻ I).

3.1 | Testing data

Ideally, external testing data will not contain MD, and imputation will not be necessary. However, If MD are present in the external testing data, additional steps may be taken to perform imputation. One may approach the imputation of testing data using (1) only the training data, (2) only the testing data, or (3) using both training and testing data. It is common to use only the training data to impute MD in the testing data. However, some imputation procedures can only impute values in the data that were used to train the imputation procedure (*e.g.,* matrix decomposition methods such as `softImpute`).[9] To apply these types of imputation procedures, approach (2) or (3) may be taken. Throughout the current study, we use only the training data to impute missing values in the testing data, CV ↻ I uses only the analysis data to impute missing values in the assessment data, and I→CV uses a stacked version of the analysis and assessment data (*i.e.,* all of the training data) to impute missing values.

4 | SIMULATED EXPERIMENTS

The goal of the current simulation study was to assess empirical differences between CV ↻ I and I→CV. Our primary objective was to measure and compare how well each strategy approximated a model's true generalization error. We assessed estimation of true oracle external R^2 using bias, variance, and root-mean-squared error (RMSE). The RMSE provides an overall assessment of



* Analysis and assessment sets play the roles of training and testing, respectively, during cross-validation

† Imputed datasets are created using 1, 2, and 3 nearest neighbors

FIGURE 3 An experimental workflow for cross-validation that applies k-nearest neighbor unsupervised imputation prior to data splitting (*i.e.*, I→CV)

estimation accuracy that depends on both bias and variance. A secondary objective was to assess the performance of downstream modeling strategies whose tuning parameters were selected using CV ↻ I and I→CV.

4.1 | Data-generating mechanisms

Consider the linear regression model, where a continuous outcome vector $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$ is generated by a linear combination of predictor variables $\mathbf{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_p\right]$. This functional relationship is often expressed as

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon,$$

where β is a $p \times 1$ vector of regression coefficients and ε is a normally distributed $N \times 1$ zero-mean random error vector. In practice, $\mathbf{X}$ often has some 'junk' variables that are not related to the outcome. We fixed the number of true predictor variables at 10, the standard error of ε at 1, and set $\beta = [-1.00, -0.78, -0.56, -0.33, -0.11, 0.11, 0.33, 0.56, 0.78, 1.00]$ throughout the simulation study. Columns of $\mathbf{X}$ were generated from a multivariate normal distribution with a first order autoregressive correlation structure. Specifically, the correlation between columns $\mathbf{x}_i$ and $\mathbf{x}_j$ was $\rho^{|i-j|}$, where ρ was set to 3/4 throughout the study. We applied this design to generate a training set of varying size (100, 500, 1000, or 5000) along with an external validation set comprising 10,000 observations in each simulated replicate.

We created three data-generation 'scenarios'. In scenario 1, the observed data are independent and identically distributed (iid). In scenario 2, the data are iid conditional on an observed grouping variable. A total of 11 groups are formed, one in the validation set and the remaining 10 in the training set. Each group is characterized by a randomly generated mean value for its predictor variables. During CV, the observed groups are separated into ten folds to mimic the prediction of outcomes in a population with different characteristics. Scenario 3 is identical to scenario 2 except that the grouping variable is latent. Consequently, CV does not break the observed groups into separate folds for scenario 3.

Amputing data

We applied the `ampute` function from the `mice` R package to generate missing values in simulated data. In each replicate, 90% of observations comprised at least one missing value. We designated up to p MD patterns randomly in each simulation replicate, where p is the number of non-outcome columns in the simulated data. A MD pattern indicates which of the p predictor variables are set to missing. For each MD pattern, the number of missing variables was randomly set to an integer ranging from 1 to $p/2$. This procedure usually induced missing values in 30-50% of the data. When data were MAR, we applied the default method for the `ampute` function (`ampute.default.weights`) to induce missingness based on the observed variables. Throughout the experiment, we applied the same missing patterns and MD mechanism in the training set and the external validation set.

Modeling procedure

We applied k-nearest-neighbor imputation to handle MD and least absolute shrinkage and selection operator (LASSO) regression to develop prediction functions throughout the simulated experiments. Nearest neighbors in the training set were used to form imputed values in the training and external validation sets. We created one imputed set for each $k \in \{1, 2, \dots, 35\}$. We selected a value for the regularization parameter $lambda$ in each imputed dataset, separately, using 10-fold CV (*i.e.,* `cv.glmnet`). The λ value selected was the one that minimized cross-validated RMSE.

Analysis plan

We varied the scenario (1, 2, or 3; described in a preceding paragraph), missing mechanism (MCAR or MAR), ratio of predictor variables to junk variables (1:1, 1:4, and 1:49), and the number of training observations (N = 100, 500, 1,000, 5,000). We present results for each of 72 settings determined by these parameters and also provide overall summary statistics for scenarios 1, 2, and 3 when data are MCAR and MAR (*i.e.,* aggregating over training sample size and predictor to noise ratio). In each simulation replication, we computed the true oracle external R^2 in the validation set for each potential value of nearest neighbors (*i.e.,* $k \in \{1, 2, \dots, 35\}$). [DEFINE ORACLE] We also estimated external R^2 for each value of k using `CV` ↻ `I` and `I` → `CV`, separately, to evaluate how well these CV procedures estimated the true oracle external R^2. We assessed the difference between estimated external R^2 according to `CV` ↻ `I` and `I` → `CV` as well as the bias, variance, and root-mean-squared error (RMSE) of

these estimates. Last, we investigated the accuracy of downstream models when CV ↻ I and I→ CV were applied to select the number of neighbors to use for imputation and the regularization parameter for a penalized regression model.

4.2 | Results

Overall, a total of 89,962 out of 90,000 (99.96%) simulation replicates were completed over a span of 33,252 computing hours. Incomplete replicates were not analyzed, as these were replicates where at least one of the amputation, imputation, or prediction models did not converge. Across all replicates, the mean number of minutes used to form imputed data using CV ↻ I and I→ CV were 7.36 and 0.80, respectively, a ratio of 9.24. As a point of reference, using the full training set, the mean number of minutes needed to tune and fit `glmnet` models was 1.34.

Across all scenarios, the mean external R^2 ranged from 0.233 to 0.443 (**Table** 1). External R^2 values were positively correlated with training set size and the ratio of predictor variables to junk variables. Notably, the mean external R^2 values in scenario 1 were uniformly greater than corresponding mean external R^2 values in scenarios 2 and 3, and the maximum difference between mean external R^2 values in scenario 2 versus scenario 3 was 0. The mean absolute difference between external R^2 estimates using CV ↻ I and I→ CV shrunk towards zero as the size of the training set increased (**Table** 2). The differences between CV ↻ I and I→ CV were lowest in scenario 1 and greatest in scenario 2. These patterns were also present in visual depictions of external R^2 portrayed as a function of k neighbors (**Figure** 4).

Bias, variance, and RMSE

For scenario 1, the overall bias of R^2 estimates under MCAR using CV ↻ I was -0.00102 versus 0.00264 using I→ CV (**Table** 3). When the data were MAR, the overall biases were 0.00325 for CV ↻ I versus -0.00056 for I→ CV . In scenarios 2 and 3, the bias of CV ↻ I was lower than that of I→ CV , and I→ CV consistently provided overly optimistics error estimates. The overall standard deviation of R^2 estimates was higher for CV ↻ I versus I→ CV in all three scenarios and both missing data mechanisms. The difference in standard deviation was most pronounced in scenario 3 when data were MCAR (0.07296 [CV ↻ I] versus 0.06728 [I→ CV]; **Table** 4). Despite the optimistic bias of I→ CV in scenario 2, the reduced variance of this approach lead to a lower overall RMSE for external R^2 compared to CV ↻ I (**Table** 5). When the data were MCAR in scenario 2, CV ↻ I and I→ CV obtained RMSEs of 0.05776 and 0.05645, respectively. Similarly, when the data were MAR in scenario 2, overall RMSE values were 0.05754 and 0.05606.

Downstream model performance

When CV ↻ I and I→ CV were applied to select tuning parameters, the overall mean external R^2 was higher using CV ↻ I in 6 out of 6 comparisons (**Table** 6). The greatest overall difference in mean R^2 between downstream models occurred in scenario 1 when the data were MCAR (absolute difference in model R^2: 0.00029; relative difference in model R^2 : 0.07%).

5 | REAL DATA EXPERIMENTS

The goal of the current resampling study was to repeat the comparisons that were summarized in Section 4 between CV ↻ I and I→ CV using real, publically accessible data. A secondary objective was to assess how much results would change if different modeling strategies were applied.

Ames, Iowa housing data

The data we use in this resampling study describe the sale of individual residential property in Ames, Iowa from 2006 to 2010. The entire set contains 2930 observations and 80 variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) that can be leveraged to predict the sale price of homes.[10] We used a cleaned version of the Ames data for our own analyses by applying the `make_ames()` function, available in the `AmesHousing` R package.[11] We also log-transformed the skewed sale price outcome.

Analysis plan

We conducted a resampling study where the Ames housing data was randomly split into training ($N = 2198$, 75%) and testing ($N = 732$, 25%) sets in each of 5,000 iterations. In each resampling replicate, we implemented two separate modeling strategies to develop prediction functions using the training set: (1) unpenalized linear regression and (2) random forests. We also implemented two imputation strategies: (1) nearest neighbor imputation using 1, 2, …, 35 neighbors and (2) mean and mode imputation for numeric and nominal variables, respectively. In addition to imputation, data were pre-processed by lumping values in discrete variables into an 'other' category if the value accounted for less than 10% of the observed values. Both CV techniques (*i.e.,* CV ↻ I and I→ CV) were implemented to estimate the external generalization error of the linear regression and random forest models when nearest neighbor imputation was applied.

Amputing data

The training and testing data were amputed in the same manner using four prototypes of missingness. The prototypes were characterized by having missing values for all variables describing the house (1) lot and garage, (2) longitude and latitude, (3) basement and year built, and (4) overall quality and general above ground square footage. We restricted our prediction models to consider only the 30 predictor variables belonging to at least one of these missing prototypes.

5.1 | Results

A total of 5000 out of 5000 (100%) resampling replicates were completed over a span of 927 computing hours. Across all replicates, the mean number of minutes used to form imputed data using CV ↻ I and I→ CV were 10.1 and 1.01, respectively. The mean (standard deviation) external R^2 for the linear regression and random forest models were 76.4 (2.65) and 80.9 (2.05), respectively. Overall, both CV techniques slightly over and under estimated the external R^2 value when linear regression and

random forests were applied, respectively. For linear regression, the mean (standard deviation) bias was -0.15 (3.43) and -0.21 (3.43) for CV ↻ I and I → CV, respectively. The standard deviation of error estimates were 1.153 and 1.145, respectively. For random forests, the mean (standard deviation) bias was 0.09 (2.55) and 0.05 (2.55) for CV ↻ I and I → CV, respectively. The standard deviation of error estimates were 0.837 and 0.857, respectively.

When CV ↻ I and I → CV were applied to select the number of neighbors used for imputation, downstream linear models obtained an external R^2 of 0.766 (0.026) and 0.766 (0.026), respectively. Similarly, downstream random forests obtained an external R^2 of 0.810 (0.020) and 0.810 (0.020), respectively. As a reference point, the mean (standard deviation) downstream external R^2 was 0.73 (0.03) and 0.81 (0.02) using linear regression and random forests, respectively.

Paragraph to contextualize and interpret the results.

6 | DISCUSSION AND RECOMMENDATIONS

We demonstrated empirical properties of CV ↻ I and I → CV using nearest-neighbor imputation prior to applying regression and random forest models. We selected these methods because they have been studied thoroughly and are widely used in applied settings. In simulated experiments, we generated outcomes using linear effects without interaction. We also studied three broad scenarios that were relevant to CV: Scenario 1 was an ideal setting where I → CV and CV ↻ I should have provided almost identical estimates of generalization error. Scenarios 2 and 3 were meant to test whether I → CV produced biased estimates of generalization error because in settings where I → CV clearly did not mimic the final application of a trained model to an external validation set. Remarkably, despite its bias in scenario 2, the reduction in variance of R^2 estimates using I → CV lead to a lower overall RMSE compared to CV ↻ I. Downstream model performance was consistently superior when CV ↻ I was used instead of I → CV. However, the increase in performance was smaller than 1% relative change (maximum overall relative difference in external R^2: 0.07%). While this difference is very small, it may be possible to find a different generative scenario where the difference is larger.

Unsupervised imputation has two interesting characteristics relevant to predictive modeling. First, it allows for imputation of testing data whose outcome data are unobserved (not missing), *e.g.,* predicting risk for incident cardiovascular disease among adults with no history of cardiovascular disease. Second, as the current analysis has shown, unsupervised imputation can be applied before CV in select settings without inducing overly optimistic estimates of model error. The benefits of this approach include (1) reduced computational overhead, (2) reduced variance in model error estimates, and (3) little difference in the performance of downstream models. If investigators are confident that training and testing data are identically distributed or are primarily concerned with selecting optimal tuning parameters, I → CV may be an extremely convenient workflow to implement. However, the drawbacks of I → CV include increased bias for model estimation, particularly in settings similar to scenario 2

(described in Section 4.1). If investigators are primarily interested in estimating model error without bias and cannot rule out the possibility that testing data are drawn from a different population or distribution than their training data, the current study clearly suggests CV ↻ I should be applied instead of I → CV. However, it is worth noting that almost all prediction modeling decisions require balancing bias and variance to optimize precision. Our results do not indicate any strong difference between I → CV and CV ↻ I with regard to precision (*i.e.,* RMSE, see **Table** 5). We suspect that in most prediction modeling applications, precision rather than bias is of primary interest.

The current study has several strengths. We implemented computational experiments using real and simulated data. We included different data-generation mechanisms, different modeling procedures, different MD patterns, and different modeling strategies to ensure our results generalized to several common analytical settings. We examined a wide variety of metrics to assess the benefits and weaknesses of applying I → CV versus CV ↻ I. Last, we have provided (1) an R package that conducts unsupervised imputation and (2) a github repository disseminating all of the code necessary to reproduce our results. Each of these supplemental components ensure that our work is easily reproduced and disseminated. There are also some gaps in the current study that can be filled by future work. We investigated v-fold CV in the current analysis. Future research may assess whether these results generalize to other forms of data-splitting such as Monte-Carlo CV or bootstrap CV. Because MNAR data present challenges that may not be overcome by imputation alone, we did not include simulations for MNAR data. Whether the current study's findings generalize to settings with MNAR data remains an interesting, unanswered question. Last, the current study has applied k-nearest neighbor imputation throughout. As many other types of imputation procedures have been established, there are numerous extensions of the current analysis that may explore whether our results hold when other imputation approaches are implemented.

TABLE 1 True external R^2 values for the modeling technique that is internally assessed using CV ↻ I and I → CV.

	Scenario 1		Scenario 2		Scenario 3	
N	MAR	MCAR	MAR	MCAR	MAR	MCAR
10 predictors, 10 junk						
100	37.9 (3.44)	37.8 (3.42)	33.5 (6.54)	33.2 (6.83)	33.5 (6.55)	33.3 (6.82)
500	42.8 (2.93)	42.7 (2.95)	40.0 (5.13)	39.8 (5.16)	40.0 (5.15)	39.8 (5.17)
1000	43.5 (2.93)	43.5 (2.94)	41.0 (4.73)	40.8 (4.78)	41.0 (4.73)	40.8 (4.78)
5000	44.3 (3.00)	44.1 (3.00)	42.0 (4.36)	41.8 (4.49)	42.0 (4.36)	41.8 (4.49)
10 predictors, 40 junk						
100	34.5 (3.79)	34.5 (3.72)	30.5 (6.80)	30.5 (6.58)	30.5 (6.86)	30.5 (6.62)
500	40.6 (2.76)	40.6 (2.76)	38.3 (4.61)	38.2 (4.70)	38.3 (4.63)	38.2 (4.71)
1000	41.5 (2.73)	41.5 (2.74)	39.3 (4.57)	39.2 (4.61)	39.3 (4.57)	39.2 (4.60)
5000	42.6 (2.74)	42.5 (2.74)	40.6 (4.21)	40.5 (4.25)	40.6 (4.21)	40.5 (4.26)
10 predictors, 490 junk						
100	27.6 (5.08)	27.7 (5.02)	23.3 (6.58)	23.3 (6.45)	23.4 (6.46)	23.3 (6.45)
500	37.5 (2.93)	37.5 (2.92)	35.8 (4.07)	35.9 (4.04)	35.8 (4.06)	35.9 (4.05)
1000	38.8 (2.83)	38.8 (2.83)	37.2 (4.34)	37.2 (4.33)	37.2 (4.35)	37.2 (4.34)
5000	39.8 (2.78)	39.8 (2.79)	38.6 (4.01)	38.6 (3.93)	38.6 (3.99)	38.6 (3.94)
Overall						
—	39.3 (5.50)	39.3 (5.45)	36.7 (7.22)	36.6 (7.21)	36.7 (7.21)	36.6 (7.21)

All values are scaled by 100 for convenience

TABLE 2 Mean absolute differences in estimates of external R^2 between CV ↻ I and I → CV.

	Scenario 1		Scenario 2		Scenario 3	
N	MAR	MCAR	MAR	MCAR	MAR	MCAR
10 predictors, 10 junk						
100	1.20 (1.02)	1.18 (0.98)	2.39 (1.71)	2.42 (1.72)	1.82 (1.49)	1.83 (1.47)
1000	0.21 (0.19)	0.21 (0.20)	1.19 (1.06)	1.25 (1.15)	0.62 (0.82)	0.64 (0.88)
500	0.31 (0.29)	0.33 (0.29)	1.32 (1.08)	1.38 (1.15)	0.74 (0.86)	0.76 (0.89)
5000	0.09 (0.08)	0.09 (0.09)	1.02 (0.89)	1.08 (1.00)	0.48 (0.76)	0.50 (0.82)
10 predictors, 40 junk						
100	1.48 (1.26)	1.48 (1.29)	2.67 (1.91)	2.68 (1.92)	2.13 (1.72)	2.15 (1.71)
1000	0.22 (0.20)	0.22 (0.20)	1.25 (1.10)	1.31 (1.18)	0.65 (0.90)	0.67 (0.95)
500	0.34 (0.30)	0.34 (0.29)	1.38 (1.20)	1.43 (1.26)	0.76 (0.92)	0.79 (0.98)
5000	0.09 (0.09)	0.10 (0.09)	1.13 (1.01)	1.21 (1.13)	0.52 (0.83)	0.55 (0.90)
10 predictors, 490 junk						
100	2.13 (1.74)	2.00 (1.64)	3.00 (2.16)	3.00 (2.14)	2.60 (1.93)	2.67 (2.00)
1000	0.21 (0.19)	0.22 (0.19)	1.14 (1.17)	1.13 (1.14)	0.59 (0.89)	0.58 (0.87)
500	0.35 (0.30)	0.34 (0.29)	1.22 (1.13)	1.20 (1.11)	0.70 (0.91)	0.70 (0.90)
5000	0.09 (0.08)	0.09 (0.08)	1.11 (1.06)	1.16 (1.13)	0.50 (0.85)	0.52 (0.90)
Overall						
—	0.56 (0.95)	0.55 (0.92)	1.57 (1.50)	1.60 (1.53)	1.01 (1.34)	1.03 (1.37)

All values are scaled by 100 for convenience

TABLE 3 Bias of external R^2 estimates using CV ↻ I and I→CV

	Missing completely at random						Missing at random					
	Scenario 1		Scenario 2		Scenario 3		Scenario 1		Scenario 2		Scenario 3	
N	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV
10 predictors, 10 junk												
100	0.41	-0.37	0.84	-1.33	0.06	-1.45	0.59	-0.21	1.11	-1.03	0.30	-1.19
500	0.14	0.04	0.20	-1.11	-0.55	-1.16	0.16	0.07	0.35	-0.90	-0.38	-0.96
1000	0.09	0.04	0.21	-1.01	-0.52	-1.06	0.08	0.03	0.33	-0.82	-0.37	-0.88
5000	-0.02	-0.02	0.12	-0.96	-0.55	-0.99	0.00	-0.01	0.27	-0.75	-0.36	-0.79
10 predictors, 40 junk												
100	0.64	-0.52	1.02	-1.45	0.27	-1.65	0.86	-0.32	1.26	-1.20	0.44	-1.42
500	0.23	0.09	0.53	-0.82	-0.23	-0.87	0.27	0.13	0.66	-0.64	-0.09	-0.70
1000	0.10	0.03	0.14	-1.13	-0.59	-1.17	0.10	0.03	0.19	-1.02	-0.51	-1.06
5000	0.07	0.04	0.08	-1.13	-0.63	-1.14	0.06	0.04	0.14	-0.99	-0.53	-1.01
10 predictors, 490 junk												
100	1.11	-0.69	1.44	-1.43	1.04	-1.49	1.04	-0.90	1.36	-1.51	0.96	-1.49
500	0.24	0.10	0.37	-0.68	-0.19	-0.71	0.37	0.21	0.49	-0.57	-0.08	-0.60
1000	0.15	0.08	0.20	-0.85	-0.42	-0.88	0.32	0.25	0.38	-0.67	-0.24	-0.71
5000	0.00	-0.02	0.31	-0.84	-0.39	-0.86	0.05	0.03	0.29	-0.80	-0.36	-0.82
Overall												
—	0.26	-0.10	0.45	-1.06	-0.22	-1.12	0.33	-0.06	0.57	-0.91	-0.10	-0.97

All values are scaled by 100 for convenience

TABLE 4 Standard deviation of external R^2 estimates using CV ↻ I and I→CV.

	Missing completely at random						Missing at random					
	Scenario 1		Scenario 2		Scenario 3		Scenario 1		Scenario 2		Scenario 3	
N	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV
10 predictors, 10 junk												
100	6.52	6.47	7.41	7.00	7.22	6.92	6.49	6.44	7.37	7.04	7.23	6.95
500	3.60	3.65	3.99	3.90	3.94	3.89	3.59	3.64	3.99	3.90	3.95	3.89
1000	3.33	3.36	3.70	3.57	3.62	3.56	3.32	3.35	3.67	3.56	3.60	3.55
5000	3.08	3.10	3.38	3.27	3.36	3.27	3.06	3.08	3.35	3.26	3.34	3.26
10 predictors, 40 junk												
100	6.76	6.63	7.23	6.85	7.07	6.80	6.66	6.57	7.21	6.84	7.08	6.77
500	3.53	3.57	3.75	3.65	3.68	3.65	3.55	3.59	3.73	3.65	3.70	3.66
1000	3.13	3.16	3.28	3.21	3.27	3.21	3.14	3.17	3.25	3.20	3.25	3.20
5000	2.82	2.83	3.00	2.88	2.96	2.88	2.81	2.83	2.96	2.88	2.95	2.88
10 predictors, 490 junk												
100	7.67	7.44	7.96	7.56	7.88	7.52	7.58	7.34	8.00	7.65	7.91	7.57
500	3.68	3.72	3.86	3.80	3.86	3.80	3.69	3.72	3.84	3.78	3.85	3.79
1000	3.25	3.27	3.27	3.20	3.26	3.20	3.25	3.28	3.29	3.23	3.29	3.23
5000	2.86	2.87	2.93	2.87	2.93	2.87	2.86	2.87	2.91	2.88	2.93	2.88
Overall												
—	6.50	6.12	7.30	6.76	7.30	6.73	6.52	6.12	7.31	6.77	7.29	6.73

All values are scaled by 100 for convenience

TABLE 5 Root-mean-squared error of external R^2 estimates using CV ↻ I and I→ CV

	Missing completely at random						Missing at random					
	Scenario 1		Scenario 2		Scenario 3		Scenario 1		Scenario 2		Scenario 3	
N	CV ↻ I	I→ CV	CV ↻ I	I→ CV	CV ↻ I	I→ CV	CV ↻ I	I→ CV	CV ↻ I	I→ CV	CV ↻ I	I→ CV
10 predictors, 10 junk												
100	5.97	5.87	9.08	8.84	8.87	8.78	5.91	5.76	8.80	8.54	8.61	8.50
500	2.23	2.22	5.09	5.03	5.06	5.04	2.18	2.18	5.07	4.97	5.03	5.00
1000	1.65	1.65	4.29	4.18	4.19	4.19	1.65	1.66	4.24	4.12	4.14	4.13
5000	0.82	0.82	3.50	3.47	3.49	3.48	0.81	0.81	3.33	3.27	3.31	3.28
10 predictors, 40 junk												
100	6.41	6.28	8.99	8.74	8.86	8.79	6.44	6.33	9.16	8.86	9.08	8.93
500	2.33	2.33	4.84	4.68	4.72	4.68	2.32	2.31	4.75	4.56	4.63	4.58
1000	1.67	1.66	4.33	4.32	4.29	4.32	1.68	1.69	4.29	4.27	4.25	4.27
5000	0.83	0.83	3.67	3.61	3.64	3.62	0.81	0.81	3.56	3.50	3.54	3.51
10 predictors, 490 junk												
100	7.29	7.22	9.01	8.86	8.93	8.89	7.30	7.28	9.10	8.98	8.85	8.80
500	2.43	2.42	4.18	4.05	4.13	4.06	2.41	2.39	4.14	4.00	4.06	3.99
1000	1.76	1.74	3.99	3.87	3.94	3.89	1.75	1.74	4.01	3.84	3.94	3.87
5000	0.85	0.85	3.23	3.10	3.16	3.11	0.86	0.86	3.31	3.19	3.23	3.17
Overall												
—	3.61	3.57	5.78	5.64	5.69	5.65	3.61	3.57	5.75	5.61	5.65	5.59

All values are scaled by 100 for convenience

TABLE 6 Mean external R^2 when CV ↻ I and I→CV were applied to tune the number of neighbors used for imputation.

	Missing completely at random						Missing at random					
	Scenario 1		Scenario 2		Scenario 3		Scenario 1		Scenario 2		Scenario 3	
N	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV	CV ↻ I	I→CV
10 predictors, 10 junk												
100	38.2	38.2	33.9	33.9	34.0	34.0	38.4	38.3	34.1	34.1	34.1	34.1
500	43.8	43.8	40.7	40.8	40.8	40.8	43.9	43.9	40.9	41.0	41.0	41.0
1000	44.8	44.8	42.0	42.0	42.0	42.0	44.9	44.9	42.1	42.2	42.1	42.2
5000	45.7	45.7	43.3	43.3	43.3	43.3	45.8	45.8	43.5	43.5	43.5	43.5
10 predictors, 40 junk												
100	35.1	35.0	31.0	31.0	31.0	31.0	35.1	35.0	31.0	30.8	31.0	30.9
500	41.8	41.8	39.2	39.2	39.2	39.2	41.8	41.8	39.2	39.3	39.3	39.3
1000	42.8	42.8	40.3	40.3	40.3	40.3	42.8	42.8	40.4	40.4	40.4	40.4
5000	44.0	44.0	41.8	41.8	41.8	41.8	44.0	44.0	41.9	41.9	41.9	41.9
10 predictors, 490 junk												
100	28.9	28.6	24.0	23.7	24.0	23.6	28.7	28.5	24.0	23.8	24.2	23.9
500	39.1	39.1	37.2	37.3	37.2	37.3	39.1	39.1	37.2	37.2	37.2	37.3
1000	40.4	40.4	38.7	38.7	38.7	38.7	40.4	40.4	38.7	38.7	38.7	38.7
5000	41.5	41.5	40.2	40.2	40.2	40.2	41.5	41.5	40.2	40.2	40.2	40.2
Overall												
—	40.5	40.5	37.7	37.7	37.7	37.7	40.5	40.5	37.8	37.8	37.8	37.8

All values are scaled by 100 for convenience

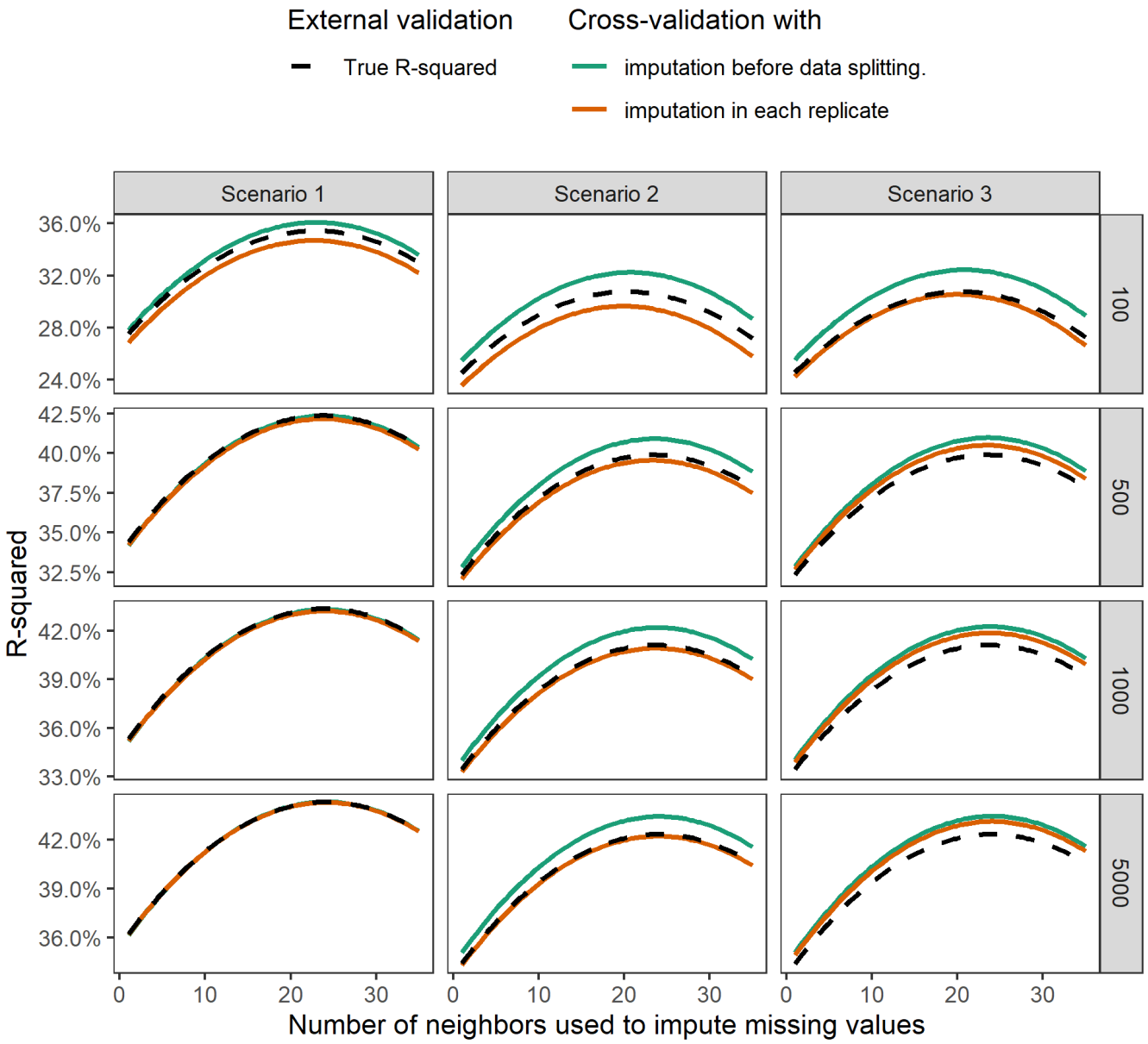


FIGURE 4 External generalization error and internal estimates of generalization error using `I`→`CV` and `CV`↻`I`.

References

1. Lang M, Binder M, Richter J, et al. mlr3: A modern object-oriented machine learning framework in R. *Journal of Open Source Software* 2019. doi: 10.21105/joss.01903

2. Hastie T, Tibshirani R, Friedman J. *The elements of statistical learning: data mining, inference, and prediction*. 2. Springer Science & Business Media . 2009.

3. Rubin DB. Inference and missing data. *Biometrika* 1976; 63(3): 581–592.

4. Twala B, Jones M, Hand DJ. Good methods for coping with missing data in decision trees. *Pattern Recognition Letters* 2008; 29(7): 950–956.

5. Twala B. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence* 2009; 23(5): 373–405.

6. Tang F, Ishwaran H. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 2017; 10(6): 363–377.

7. Van Buuren S. *Flexible imputation of missing data*. CRC press . 2018.

8. Jerez JM, Molina I, García-Laencina PJ, et al. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial intelligence in medicine* 2010; 50(2): 105–115.

9. Hastie T, Mazumder R. softImpute: Matrix Completion via Iterative Soft-Thresholded SVD. 2015. R package version 1.4.

10. De Cock D. Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project. *Journal of Statistics Education* 2011; 19(3).

11. Kuhn M. AmesHousing: The Ames Iowa Housing Data. 2017. R package version 0.0.3.