`MLIM` Multiple Imputation Handbook

Version `0.0.8`

E. F. Haghish
Department of Psychology
University of Oslo, Norway
haghish@uio.no

September 1, 2022

Contents

List of Code Listings

1 Introduction

`mlim` is a multiple imputation software package for **R** language, which uses automated machine learning to fine-tune the models and minimize the imputation error. In addition, `mlim` uses state-of-the-art computer science procedures to auto-balance imbalance data (see below), which results to fairer imputation, when one category is less represented or appears with lower prevalence. The latter can increase the coefficients estimated from imputed data, which is a big advantage for moving toward automated machine learning solutions for multiple imputations. The difference between `mlim` `mlim` automatically fine-tunes the model for imputing each variable, hence, the user is not required to define the parameters of the model and instead, the model is optimized for each variable, while avoiding overfitting. These features, to the author's knowledge, had not been implemented in any R package before.

This document provides a nice and clean documentation of the package and is occasionally updated with new stable releases. Visit the `mlim` repository on GitHub[1] to access more materials and tutorials about the package.

1.1 Automated machine learning for missing data imputation

In recent years, there have been several attempts for using machine learning for missing data imputation. Yet, `mlim` is unique because it is the first **R** package to implement automated machine learning for multiple imputation and brings the state-of-the-arts of machine learning to provide a versatile missing data solution. But how is automated machine learning any different from other machine learning imputation models? The difference is that automated machine learning algorithms *fine-tune* the models, resulting in more accurate predictions, in contrast to classical (static) machine learning models, where the model's parameters should be specified by the user. When it comes to missing data imputation, instead of setting up an algorithm with pre-specified parameters to impute all the variables of the dataset with the same parameters, we can allow the algorithm to effectively search for the optimal parameters for each variable. This dramatic change in approaching missing data imputation should explain why `mlim` outperforms other **R** packages because it optimizes the imputation of each variable. Compared to other **R** packages, `mlim` provides a better multiple imputation procedure because:

1. It automatically fine-tunes the parameters of each machine learning model for each variable

[1] `https://github.com/haghish/mlim`

2. It delivers a higher prediction accuracy, hence, lower imputation error
3. It is a fairer algorithm to minorities, correcting for biases emerging from imbalanced data
4. Does not make any assumption about the distributions of the variables
5. Takes the interactions between the variables into account
6. Does not enforce linear relations between the predictors
7. Uses a blend of different machine learning models
8. Implements a sophisticated procedure for optimizing imputed data with different algorithms

1.2 Installation

`mlim` is under fast development. The package receive monthly updates on CRAN. Therefore, it is recommended that you install the GitHub version until version 0.1 is released. To install the latest development version from GitHub:

```
# install mlim from GitHub
library(devtools)
install_github("haghish/mlim")

# install mlim from CRAN
install.packages("mlim")
```

2 Procedure

`mlim` support several algorithms. These algorithms have different computational needs and `mlim` optimizes the imputation procedure without requiring excessive computational resources. These algorithms, their computational requirements, and `mlim` optimization procedure is further explained below.

2.1 Imputation steps

`mlim` follows three steps to carry out the imputation:

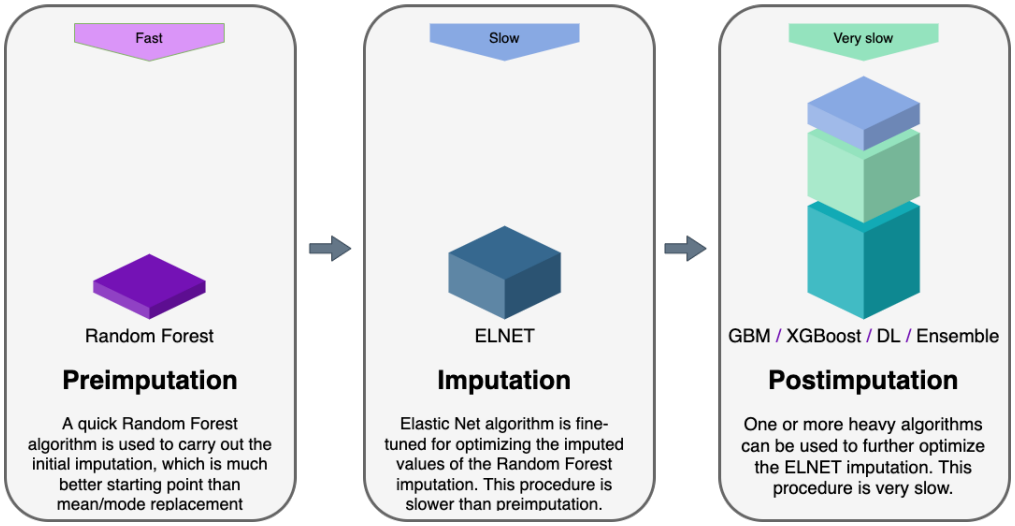
1. **Preimputation** : Instead of replacing missing observations with mean or mdoe, a very fast **RF** (Random Forest) algorithm is used to carry out a primary single imputation. The imputed data will be further optimized in the following steps.

2. **Reimputation** : The preimputed data will be reimputed with **ELNET** (default) or another algorithm.

3. **Postimputation** : The reimputed data can be even further optimized using a heavier algorithm that requires much more computational resources such as **GBM**, **XGB**, **DL**, or all of them via **Ensemble**. The latter trains a stacked ensemble from all of the available trained algorithms to minimize imputation error as much as possible, within the given fine-tuning time. **Note: Using 'Ensemble' is not recommended for general imputation practice and is intended for particular cases or smaller datasets were minimizing imputation error can make a significant difference.**

2.2 Preimputation, reimputation, and postimputation

although the third step is optional. The algorithm begins with a preimputation, which is a quick imputation with Random Forest. The reason for pre-imputation is to save time on the reimputation. Therefore, instead of replacing missing observations with mean and mode and starting a slow procedure for optimizing the imputed values, a fast and efficient algorithm is used for initial imputation and then, the imputed values are optimized. For the second step, the `ELNET` algorithm is fine-tuned. Note that what makes `ELNET` slower than Random Forest is the fact that `ELNET` is being fine-tuned but the preimputation procedure is not fine-tuned and a fixed model is used for any variable, regardless of variable type or data size.

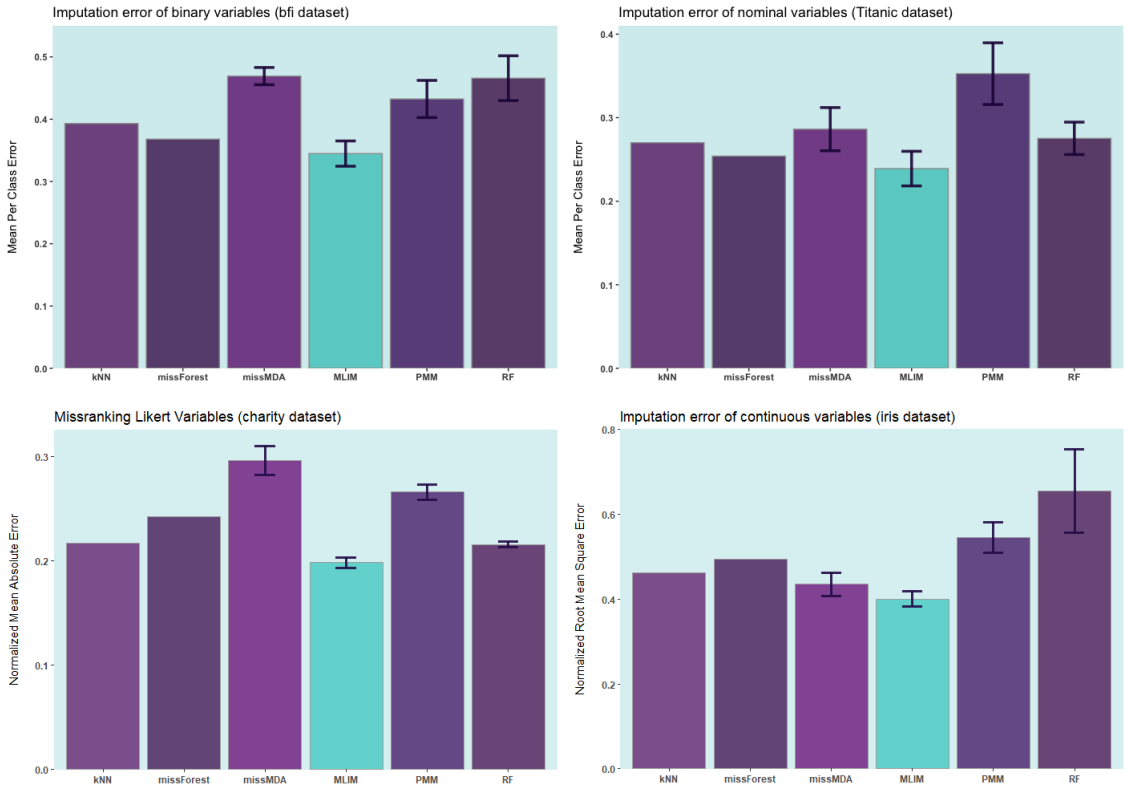


2.3 Supported algorithms

- **RF** (Random Forest and Extreme Randomized Trees). This algorithm is only used in *preimputation*, but it can also be used as the main imputation algorithm or *postimputation* (see below for definitions)
- **ELNET** (Elastic Net). This is the main default imputation algorithm
- **GBM** (Gradient Boosting Machine).
- **XGB** (Extreme Gradient Boosting, available in Mac OS and Linux).
- **DL** (Deep Learning).
- **Ensemble** (Stacked Ensemble, only works if other algorithms are also present).

3 Fast optimization with ELNET

As noted earlier, `mlim` carries out optimization process on preimputed data. The fastest algorithm for optimizing imputed missing data is `ELNET`, which is also more fair towards imbalanced data and categories with lower prevalence. As shown in the figures below, `ELNET` optimization already outperforms all other statistical and machine learning single or multiple imputation algorithms available in **R**.



The analysis above uses four different datasets and adds artificial missingness at the rate of 15% to all variables. Then several famous **R** algorithms are used to carry out either single or multiple imputation and the results are compared with `mlim` multiple imputation. There are many imputation algorithms that do not support multiple imputation. In fact, there are many situations where multiple imputation is not feasible or even needed. But as evident in the figures, in all four examples, `mlim` outperforms all other algorithms for binary, multinomia, ordinal (Likert), and continuous variables.

> `ELNET` outperforms other R packages and provides fairer imputations and thus, can result it more reliable single or multiple imputation with less biased estimates, particularly for imbalanced categorical variables. Next, you will see that the performance of `ELNET` can be further improved with more computationally intensive algorithms.

4 Postimputation with heavier algorithms

The reason `mlim` uses `ELNET` algorithm by default to optimize the imputed observations is that it is easy to fine-tune `ELNET`, without requiring a huge

amount of computing resources or runtime. Moreover, ELNET is a very stable algorithm and generalizes very well, which is a big advantage. On small datasets, other algorithms might simply overfit and even after hours of training, yield less accurate imputed values than ELNET does. However, when desired or needed, the imputed observations can be further optimized with heavier algorithms such as GBM, XGB, DL, RF, or a combination of these algorithms by adding Ensemble (in general, Ensemble is *highly recommended* for stabilizing the postimputation optimization).

The reader should be aware that postimputation procedure is extremely computation extensive and might not be suitable for standard laptops. If you decide to use strong optimization, make sure you run your analysis on a powerful computer, with a lot of CPU and RAM. The more RAM, the better! If you have 16 GB of RAM, you can still run postimputation, but keep a close eye on the RAM. However, when the data is large enough and the imputation models are well-tuned, the user will be rewarded with significantly lower imputation error and even fairer imputations, which can be of great incentive in some cases.

> **Note:** The fact that mlim can do Deep Learning imputation might seem intriguing, i.e., you might think that Deep Learning multiple imputation is more likely to result in a much more accurate single or multiple imputation. Here is a piece of advice: do not underestimate ELNET for its simplicity and speed. Very often, it outperforms other algorithms, while it can be fine-tuned at much faster speed. Comparing the errors of training, other algorithms outperform ELNET, but is there any guarantee that such a reduction of error will be generalizable to the unseen data or the missing values? Answering this question is not that easy. Overfitting can still happen, eventhough mlim performs 10-fold cross validation. Therefore, missing data imputation with heavy machine learning algorithms is a feature intended for advanced users. For general purpose, just use ELNET, which is the default algorithm and you are much more likely to get high efficiency, high accuracy, and even fairer imputed values.

5 Single imputation

Traditionally, multiple imputation have been the golden standards if treating missing data because they allow to estimate the uncertainty in guessing the missing observations and that uncertainty is desired. However, multiple imputation is not the only way of making such assessment about the imputed

values. Random Forest machine learning algorithm can provide Out Of Bag (OOB) error estimate, which also estimate the prediction error.

Although `mlim` can carry out multiple imputation, it can also be used to carry out a single imputation. Yet, `mlim` can provide estimates of imputation accuracy via Cross Validation metrics. The `mlim.summarize` function can print the accuracy of the imputation for each imputed variable:

6 Syntax

`mlim` is designed to be suitable for large datasets, despite its computational demands. However, for large datasets, you should be familiar with the syntax of the package as well as the machine you run your analysis on. There are settings for specifying number of CPU cores and RAM. The best place for reading the syntax is the package Vignette on CRAN[2]. I avoid reexplaining the syntax of the package because maintaining this document becomes difficult. Instead, I will focus on the practical aspects of using `mlim` and thus, for the rest of this document, I will assume that you are familiar with the package syntax.

[2] `https://cran.r-project.org/web/packages/mlim/mlim.pdf`