Mr. Clean: An Ensemble of Data Cleaning Algorithms for Increased Data Retention

Kenneth Smith
Department of Computer Science
University of Missouri - St. Louis
St Louis, MO , USA
kpsc59@umsystem.edu

Sharlee Climer
Department of Computer Science
University of Missouri - St. Louis
St Louis, MO, USA
climer@umsl.edu

***Abstract*—Handling missing data is a critical issue in nearly all analytical fields. Techniques that address this problem are generally categorized as either imputation or deletion algorithms. Imputation techniques replace missing data based on observed values and an assumed relationship, which can lead to biases in the imputed values and analysis results. Deletion programs remove missing data by deleting the corresponding sample or feature. This manuscript focuses on partial deletion, where some missing data is allowed, but samples and features with excessive missing data are removed. By intelligently selecting which rows and columns to delete, more valid data can be retained than with tradition deletion techniques. We developed three new algorithms for partial deletion: a greedy algorithm and two mathematical optimization programs. We compare these methods against the *DataRetainer*, *Auto-miss*, *list-wise*, and *feature-wise* programs, using several real-world data sets and a range of allowed missingness values. Our *Greedy* algorithm outperforms or ties existing algorithms in terms of run time and valid elements kept in nearly all scenarios. Our mathematical optimization programs further increase the number of valid elements kept, but require additional computational costs. These programs will allow researchers to retain more of their precious data thereby strengthening downstream analyses.**

***Index Terms*—Data Cleaning, data deletion, imputation**

I. Introduction

Missing data is common in real-world data sets across a wide range of domains, such as psychology [1], gene expression analysis [2, 3, 4], genomics [5], and epidemiology [6]. The mechanisms leading to missing data have been classified as either Missing Completely at Random (MCAR), Missing at Random (MAR), or Missing Not at Random (MNAR) [7]. In a MCAR process, the probability that a data element is missing does not depend on observed data, nor the value of the missing data. For a MAR process, the missing data is dependent, at least in part, on observed data, but not on the value of the missing data itself. Finally, in MNAR the probability that a value is missing depends on the missing value. While it is not always possible to determine which mechanism or mechanisms are at play, it is useful to identify how the various algorithms that handle missing data operate under each process.

The results from analyses that tolerate missing data can be greatly affected by the amount and locations of missing values, while the user may be unaware of the impact. Furthermore, many analysis techniques do not allow *any* missing data and require either deletions or imputation. In these cases, partial deletion can be performed prior to imputation to reduce the introduction of detrimental biases.

Missing data deletion is commonly carried out in either a list-wise or pairwise manner. In list-wise deletion, also called complete case analysis, every sample with missing data is removed. If the data is not MCAR, list-wise deletion results in bias [8]. In all scenarios, the power of the analysis may be reduced as there are fewer samples. Despite the reduction in power and possible bias, list-wise deletion was the default method in most statistical software packages [9]. A less common alternative to list-wise deletion involves removing all features with missing data. This helps to maximize the power of any test, but may remove features with strong predicting capabilities, as well as introducing bias, depending on the missing mechanism [10]. Pairwise deletion is applicable when researchers are testing a model on a specific subset of features. This may be part of a feature selection algorithm or may be used after the features have been selected. In pairwise deletion, a sample is only removed if it contains missing data in the current feature subset, regardless of missing data in other features. Pairwise deletion keeps more samples than list-wise deletion, but can suffer from a covariance matrix that is not positive definite [8], which causes problems for some analysis techniques.

Data sets can also be cleaned by removing a combination of samples and features. The *DataRetainer* [11, 12] and *Auto-miss* [13] algorithms allow researchers to clean data sets to a desired amount of missingness. In some studies, all missing data was removed before analysis [14]. While in other studies, excessive missing data (e.g. samples or features with more than 10% missing data), was removed before analysis [11,

Alzheimer's Association Research Grant (AARG-22-925002), University of Missouri - St. Louis Research Grants

979-8-3503-3748-8/23/$31.00 ©2023 IEEE

14, 15]. These studies demonstrate that significant patterns can be found in real-valued and categorical data with such an approach, even in highly studied data sets [15]. Unfortunately, the *DataRetainer* program requires large amounts of user interaction when cleaning data files. Furthermore, we are unaware of any detailed analysis of the run time or data retention rate of *DataRetainer* or *Auto-miss*.

Researchers who choose to handle missing data through imputation have several methods at their disposal, including mean, regression, local similarity, and multiple imputation. In mean imputation, any missing data is replaced with the mean value of the feature. This can be performed on the raw data, or missing data can be replaced with a value of 0 after normalization. Both cases bias the variance and covariance downward [10]. In regression imputation, researchers fit a regression model to the missing data using the observed data as predictors. This technique biases the variance downward and covariance upward [10]. Stochastic regression attempts to address the biases of regression imputation by including a random error term. However, biases to the p-value and confidence intervals (CI) are introduced [10]. Also, a survey by Musil [16] showed an increase in correlation between imputed values, even when an error term was added to the imputation model. Local similarity techniques impute missing values based on similar features. Similarity can be measured by L2 norm or Pearson's correlation. Often the top k-nearest features are then used to impute missing data [3]. Despite the more complex nature of some local similarity algorithms, Souto [4] found negligible impact to gene expression clustering and classification between them and simpler techniques.

Multiple Imputation (MI) is a sophisticated form of imputation that includes the advantages of stochastic regression, but better estimates the p-value and CI parameters. MI works in three steps. First several imputed versions of the data set are created. Austin [17] recommends using between 5 and 20 imputed versions. Second, each imputed data set is analyzed. Third, the results of each analysis are combined [10]. If an accurate imputation model is selected, MI will result in unbiased estimates for the MCAR and MAR mechanisms. However, MI can be computationally excessive, and the increased efforts and computation time may outweigh the benefits, especially for large data sets with relatively low numbers of missing values [18]. Additionally, there could still exist bias due to MNAR mechanisms. The bias may be reduced by using the missing indicator method [19], but this increases the number of features in the imputation model.

In this manuscript, we propose an ensemble of cleaning techniques, collectively called Mr. Clean, which remove rows and/or columns from a data matrix so that each remaining row and column contain less than a prescribed percentage of missing data. Unlike common deletion techniques, Mr. Clean aims to maximize the number of valid elements kept from the original data matrix, instead of maximizing the number of samples or features kept. The user may create a cleansed matrix with no missing data or may remove exceedingly large amounts of missing data as a preprocessing step before imputation or feature selection.

Mr. Clean is composed of three algorithms: a greedy solver and two Integer Linear Program (ILP) solvers, called *RowCol* and *Element*. A greedy algorithm is an approximation technique that chooses the locally 'best' option at every decision point. While greedy algorithms tend to run fast, they are not guaranteed to find the optimal solution and sometimes perform quite poorly regarding accuracy. An ILP is a mathematical optimization program in which all the variables are restricted to integers and the objective function and constraints are linear. ILPs are generally NP-complete and may require unacceptable time or memory to converge. However, given an efficient implementation and ILP definition, they can usually be executed for small to medium sized problems and are guaranteed to find the optimal solution, if allowed to run to completion.

II. Methods

A. Experimental Design

We begin by considering a data matrix, $\mathbf{D}_{m\times n}$, where each element in the matrix is either valid (number, string, etc.) or missing. Invalid data is treated as missing. From the data matrix we create a binary matrix, $\mathbf{B}_{m\times n}$, where $b_{ij} = 1$ if d_{ij} is valid and 0 if d_{ij} is missing. Let γ, the desired maximum percentage of missing data in each row and column be represented as a decimal (i.e., $\gamma = 0.1$). Then, each row in the clean data matrix should meet the following property:

$$\frac{\#\ of\ missing\ elements\ in\ row}{\#\ of\ elements\ in\ row} \leq \gamma \tag{1}$$

A similar property is met by each column. Here we introduce two vectors $r = [r_1, r_2, \ldots, r_m]$ and $c = [c_1, c_2, \ldots, c_n]$, that are used to indicate which rows and columns from the original data matrix are preserved in the clean data matrix. In particular, $r_i = 0$ if none of the elements in d_{i*} are in the clean data matrix and 1 if at least one element from d_{i*} is preserved in the clean data matrix. The elements of c operate in a similar fashion for columns. Eq. (1) can be written in terms of c and $\mathbf{B}$ for the i-th row as:

$$\frac{\sum_{j=1}^{n} c_j(1 - b_{ij})}{\sum_{j=1}^{n} c_j} \leq \gamma \tag{2}$$

The denominator counts the number of columns from the original data matrix preserved in the clean data matrix. The numerator counts the number of missing elements in row i that are in preserved columns. If missing elements are removed, they do not count toward the amount of missing data in the clean data matrix. The corresponding equation for the j-th column is:

$$\frac{\sum_{i=1}^{m} r_i(1 - b_{ij})}{\sum_{i=1}^{m} r_i} \leq \gamma \tag{3}$$

With the reformulated properties of Eq. (2) and (3) we now turn our attention to selecting which rows and columns to preserve.

B. Greedy Algorithm

The simplest approach to cleaning a matrix is to remove a row or column if it contains more missing data than allowed. This can be done all at once, or in an iterative fashion, with the row or column with the most missing data removed at each step. We will consider removing all of the rows and columns with too much missing data at once as the *naive* method. However, if our goal is to maximize the number of valid elements kept, this approach can easily fail. Consider the problem when $\gamma = 0.1$ and the matrix $\mathbf{B}_{11\times15}$ is all 1's except for two 0's in columns 1 and 2 of row 1. The only row/column with too much missing data is row 1. If row 1 is removed, the clean matrix will contain 10*15 = 150 valid elements. However, if column 1 is removed, the clean matrix has 10*14 + 13 = 153 valid elements. Removing column 2 results in a similar solution. The sub-optimal solution occurs because the rows contain more valid elements than the columns and row 1 has more missing data than is allowed.

Algorithm 1 Greedy Algorithm

Require: $\mathbf{B}, \gamma$

$r \leftarrow 1_m, c \leftarrow 1_n$ // Initialize all rows / columns to kept
while true **do**
  $\alpha \leftarrow$ calculateAlpha($\mathbf{B}, c$)
  $\beta \leftarrow$ calculateBeta($\mathbf{B}, r$)
  maxMiss $\leftarrow$ largest % missing in preserved row/col
  if maxMiss $\leq \gamma$ **then**
    break
  end if
  if maxMiss is from rows **then**
    $i \leftarrow$ row with maxMiss
    $k \leftarrow$ Calculate number of columns to remove
    sumBeta $\leftarrow$ Number of valid elements in k columns
    if $\alpha_i <$ sumBeta **then**
      Remove row i
    else
      Remove k columns
    end if
  else
    $j \leftarrow$ column with maxMiss
    $k \leftarrow$ Calculate number of rows to remove
    sumAlpha $\leftarrow$ Number of valid elements in k rows
    if $\beta_j <$ sumAlpha **then**
      Remove column j
    else
      Remove k rows
    end if
  end if
end while
return r, c

Based on the example above, the goal of our algorithm should be to find the elements contributing to invalid rows or columns, and to remove those elements along with the fewest number of valid elements. Our greedy algorithm iteratively removes rows and/or columns until the percentage of missing data in each remaining row and column is no more than γ. At each iteration, the percentage of missing data in each preserved row and column is calculated. The row/column with the largest percentage of missing data is selected. If a row is selected, the algorithm calculates k, the number of columns with missing data in the selected row that need to be removed so that the row has no more than γ percent missing data. The k columns are selected so that the smallest amount of valid data would be removed. If the number of valid elements in the row is less than the sum of valid elements in the selected k columns, the row is removed. Otherwise, the k columns are removed. If a column has the largest percentage of missing data, the above process is repeated, but the rows and columns are swapped. After removing the row(s) or column(s), the number of valid elements in each preserved row and column are recalculated. Pseudo code for our algorithm is shown in Algorithm 1. We let α_i and β_j indicate the number of valid elements in the i-th row and j-th column, respectively.

C. Maximize Retained Rows and Columns (RowCol)

The *DataRetainer*, *Auto-miss*, and our *Greedy* algorithm clean the matrix iteratively. However, it is possible to clean the matrix with exact methods as well. Intuitively, any cleaning algorithm will want to keep rows and columns with a large number of valid elements. Using the original number of valid elements in each row and column, we can clean a matrix by maximizing the weighted sum of the retained rows and columns. Let $\alpha_i = \sum_{j=1}^{n} b_{ij}$ be the original number of valid elements in row i and let $\beta_j = \sum_{i=1}^{m} b_{ij}$ be the original number of valid elements in column j. Then the weighted sum of the retained rows and columns is:

$$\sum_{i=1}^{m} \alpha_i r_i + \sum_{j=1}^{n} \beta_j c_j \tag{4}$$

In order to optimize Eq. (4) using an ILP, Eq. (2) needs to be reformulated to create constraints on retained rows. We begin by rearranging the inequality, in terms of c_j.

$$0 \leq \sum_{j=1}^{n} (\gamma + b_{ij} - 1)\, c_j$$

When the *rhs* of the inequality is non-negative, the row contains an acceptable amount of missing data. The *rhs* ranges in value between $[-n, n]$. By dividing both sides by n, we change the range to $[-1, 1]$, and the conditions under which the inequality is met remain the same. Using the reformulated inequality, a constraint can be created to delete row i when the *rhs* is negative.

$$r_i \leq 1 + \frac{1}{n} \sum_{j=1}^{n} (\gamma + b_{ij} - 1)\, c_j$$

By scaling the sum on the *rhs*, this constraint forces r_i to never be less than 0. A similar constraint can be derived for each column. These two constraint sets can be combined with the weighted sum objective function and integer constraints for

the decision variables to create the *RowCol* IP as described below:

$$\underset{r,\,c}{\text{maximize}} \quad \sum_{i=1}^{m} \alpha_i r_i + \sum_{j=1}^{n} \beta_j c_j \tag{5a}$$

$$\text{subject to} \quad r_i \leq 1 + \frac{1}{n} \sum_{j=1}^{n} (\gamma + b_{ij} - 1)\, c_j \quad \forall\, i, \tag{5b}$$

$$c_j \leq 1 + \frac{1}{m} \sum_{i=1}^{m} (\gamma + b_{ij} - 1)\, r_i \quad \forall\, j, \tag{5c}$$

$$r_i, c_j \,\epsilon\, \{0,1\}\ \forall\, i, j \tag{5d}$$

After determining the rows and columns in the optimal solution, we clean the matrix by keeping elements for which the corresponding row and column are kept.

D. Maximize Retained Elements (Elements)

The *RowCol* IP objective function is based on the number of valid elements in each row and column in the original data matrix. However, as rows are removed, the number of valid or missing elements in each column decreases. Likewise, as columns are removed, the number of valid or missing elements in each row decreases. To account for this, we replace the α_i's and β_j's values from Eq. (5a) with the number of valid elements based on the current decision variable values.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij} c_j r_i + \sum_{j=1}^{n}\sum_{i=1}^{m} b_{ij} r_i c_j = 2 \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij} r_i c_j \tag{6}$$

After removing the constant, the new objective function counts the number of valid elements in the cleaned matrix. Eq. (6) is a quadratic equation. By introducing a new variable and constraint for each element, we convert the equation into a linear one and create the *Element* IP.

$$\underset{x}{\text{maximize}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij} x_{ij} \tag{7a}$$

$$\text{subject to} \quad x_{ij} \leq \frac{1}{2}\,(r_i + c_j) \quad \forall i, j, \tag{7b}$$

$$r_i \leq 1 + \frac{1}{n} \sum_{j=1}^{n} (\gamma + b_{ij} - 1)\, c_j \quad \forall\, i, \tag{7c}$$

$$c_j \leq 1 + \frac{1}{m} \sum_{i=1}^{m} (\gamma + b_{ij} - 1)\, r_i \quad \forall\, j, \tag{7d}$$

$$r_i, c_j, x_{ij} \,\epsilon\, \{0,1\}\ \forall\, i, j \tag{7e}$$

This algorithm is guaranteed to maximize the number of valid elements in the cleaned matrix, while satisfying the allotted percentage of missing data in each row and column.

E. Configuring Other Programs

We compared Mr. Clean to several other available programs. *List-wise* deletion removes any individual (row) with missing data and requires no configuration. Similarly, *feature-wise* removes any feature (column) with missing data and requires no configuration. The *naive* method removes all rows and columns with a percent of missing data greater than γ. *Auto-miss* was configured to remove all missing data by setting MAX_PERCENT_MISSING=0 in the configuration file. Other values could be used; however *Auto-miss* applies the percent missing to the entire data matrix and not to each row and column separately. *DataReatiner* requires manual interaction to determine how many rows and columns to remove. At each iteration, *DataRetainer* creates a table indicating the number of rows and columns that would be retained at various levels of missingness. Afterwards, the user inputs the maximum percent of missing data allowed for both rows and columns, separately. This table, along with the following logic, was used to determine the order in which rows / columns were removed.

DataRetainer Procedure

1) Determine the desired percent missing.
2) Starting from the bottom of table, find the percentage missing where the number of rows first changes. Likewise, find the percentage missing where the number of columns first changes.
3) If both rows and columns change at a percentage less than our desired amount, then we exit *DataRetainer*.
4) If the rows and columns change at different percentages, remove the rows/columns corresponding to the larger percentage.
5) If the rows and columns change at the same percentage, remove the one that corresponds to less elements. The number of elements removed for the rows is calculated by multiplying the difference in the number of rows by the number of columns.
6) Repeat steps 2-4 until the desired level of missingness is reached.

III. Results

We evaluated the performance of Mr. Clean using 17 real-world data sets. Continuous values were cleaned from the kamyr-digester data set [20], along with gene expression values from AD_CSF [21], GSE54456 [22], GSE215307 [23], and single cell RNA-seq values from GSE146026 [24] and GSE146264 [25]. Discrete data was obtained from chromosome Y of all 11 population groups in the HapMap3 project [26]. Details on the original data sets are provided in Table I, including the number of rows, columns, and valid elements. We cleaned each data set for γ = 5% and 0%, as defined in the Section II. Since *list-wise* and *feature-wise* deletion removes rows/columns with any missing data, they were only executed for the $\gamma = 0\%$ scenarios. Due to run-time or memory limitations, the ILPs were not run to completion, or at all, for some scenarios.

Table II shows the percentage of valid elements retained by each algorithm, compared to the best solution, along with the maximum number of valid elements retained for each data set, for the $\gamma = 0\%$ scenarios. *Element* IP was not able to run on the four largest data sets, and was executed with a 1% MIP gap on the AD_CSF data set. When *Element* IP was able to converge, it preserved the largest number of elements, as expected. The *DataRetainer*, *Greedy*, and *RowCol*

TABLE I
SUMMARY OF DATA SETS BEFORE CLEANING

Data Set	# Rows	# Cols	# Valid Elements	Percent Missing				
				Total	Min in Row	Max in Row	Min in Col	Max in Col
kamyr-digester	301	22	6,270	5.3%	0.0%	40.9%	0.0%	46.8%
MEX	605	86	51,604	0.7%	0.0%	4.2%	0.0%	12.9%
ASW	616	87	53,338	0.4%	0.0%	4.1%	0.0%	2.4%
TSI	556	102	55,376	2.1%	0.0%	4.5%	0.0%	51.1%
GIH	606	101	60,775	0.6%	0.0%	4.5%	0.0%	5.1%
CHD	576	109	62,283	0.7%	0.0%	4.2%	0.0%	8.0%
LWK	620	110	67,422	1.0%	0.0%	4.2%	0.0%	55.6%
JPT	926	116	101,268	29.6%	2.4%	77.8%	0.0%	94.5%
MKK	585	184	106,569	0.9%	0.0%	4.6%	0.0%	48.9%
CHB	930	139	123,140	30.2%	1.3%	81.2%	0.0%	94.9%
CEU	943	174	160,132	29.5%	4.9%	73.9%	0.0%	94.8%
YRI	922	209	186,895	28.8%	2.7%	76.7%	0.0%	94.8%
AD_CSF	489	705	337,079	2.2%	0.0%	13.8%	0.0%	12.9%
GSE54456	174	21099	3,213,955	12.5%	8.6%	23.3%	0.0%	99.4%
GSE215307	375	57905	16,184,636	25.5%	0.0%	46.7%	0.0%	50.1%
GSE146026	11548	9609	17,362,239	84.4%	0.0%	99.9%	39.3%	95.9%
GSE146264	7303	58129	16,757,950	96.1%	37.3%	100.0%	1.0%	100.0%

TABLE II
PERCENTAGE OF MAXIMUM NUMBER OF ELEMENTS FOR $\gamma = 0$

Data Set	List-wise	Feature-wise	Auto-miss	DataRetainer	Greedy	RowCol IP	Element IP	Max # Elements
Kamyr-digester	53.60	5.60	88.38	100.00	100.00	100.00	100.00	5,377
MEX	82.84	15.61	52.39	100.00	100.00	98.99	100.00	34,880
ASW	99.22	24.38	69.23	99.81	99.89	99.22	100.00	40,421
TSI	32.97	25.277	78.25	99.77	99.77	100.00	100.00	39,600
GIH	92.41	15.72	71.18	99.57	99.57	99.80	100.00	42,406
CHD	89.60	20.23	73.63	99.73	100.00	100.00	100.00	42,700
LWK	40.42	10.91	71.34	99.33	100.00	99.87	100.00	45,450
JPT	0.00	2.04	14.14	100.00	99.67	98.80	100.00	45,492
MKK	47.25	23.18	84.30	98.19	99.17	98.94	100.00	63,081
CHB	0.00	0.00	5.15	99.76	100.00	99.85	100.00	53,550
CEU	0.00	1.54	16.56	99.14	99.74	99.48	100.00	61,056
YRI	0.00	3.69	28.43	99.01	99.46	99.12	100.00	74,958
AD_CSF	6.33	10.43	93.22	86.07	100.00	92.81	100.00	89,100
GSE54456	0.00	98.47	0.74	100.00	99.92	100.00	NA	2,599,371
GSE215307	1.22	99.38	1.22	99.47	100.00	100.00	NA	4,735,444
GSE146026	0.00	0.00	2.63	66.30	100.00	NA	NA	365,292
GSE146264	0.00	0.00	24.34	100.00	92.60	NA	NA	138,645

TABLE III
PERCENTAGE OF MAXIMUM RUN TIME FOR $\gamma = 0$

Data Set	List-wise	Feature-wise	Auto-miss	DataRetainer	Greedy	RowCol IP	Element IP	Max Time (s)
Kamyr-digester	0.01	0.01	0.90	2.86	0.05	2.67	100.00	0.35
MEX	0.01	0.01	0.33	4.87	0.08	0.29	100.00	2.88
ASW	0.02	0.02	0.77	10.15	0.15	0.48	100.00	1.18
TSI	<0.01	<0.01	0.08	2.28	0.02	0.09	100.00	12.28
GIH	0.01	0.01	0.28	4.69	0.07	0.23	100.00	4.05
CHD	0.01	0.01	0.25	4.90	0.05	0.24	100.00	4.49
LWK	<0.01	0.01	0.08	1.34	0.03	0.14	100.00	14.97
JPT	<0.01	<0.01	<0.01	0.10	<0.01	0.09	100.00	345.72
MKK	<0.01	<0.01	0.02	0.23	<0.01	0.02	100.00	156.12
CHB	<0.01	<0.01	<0.01	0.03	<0.01	0.03	100.00	1273.23
CEU	<0.01	<0.01	<0.01	0.02	<0.01	0.02	100.00	2388.57
YRI	<0.01	<0.01	<0.01	0.02	<0.01	0.02	100.00	3297.15
AD_CSF	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	100.00	*86400
GSE54456	<0.01	0.12	1.02	10.98	3.35	100.00	NA	35.25
GSE215307	<0.01	0.32	3.72	100.00	62.90	62.90	NA	22.36
GSE146026	<0.01	<0.01	0.42	100.00	0.71	NA	NA	1982.12
GSE146264	<0.01	<0.01	1.50	100.00	3.65	NA	NA	2056.93

algorithms were generally within 2% of *Element*, with the exception of the AD_CSF data set. *Greedy* kept significantly more elements than *DataRetainer* for the GSE146026 data set, while performing slightly worse than *DataRetainer* on the GSE146264 data set. The *list-wise*, *feature-wise*, and *Auto-miss* retained the smallest amounts of data, with *list-wise* and *feature-wise* removing all elements in 7 and 3 scenarios, respectively. The relative run times between the algorithms for the $\gamma = 0\%$ scenarios are listed in Table III. The *Element* IP required several orders of magnitude more run time than the other algorithms, even timing out at 2 days for the AD_CSF data set (noted as *86400). The *Greedy* algorithm executed faster than the *DataRetainer* on all data sets, generally by a few orders of magnitude. The *list-wise* and *feature-wise* algorithms ran the fastest, while *auto-miss* performed slower than *Greedy* for small data sets, but faster for large data sets.

The difference in the number of elements kept between *DataRetainer*, *Greedy*, *RowCol*, and *Element* was even smaller for the $\gamma = 5\%$ scenarios, as seen in Table IV. Similar to the $\gamma = 0\%$ scenarios, the *Element* IP was not run on the four largest data sets due to memory limitations and the *RowCol* IP was not run on the two largest data sets as well. For the data sets cleaned with *Element* IP, *Greedy* and *RowCol* kept at least 99% of the valid elements kept by *Element*. *DataRetainer* performed fairly well, staying within 89% of the maximum number of valid elements retained. The *naive* approach varied greatly. In some cases it obtained an optimal solution, while in others it removed all of the valid elements. The maximum run time for the $\gamma = 5\%$ scenarios are shown in Table V. The *Element* IP required 15 to 500 times as long as the *RowCol* IP to clean the data sets, and the *RowCol* IP took at least 38 times as long as the *Greedy* algorithm, with *DataRetainer* falling in between *RowCol* and *Greedy*. The *Element* IP timed out at 2 days on the AD_CSF data set, while the *RowCol* IP timed out on the GSE215307 data set (indicated with *86400).

IV. Discussion

In this manuscript, we compared three new data cleaning algorithms to the existing *list-wise*, *feature-wise*, and *naive* deletion techniques, along with the *DataRetainer* and *Auto-miss* algorithms. *List-wise* was the fastest, but was only used for the $\gamma = 0\%$ scenarios. It, along with *feature-wise* deletion, removed substantially more valid elements than all other approaches. *Auto-miss* was only used for $\gamma = 0\%$ scenarios as well. In general, *Auto-miss* retained less data than *DataRetainer* and the Mr. Clean algorithms. Based on our results we recommend not using *list-wise*, *feature-wise* or *Auto-miss* unless your data is comprised of an adequate number of rows or columns without any missing data. For example, a large survey in which most of the participants complete all questions, might be a good candidate for list-wise deletion. Further investigation is required to analyze *Auto-miss* performance if a global percent missing greater than 0 is desired.

The *DataRetainer* algorithm performed well in all scenarios, except the $\gamma = 0\%$ on the AD_CSF and GSE146026 data sets. Since this program requires the user to interactively decide how to clean the data, the results may be sub-optimal due to user error. Additionally, *DataRetainer* only outperformed both *Greedy* and *RowCol* in a few scenarios. Therefore, using *Greedy* and *RowCol* in tandem might remove the user burden of running *DataRetainer*.

In our trials, the *Greedy* algorithm struck the best balance between speed and the number of valid elements kept. None of the data sets proved problematic in terms of size. In fact, *Greedy* can be executed on larger data sets than what we considered here. Our team recommends running *Greedy* for every data set you wish to clean, either as the sole method or as a comparison with other methods. The *Greedy* algorithm can also provide a starting solution for both ILPs.

The *RowCol* ILP was the faster ILP and slightly outperformed *DataRetainer* in most scenarios, in terms of valid elements kept. Its performance against *Greedy* was not as clear cut. In some scenarios it kept more elements than *Greedy*, while in others it kept less. However, it required substantially more run time. This algorithm is optimal based on the original number of valid elements in each row and column. When deletions change this number, the results can be sub-optimal. *RowCol* is best suited for small and medium size data sets, or ones where the optimal answer is achieved by removing mostly rows or mostly columns.

The *Element* ILP will always find the optimal answer when it runs to completion. However, it requires significantly more memory and run time than the other algorithms. As a result, we were unable to run it on larger data sets. It is best suited for small data sets or when retention of data is of high importance.

A. Limitations

In this paper, we explored the number of valid elements kept from an original data matrix by various algorithms. We have not analyzed the impact to test power or imputation accuracy from any deletion technique. Nor have we recommended a maximum amount of missing data to allow. Both considerations are specific to the given application. The *DataRetainer* program that we tested requires user interaction. While we followed a predefined cleaning process, we acknowledge that a different process may achieve more retained elements for the same level of missingness in the final dataset. Finally, we were not able to prove the optimal number of elements kept for all scenarios. The *Element* ILP will find this solution, but due to its time and space complexity, the program did not run to completion within our given time and memory limits for the medium to large data sets in our trials.

B. Future Work

The *Greedy* algorithm decides on the next 'best' move as described in Section II. While we tested several definitions of 'best,' further exploration may result in an improvement in the number of valid elements kept. Another opportunity for future work deals with the two ILPs. Additional constraints or more restrictive constraints may exist that reduce run-time. Other opportunities are the development of distributed versions for

TABLE IV
PERCENTAGE OF MAXIMUM NUMBER OF ELEMENTS FOR $\gamma = 0.05$

Data Set	Naive	DataRetainer	Greedy	RowCol IP	Element IP	Max elements
Kamyr-digester	51.57	97.30	100.00	100.00	100.00	5,526
MEX	100.00	100.00	100.00	100.00	100.00	51,077
ASW	100.00	100.00	100.00	100.00	100.00	53,338
TSI	99.18	99.18	99.99	100.00	100.00	52,714
GIH	99.21	99.21	99.99	100.00	100.00	60,679
CHD	100.00	100.00	100.00	100.00	100.00	60,683
LWK	100.00	99.49	100.00	100.00	100.00	67,147
JPT	17.23	99.47	99.99	100.00	100.00	61,919
MKK	100.00	100.00	100.00	100.00	100.00	105,722
CHB	15.60	99.80	99.99	100.00	100.00	76,899
CEU	0.00	100.00	100.00	100.00	100.00	93,196
YRI	21.03	100.00	100.00	100.00	100.00	112,299
AD_CSF	92.80	98.73	99.00	99.99	100.00	292,861
GSE54456	0.00	99.77	99.80	100.00	NA	2,854,035
GSE215307	0.27	99.82	100.00	100.00	NA	6,986,514
GSE146026	0.00	89.02	100.00	NA	NA	944,121
GSE146264	0.00	94.12	100.00	NA	NA	529,981

TABLE V
PERCENTAGE OF MAXIMUM RUN TIME FOR $\gamma = 0.05$

Data Set	Naive	DataRetainer	Greedy	RowCol IP	Element IP	Max Time (s)
Kamyr-digester	0.08	4.28	0.07	6.14	100.00	0.23
MEX	0.40	2.87	0.08	5.79	100.00	1.39
ASW	0.44	3.06	0.16	6.58	100.00	0.65
TSI	0.08	2.82	0.03	2.57	100.00	5.32
GIH	0.33	2.45	0.14	5.36	100.00	1.63
CHD	0.12	2.54	0.05	4.41	100.00	2.36
LWK	0.08	0.36	0.03	3.23	100.00	5.54
JPT	0.01	0.20	0.01	0.55	100.00	109.64
MKK	0.07	1.03	0.02	1.50	100.00	9.70
CHB	0.01	0.33	0.01	1.20	100.00	75.05
CEU	<0.01	0.10	<0.01	0.21	100.00	346.77
YRI	<0.01	0.10	<0.01	0.20	100.00	451.86
AD_CSF	<0.01	<0.01	<0.01	0.40	100.00	*86400
GSE54456	0.36	8.73	2.55	100.00	NA	34.14
GSE215307	<0.01	0.03	0.02	100.00	NA	*86400
GSE146026	0.09	100.00	0.77	NA	NA	1772.80
GSE146264	0.35	100.00	4.32	NA	NA	1826.85

both *RowCol* and *Element* which allow different paths to be searched in parallel. Such an improvement should improve run-time, space complexity, and facilitate the execution of larger data sets. Finally, studying the impact of down stream analysis based on these deletion algorithms is warranted.

C. Source Code

The source code for all algorithms is publicly available on GitHub. Greedy: https://github.com/ClimerLab/MrClean-Greedy. RowCol and Element IP: https://github.com/ClimerLab/MrClean.

D. Computation Environment

The experiments in this manuscript were conducted on a Linux Mint 21.1 machine, with an Intel i5-4690K CPU @ 3.50Ghz processor and 16 GB of memory.

REFERENCES

[1] Margot Peeters et al. "How to handle missing data: A comparison of different approaches". In: *European Journal of Developmental Psychology* 12 (4 July 2015), pp. 377–394. ISSN: 17405610. DOI: 10.1080/17405629.2015.1049526.

[2] Akanksha Farswan et al. "Imputation of Gene Expression Data in Blood Cancer and Its Significance in Inferring Biological Pathways". In: *Frontiers in Oncology* 9 (Jan. 2020). ISSN: 2234943X. DOI: 10.3389/fonc.2019.01442.

[3] Aditya Dubey and Akhtar Rasool. "Efficient technique of microarray missing data imputation using clustering and weighted nearest neighbour". In: *Scientific Reports* 11 (1 Dec. 2021). ISSN: 20452322. DOI: 10.1038/s41598-021-03438-x.

[4] Marcilio C.P.D. Souto, Pablo A. Jaskowiak, and Ivan G. Costa. "Impact of missing data imputation methods on gene expression clustering and classification". In: *BMC*

Bioinformatics 16 (1 Feb. 2015). ISSN: 14712105. DOI: 10.1186/s12859-015-0494-3.

[5] Ben Omega Petrazzini et al. "Evaluation of different approaches for missing data imputation on features associated to genomic data". In: *BioData Mining* 14 (1 Dec. 2021). ISSN: 17560381. DOI: 10.1186/s13040-021-00274-7.

[6] Paul Madley-Dowd et al. "The proportion of missing data should not be used to guide decisions on multiple imputation". In: *Journal of Clinical Epidemiology* 110 (June 2019), pp. 63–73. ISSN: 18785921. DOI: 10.1016/j.jclinepi.2019.02.016.

[7] Donald B. Rubin. "INFERENCE AND MISSING DATA". In: *ETS Research Bulletin Series* 1975 (1 June 1975), pp. i–19. ISSN: 2333-8504. DOI: 10.1002/J.2333-8504.1975.TB01053.X. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/j.2333-8504.1975.tb01053.x.

[8] Daniel A. Newman. "Missing Data: Five Practical Guidelines". In: *Organizational Research Methods* 17 (4 Oct. 2014), pp. 372–411. ISSN: 15527425. DOI: 10.1177/1094428114548590.

[9] Andrew Briggs et al. "Missing.... presumed at random: cost-analysis of incomplete data". In: *Health Economics* 12 (5 2003), pp. 377–392. DOI: 10.1002/hec.766. URL: https://onlinelibrary.wiley.com/doi/10.1002/hec.766.

[10] Joost R. van Ginkel et al. "Rebutting Existing Misconceptions About Multiple Imputation as a Method for Handling Missing Data". In: *Journal of Personality Assessment* 102 (3 May 2020), pp. 297–308. ISSN: 00223891. DOI: 10.1080/00223891.2018.1530680.

[11] Sharlee Climer, Alan R. Templeton, and Weixiong Zhang. "Allele-Specific Network Reveals Combinatorial Interaction That Transcends Small Effects in Psoriasis GWAS". In: *PLoS Computational Biology* 10 (9 2014). ISSN: 15537358. DOI: 10.1371/journal.pcbi.1003766.

[12] Sharlee Climer. "Connecting the dots: The boons and banes of network modeling". In: *Patterns* 2 (12 Dec. 2021). ISSN: 26663899. DOI: 10.1016/j.patter.2021.100374.

[13] Michael Chan. *auto-miss*. 2019. URL: https://github.com/ClimerLab/auto-miss.

[14] Sharlee Climer et al. "Synchronized genetic activities in Alzheimer's brains revealed by heterogeneity-capturing network analysis". In: (2020). DOI: 10.1101/2020.01.28.923730.

[15] Sharlee Climer, Alan R. Templeton, and Weixiong Zhang. "Human gephyrin is encompassed within giant functional noncoding yin-yang sequences". In: *Nature Communications* 6 (2015), pp. 1–11. ISSN: 20411723. DOI: 10.1038/ncomms7534. URL: http://dx.doi.org/10.1038/ncomms7534.

[16] Carol M Musil Camille B Warner Piyanee Klainin Yobas Susan L Jones and Susan L Jones. "A Comparison of ImputationTechniques for Handling Missing Data". In: *Western Journal of Nursing Research* 24 (7 2002), pp. 815–829. DOI: 10.1177/019394502237390.

[17] Peter C. Austin et al. *Missing Data in Clinical Research: A Tutorial on Multiple Imputation*. Sept. 2021. DOI: 10.1016/j.cjca.2020.11.010.

[18] Adrienne D Woods et al. "Missing Data and Multiple Imputation Decision Tree". In: *PsyArXiv* (2021). DOI: 10.31234/osf.io/mdw5r. URL: https://doi.org/10.31234/osf.io/mdw5r.

[19] Jungyeon Choi, Olaf M. Dekkers, and Saskia le Cessie. "A comparison of different methods to handle missing data in the context of propensity score analysis". In: *European Journal of Epidemiology* 34 (1 Jan. 2019), pp. 23–36. ISSN: 15737284. DOI: 10.1007/s10654-018-0447-z.

[20] Kevin Dunn. *Kamyr digester-OpenMV.net Datasets.* Oct. 2011. URL: https://openmv.net/info/kamyr-digester.

[21] Chengran Yang et al. "Genomic atlas of the proteome from brain, CSF and plasma prioritizes proteins implicated in neurological disorders". In: *Nature Neuroscience* 24 (Sept. 2021), pp. 1302–1312. DOI: 10.1038/s41593-021-00886-6. URL: https://doi.org/10.1038/s41593-021-00886-6.

[22] Bingshan Li et al. "Transcriptome analysis of psoriasis in a large case-control sample: RNA-seq provides insights into disease mechanisms". In: *Journal of Investigative Dermatology* 134 (7 2014), pp. 1828–1838. ISSN: 15231747. DOI: 10.1038/jid.2014.28. URL: http://dx.doi.org/10.1038/jid.2014.28.

[23] Peter M Szabo et al. "Cancer-associated fibroblasts are the main contributors to epithelial-to-mesenchymal signatures in the tumor microenvironment". In: *Scientific Reports* 13 (3051 2023). DOI: 10.1038/s41598-023-28480-9. URL: https://doi.org/10.1038/s41598-023-28480-9.

[24] "A single-cell landscape of high-grade serous ovarian cancer". In: *Nature Medicine* 26 (8 Aug. 2020), pp. 1271–1279. ISSN: 1546170X. DOI: 10.1038/s41591-020-0926-0.

[25] Jared Liu et al. "Single-cell RNA sequencing of psoriatic skin identifies pathogenic Tc17 cell subsets and reveals distinctions between CD8+ T cells in autoimmunity and cancer". In: *Journal of Allergy and Clinical Immunology* 147 (6 June 2021), pp. 2370–2380. ISSN: 10976825. DOI: 10.1016/j.jaci.2020.11.028.

[26] David M. Altshuler et al. "Integrating common and rare genetic variation in diverse human populations". In: *Nature* 467 (7311 Sept. 2010), pp. 52–58. ISSN: 14764687. DOI: 10.1038/nature09298.