Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 80 (2007) 51–62
The Journal of Systems and Software
www.elsevier.com/locate/jss

A new imputation method for small software project data sets

Qinbao Song [a,*], Martin Shepperd [b]

[a] Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
[b] Brunel University, Uxbridge, UB8 3PH, UK

Received 17 March 2005; received in revised form 30 April 2006; accepted 3 May 2006
Available online 16 June 2006

Abstract

Effort prediction is a very important issue for software project management. Historical project data sets are frequently used to support such prediction. But missing data are often contained in these data sets and this makes prediction more difficult. One common practice is to ignore the cases with missing data, but this makes the originally small software project database even smaller and can further decrease the accuracy of prediction. The alternative is missing data imputation. There are many imputation methods. Software data sets are frequently characterised by their small size but unfortunately sophisticated imputation methods prefer larger data sets. For this reason we explore using simple methods to impute missing data in small project effort data sets. We propose a class mean imputation (CMI) method based on the *k*-NN hot deck imputation method (MINI) to impute both continuous and nominal missing data in small data sets. We use an incremental approach to increase the variance of population. To evaluate MINI (and *k*-NN and CMI methods as benchmarks) we use data sets with 50 cases and 100 cases sampled from a larger industrial data set with 10%, 15%, 20% and 30% missing data percentages respectively. We also simulate Missing Completely at Random (MCAR) and Missing at Random (MAR) missingness mechanisms. The results suggest that the MINI method outperforms both CMI and the *k*-NN methods. We conclude that this new imputation technique can be used to impute missing values in small data sets.

Keywords: Software effort prediction; Missing data; Data imputation; Class mean imputation; *k*-NN imputation

1. Introduction

Given that 75% of software projects reported overruns (Moløkken and Jørgensen, 2003), there is considerable demand from industry for accurate software project effort prediction. Unfortunately, a barrier to accurate effort prediction is incomplete and small (in terms of the number of cases) software engineering data sets. Therefore, in order to improve effort prediction, we must first carefully deal with missing data. Although a wide range of missing data techniques have been proposed, none of them specifically focuses upon missing values in small data sets with both nominal and continuous values. For this reason we wish to develop an imputation method specifically to address this problem, that is so typical in software engineering data sets.

There are three types of missing data techniques. Identifying the proper missing data method from these techniques for incomplete small software data sets is the precondition of tackling missing software engineering data. Therefore, in this section, we firstly introduce the taxonomy of missing data techniques, present a big picture of missing data techniques to readers; then briefly summarize the related work in the software engineering field; and lastly raise the research issue of this paper.

1.1. Missing data techniques taxonomy

The missing data problem has been studied by researchers in many fields for more than 30 years. There are three approaches to this problem. First, there are missing data ignoring techniques, e.g. (Haitovsky, 1968; Roth, 1994).

* Corresponding author. Tel.: +86 29 82668645; fax: +86 29 82668971.
E-mail addresses: qbsong@mail.xjtu.edu.cn (Q. Song), martin.shepperd@brunel.ac.uk (M. Shepperd).

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.05.003

Second, there are missing data toleration techniques (Aggarwal and Parthasarathy, 2001; Schuurmans and Greiner, 1997). Third, there are missing data imputation techniques which are the emphasis of this paper, e.g. (Friedman, 1998; Little, 1988; Schafer and Olsen, 1998; Shirani et al., 2000; Troyanskaya et al., 2001).

The *missing data ignoring techniques* simply delete the cases that contain missing data. Because of their simplicity, they are widely used (Roth, 1994) and tend to be the default for most statistics packages, but this may not lead to the most efficient utilization of the data and incurs a bias in the data unless the values are missing completely at random. Consequently they should be used only in situations where the amount of missing values is very small. This approach has two forms:

- *Listwise deletion* (LD) is also referred to as case deletion, casewise deletion or complete case analysis. This method omits the cases containing missing values. It is easy, fast, does not 'invent data', commonly accepted and is the default of most statistical packages. The drawback is that its application may lead to a large loss of observations, which may result in too small data sets if the fraction of missing values is high and particularly if the original data set is itself small, as is often the situation for software project estimation (Myrtveit et al., 2001).
- *Pairwise deletion* (PD) is also referred to as the available case method. This method considers each feature separately. For each feature, all recorded values in each observation are considered (Strike et al., 2001) and missing data are ignored. This means that different calculations will utilise different cases and will have different sample sizes, an undesirable effect. The advantage is that the sample size for each individual analysis is generally higher than with complete case analysis. It is necessary when the overall sample size is small or the number of cases with missing data is large.

The *missing data toleration techniques* use a probabilistic approach to handle missing data. They do not predicte missing data but assign a probability to each of the possible values. Thus they are internal missing data treatment strategies, which perform analysis directly using the data set with missing values.

Breiman et al. (1984) proposed the CART algorithm which may be used to address the missing data problem in the context of a decision tree classifier. If some cases contain missing values, CART uses the best surrogate split to assign these cases to branches of a spilt on a feature where these cases' values were missing. C4.5 (Quinlan, 1993) is an alternative method to CART. C4.5 uses a probabilistic approach to handle missing data. Missing values can be present in any variables except the class variable. This method calculates the expected information gain by assuming that the missing value is distributed according to the observed values in the subset of the data at that node of the tree. From amongst the simpler methods, it seems to be one of the better techniques to deal with missing values (Grzymala-Busse and Hu, 2000).

If the objective is not to predict the missing values, missing data toleration is a nice choice. This is because any prediction of missing values will incur bias thereby making prediction results doubtful. However, most data analysis methods only work with a complete data set, so first we must fill in missing values or delete the cases with missing values, and then use the resulting data set to perform subsequent analysis. In this case, toleration techniques cannot be used. Moreover, in cases where the data set contains large amounts of missing data, or the mechanism causing to the missing data is non-random, imputation techniques are likely to perform better than ignoring techniques (Haitovsky, 1968).

The *missing data imputation techniques* estimate missing values for the missing cases and insert estimates obtained from other reported values to produce an estimated complete case. The common forms are as follows:

- *Mean imputation* (MI) is also referred to as unconditional mean imputation. This method imputes each missing value with the mean of reported values. It is fast, simple, easily implemented and no observations are excluded. The disadvantage is that it leads to underestimation of the population variance. It is also a rather naïve approach.
- *Regression imputation* (RI) is also referred to as conditional mean imputation. This method replaces each missing value with a predicted value based on a regression model. The regression model is built using the complete observations. It tends to perform better than MI, but still underestimates variance.
- *Hot-deck imputation* (HDI) methods fill in missing data by taking values from other observations in the same data set. The choice of which value to take depends on the observation containing the missing value. Randomly choosing observed values from donor cases is the simplest hot-deck method. The similar response pattern imputation (SRPI) (Joreskog and Sorbom, 1993), which identifies the most similar case without missing observations and copies the values of this case to fill in the holes in the cases with missing data, and the k nearest neighbours (k-NN) imputation (Fix and Hodges, 1952; Cartwright et al., 2003; Song et al., 2005; Jönsson and Wohlin, 2004), which searches for the k most similar cases to the missing value and replaces the missing value by the mean or modal value of the corresponding feature values of the k nearest neighbours all belong to this class. This approach preserves the sample distribution by substituting different observed values for each missing observation, but the data set must be large enough to find appropriate donor cases.
- *Multiple imputation* means that the missing data are imputed $m > 1$ times, with a different randomly chosen error term added in each imputation. In this method, each missing value is replaced by a set of m plausible

values drawn from their predictive distribution. After performing multiple imputation, there are m complete, imputed data sets.[1] Then each imputed data set can be analyzed by complete-data methods. After performing identical analyses, the results are combined and overall estimates are produced (Schafer and Olsen, 1998). This method can relieve the distortion of the sample variance, but it needs to include all features, data must meet normal distribution assumptions, as well as computing and storage requirements.

From previous introduction, we see that ignoring techniques make small software engineering data sets even smaller and further make software effort prediction more difficult; while toleration techniques cannot provide complete data sets for software effort prediction methods, so only imputation techniques can be used for the purpose of completing small and incomplete software data sets.

1.2. Related work

Dealing with missing software project data is a recent phenomenon, and is entirely focused on the empirical evaluation of existing missing data processing techniques (Strike et al., 2001; Myrtveit et al., 2001; Cartwright et al., 2003; Song et al., 2005; Jönsson and Wohlin, 2004).

Strike et al. (2001) evaluated three different techniques for dealing with missing data in the context of software cost modelling. These techniques include listwise deletion, mean imputation and eight different types of hot-deck imputation. Their results indicate that all the missing data techniques perform well, with small biases and high precision. However, they recommend the simplest technique, listwise deletion, as a reasonable choice.

Myrtveit et al. (2001) applied missing data techniques to a software project data set, and evaluated four missing data techniques: listwise deletion (LD), mean imputation (MI), similar response pattern imputation (SRPI) and full information maximum likelihood (FIML). They found that prediction models constructed on LD, MI and SRPI data sets were biased unless the data were missing completely at random (MCAR), and only FIML was appropriate when the data set was not MCAR, but in this situation one must have sufficient data for the technique. Compared to LD, MI and SRPI seem appropriate when the LD data set is too small to enable the construction of a meaningful regression-based prediction model.

Cartwright et al. (2003) used two industrial data sets containing a number of missing values to assess the potential value of imputation. The relative performance was compared of effort prediction models derived by stepwise regression methods on the raw data and data sets with values imputed by various techniques. For both data sets they found that k-NN and sample mean imputation (SMI) significantly improved effort prediction accuracy with the k-NN method giving the best results.

Song et al. (2005) explored the entry validation issue by determining what is the safest default assumption about the missingness mechanism for imputation techniques when dealing with small software project data sets. They found that both CMI and k-NN imputation techniques have practical applications for small software engineering data sets with missing values.

Jönsson and Wohlin (2004) evaluated the k-NN method using Likert data in a software engineering context. They found that it is feasible to use the k-NN method with Likert data and the ability of the method remains high for large amount of missing data without affecting the quality of the imputation.

Although the empirical evaluation of existing general-purpose missing data processing techniques is important, missing data imputation must be developed within the context of the specific analysis. Since different analysts are concerned with different contexts, no single imputation technique can satisfy all interests. On the other hand, the general-purpose methods are not always the best choice for the problem of a specific field. Therefore, we place our work in the context of software project effort prediction with small data sets and propose a new robust method to impute missing software project data.

Dealing with small size – in terms of the number of cases – coupled with both nominal and continuous values is usually an important characteristic of historical software project data sets, but the more sophisticated imputation methods benefit from larger data sets, so we decided to explore how to use simple methods to impute missing data for situations of relatively few cases (i.e. 100 or less). Class mean imputation (CMI) offers clear advantages over mean imputation but k nearest neighbours (k-NN) imputation gives better results when the homogeneity of the data set is high. However, CMI reduces the variance of population, while k-NN may select a non-relevant object because of its heavy dependance on the distance between two cases. Therefore, in order to keep the advantages of both methods and avoid their individual limitations, we integrate CMI and k-NN to address the missing data problem in small data set size environments containing both nominal and continuous features.

When we use the CMI and k-NN methods, one important problem is determining which features to use to compute distance when we search for similar cases: all or just a subset and in that case which subset? These questions will also be discussed in this paper. One ubiquitous problem of most data imputation methods is that usually they underestimate the variance of feature values. We try to relieve this problem by using an incremental method to fill in the missing data.

The remainder of this paper is organized as follows. Section 2 describes the problem, gives some definitions and introduces relevant terms. Section 3 presents the

[1] Across these complete data sets, the observed values are the same, but the missing values are filled in with different imputations that reflect the uncertainty about the missing data.

proposed missing data imputation algorithms. Section 4 shows the experimental results. Section 5 summarises the contribution of this paper.

2. Problem statement

In this section we first formally define the problem and then introduce the missing mechanisms and missing patterns.

2.1. Problem definition

2.1.1. Data set, complete and incomplete data sets

A dataset is a number of observations on $n \in N$ software projects, and is denoted as $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ where X_i $(i = 1, 2, \ldots, n)$ is one observation or case. We assume that the data set $\mathbf{X}$ is divided into an observed component $\mathbf{X}_{\text{obs}}$ and a missing component $\mathbf{X}_{\text{mis}}$, thus $\mathbf{X} = \{\mathbf{X}_{\text{obs}}, \mathbf{X}_{\text{mis}}\}$. If the missing component is null, then the data set is complete, otherwise it contains missing values.

Each element or project $X_i \subset \mathbf{X}$ where $i = 1, 2, \ldots, n$ is a case, and if given p as the dimensionality of the feature space, case X_i can be denoted as $(x_{i1}, x_{i2}, \ldots, x_{ip})$, $p > 1$. Given $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant p$ then x_{ij} is the value of the jth feature of the ith case from data set $\mathbf{X}$. Features are a set of variables used to characterize the corresponding projects. We denote the jth feature of the data set $\mathbf{X}$ as F_j where $j = 1, 2, \ldots, p$. Within the data set $\mathbf{X}$ we assume each case has a special feature c that acts as a class label,[2] so we have $N_c \in N$ distinct classes. In addition for each case, X_i $(1 \leqslant i \leqslant n)$, we denote the missing feature values as $X_i^{\text{mis}} = \{x_{ij}, ?\}^f$ subject to $0 \leqslant f < p \wedge 1 \leqslant j \leqslant p$. Thus, $X_i = (X_i^{\text{obs}}, X_i^{\text{mis}})$, where X_i^{obs} are the observed feature values. Note that each case may have different missing features.

2.1.2. Missing data imputation

This involves using some technique to complete the missing data '?' in $X_i^{\text{mis}} \subset X_i$, produce an estimated complete case $\hat{X}_i$ and an estimated complete data set $\hat{\mathbf{X}}$, so that the latter can be used for further analysis.

This is best illustrated by an example. Suppose there is a data set, which can be used to predict software effort, consisting of three variables and 16 cases, see Table 1 for details. The first two columns of the table show the complete data for variables V_1 and V_2, and the last column shows the values for variable V_3. As variable V_3 contains a missing value in case 2, we cannot directly use this data set to predict software effort. First we have to impute this missing value. Because the similarity between the values of variables V_1 and V_2 in case 2 and the corresponding values in case 1 is very high, the missing data '?' can be assumed to be 30 as that is the value of variable V_3 in case 1. Now we've obtained an imputed complete data set, thus can be used to predict software effort.

Table 1
A data set and the simulation of three missingness mechanisms

V_1	V_2				
	Complete	MCAR	MAR	NI	V_3
A	85	85	85	?	30
A	94	?	94	?	?
A	111	111	111	111	45
A	130	130	130	130	58
B	80	80	?	?	60
B	97	97	?	?	67
B	117	117	?	117	70
B	125	?	?	125	80
C	88	?	88	?	81
C	91	91	91	?	84
C	123	123	123	123	94
C	132	?	132	132	97

2.2. Missingness mechanisms and patterns

Both missingness mechanisms and patterns have great impact on research results, both are critical issues a researcher must address before choosing an appropriate method to deal with missing data.

2.2.1. Missingness mechanisms

Missing values introduce complexities to data analysis, so it is important to choose an appropriate method. However, choosing the proper method depends on the assumptions one makes about the missingness mechanism. In general, the missingness mechanism is concerned with whether the missingness is related to the study variables or not. This is extremely significant as it determines how difficult it may be to handle missing values and at the same time how risky it is to ignore them.

Little and Rubin (2002) divide these mechanisms into three classes: Missing Completely at Random (MCAR), Missing at Random (MAR), and Non-ignorable (NI). Viewing response as a random process (Rubin, 1976), the missingness mechanisms can be introduced as follows.

Suppose Z is a data matrix that includes observed and missing data, let Z_{obs} be the set of observed values of Z, let Z_{mis} be the set of missing values of Z and let R be the missing data indicator matrix, i be the ith case and j the jth feature:

$$R_{ij} = \begin{cases} 1 & \textit{if } Z_{ij} \text{ is missing,} \\ 0 & \text{if } Z_{ij} \text{ is observed.} \end{cases}$$

Missing Completely At Random (MCAR) indicates that the missingness is unrelated to the values of any variables, whether missing or observed, thus

$$p(R|Z) = p(R) \quad \text{for all } Z.$$

This can be illustrated by an example. Suppose columns 3–5 of Table 1 show the values of V_2 that remain after imposing three missingness mechanisms. In the third

[2] For example we might use application domain or a coarse grained view of project size as potential class labels.

column, the missing values appeared in the cases whose values of variable V_1 are A, B, and C, which are the all possible values of variable V_1, and the missing values of variable V_2 and variable V_3 distributed from small to large. This means the values were missing randomly and have no any relation with variable V_1, variable V_2 and itself, thus it is MCAR.

MCAR is an extreme condition and from an analyst's point of view, it is an ideal condition. Generally you can test whether MCAR conditions can be met by showing there is no difference between the distribution of the observed data of the observed cases and missing cases, this is Little's (Little, 1988; Little et al., 1995) multivariate test which is implemented in SYSTAT and the SPSS Missing Values Analysis module. Unfortunately this is hard when there are few cases as there can be a problem with Type I errors.

Non-Ignorable (NI) is at the opposite end of the spectrum. It means that the missingness is non-random, it is related to the missing values, and it is not predictable from any one variable in the data set. That is

$$p(R|Z) \neq p(R) \quad \text{for all } Z, \quad p(R|Z) \text{ depends on } Z_{\text{mis}}.$$

NI is the worst case since, as the name implies, the problem cannot be avoided by a deletion technique nor are imputation techniques in general effective unless the analyst has some model of the cause of missingness. For example, in the fifth column of Table 1, all the values that are smaller than 100 were missing, the corresponding values of variable V_3 distributed from small to large and the corresponding values of the V_1 can be A, B, and C. This means the values that were missing only depend on the variable V_2 itself, thus it is NI. This also can be illustrated by another example. Suppose software engineers are less likely to report high defect rates than low rates, perhaps for reasons of politics. Merely to ignore the incomplete values leads to a biased sample and an overly optimistic view of defects. On the other hand imputation techniques do not work well either since they attempt to exploit known values and (as we have already observed) this is a biased sample. Unless one has some understanding of the process and can construct explanatory models there is little that can be done with NI missingness.

Missing At Random (MAR) lies between these two extremes. It requires that the cause of the missing data is unrelated to the missing values, but may be related to the observed values of other variables, that is

$$p(R|Z) = p(R|Z_{\text{obs}}) \quad \text{for all } Z_{\text{mis}}.$$

For example, in the fourth column of Table 1, all the missing values appeared in the cases whose values of variable V_1 are B, and the missing values distributed from small to large. This means the missing values depend only on the variable V_1, and have no relation with itself, thus it is MAR.

Most missing data methods assume MAR. Whether the MAR condition holds can be examined by a simple *t*-test of mean differences between the groups with complete data and that with missing data (Kim and Curry, 1977; Tabachnick and Fidell, 2001). MAR is less restrictive than MCAR because MCAR is a special case of MAR. MAR and MCAR are both said to be ignorable missing data mechanisms, which is coined in (Rubin, 1976) and is fully explained in the context of multiple imputation in (Rubin, 1987).

In practice it is usually difficult to meet the MCAR assumption. MAR is an assumption that is more often, but not always, tenable.

2.2.2. Missingness patterns

The missing data indictor matrix R reveals the missing data pattern. By rearranging the cases and the variables of a data set, we can get the missing data patterns. Generally, there are two types of missing data patterns, they are the *univariate pattern* and *multivariate pattern*.

In the *univariate missingness pattern*, only one variable contains missing values. While in the *multivariate missingness pattern*, more than one variable contains missing data. We can refine this pattern into two types: *monotone pattern* and *arbitrary pattern*. In *monotone pattern*, variables can be arranged so that for a set of variables $x_1, x_2, \ldots, x_n$, if x_i is missing, then so are $x_{i+1}, \ldots, x_n$. In *arbitrary pattern*, missing data can occur anywhere and no special structure appears regardless of how the variables are arranged.

The SPSS Missing Values Analysis module has the function of assessing missing data patterns. The type of missing data pattern may affect the selection of missing data methods, because some missing data methods are sensitive to the missing data patterns. Therefore, we will discuss this issue when introducing a specific missing data imputation method if applicable.

We can conclude that the missingness mechanism concerns the distribution of R given Z. Since the missingness pattern concerns which values are missing, it solely concerns the distribution of R. Some imputation methods cannot be used for some missing patterns or some missing mechanism. Thus, before imputing, we must know the missingness mechanism and pattern of the given data set, and choose a suitable imputation method that can work well under the missingness mechanism and pattern.

3. Proposed imputation approach

In this section, we provide one information theoretic key features selection method, introduce the distance measures used by the proposed imputation algorithms and describe our new imputation algorithm.

3.1. Key features selection

The purpose of imputing missing data in the empirical software engineering domain is to obtain a complete data set and use this complete data set to make a software project effort prediction or other predictions. Therefore,

although all features are necessary for characterising a set of software projects, not all of them are necessary for predicting one specific feature since not all features will be required by that prediction model. On the other hand, different prediction tasks may need different feature subsets. When performing a specific prediction task, we must first decide which feature subset is useful for the prediction task, and then impute the missing data only for that feature subset. Otherwise, if we use all of the features for the prediction task and impute all of the missing values, it is not only redundant, but also useless work that reduces the prediction accuracy to a certain extent. So we argue that not every missing value but merely those contained in key features need to be imputed.

Feature subset selection (Kohavi and John, 1997; Jain and Zongker, 1997), which we refer to as key feature selection, is the process of identifying and removing as much irrelevant and redundant information as possible. It is a problem which occurs in many contexts (Han et al., 1996; Provost and Kolluri, 1999). It is also an important problem in software project effort prediction. Kirsopp et al. (2002) explored this problem. They and others think it is a combinatorial problem and therefore NP-hard. In this paper, we will discuss this problem from the aspect of informatics. As stated previously, in the field of software project effort prediction, we argue that not all the missing data, but only the key features, need to be imputed. First, we should know what are the key features and how to identify them from a given data set for a specific prediction task.

The feature subsets that play important roles in specific prediction purpose are key features. We use an information-theoretic (Quinlan, 1993) approach to select them. The features with higher information gain are chosen as the key features.

Suppose s_i $(i = 1,2,\ldots,N_c)$ is the number of cases of $\mathbf{X}$ in class C_i $(i = 1,2,\ldots,N_c)$. The expected information needed to classify the given data set $\mathbf{X}$ is given by

$$I(s_1,s_1,\ldots,s_{N_c}) = -\sum_{i=1}^{N_c}\frac{s_i}{n}\log_2\frac{s_i}{n}. \quad (1)$$

Note that a log function to the base 2 is used since the information is encoded in bits.

Let Feature F have v distinct values, $\{a_1,a_2,\ldots,a_v\}$. Feature F can be used to partition $\mathbf{X}$ into v subsets, $\{X_1,X_2,\ldots,X_v\}$, where X_j $(j = 1,2,\ldots,v)$ contains those cases in $\mathbf{X}$ that have value a_j $(j = 1,2,\ldots,v)$ of F. If X_j contains s_{ij} cases of class C_i, the information needed to classify cases in all subsets X_j by Feature F, also referred to as Entropy, is given by

$$E(F) = \sum_{j=1}^{v}\frac{s_{1j}+s_{2j}+\cdots+s_{N_cj}}{n}I(s_{1j},s_{2j},\ldots,s_{N_cj}). \quad (2)$$

Note that for a given subset X_j,

$$I(s_{1j},s_{2j},\ldots,s_{N_cj}) = -\sum_{i=1}^{N_c}\frac{s_{ij}}{s_j}\log_2\frac{s_{ij}}{s_j}. \quad (3)$$

The information gain of feature F is

$$\mathrm{Gain}(F) = I(s_1,s_2,\ldots,s_{N_c}) - E(F). \quad (4)$$

We compute the information gain of each feature according to Eqs. (1)–(4). The feature with the highest information gain is chosen as one key feature for the given data set. By repeating the procedure for all subsets, we can obtain all the key features. The algorithm of learning key features from the training data set is as follows:

Algorithm: KeyFeaturesExtraction
Input: *feature-list, cases*
Output: *KeyFeatures*
(1.1) if *cases* are all of the same class then return *KeyFeatures*;
(1.2) if *feature-list* is empty then return *KeyFeatures*;
(1.3) for each feature F_i among *feature-list* do
(1.4) Compute information gain (F_i) according to formula (4);
(1.5) end for;
(1.6) Search for feature F with maximum information gain;
(1.7) Let *feature-list* - = F;
(1.8) Let *KeyFeatures* $\cup = F$;
(1.9) for each known value a_i of feature F do
(1.10) Let X_i be the set of cases from cases for which $F = a_i$;
(1.11) if X_i is not empty then call KeyFeaturesExtraction (*feature-list*, X_i);
(1.12) end for;

3.2. Proposed imputation algorithm

The MI method is fast, simple to implement, no observations are excluded and it is widely be used. For both MCAR and MAR missingness mechanisms, bias caused by using this method tends to be almost zero (Strike et al., 2001). Therefore, we base our proposed imputation algorithm on the naïve mean imputation method.

When we use this method, however, all the missing data are imputed at the centre of the distribution so the variance for the feature that contains missing data will be underestimated. The reason is that it uses the mean of the observations to impute all the missing vales. In order to overcome this problem, we classify the data set according to the specific prediction task and use the features mean of the corresponding class instead of the features mean of all cases and then use an incremental method to enhance it. Unfortunately, the variance still needs to be improved, nonetheless, we believe MI offers a useful benchmark for comparison purposes.

k-NN is a commonly used imputation technique (Fix and Hodges, 1952; Cartwright et al., 2003; Song et al., 2005; Jönsson and Wohlin, 2004), where k is the number of cases sought. k-NN works by finding the k most similar complete cases to the target case to be imputed where sim-

ilarity is measured by a given distance function. The known values of the most similar cases are then used to derive a value for the missing data. Because of the high variance and low bias when k is small, a number of studies have reported good results using k-NN. But k-NN will give misleading results when the heterogeneity of the data set is high since it heavily depends on the distances between cases to select the donors so it may identify the donors from incorrect classes.

Based on the above analysis, we integrate the k-NN method with the MI method, using the latter to provide a correct searching range for the former, and the former to improve the total variance and reduce the bias posed by the latter. We refer to the proposed algorithm as Mean Imputation based k Nearest neighbours hot-deck Imputation (MINI).

3.2.1. Distance measures

Distance is an important measure when we classify cases and look for donors for missing values. When we calculate the distance between any two cases or one case and one class, both nominal and continuous values are used because in practice both types of feature are to be found in real world data sets. In this subsection, first we describe the distance measures between nominal values and continuous values respectively, and then we merge them together to obtain a generalized distance measure between two cases or one case and one class.

Definition 1. (*Distance between continuous values*). Suppose jth feature F_j $(j=1,2,\ldots,p)$ of data set $\mathbf{X}$ is a continuous feature, let $D(x_{ij},x_{i'j})$ be the distance between values x_{ij} and $x_{i'j}$ $(1\leqslant i,i'\leqslant n)$ of feature F_j, we define it as

$$D(x_{ij},x_{i'j})=\frac{1}{\max\{x_{rj}|r=1,2,\ldots,n\}}|x_{ij}-x_{i'j}|. \tag{5}$$

Thus, all the distances between two continuous values are mapped into interval $[0,1]$, this cancels the impact of different scales.

Definition 2. (*Mean of continuous values of a given class*). Suppose the jth feature F_j $(j=1,2,\ldots,p)$ of data set $\mathbf{X}$ is a continuous feature, let M_j^c be the mean of the jth feature of class C_c $(c=1,2,\ldots,N_c)$, we define it as

$$M_j^c=\frac{1}{\|C_c\|}\sum_{r=1}^{n}x_{r,j}^c, \tag{6}$$

where $\|C_c\|$ is the number of cases in class C_c; $c\in\{1,2,\ldots,N_c\}$; $x_{r,j}^c$ indicates that the feature value $x_{r,j}$ belongs to class C_c.

Nominal features frequently appear in historical software project data sets along with continuous features. Usually, each case is partly continuous values and partly nominal values. When we search for similar cases or construct distance functions, we must tackle nominal values together with continuous values.

Definition 3. (*Distance between nominal values*). Suppose jth feature F_j $(j=1,2,\ldots,p)$ of data set $\mathbf{X}$ is a nominal feature. Let $d(x_{ij},x_{i'j})$ be the distance between values x_{ij} and $x_{i'j}$ $(1\leqslant i,i'\leqslant n)$ of feature F_j, it is defined as

$$d(x_{ij},x_{i'j})=\begin{cases}1 & \text{if } x_{ij}\neq x_{i'j},\\ 0 & \text{if } x_{ij}\equiv x_{i'j}.\end{cases} \tag{7}$$

Definition 4. (*Mode of nominal values*). Given v the known feature value, the mode M_j^c of the jth nominal feature F_j $(j=1,2,\ldots,p)$ of class C_c $(c=1,2,\ldots,N_c)$ is defined as

$$M_j^c=\max_{1\leqslant i\leqslant n}\left\{P\left(x_{ij}^c\right)\right\}, \tag{8}$$

where

$$P\left(x_{ij}^c\right)=\frac{\left\|x_{ij}^c\middle|x_{ij}^c\right)\equiv v\right\|}{N_c}\quad(i=1,2,\ldots,n).$$

Definition 5. (*Generalized distance between two cases*). For any two cases X_i, $X_j\subset\mathbf{X}$ $(ij=1,2,\ldots,n\wedge i\neq j)$, we define the generalized distance between them as

$$D(X_i,X_j)=\sum_{k=1}^{p}|x_{i,k}-x_{j,k}|, \tag{9}$$

where

$$|x_{i,k}-x_{j,k}|=\begin{cases}D(x_{i,k},x_{j,k}) & \text{if } k\text{th feature is continuous value}\\ d(x_{i,k},x_{j,k}) & \text{if } k\text{th feature is nominal value.}\end{cases}$$

Definition 6. (*Generalized distance between one case and one class*). Suppose the jth feature of data set $\mathbf{X}$ contains missing data, cases X_i, $X_j\subset\mathbf{X}$ $(ij=1,2,\ldots,n|i\neq j)$, let $D(i,c)$ be the distance between case X_i with missing data and class C_c $(c=1,2,\ldots,N_c)$, we define it as

$$D(i,c)=\frac{1}{NoC_c}\sum_{j\in NoC_c}D(X_i,X_j), \tag{10}$$

where, $c\in\{1,2,\ldots,N_c\}$; X_j is a case that belongs to class C_c; NoC_c is the set of case numbers of class C_c.

3.2.2. Algorithm

The MINI algorithm is a three-step procedure: key features selection, key data classification, and missing values filling in. The key features selection step learns key features f_k from the training data set, see Section 3.1 for details. The key data classification step extracts the key data set $\hat{\mathbf{X}}$ according to key features f_k from real data set $\mathbf{X}$ and uses an existing method to classify the data set $\hat{\mathbf{X}}$ according to the particular prediction task. The missing values filling in step searches for the most similar cases and assigns the mean or mode of the feature values to the missing data. This step computes the distances between the first case containing missing data and each class, searches for the minimum distance among these distances and obtains the corresponding class. Then it uses the k-NN method to look

for donor cases for each missing data vaue in the range of the given class, and fills in the first missing value with the feature mean or mode of donor cases. After that, the missing value is written back and the algorithm moves on to impute the second missing value using the same method as the first one, and so on.

Suppose that the jth feature value x_{ij} of case X_i is missing, let f_k be the key features and F_j be the jth feature. The imputation algorithm is as follows:

Algorithm: MINI

Input: Key data set $\hat{\mathbf{X}}$ with class label set C_k $(k = 1, 2, \ldots, N_c)$, key features f_k.

Output: The estimated value of missing data x_{ij}

(3.1) Search data set $\hat{\mathbf{X}}$ for missing values index set $S_m = \{(ij)\}^{N_m}$,
where i and j are respectively the case number and feature number of missing data x_{ij}, and N_m is the number of missing values;
(3.2) for each $(ij) \in S_m$ do
(3.3) for each class $c \in C_k$ do
(3.4) Use formula (10) to compute the distance $D(i, c)$ between case X_i and class c according to the key features f_k;
(3.5) end for
(3.6) $d(i, c) = \min_{1 \leqslant c \leqslant N_c}\{D(i, c)\}$;
(3.7) Search for donor set DoN in class c using k-NN method;
(3.8) if F_j is a continuous feature then
(3.9) Use formula (6) to compute the mean M_j^c of the jth feature of donor set DoN;
(3.10) else if F_j is a nominal feature then
(3.11) Use formula (8) to compute the mode M_j^c of the jth feature of donor set DoN;
(3.12) end if
(3.13) $\hat{x}_{ij} = M_j^c$;
(3.14) $X_m = \cup \hat{x}_{ij}$;
(3.15) end for
(3.16) return X_m;

4. Experiments and results

4.1. Data source

The data set used in this study is the International Software Benchmarking Standards Group (ISBSG) database, Release 7. The ISBSG database contains 1238 projects from insurance, government, banking, business services, manufacturing, communications, and utilities organisations of 20 different countries. The ISBSG data set has a large proportion of missing data, exceeding 40% for some features.

In order to simulate various missing patterns and compare the accuracy of different imputation methods, we first cleaned the ISBSG database as follows. If a feature contained missing values and was not a key feature for predicting software project effort, it was deleted, otherwise, the corresponding case was deleted. Thus we obtained a new data set that comprised 363 complete cases, named ISBSG363.

In order to validate that the proposed method can be used to impute missing data in small data sets, we randomly extracted 100 and 50 cases from data set ISBSG363, and named them ISBSG100 and ISBSG50 respectively. The ISBSG database has 55 features, some of which have no relation with software effort prediction, so we used the key features selection method to identify eight features and obtain the final data sets. The general information of these two data sets is shown in Table 2.

We used these two data sets for the experiments.

4.2. Simulation approach and evaluative measures

The basic idea of our simulation approach is the following. For data sets ISBSG100 and ISBSG50, by deleting the known values from each of them according to selected missingness mechanisms, we created five data sets containing missing values for each given missing pattern and missing data percentage. We simulated five missingness mechanisms, four missing patterns and five missing data percentages, so we generated 160 data sets for each of the two basic data sets, so a total of 320 data sets was obtained. Then three different methods, CMI, k-NN[3] and MINI, were applied and their performance compared to the true values.

When generating incomplete data sets, we considered the following three aspects:

- *Missingness mechanisms*
  MCAR: The implementation of the MCAR mechanism is to induce missing values for the desired feature or features completely at random.MAR: The implementation of the MAR mechanism is based on the size of the project. The bigger the project is, the greater is the probability of missing data. First we order the cases according to project size, then divide the data set into four parts with different percentages of missing data. The missing data percentage is in proportion to the mean of project size of each part. That is, for ith part, the missing data percentage is $\frac{M_i}{\sum_{i=1}^{4} M_i} \times p\% \times n$, where M_i is the mean of project size of the ith part, $p\%$ is the total missing data percentage, and n is the number of data items.
- *Missingness patterns*
  Four missing data patterns were simulated: univariate missing data in both nominal feature and continuous feature, bivariate missing data in both nominal features and continuous features.

[3] In the experiment, for the proposed method MINI and k-NN method, we used the two nearest cases ($k = 2$) since Kadoda et al. (2001) suggested this to perform consistently better than higher values of k for this particular problem domain.

Table 2
Descriptive statistics for data sets ISBSG100 and ISBSG50

Feature	Type	Description
Development type	Nominal	Enhancement, new development or re-development
Development platform	Nominal	MainFrame, MidRange or PC
Programming language	Nominal	ACCESS, Ada, C,…, other
Organisation type	Nominal	Aerospace, automotive, banking,…, other
Application type	Nominal	Administrative support system,…, other
Project elapsed time	Continuous	For ISBSG100: min = 1, mean = 10.46, median = 7, max = 84 For ISBSG50: min = 1, mean = 10.998, median = 8, max = 45
Function points	Continuous	For ISBSG100: min = 11.0, mean = 498.15, median = 278.5, max = 5684 For ISBSG50: min = 52.0, mean = 549.7, median = 318.5, max = 3460
Summary work effort	Continuous	For ISBSG100: min = 30.0, mean = 4448.63, median = 2044, max = 28,855 For ISBSG50: min = 30.0, mean = 6389.4, median = 2570, max = 32,760

- *Missing data percentages*
  For each missing data pattern of each missing data mechanism, four different percentages of missing values (10%, 15%, 20%, and 30%), were simulated.

When evaluating the different methods, for continuous values, we used Mean Magnitude of Relative Error (MMRE) (Conte et al., 1986) as the accuracy measure of different imputation methods. MMRE is a widely used indicator of prediction accuracy in the software engineering domain and is defined as

$$MMRE = \frac{100}{n}\sum_{i=1}^{i=n}\frac{|x_i - \hat{x}_i|}{x_i}, \tag{11}$$

where n is the number of predictions $\hat{x}$ of x.

For nominal values, we used the following formula to compute the imputation error, ϵ:

$$\epsilon = \left(1 - \frac{N_{\text{cor}}}{N_{\text{mis}}}\right) \times 100, \tag{12}$$

where N_{cor} is the number of missing values correctly imputed, N_{mis} is the total number of missing values.

4.3. Experimental results

In this subsection, we present the experimental results (in terms of MMRE or ϵ, the variance of MMRE or ϵ, p values, and the rate γ of growth of MMRE or ϵ.) for the data sets ISBSG100 and ISBSG50 with different missing patterns under MCAR and MAR missingness mechanisms for three different imputation methods: MINI, k-NN and CMI. For each missing data percentage of each missing pattern, we let the average MMRE (for the continuous values) or the average ϵ (for the nominal values) of the corresponding five data sets be the final result.

From Table 3 we observe a trend for all three techniques to deteriorate in performance as the proportion of missing values increases. This occurs for both nominal and continuous features. We also see that MINI consistently has lower error rates than either of the other techniques.

Table 3
The accuracy of the three methods with different missing data percentages

Feature type	Missing data percentage (%)	Imputation method		
		CMI	MINI	k-NN
Nominal	10	40.5	28.1	32.3
	15	44.7	32.9	36.7
	20	47.6	35.4	40.3
	30	50.0	38.0	43.3
Continuous	10	66.1	58.2	74.0
	15	68.6	62.0	76.8
	20	70.9	65.6	78.6
	30	72.6	67.3	82.4

From Table 4 we also observe that the larger data set leads to better imputation but that the effect is not very large. This occurs for both nominal and continuous features. Again we see that MINI consistently has lower error rates than either of the other techniques for both of the data sets.

Table 5 shows the impact of imputation when one or two features are missing. Two missing features reduce performance. This occurs for both nominal and continuous features, however, the effect is far more pronounced for nominal than continuous features. Again we see that MINI consistently has lower error rates than either of the other techniques.

In Table 6 we consider the impact of different missingness mechanisms. Unsurprisingly, MCAR is easier to deal with, however the differences are not very pronounced which is an encouraging result from a practical point of view.

Table 4
The accuracy of the three methods with different data set sizes

Feature type	Data set size	Imputation method		
		CMI	MINI	k-NN
Nominal	50	48.1	34.5	39.7
	100	43.3	32.7	36.6
Continuous	50	69.9	64.1	80.9
	100	69.2	62.4	75.0

Table 5
The accuracy of the three methods with different number of features with missing data

Feature type	No. of features with missing data	Imputation method		
		CMI	MINI	*k*-NN
Nominal	1	39.4	29.4	33.1
	2	52.0	37.9	43.2
Continuous	1	67.6	61.3	76.6
	2	71.5	65.2	79.3

Table 6
The accuracy of the three methods with different missingness mechanisms

Feature type	Missingness mechanism	Imputation method		
		CMI	MINI	*k*-NN
Nominal	MAR	47.8	36.2	39.1
	MCAR	43.9	31.1	37.2
Continuous	MAR	70.3	62.7	79.5
	MCAR	68.8	61.6	76.7

The overall comparison of the performance of the three imputation methods is given by Table 7. From the above results we can observe that:

1. The MINI method exhibits a lower error rate than either the CMI or *k*-NN method.
2. The MINI's variance of accuracy lies between CMI and *k*-NN. Here a lower value implies the method is more reliable.
3. The CMI method is most suited for imputing missing data in continuous features, *k*-NN to nominal features and MINI to both conditions.

Both the missingness mechanism and missing data percentage are important factors affecting the performance of imputation methods. For the purpose of more formally determining the significance of the results, we used a Wilcoxon test to compare sample medians of the accuracy values given missingness mechanism and data set size respectively. We have five alternate hypotheses:

- MINI imputation is more accurate than CMI/*k*-NN when the missingness mechanism is MAR;
- MINI imputation is more accurate than CMI/*k*-NN when the missingness mechanism is MCAR;
- MINI imputation is more accurate than CMI/*k*-NN when the data set size is 50;
- MINI imputation is more accurate than CMI/*k*-NN when the data set size is 100;
- MINI imputation is more accurate than CMI/*k*-NN.

Tables 8–10 are the results of Wilcoxon tests. The null hypotheses are that there is no difference with ($\alpha = 0.05$). From the tables we can see that all five alternate hypotheses are accepted. This suggests that MINI outperforms both CMI method and *k*-NN method.

Lastly we consider the problem of reduced feature variance. An unfortunate side effect of many imputation techniques is to reduce the variance of features that contain imputed values. This is particularly acute for very simple methods such as mean imputation since all replaced values will be the sample mean leading to a potentially substantial reduction in variance. Table 11 indicates that MINI fares the best of the three methods with only 5% reduction com-

Table 7
The accuracy of the three methods

Feature type	Imputation method	Accuracy	Accuracy variance	
			Mean	Median
Nominal	CMI	45.7	62.1	58.0
	MINI	33.6	62.3	56.3
	k-NN	38.1	96.4	76.6
Continuous	CMI	69.6	95.0	67.5
	MINI	63.3	104.5	82.7
	k-NN	77.9	162.3	154.3

Table 8
Wilcoxon test of imputation accuracy differences between MINI and *k*-NN and CMI under MAR and MCAR

Missingness mechanism	Method pair	*p* of MMRE	*p* of ϵ
MAR	MINI vs. CMI	<0.0001	<0.0001
	MINI vs. *k*-NN	<0.0001	<0.0001
MCAR	MINI vs. CMI	<0.0001	<0.0001
	MINI vs. *k*-NN	<0.0001	<0.0001

Table 9
Wilcoxon test of imputation accuracy differences between MINI and *k*-NN and CMI with ISBSG100 and ISBSG50

Data set size	Method pair	*p* of MMRE	*p* of ϵ
50	MINI vs. CMI	<0.0001	<0.0001
	MINI vs. *k*-NN	<0.0001	<0.0001
100	MINI vs. CMI	<0.0001	<0.0001
	MINI vs. *k*-NN	<0.0001	<0.0001

Table 10
Wilcoxon test of imputation accuracy differences between MINI and *k*-NN and CMI

Method pair	*p* of MMRE	*p* of ϵ
MINI vs. CMI	<0.0001	<0.0001
MINI vs. *k*-NN	<0.0001	<0.0001

Table 11
The average impact of imputation upon feature variance

Method	Variance	Percentage change
CMI	341.72	−14.35
k-NN	363.28	−8.95
MINI	377.71	−5.33
Original variance	398.99	0

pared with 14% for the class mean imputation (which of course is closely related to mean imputation).

Consequently, these results suggest that, overall, the MINI method may be more suitable for imputing missing data in small data sets than the *k*-NN or CMI methods.

5. Conclusions

We have presented a new, class mean imputation based, *k*-NN hot deck imputation method called MINI, for the imputation of missing values for small software project data sets. We compared MINI with two other widely used imputation methods, CMI and *k*-NN using small real world software project data sets of 50 and 100 cases respectively. In the experiments we set four different missing patterns: missing data in one continuous feature, missing data in two continuous features, missing data in one nominal feature, and missing data in two nominal features. We also examined four different levels of missing data (10%, 15%, 20% and 30%), using both MCAR and MAR missingness mechanisms. Then we compared the proposed MINI method with CMI and *k*-NN for a total of 320 data sets.

Our results show that the MINI method is superior to either the CMI or the *k*-NN method, in terms of imputation error and also in terms of the least impact upon feature variance. The results of the Wilcoxon test confirm the conclusion, and suggest that the proposed approach might be considered by data analysts when imputing missing data values in small software project data sets. Clearly, there is the question as to what extent these results might generalise? We have focused upon a specific domain, i.e. software engineering project data sets. These tend to contain few cases with many features that exhibit complex interactions. We conducted our experiments using data drawn from a real world data set collected by the ISBSG. Nevertheless, it would be useful for other research groups using different data sets to endeavour to replicate our findings.

Acknowledgements

The authors would like to thank the International Software Benchmarking Standards Group (ISBSG) for providing the data set for this analysis. The authors also would like to thank the anonymous reviews and the editor for their insightful and helpful comments. This work was partly supported by the UK, Engineering and Physical Sciences Research Council under grant GR/S55347.

References

Aggarwal, C.C., Parthasarathy, S., 2001. Mining massively incomplete data sets by conceptual reconstruction. In: Proceedings of the Seventh ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 227–232, San Francisco, California, USA.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Wadsworth International Group, Belmont, CA, 1984.

Cartwright, M., Shepperd, M., Song, Q., 2003. Dealing with Missing Software Project Data, Proc. 9th IEEE International Software Metrics Symposium. IEEE Computer Society, Sydney, Australia.

Conte, S., Dunsmore, H., Shen, V.Y., 1986. Software Engineering Metrics and Models. Benjamin Cummings, Menlo Park, CA.

Fix, E., Hodges, J.L., 1952. Discriminatory analysis: nonparametric discrimination: small sample performance. Technical Report Project 21-49-004, Report Number 11, USAF School of Aviation Medicine, Randolf Field, Texas.

Friedman, N., 1998. The Bayesian structural EM algorithm. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers, San Francisco, CA, pp. 129–138.

Grzymala-Busse, J.W., Hu, M., 2000. A comparison of several approaches to missing attribute values in data mining. In: RSCTC'2000, pp. 340–347.

Haitovsky, Y., 1968. Missing data in regression analysis. Journal of the Royal Statistical Society B30, 67–81.

Han, J., Chen, M., Yu, P., 1996. Data mining: an overview from database perspective. IEEE Transactions on Knowledge and Data Engineering 8 (6), 833–866.

Jain, A.K., Zongker, D., 1997. Feature selection: evaluation, application and small sample performance. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (2), 153–158.

Jönsson, P., Wohlin, C., 2004. An evaluation of *k*-nearest neighbour imputation using likert data. In: Proceedings of the 10th IEEE International Software Metrics Symposium. IEEE Computer Society, Los Alamitos, California, USA, pp. 108–118.

Joreskog, K.G., Sorbom, D., 1993. LISREL 8 User's Reference Guide. Scientific Software Int'l Inc., Chicago.

Kadoda, G., Cartwright, M., Shepperd, M.J., 2001. Issues on the effective use of CBR technology for software project prediction. In: 4th Intl. Conf. on Case Based Reasoning, Vancouver, 2001.

Kim, J.-O., Curry, J., 1977. The treatment of missing data in multivariate analysis. Sociological Methods and Research 6 (2), 215–240.

Kirsopp, C., Shepperd, M.J., Hart, J., 2002. Search Heuristics, Cased-Based Reasoning and Software Project Effort Prediction, GECCO 2002: Genetic and Evolutionary Computation Conf., New York, AAAI.

Kohavi, R., John, G., 1997. Wrappers for feature subset selection. Artificial Intelligence 97 (1–2), 273–324.

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

Little, R.J.A., Rubin, D.B., 2002. Statistical Analysis with Missing Data, Second ed. John Wiley & Sons, New York.

Little, R.C., Roderick, J.A., Schenker, N., 1995. Missing data. In: Arminger, G., Clogg, C., Sobel, M. (Eds.), Handbook of Statistical Modeling for the Social and Behavioral Sciences. Plenum, New York, pp. 39–75.

Moløkken, K., Jørgensen, M., 2003. A review of surveys on software effort estimation. In: 2nd International Symposium on Empirical Software Engineering. IEEE Computer Society, Rome, pp. 223–230.

Myrtveit, I., Stensrud, E., Olsson, U.H., 2001. Analyzing data sets with missing data: an empirical evaluation of imputation methods and likelihood-based methods. IEEE Transactions on Software Engineering 27 (11), 999–1013.

Provost, F., Kolluri, V., 1999. A survey of methods for scaling-up inductive algorithms. Data Mining and Knowledge Discovery 2, 131–169.

Quinlan, J.R., 1993. C4.5 Programs for Machine Learning, CA, Morgan Kaufmann.

Roth, P., 1994. Missing data: a conceptual review for applied psychologists. Personnel Psychology 47, 537–560.

Rubin, D.B., 1976. Inference and missing data. Biometrika 63, 581–592.

Rubin, D.B., 1987. Multiple Imputation. John Wiley & Sons, New York.

Schafer, J.L., Olsen, M.K., 1998. Multiple imputation for multivariate missing-data problems: a data analyst's perspective. Multivariate Behavioral Research 35, 545–571.

Schuurmans, D., Greiner, R., 1997. Learning to Classify Incomplete Examples. In: Computational Learning Theory and Natural Learning Systems, Making Learning Systems Practical, vol. IV. MIT Press, pp. 87–105 (chapter 6).

Shirani, S., Kossentini, F., Ward, R., 2000. Reconstruction of baseline JPEG Coded Images in error prone environments. IEEE Transactions on Image Processing 9 (7), 1292–1298.

Song, Q., Shepperd, M., Cartwright, M., 2005. A short note on safest default missingness mechanism assumptions. Empirical Software Engineering: An International Journal 10 (2), 235–243.

Strike, K., El Emam, K., Madhavji, N., 2001. Software cost estimation with incomplete data. IEEE Transactions on Software Engineering 27 (10), 890–908.

Tabachnick, B.G., Fidell, L.S., 2001. Using Multivariate Statistics, fourth ed. Allyn & Bacon, Needham Heights, MA.

Troyanskaya, O., Cantor, M., Sherlock, G., et al., 2001. Missing value estimation methods for DNA microarrays. Bioinformatics 17, 520–525.