[[1]](#footnote-1)

Missing Value Imputation Methods for Gene-Sample-Time Microarray Data

Yifeng Li, Alioune Ngom, Luis Rueda

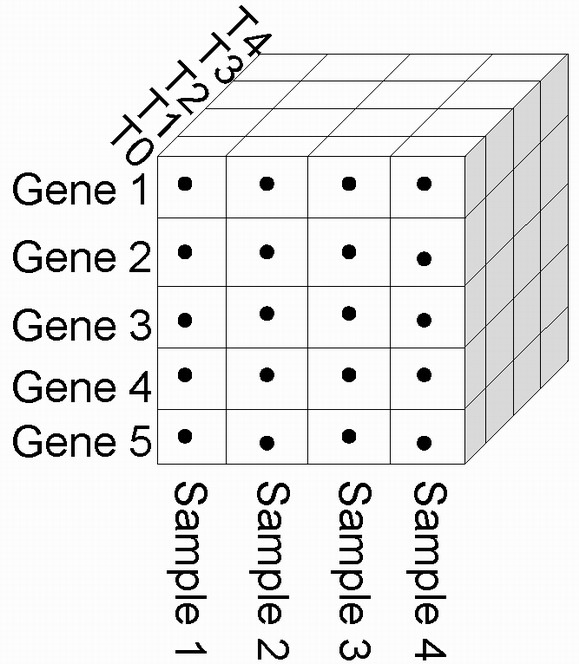
*Abstract*—With the recent advances in microarray technology, the expression levels of genes with respect to samples can be monitored synchronically over a series of time points. Such three-dimensional microarray data, termed *gene-sample-time* microarray data or *GST* data for short, may contain missing values. Current microarray analysis methods require complete data set, and thus, either each row, column or tube containing missing values must be removed from the original GST data, or these missing values must be estimated before analyses. Imputation of missing values is, however, more recommended than removal of data in order to increase the effectiveness of analysis algorithms. In this report, we extend automated imputation methods, devised for two-dimensional microarray data, to GST data. We implemented imputation methods for GST data based on Singular Value Decomposition (3SVDimpute), K-Nearest Neighbor (3KNNimpute), and gene and sample average methods (3Aimpute), and show that methods based on KNN present best results with lowest NMRS error.

# INTRODUCTION

DNA microarray technology can monitor thousands of genes in parallel, dramatically accelerating molecular biology experiments and providing a huge amount of data to find co-regulated genes, functions of genes, genetic networks, and marker genes, for instance. There are two types of microarray data: *gene-sample* data sets, which compile the expression levels of various genes over a set of biological samples; and *gene-time* data sets, which record the expression levels of various genes over a series of time-points. Both types of data are represented by a two-dimensional (2D) gene expression matrix, where genes correspond to rows in the matrix and each matrix entry contains the expression level of a given gene for some sample or at certain time-point.

Within the last few years in medical research, the expression levels of genes with respect to biological samples have been monitored synchronically over a series of time-points [13]. This corresponds to a three-dimensional (3D) data set, termed *gene-sample-time* (*GST*) microarray data [14]; which can be viewed as a collection of gene-sample data over a series of time-points, or a collection of gene-time data across some samples (see Fig.1).

Microarray data often suffer from missing expression level values due to many factors such as, insufficient resolution, image corruption, artifacts, systematic error, or incomplete experiments. Microarray analysis methods based on machine learning approaches (such as clustering or classification) require complete gene expression data sets, in order to perform robustly and effectively. Incomplete data sets must first be pre-processed before analysis in order to deal with their missing values. For 2D microarray data, current strategies for dealing with missing values include: either (1) removing from the original data all rows and columns containing missing values, or (2) imputing all missing values of the original data. Imputation is the process of estimating missing values to obtain a complete data set. In either way, the pre-processing step is meant to construct a *complete* matrix for use in the subsequent analysis stage. The first strategy can result in important loss of data for small data sets, and in particular for data sets with small sample size or with short time-series (this is generally the case for 2D microarray data). The alterative strategy is more reasonable for increasing the reliability, effectiveness and robustness of the subsequent analysis methods, since no information is lost from the original data.



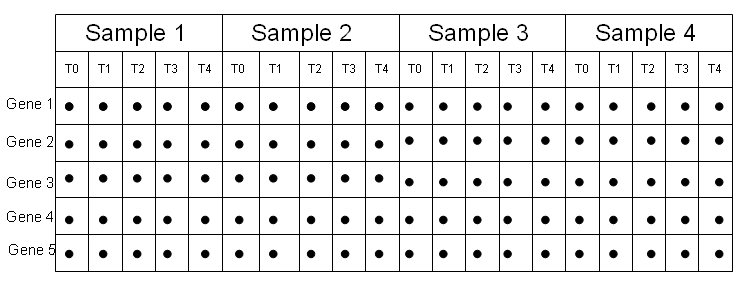


Fig. 1. Two representation of a gene-sample-time microarray data.

GST microarray data sets also suffer from missing values. It is even more important to estimate missing values for such data, because many more tubes (along the time-axis) are deleted for each deleted row or column of the 3D matrix. Current GST data sets have a very small number of samples or time-points, relative to the number of genes. Also, even if there are enough samples or time-points in the data, it does not seem reasonable to remove any missing values along the time-axis. Current approaches for analyzing GST data sets include for instance *gene selection* methods [12] or *tri-clustering* methods [13] for bio-marker prediction, which are both used to search for subsets of genes as candidate bio-markers for specific drug therapy. In [12], the authors limited their analysis only on the baseline time-point *t* = 0 (that is, the frontal slice in Fig.1, corresponding to the gene-sample matrix at *t* = 0), for finding markers genes. This amounted to remove all time-slices for *t* > 0, and analyzing only the time-slice *t* = 0. It should be noted that there are no missing values in their GST data set at time-slice *t* = 0, however, their gene selection approach could have benefited by utilizing the information provided at all time-slices. Tri-clustering, that is simultaneous clustering of genes, samples and time-points (i.e., finding subsets of genes that have similar expression patterns across subsets of samples and over certain time-intervals), is applied on complete pre-processed GST data [13]. Their initial GST data contains missing values, at many time-slices, which have been estimated before performing the tri-clustering algorithm. It is important to note that, in tri- and bi-clustering, the time-intervals must not contain missing values, since they correspond to cellular processes or co-regulated genes to be determined by bi- or tri-clustering. Also, a similarity metric for genes cannot be readily determined when there are missing values within a time-interval.

In [13], the initial incomplete GST data was imputed by applying the KNNimpute algorithm [1] on either each time-slice (gene-sample matrices) or each sample-slice (gene-time matrices). This is done independently of the other slice imputations; that is, imputation at time-slice *t* does not affect (or is not used for) imputation at other time-slices. KNNimpute was originally devised for 2D microarray data, and its efficiency and effectiveness on GST data is not clear and has not been studied. In this paper, popular but simple imputation methods devised for 2D microarray data, such as *average*, *k-nearest neighbor* (KNN), and *singular value decomposition* (SVD) methods, are modified for pre-processing GST microarray data. Furthermore, we investigate the benefit of applying multiple time-series profile alignment as a pre-processing step for the GST imputation stage, if any. To the best of our knowledge, this is the only paper focusing on estimating missing values for incomplete GST microarray data sets.

# Related works

Many imputation methods have been proposed specifically for 2D microarray data sets, since 2001. An imputation method that estimates a missing value g*ij* in a row *i* by the average value in column *j* was investigated in [1], and was shown to give the worst performance among many imputation methods. In [1], a k-nearest neighbor based imputation methods, KNNimpute, have been proposed using Euclidean, Pearson correlation and variance minimization, as similarity metrics, and Euclidean metric gave best results. For each incomplete gene *gi* with missing value *gij*, imputation is done by first finding its *k* nearest complete genes, and then taking the weighted average value at column *j* of those *k* genes as estimation of *gij*. Paper [1] also described a *singular value decomposition* based imputation method, SVDimpute. A set of eigen-genes is found by applying SVD on the complete genes only. Each incomplete gene is then represented as a linear combination of those eigen-genes. Linear regression using expectation-maximization (EM) algorithm is performed on those eigen-genes to estimate the missing values of given incomplete genes. LLSimpute, a *local least square*s method [2], represents an incomplete gene as a linear combination of its *k* nearest complete neighbors. Least square optimization is used to find the coefficients of the linear combination, which are then used for estimating the missing values of the incomplete gene. Other methods based on least squares regression are also introduced in [3,4,5]. The *Bayesian principal component analysis* method, BPCAimpute, applies PCA similarly to the SVD method. However an EM-like Bayesian estimation algorithm is used to estimate the coefficients of the linear combinations for each incomplete gene [8] to impute. In [6], genes are represented as cubic spline functions first, and then missing values for incomplete genes are estimated by resampling the continuous curve. ARLSimpute [7] first, applies auto-regression on a set of *k* similar genes (missing values are set to zero initially) to estimate their AR coefficients by means of a least square method. Then, using the AR coefficients, missing values for all incomplete genes are imputed by means of another least square regression method. ARLSimpute is the only method devised specifically for time-series profiles; however it works only on long stationary time-series data. It is also the only method that is able to impute a time-point (entirely missing column of a gene-time microarray data).

Except the approach in [6], imputation methods for 2D microarray data are all based on similar principles: they either find nearest complete genes to impute on incomplete genes, or find eigen-genes to impute on incomplete genes, or combine these two principles. Current methods, except the BPCA method, are devised for non time-series data, though they have been applied to time-series data. They also do not work when an entire column of a 2D microarray data is missing. In the next section, we describe variants of Average, KNNimpute and SVDimpute methods, for GST microarray data, that can impute missing values even for an empty row, column or tube of the 3D data.

# Methods

In this paper, we modify the Average, KNNimpute and SVDimpute methods to pre-process incomplete GST data. In particular for the time-series in the GST data, we also investigate a pre-processing method that takes into account the temporal relationships within and between the time-series. In the following sections, *m*, *n* and *p* are respectively, the number or genes, samples and time-points in the GST data; *gi*, *sj* and *tk* are the gene, sample and time-point at row *i*, column *j* and tube *k*, respectively; *gij* is the time-series of gene *gi* at sample *sj*, or equivalently, *sij* is the time-series of sample *sj* at gene *gi*; *gijk* is a value at gene *gi*, sample *sj* and time-point *tk*; *γ* is the estimate of a missing value. A time-series is incomplete if it has a missing value. In the following, we assume that the gene time-series *gij* (i.e., sample time-series *sij*) has a missing value *gijk* at some time-point *tk*. In the data set we used, each gene *gi* or sample *sj* contains at least one complete time-series.

## Average Methods

We implemented three average imputation methods, GAimpute (gene-average), SAimpute (sample-average) and GSAimpute (gene-sample-average). In GAimpute, the estimate *γ* of *gijk* is the average value at time-point *tk* of the *m* gene time-series of sample *sj*. In SAimpute, *γ* is the average of the *n* sample time-series of gene *gi* at time-point *tk*. In GSAimpute, estimate *γ* of *gijk* is the average of the estimates obtained by GAimpute and SAimpute. These three methods are generically termed 3Aimpute, in subsequent sections.

## k-Nearest Neighbors Methods

Similarly to the averaging methods above, we devised three KNNimpute methods on incomplete gene or sample time-series. For each incomplete time-series *gij* or *sij*, *c* nearest neighbors are computed only from complete time-series. The estimate *γ* of *gijk* is then taken as the weighted average at time-point *tk* of these nearest neighbors.

In GKNNimpute, the *c* neighbors of *gij* are obtained from all the complete gene time-series of sample *sj*. In SKNNimpute, the neighbors are found from all the complete sample time-series of gene *gi*. When the samples have class-labels, then the *c* complete sample time-series must be of the same class as the incomplete time-series *sij*. In GSKNNimpute, *γ* is the average of the estimates from GKNNimpute and SKNNimpute. We denote these methods by the generic term 3KNNimpute.

For missing value estimation, Euclidean distance performed better than other metrics, in [1,7]; hence we have used this metric to find the nearest neighbours. We have also used the new integral-distance metric of [9,10] which is was demonstrated to be more robust than the Pearson correlation distance on time-series data [9]. When using Euclidean distance, we will add letter ‘E’ after KNN, otherwise we add letter A.

## k-Nearest Neighbor with Multiple Time-Series Alignment

The 3KNNAimpute methods based on the integral-distance function [9,10] are quite slow due to computing the pair-wise alignment between two time-series, for computing their distance. Pair-wise alignment was introduced in [9,10] in order to take into account the temporal relationships between time-series and within times, and thus to perform a more robust analysis of time-series data. Multiple time-series alignment (MA) was introduced in [10] as a pre-processing stage for clustering gene time-series, and was shown to yield much faster clustering time. MA is a series of transformations performed on all time-series profiles, at once, such that the area between any two transformed profiles is minimal, and thus the subsequent clustering method needs not to apply pair-wise alignments for computing the integral-distance between time-series profiles.

For imputation with MA, we first apply GKNNimpute, with Euclidean distance, on the original data *M* in order to obtain a complete data set *C*; this, because MA requires a complete data set. We then perform the MA transformations of [10] on *C* to yield new data set *A*. For each incomplete time-series *gij* in *M* with missing value *gijk* at *tk*, we find the *c* nearest neighbors of its counterpart in *A* using integral-distance (no alignment is necessary here). We then estimate *gijk* in *A* as the weighted average of these neighbors at time-point *tk*. This is essentially GKNNimpute applied on *A*, except that the nearest neighbors in *A* must be complete in *M*. After imputing *gijk* in *A*, we apply the inverse transformation MA-1 [10] to obtain the estimation of *gijk* in *M*. SKNNimpute and GSKNNimpute can also be applied in similar way. We use the generic term 3MAimpute for methods based on MA.

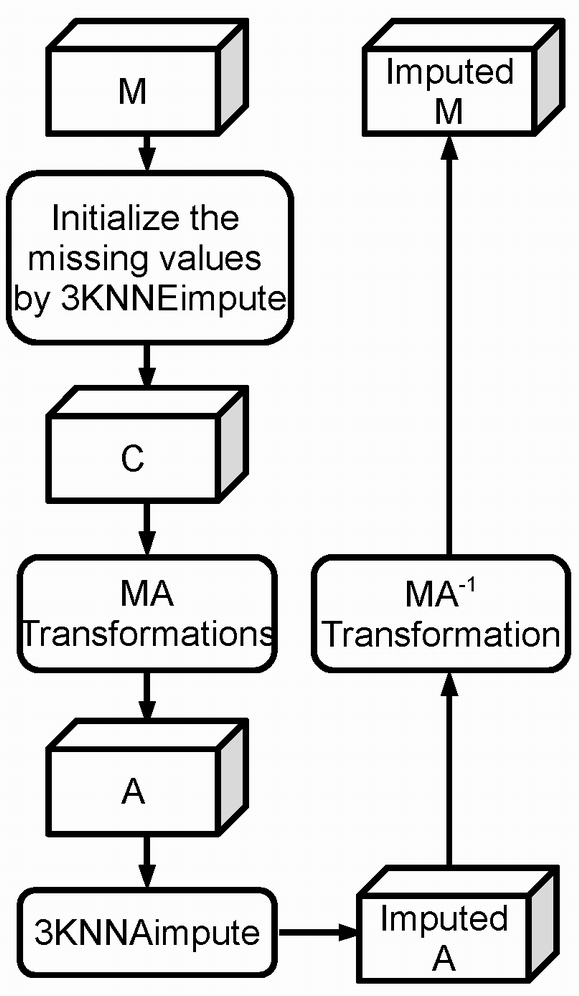


Fig. 2. KNN Imputation with Multiple-Alignment.

## Singular Value Decomposition

We modified the SVDimpute method of [11] for GST data. First, SVD is performed on all complete time-series of gene *gi* or sample *sj* to find a set of *k* eigen-samples or eigen-genes. Each incomplete gene or sample time-series is then a linear combination of these eigenvectors. We apply linear least squares regression to obtain the coefficients of these eigenvectors for each incomplete time-series. Missing values are then imputed by linear combination of the corresponding values of eigenvectors. These methods are generically called 3SVDimpute, including: GSVDimpute for imputing on sample *sj* at time-point *tk*, SSVDimpute for imputing on gene *gi* at time-point *tk*, and the combined GSSVDimpute.

## Time-Point Estimation for GST Data

In some GST data set, a sample *sj* may have a missing value at time-point *tk* for all its gene time-series *gij*. Thus the entire tube *tk* is missing for that sample. Most existing imputation algorithms on 2D microarray data, such as Average, KNNimpute and SVDimpute, data are not able to deal with this situation. One exception is ARLSimpute [7].

Assume sample *sj*, has missing values *g*1*jk*, *g*2*jk*, …, *gmjk* at time-point *tk* (in all its gene time-series *gij*). Our G\*impute methods cannot estimate these values. However, our S\*impute methods has no such limitation: since an incomplete gene time-series *gij* of *sj* will be imputed from the other complete sample time-series *sij* of *gi* (since *gi* contains at least one complete sample time-series).

## Evaluation of Imputation Methods

Let *G* = [*gijk*] be a complete GST data set and *H* = [*hijk*] be an incomplete data set obtained from *G* by removing some values. Let *Γ* = [*γijk*] be an imputation of *H*: that is, obtained after applying an imputation algorithm on *H*. We can assess the performance of an imputation method by measuring the error *E*(*G*, *Γ*) between *G* and *Γ*. We use the normalized root mean squared (NRMS) error defined as:

 (1)

When *G* is known, then the missing value imputation problem is a supervised learning problem. Most microarray data sets are initially incomplete; thus *G* = *H*, and therefore, the missing value imputation problem becomes an unsupervised learning problem to find a true completion of *G*.

In the next section, we test our algorithms only on complete data sets *G*. That is given an initial but incomplete GST data set *M*, we first remove from *M* every gene, sample and time-point containing missing values; thus, producing a complete sub-structure *G* of *M*. We then generate a new data set *H* from *G* by randomly deleting some values from *G* (*G* and *H* have the same dimensions). Next, we perform an imputation on *H* to obtain a complete *Γ* and compute the error *E*(*G*, *Γ*). We repeat this process again 100 times: randomly altering *G* then computing *Γ*. In the experiments below, we report the average error over 100 iterations. See Fig.3 for the experimental procedure.

# Experiments and Discussions

In this study, we used the interferon-β (INFβ) data set of [12]. This GST data contains 70 genes, 52 samples (from 31 good responders and 21 bad responders), and 7 time-points (at 0, 3, 6, 9, 12, 18, 24 months). The samples are taken from patients suffering with relapsing-remitting multiple sclerosis and treated with INFβ as initial therapy. Their blood samples are obtained by veni-puncture every 3 months, to produce the microarray data. After two years, the patients are classified as either good or bad responders according to strict experimental criteria.

We performed our experiments on this data set; using the experimental procedure described in the previous section (Fig.3), and evaluated our imputations algorithms. The procedure of the experiment is shown in Fig.3. Here, *M* is the original incomplete INFβ data set, and removing incomplete genes, samples and time-points resulted in a complete data set *G* containing 53 genes, 27 samples (18 good and 9 bad responders) and 7 time-points.

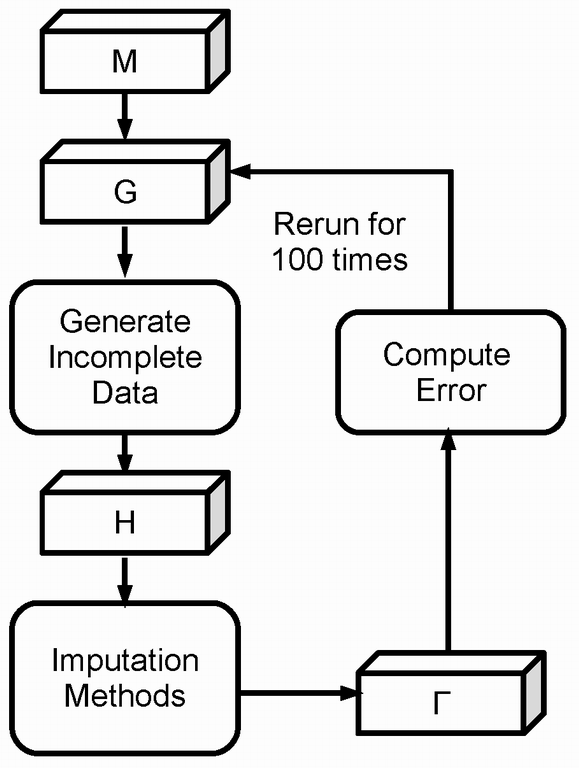


Fig. 3. Experimental procedure.

The incomplete data set *H* is obtained from *G* in the following manner: we randomly select *g*% of all genes time-series in *G* to be altered, then for each such time-series *gij*, we randomly select at most *t* values to be deleted; we randomly select a sample *sj* then remove all values at time-point *tk*., that is we delete the time-point *tk*. for some random *k*. After performing an imputation method on *H* we obtain the estimated *Γ*, then compute the error *E*(*G*, *Γ*).

Table I shows for each method, the number of times it has produced the lowest estimation error for *g%* = 0.1, 0.2, …, 0.9, 1, 2, …, 20 (that is for 29 values of *g*%) when at most *t* = 1, 2, 3, 4, 5 values are missing in selected gene time-series (columns of the table). In Table I, we did not remove any time-point. As shown in the table, GMAimpute is the best method among the G\*impute methods (which use the gene time-series from the same samples to estimate the missing values), while SKNNAimpute is the best method among the S\*impute methods (that use the sample time-series to estimate missing values). For instance, for *t* = 4, GMAimpute outperforms the other G\*impute methods for 25 times out of 29 value of *g*%. SKNNAimpute outperforms the other S\*impute methods for all values of *t*.

TABLE I

Statistics of which method is the best among the five methods for 29 different percentages of (from 0.1% to 20%) genes which contain at most 1, 2, 3, 4, or 5 missing values, respectively. There is no time point missing in this case.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Methods | At Most | | | | | |
| 1 | 2 | 3 | 4 | 5 |
| GAimpute | 0 | 0 | 0 | 0 | 0 | |
| GKNNEimpute | 5 | 8 | 8 | 3 | 6 | |
| GKNNAimpute | 3 | 5 | 0 | 1 | 1 | |
| GSVDimpute | 0 | 0 | 0 | 0 | 0 | |
| GMAimpute | 21 | 16 | 21 | 25 | 22 | |
| SAimpute | 0 | 0 | 1 | 0 | 0 | |
| SKNNEimpute | 2 | 5 | 7 | 4 | 3 | |
| SKNNAimpute | 17 | 15 | 12 | 10 | 15 | |
| SSVDimpute | 0 | 0 | 0 | 0 | 0 | |
| SMAimpute | 10 | 9 | 9 | 15 | 11 | |
| GSAimpute | 0 | 0 | 0 | 0 | 0 | |
| GSKNNEimpute | 4 | 5 | 4 | 7 | 3 | |
| GSKNNAimpute | 16 | 13 | 12 | 5 | 12 | |
| GSSVDimpute | 0 | 0 | 0 | 0 | 0 | |
| GSMAimpute | 9 | 11 | 13 | 17 | 14 | |

Fig. 4 shows the performance of G\*impute methods for *t* = 1. Clearly, GMAimpute gives the lowest error on average as *g*% increase. In general for all values of *t*, we found that KNN-based methods gave best results and SVD-based methods gave worst results. KNN methods use local information around an incomplete time-series, for estimation: i.e, a subset of time-series similar to the given incomplete time-series. The averaged methods estimate an incomplete time-series by the centroid of either a gene or a sample, but unlike in KNN, the centroid may be very far from the incomplete time-series to estimate. In our data set however, SVD performed worst simply because there are not enough complete time-series, and thus is outperformed by average method. In general in all our experiments with G\*impute, S\*impute and GS\*impute, SVD underperformed all other methods. See Figures 5 and 6 for instance, for increasing *g*% with *t* = 1.

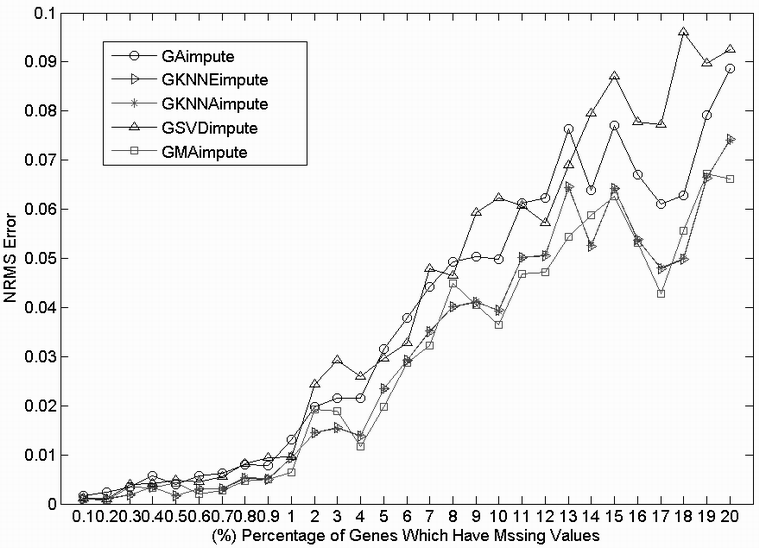


Fig. 4. Errors of G\*impute methods, for *g*% = 0.1% to 20% and *t* = 1.

Next, we performed experiments for the case where data set *H* contains a sample with one time-point that is entirely missing. Tables II, III, and IV show the NRMS error of G\*impute, S\*impute, and SG\*impute methods, respectively, when one time-point missing and that at most *t* = 1 value missing for each of the *g*% selected time-series. In Table II, GKNNAimpute outperforms all the other G\*impute methods in 17 out of 29 values of *g*%. Likewise in Tables III and IV, SKNNAimpute and GSKNNAimpute outperform their respective competitors. Thus, the KNN-based methods are best in all experiments. In all tables, best results are in bold.

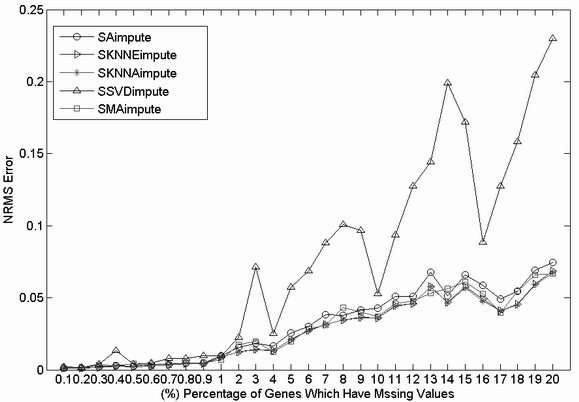


Fig. 5. Errors of S\*impute methods, for *g*% = 0.1% to 20% and *t* = 1.

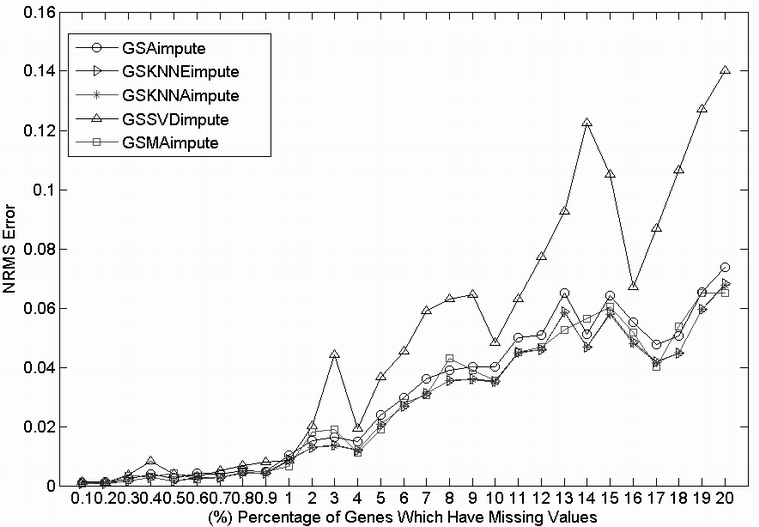


Fig. 6. Errors of GS\*impute methods, for *g*% = 0.1% to 20% and *t* = 1.

TABLE II

The mean NRMS error of G\*impute with at most one missing value for each selected time-series and one missing time point.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Percent | GA  impute | GKNNE  impute | GKNNA  impute | GMA  impute |
| 0.1% | 0.0394 | 0.0338 | **0.0330** | 0.0360 |
| 0.2% | 0.0346 | 0.0270 | 0.0271 | **0.0249** |
| 0.3% | 0.0386 | 0.0329 | 0.0330 | **0.0329** |
| 0.4% | 0.0276 | 0.0237 | **0.0235** | 0.0279 |
| 0.5% | 0.0333 | 0.0251 | **0.0249** | 0.0314 |
| 0.6% | 0.0365 | 0.0308 | **0.0308** | 0.0315 |
| 0.7% | 0.0433 | 0.0377 | 0.0380 | **0.0358** |
| 0.8% | 0.0293 | 0.0248 | **0.0247** | 0.0303 |
| 0.9% | 0.0370 | 0.0285 | 0.0287 | **0.0277** |
| 1% | 0.0344 | 0.0265 | **0.0259** | 0.0262 |
| 2% | 0.0541 | 0.0470 | 0.0466 | **0.0425** |
| 3% | 0.0358 | 0.0292 | **0.0288** | 0.0425 |
| 4% | 0.0517 | 0.0435 | **0.0429** | 0.0479 |
| 5% | 0.0439 | 0.0341 | **0.0339** | 0.0394 |
| 6% | 0.0435 | 0.0350 | 0.0351 | **0.0346** |
| 7% | 0.0465 | 0.0379 | **0.0374** | 0.0430 |
| 8% | 0.0591 | **0.0484** | 0.0486 | 0.0531 |
| 9% | 0.0641 | 0.0522 | **0.0521** | 0.0533 |
| 10% | 0.0660 | 0.0546 | **0.0540** | 0.0591 |
| 11% | 0.0792 | 0.0666 | **0.0661** | 0.0706 |
| 12% | 0.0799 | 0.0662 | **0.0660** | 0.0735 |
| 13% | 0.0780 | 0.0664 | **0.0659** | 0.0664 |
| 14% | 0.0707 | 0.0565 | 0.0569 | **0.0553** |
| 15% | 0.0860 | **0.0717** | 0.0719 | 0.0735 |
| 16% | 0.0774 | 0.0632 | **0.0627** | 0.0636 |
| 17% | 0.0893 | 0.0766 | 0.0764 | **0.0706** |
| 18% | 0.0821 | **0.0687** | 0.0690 | 0.0821 |
| 19% | 0.0961 | 0.0811 | **0.0809** | 0.0823 |
| 20% | 0.1037 | 0.0903 | 0.0894 | **0.0881** |

The first column is the percentage of the time-series having missing values.

TABLE III

The mean NRMS error of S\*impute with at most one missing value for each selected gene and one missing time point.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Percent | SA  impute | SKNNE  impute | SKNNA  impute | SMA  impute |
| 0.1% | 0.0393 | 0.0338 | **0.0330** | 0.0360 |
| 0.2% | 0.0343 | **0.0272** | 0.0273 | 0.0250 |
| 0.3% | 0.0382 | **0.0326** | 0.0327 | 0.0331 |
| 0.4% | 0.0266 | 0.0231 | **0.0230** | 0.0277 |
| 0.5% | 0.0335 | 0.0250 | **0.0247** | 0.0315 |
| 0.6% | 0.0363 | **0.0307** | 0.0308 | 0.0315 |
| 0.7% | 0.0428 | 0.0375 | 0.0380 | **0.0356** |
| 0.8% | 0.0282 | 0.0247 | **0.0245** | 0.0302 |
| 0.9% | 0.0368 | 0.0280 | 0.0283 | **0.0276** |
| 1% | 0.0340 | 0.0263 | **0.0257** | 0.0261 |
| 2% | 0.0517 | 0.0457 | 0.0455 | **0.0427** |
| 3% | 0.0331 | 0.0277 | **0.0273** | 0.0412 |
| 4% | 0.0494 | 0.0442 | **0.0433** | 0.0486 |
| 5% | 0.0410 | 0.0320 | **0.0315** | 0.0387 |
| 6% | 0.0380 | **0.0324** | 0.0329 | 0.0337 |
| 7% | 0.0424 | 0.0355 | **0.0348** | 0.0424 |
| 8% | 0.0532 | 0.0472 | **0.0470** | 0.0534 |
| 9% | 0.0584 | 0.0514 | **0.0507** | 0.0553 |
| 10% | 0.0624 | 0.0526 | **0.0520** | 0.0592 |
| 11% | 0.0719 | 0.0634 | **0.0623** | 0.0686 |
| 12% | 0.0738 | 0.0649 | **0.0646** | 0.0728 |
| 13% | 0.0713 | **0.0615** | 0.0616 | 0.0656 |
| 14% | 0.0610 | **0.0497** | 0.0499 | 0.0537 |
| 15% | 0.0739 | 0.0648 | **0.0642** | 0.0726 |
| 16% | 0.0665 | 0.0571 | **0.0563** | 0.0621 |
| 17% | 0.0790 | 0.0695 | 0.0694 | **0.0690** |
| 18% | 0.0727 | 0.0645 | **0.0645** | 0.0822 |
| 19% | 0.0850 | 0.0759 | **0.0757** | 0.0811 |
| 20% | 0.0927 | 0.0804 | **0.0786** | 0.0844 |

The first column is the percentage of the genes having missing values.

TABLE IV

The mean NRMS error of GS\*impute with at most one missing value for each selected time-series and one missing time point.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Percent | GSA  impute | GSKNNE  impute | GSKNNA  impute | GSMA  impute |
| 0.1% | 0.0393 | 0.0338 | **0.0329** | 0.0360 |
| 0.2% | 0.0344 | 0.0271 | 0.0272 | **0.0249** |
| 0.3% | 0.0382 | **0.0327** | 0.0328 | 0.0330 |
| 0.4% | 0.0268 | 0.0234 | **0.0232** | 0.0277 |
| 0.5% | 0.0331 | 0.0249 | **0.0247** | 0.0314 |
| 0.6% | 0.0360 | 0.0307 | **0.0307** | 0.0314 |
| 0.7% | 0.0425 | 0.0375 | 0.0379 | **0.0356** |
| 0.8% | 0.0282 | 0.0245 | **0.0244** | 0.0301 |
| 0.9% | 0.0363 | 0.0281 | 0.0283 | **0.0276** |
| 1% | 0.0338 | 0.0263 | **0.0256** | 0.0261 |
| 2% | 0.0516 | 0.0460 | 0.0456 | **0.0425** |
| 3% | 0.0328 | 0.0278 | **0.0272** | 0.0416 |
| 4% | 0.0483 | 0.0429 | **0.0422** | 0.0479 |
| 5% | 0.0394 | 0.0319 | **0.0317** | 0.0387 |
| 6% | 0.0378 | **0.0325** | 0.0327 | 0.0336 |
| 7% | 0.0412 | 0.0355 | **0.0348** | 0.0421 |
| 8% | 0.0525 | 0.0466 | **0.0465** | 0.0528 |
| 9% | 0.0573 | 0.0503 | **0.0499** | 0.0538 |
| 10% | 0.0585 | 0.0517 | **0.0510** | 0.0584 |
| 11% | 0.0714 | 0.0636 | **0.0628** | 0.0691 |
| 12% | 0.0724 | 0.0638 | **0.0634** | 0.0725 |
| 13% | 0.0697 | 0.0623 | **0.0621** | 0.0653 |
| 14% | 0.0607 | **0.0507** | 0.0509 | 0.0535 |
| 15% | 0.0747 | 0.0661 | **0.0659** | 0.0722 |
| 16% | 0.0661 | 0.0582 | **0.0574** | 0.0620 |
| 17% | 0.0786 | 0.0711 | 0.0710 | **0.0692** |
| 18% | 0.0709 | **0.0638** | 0.0640 | 0.0810 |
| 19% | 0.0844 | 0.0758 | **0.0754** | 0.0805 |
| 20% | 0.0915 | 0.0831 | **0.0813** | 0.0854 |

The first column is the percentage of the time-series having missing values.

In all experiments, GMAimpute performed best with small *g*% and no missing time-points. MA requires a complete data for transformation, and hence, the initialization process Fig.2 will essentially affect the performance of MA-based imputation methods. For large value of *g*%, such initialization would lead more imprecise alignment which will eventually deteriorate the estimation of missing values.

The 3KNNAimpute methods, which use the integral-distance of [9,10], outperform the KNN methods, based on Euclidean distance; in most experiments (see the tables, above). In the context of time-series analysis, it has been shown in [9] that the integral-distance was more robust than Pearson correlation distance. It remains to investigate why integral-distance performed better than Euclidean. 3KNNA has the advantage of estimating both missing values and missing time-points. Euclidean distance does not consider the order of time points, i.e. exchanging the order of time-points will result in the same distance. Area based distance takes into the account the order of time points. Exchanging the time points would likely produce different distances. Moreover, the area based distance measure can find the similar genes which might be dissimilar in the context of Euclidean distance. The 3MAimpute methods outperform the 3KNNEimpute algorithms for small values of *g*%.

# Conclusions

GST data provide more information to mine principles related to genome by mean of statistical and machine learning algorithms, such as clustering, classification, feature selection, and so on. Due to diverse reasons, GST data often suffer from missing values and missing time points. However these analysis algorithms require a complete GST data. Missing value estimation methods therefore have to be developed as preprocessing tool of GST data before further analysis. The contributions of this study are that it extends three existing missing value estimation methods from 2D microarray data to 3D GST data, and investigate the MA based method. Their performances on GST data are investigated. The extension strategy involves sample-by-sample estimation and gene-by-gene estimation. Not only can the extended methods address the missing value problem, but also missing time points. Experiment shows that GMAimpute is the best method for a small number of missing values, while 3KNNAimpute is the best when the GST data suffer from both missing values and missing time points.

More statistical and heuristic missing values estimation methods are being studied by the authors. Tensor analysis algorithms are under investigation to be specifically used for GST data, because tensor tools can extract substantial features for 3D data, even noisy data. These features can be used to represent the data, which is a possible way to devise tensor based missing value imputation algorithms. *Nonnegative Matrix Factorization* (*NMF*) has been shown in literatures that it has better clustering performance than k-mean clustering. Since GST data are nonnegative, so it is interesting to employ NMF to design clustering based missing value estimation methods. Furthermore, EM based approaches will also be devised for GST data. Time-series regression and analysis tools, for example wavelets should also be emphasized for missing value imputation of GST data. Additionally, more robust time-order preserving distance metrics will be studied

References

1. O. Troyanskaya, M. Cantor, G. Sherlock , P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman1, “Missing value estimation methods for DNA microarrays,” Bioinformatics, vol. 17 no. 6, pp: 520–525, 2001.
2. H. Kim, G. H. Golub, and H. Park, "Missing value estimation for DNA microarray gene expression data: local least squares imputation," Bioinformatics, vol.21, no.2, pp:187-198, 2005.
3. T. H. Bø, B. Dysvik, and I. Jonassen, “LSimpute accurate estimation of missing values in microarray data with least squares methods,” Nudeic Acids Res, vol. 32, no. 3, pp:e34, 2004.
4. G. N. Brock, J. R. Shaffer, R. E. Blakesley, M. J. Lotz, and G. C. Tseng, “Which missing value imputation method to use in expression profiles: a comparative study and two selection schemes,” BMC Bioinformatics, vol. 9:12, 2008.
5. D. V. Nguyen, N. Wang, and R. J. Carroll, "Evaluation of missing value estimation for microarray data," Journal of Data Science, vol. 2, pp: 347-370, 2004.
6. Z. Bar-Joseph, G. Gerber, D. K. Gifford “Continuous representations of time-series gene expression data,” Journal of Computational Biology, vol. 10, no.3-4, pp: 341-356, June, 2003.
7. M. K. Choong, M. Charbit and and H. Yan, “Autoregressive model based missing value estimation for DNA microarray time series data,” *IEEE Trans. on Information Technology in Biomedicine*, vol.13, no. 1, pp:131–137, 2009.
8. S. Oba, M. A. Sato, I. Takemasa, M. Monden, K. Matsubara, and S. Ishii, "A Bayesian missing value estimation method for gene expression profile data," Bioinformatics, vol. 19, no. 16, pp:2088-2096, 2003.
9. L. Rueda, A. Bari, and A. Ngom, Clustering time-series gene expression data with unequal time intervals," Springer Trans. on Compu. Systems Biology X LNBI 5410, 2008, pp:100-123.
10. N. Subhani, A. Ngom, L. Rueda, and C. Burden, "Microarray Time-Series Data Clustering via Multiple Alignment of Gene Expression Profiles", LNCS, vol. 5780/2009, pp: 377-390, 2009.
11. T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown, and D. Botstein, "Imputing Missing Data for Gene Expression Arrays," Stanford Statistics Department, Technical report, 1999.
12. S.E. Baranzini, P. Mousavi, J. Rio, S. J. Caillier, A. Stillman, P. Villoslada, M. M. Wyatt, M. Comabella, L. D. Greller, R. Somogyi, X. Montalban, and J. R. Oksenberg, “Transcription-based prediction of response to INFβ using supervised computational methods”, PLOS Biology, vol. 3, no. 1, pp: 166-176, 2005.
13. B. Weinstock-Guttman, D. Badgett, K. Patrick, L. Hartrich, D. Hall, M. Baier, J. Feichter and M. Ramanathan, “Genomics effects of interferon-β in multiple-sclerosis patients”, Journal of Immunology, vol.171, pp:2694-2702, 2003.
14. A. Zhang, “Advanced analysis of gene expression microarray data”, World Scientific Publishing Co., ISBN 981-256-645-7, pp:214-238, 2009

1. Manuscript received December 14, 2009. This work is supported in part by NSERC Grant RG-PIN 228117.

   The authors are with the School of Computer Science, 5115 Lambton Tower, University of Windsor, 401 Sunset Avenue, N9B 3P4, Windsor, Ontario, Canada. E-mail: {li11112c,angom,lrueda}@uwindsor.ca. [↑](#footnote-ref-1)