University of Sunderland

A STOCHASTIC METHOD FOR ESTIMATING IMPUTATION ACCURACY

Norman Solomon

School of Computing and Technology
University of Sunderland

A thesis submitted in partial fulfilment of the requirements of the University of Sunderland for the degree of Master of Philosophy

The research programme was carried out in collaboration with Trends Business Research of Newcastle-Upon-Tyne

May 2008

Abstract

This thesis describes a novel imputation evaluation method and shows how this method can be used to estimate the accuracy of the imputed values generated by any imputation technique. This is achieved by using an iterative stochastic procedure to repeatedly measure how accurately a set of randomly deleted values are "put back" by the imputation process. The proposed approach builds on the ideas underpinning uncertainty estimation methods, but differs from them in that it estimates the *accuracy* of the imputed values, rather than estimating the *uncertainty* inherent within those values. In addition, a procedure for comparing the accuracy of the imputed values in different data segments has been built into the proposed method, but uncertainty estimation methods do not include such procedures.

This proposed method is implemented as a software application. This application is used to estimate the accuracy of the imputed values generated by the expectation-maximisation (EM) and nearest neighbour (NN) imputation algorithms. These algorithms are implemented alongside the method, with particular attention being paid to the use of implementation techniques which decrease algorithm execution times, so as to support the computationally intensive nature of the method. A novel NN imputation algorithm is developed and the experimental evaluation of this algorithm shows that it can be used to decrease the execution time of the NN imputation process for both simulated and real datasets. The execution time of the new NN algorithm was found to steadily decrease as the proportion of missing values in the dataset was increased.

The method is experimentally evaluated and the results show that the proposed approach produces reliable and valid estimates of imputation accuracy when it is used to compare the accuracy of the imputed values generated by the EM and NN imputation algorithms. Finally, a case study is presented which shows how the method has been applied in practice, including a detailed description of the experiments that were performed in order to find the most accurate methods of imputing the missing values in the case study dataset. A comprehensive set of experimental results is given, the associated imputation accuracy statistics are analysed and the feasibility of imputing the missing case study data is assessed.

Acknowledgements

Thanks to my supervisors, Dr. Giles Oatley and Dr. Ken McGarry, for their help and support. Especially for their practical and positive suggestions for improving the structure and content of this thesis.

I would also like to thank everyone at Trends Business Research (the collaborating company) for the help and support they provided throughout the entire project lifecycle. In particular, for helping to define the project objectives and for providing the case study dataset described in chapter six.

Thanks to the UK Engineering and Physical Sciences Research Council who funded the work under the Cooperative Awards in Science and Engineering scheme.

Finally, thanks to the late Richard P. Feynman whose inspirational writings kept me going when progress was slow. Especially for the book *"The Pleasure of Finding Things Out"*, Feynman, (2001).

Contents

List of Figures

List of Tables

Chapter One

Introduction

1. Introduction

Non-response in surveys is perhaps the most prevalent missing data problem (Rubin, 1996a) and it is often found that several of the variables in a survey dataset - such as a set of questionnaires - have some missing values (Allison, 2001). Many statistical software packages simply omit all cases that have one or more missing values (referred to as “case deletion”) when computing the statistics that describe the dataset. This can bias the results of the data analysis process and cause misleading conclusions to be drawn, as Schafer (1997) points out;

> *“When the incomplete cases comprise only a small fraction of all cases (say, five percent or less) then case deletion may be a perfectly reasonable solution to the missing data problem. In multivariate settings where missing values occur on more than one variable, however, the incomplete cases are often a substantial portion of the entire dataset. If so, deleting them may be inefficient, causing large amounts of information to be discarded. Moreover, omitting them from the analysis will tend to introduce bias, to the extent that the incompletely observed cases differ systematically from the completely observed ones”*

Imputation methods attempt to solve the problem of missing data by replacing missing values with plausible estimates, which avoids the problems described above. Essentially, all imputation methods have the same basic objective. That is, they try to make the best possible use of the information content (the patterns and so on) within the *known* values in a particular dataset, to generate the best possible estimates for the *missing* values in that dataset.

Rubin (1996a) points out that the primary (usually achievable) objective of imputation is to ensure that data analysis tools *“can be applied to any dataset with missing values using the same command structure and output standards as if there were no missing data”*, and that a further, desirable (but not always achievable) objective is to allow statistically valid inferences to be drawn when analysing imputed datasets.

Chambers (2001) lists five *“desirable properties for an imputation procedure”* - i.e. a set of criteria that can be used to evaluate the performance of any imputation method, as follows;

1. ***Predictive Accuracy*** - The imputation procedure should maximise preservation of true values. That is, it should result in imputed values that are "close" as possible to the true values.

2. ***Ranking Accuracy*** - The imputation procedure should maximise preservation of order in the imputed values. That is, it should result in ordering relationships between imputed values that are the same (or very similar) to those that hold in the true values.

3. ***Distributional Accuracy*** **-** The imputation procedure should preserve the distribution of the true data values. That is, marginal and higher order distributions of the imputed data values should be essentially the same as the corresponding distributions of the true values.

4. ***Estimation Accuracy*** - The imputation procedure should reproduce the lower order moments of the distributions of the true values. In particular, it should lead to unbiased and efficient inferences for parameters of the distribution of the true values (given that these true values are unavailable).

5. ***Imputation Plausibility*** **-** The imputation procedure should lead to imputed values that are plausible. In particular, they should be acceptable values as far as the editing procedure is concerned.

The list is taken directly from Chambers (2001), who explains that *"The list itself is ranked from properties that are hardest to achieve to those that are easiest"*. This dissertation describes a novel method for estimating the *"predictive accuracy"* of imputation techniques, and as such it focuses on evaluating the performance of imputation methods using the first criteria given above.

It is important to emphasise at the outset that the *"true values"* referred to above are the actual, real values of the missing data items, which are by definition, unknown. Therefore, it is impossible to *prove* that any imputation procedure has imputed values accurately, since the true values can never be compared with the imputed values. Consequently, *general purpose methods* for evaluating the accuracy of the imputed values generated by imputation procedures have received very little attention in the literature.

However, the accuracy of the imputed values generated by imputation procedures *can be estimated* and this dissertation describes the development of a *novel, general purpose* imputation evaluation method that can be used to achieve this goal.

1.1 Description of the Work Undertaken

This section explains why the work was undertaken and describes how the collaboration with the partner company led to the formulation of the project objectives.

1.1.1 Motivation for the Work

The work was funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under the Cooperative Awards in Science and Engineering (CASE) scheme. This scheme allows students to collaborate with commercial organisations, so that the results of the work will benefit the student, the academic institution to which that student belongs and the commercial organisation involved. In this case the work was undertaken in an attempt to solve the collaborating company's missing data problem, as described below.

The collaborating company were Trends Business Research (TBR), who are based in Newcastle-upon-Tyne. TBR offer business and economic research consultancy to clients in the private and public sectors at local, regional, national and international levels. TBR's activities are based upon the collection, enrichment, analysis and reporting of information describing UK business organisations, so as to further the strategic aims of their clients. This information is stored in the Trends Central Database (TCD), which describes approximately 1.48 million UK business organisations (referred to as "Firms"), ranging from sole traders to conglomerates. The TCD tables contain descriptions of each Firm, including its financial situation, number of employees, business activities and geographical location. This allows detailed statistical analysis of the data to be performed at various geographical levels - such as postcode regions or political areas, such as constituencies and wards.

However, the TCD variables that describe each Firm's financial situation all have missing values - which constantly hampers the data analysis described above.

This problem is exacerbated by the following factors. Firstly, the proportions of missing values for each financial variable are unusually large - i.e. they range from 27 to 96 percent, depending on the variable. Secondly, 71 percent of the Firms described in the TCD have no known financial figures whatsoever (all of the values are missing). Thirdly, the missingness pattern structure (see the following section for a definition of missingness patterns) for the financial variables is extremely unbalanced. Finally, the known values for each of the financial variables all contain small proportions of *very extreme* outlier values.

However, the missing data problem is somewhat alleviated by the fact that larger Firms (those with more employees) generally have smaller proportions of missing financial data - i.e. the probability of a Firm's financial figures being missing decreases as the Firm's size increases. And some of the variables that describe each Firm are fully observed, such as the variables that specify each Firm's geographical location. A more detailed description of the TCD dataset and TBR's missing data problem is given in chapter six.

1.1.2 Objectives

1. To discover whether imputation of the missing values in the collaborating company's database was feasible, given the overall poor quality of the dataset. The criterion used to assess the feasibility of the imputation process was to be the predictive accuracy of the imputed values.

2. To devise a new method for estimating the predictive accuracy of the imputed values generated by any imputation technique. The method should build on the ideas underpinning existing imputation evaluation methods.

3. To implement the method in the form of a software application that will allow users to estimate the predictive power of any imputation technique.

4. To use the software application to experimentally evaluate the reliability and the validity of the new method and to achieve the first objective.

1.2 Description of the Proposed Imputation Evaluation Method

This section gives an overview of the imputation evaluation method devised by the author (a more detailed description is given in chapter four). Section 1.2.1 gives an informal description of the method. Section 1.2.2 explains how the method can be used to estimate the predictive *power* of imputation techniques. Section 1.2.3 gives a functional overview of the method with reference to the contents of the rest of the thesis.

1.2.1 Informal Description of the Method

The method can be used to estimate the predictive accuracy of the imputed values for any variable in the dataset (only one variable can be evaluated each time the method is employed), where the required variable is chosen by the user of the software that implements the method. However, the evaluation process can be repeated for all of the variables in the dataset, if this is required. The functional steps of the method are summarised below.

1. A small proportion (perhaps up to 5%) of the known values are deleted *at random* from within the variable to be evaluated (which will already have some missing values).

2. Deleted values are recorded just before they are deleted, and a measure of how accurately they have been "put back" is taken when the imputation process is complete.

3. Steps 1 and 2 are repeated several times and the accuracy statistics computed at step 2 are stored after each repetition.

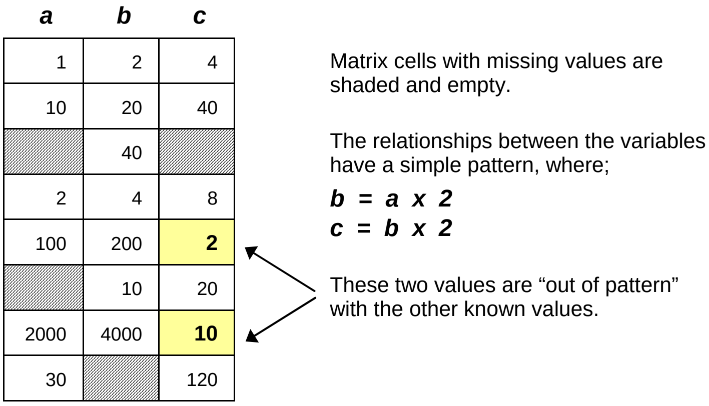
4. The stored statistics are aggregated so that the estimates of imputation accuracy produced will be more statistically reliable.

This method is described as "stochastic" in this thesis because the known values are *randomly deleted* at step 1. The repetition of steps 1 and 2 forms an essential part of the method, because this process will produce more statistically reliable estimates of imputation accuracy. The reason why this is true can be explained using the following example. Suppose an unbiased coin was thrown twice and fell on heads both times. A maximum-likelihood based statistical analysis of this small sample (see chapter two for a discussion of maximum likelihood theory) would *estimate* the probability of the coin falling on a head as 100%. However, if the coin was thrown ten times, then the *estimate* of the probability of a head being thrown should move closer to the true value of 50%. And as the number of throws is increased the estimate of the probability of a head being thrown should move closer and closer to the true value. This principle applies to any stochastic process which attempts to estimate unknown quantities, such as proposed imputation evaluation method

1.2.2 Estimating the Predictive Power of Imputation Techniques

The process used to estimate the predictive *accuracy* of imputed values is described in the previous section. This process also estimates the predictive *power* of the imputation technique used to generate the imputed values, which in turn allows the feasibility of using that technique to be assessed.

The proposed method also allows the predictive power of candidate imputation techniques to be *compared*, so that the technique that generates the most accurately imputed values can be chosen (as described in chapter five). Estimating the predictive power of an imputation technique is equivalent to measuring how well that technique has utilised the patterns within the known values in the dataset. This idea is fundamental to the proposed approach and it is discussed further below.



a	***b***	***c***
1	2	4
10	20	40
▨	40	▨
2	4	8
100	200	**2**
▨	10	20
2000	4000	**10**
30	▨	120

Matrix cells with missing values are shaded and empty.

The relationships between the variables have a simple pattern, where;

b = a x 2
c = b x 2

These two values are "out of pattern" with the other known values.

Fig 1.1 – Numeric patterns in a data matrix that has some missing values

Consider the values in the data matrix shown in Fig 1.1. The relationships between the variables (the values in the matrix columns) ***a, b*** and ***c*** are very strong, with the exception of the two "out of pattern" values (any other relationships found within the values in the Fig 1.1 matrix should be ignored here, since the idea of this section is to illustrate how data patterns would be utilised by regression based imputation procedures).

Any regression based imputation procedure should produce accurate estimates for the missing values, because the patterns within a large majority of the known values are so strong. The important point to note is that ***the only information available to any imputation procedure is contained within the patterns that exist among the known values in the dataset***. However, these patterns will degrade and weaken as the proportion of "out of pattern" values increases, and ultimately the imputation process will become infeasible - i.e. this will occur when the proportion of "out of pattern" values exceeds a certain critical value.

Consider the following two extreme theoretical examples. **(1)** If every matrix cell with a known value contained the same value, then imputation would be easy to achieve and there would be very little uncertainty within the imputed values. **(2)** If every matrix cell with a known value contained a randomly generated integer in the range one to a billion, then no patterns would exist within the known values and imputation would be completely infeasible. However, in practice the patterns within most datasets will fall somewhere towards the centre of these two theoretical extremes.

Using the predictive power of the patterns in the dataset to assess imputation feasibility

The proposed method assesses the feasibility of employing a particular imputation technique by measuring how well that technique utilises the predictive power of the patterns within the known values in the dataset. That is, if the technique being evaluated by the method has been devised to utilise the type of patterns that *actually do exist*, then that technique should generate reasonably accurate estimates for the missing values.

The key point to note is that different types of imputation technique will utilise different types of patterns when generating estimates for missing values. For example, regression based techniques will utilise the relationships *between variables* to generate regression equations, whereas nearest neighbour techniques will utilise the relationships *between observations* to find similar matrix rows (using distance functions). However, the proposed method will work equally well regardless of the types of patterns utilised by the imputation techniques it evaluates - ***i.e. one of the strengths of the proposed method is that it does not need to "know" how the imputation technique it is evaluating actually works***. The following example explains the reasoning underpinning this general purpose approach, with reference to the functional description of the method given in the previous section.

If imputation method X *repeatedly* "puts back" the randomly deleted values inaccurately, then the deleted values must *not have* fallen into the patterns that imputation method X used to generate estimates for the missing values. Consequently, the patterns within the known values in the dataset must not be strong enough to support imputation method X (but these patterns might be much better utilised by imputation method Y or Z, depending on how these methods work). Therefore, the feasibility of employing imputation method X to impute the missing values is questionable. This approach can be used to evaluate and compare any imputation method. The consequences of this approach are discussed in considerable detail in the more suitable context of section 5.1.4.

1.2.3 Functional Overview of the Method: Structure of the Thesis

The diagram below gives an overview of the sequence of steps that are performed whenever the proposed method is employed. The overall process can be used to estimate the accuracy of the imputed values generated by any imputation technique.

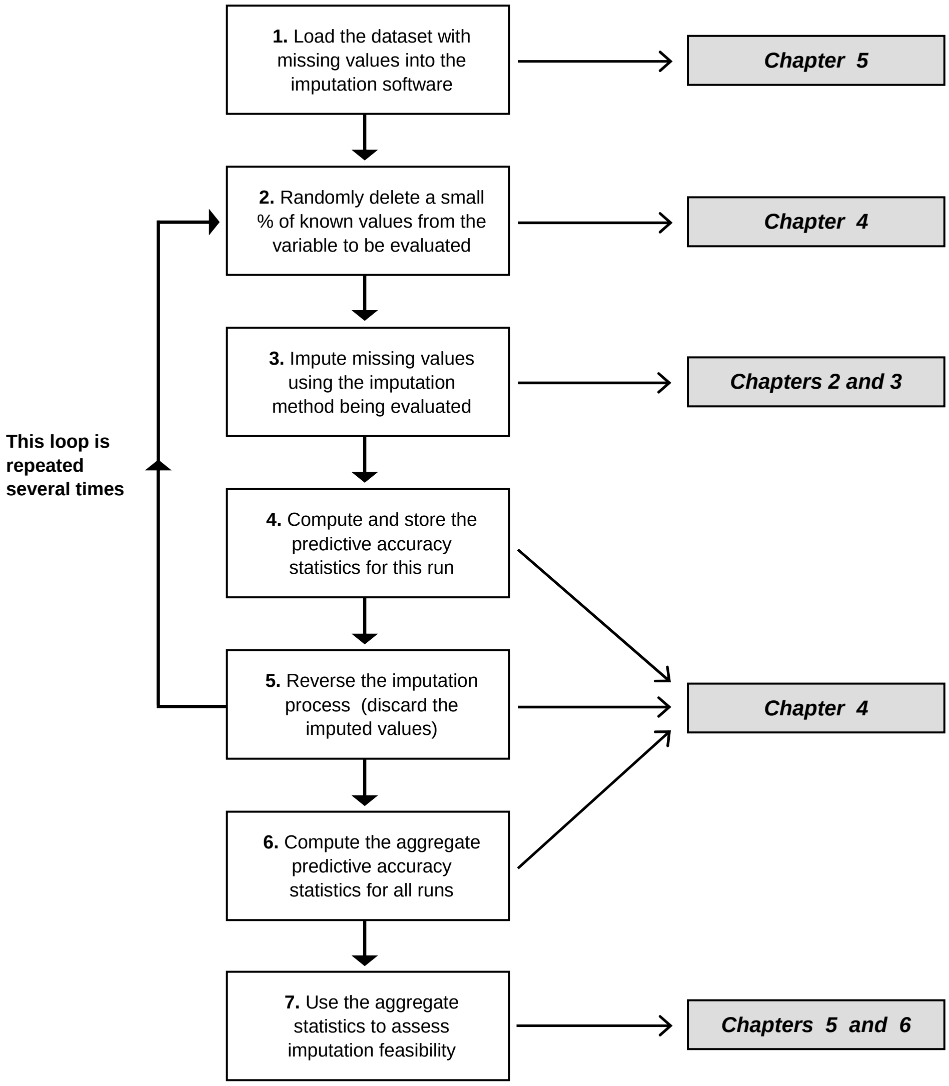

1. Load the dataset with missing values into the imputation software
Chapter 5
2. Randomly delete a small % of known values from the variable to be evaluated
Chapter 4
3. Impute missing values using the imputation method being evaluated
Chapters 2 and 3
This loop is repeated several times
4. Compute and store the predictive accuracy statistics for this run
5. Reverse the imputation process (discard the imputed values)
Chapter 4
6. Compute the aggregate predictive accuracy statistics for all runs
7. Use the aggregate statistics to assess imputation feasibility
Chapters 5 and 6

Fig 1.2 – Estimating imputation accuracy : Structure of the thesis

The sequence of steps shown in Fig. 1.2 reflects the structure of the thesis. The following points are important in this respect;

- ***Chapters 2 and 3*** discuss the theory underpinning the imputation techniques that have been implemented as part of the software that implements the proposed method. These techniques have been implemented alongside the method in the form of an integrated software application. This was essential, because it would impractical to implement the repetitive process shown in steps 2 to 5 of Fig. 1.2 in any other way.

- ***Chapter 4*** gives a formal explanation of how the proposed method can be used to estimate the predictive accuracy of the imputed values generated by *any imputation technique* (e.g. the techniques described in chapters 2 and 3, among others). The ideas presented in this chapter form the principal contribution made in this dissertation.

- ***Chapters 5 and 6*** Chapter 5 explains how the reliability and the validity of the proposed method was experimentally evaluated. Chapter 6 assesses the feasibility of imputing the missing values in the collaborating company's dataset.

1.3 Missing Data Mechanisms

An understanding of the "mechanisms" that lead to missing data is an essential prerequisite for an understanding of missing data problems and these ideas are referred to throughout the thesis. This section defines the terminology used when discussing missing data mechanisms, explains the theory underpinning these concepts and describes how this theory can be applied in practice.

Many imputation methods will find the most reliable estimates for missing values when these values are "missing at random" (MAR), rather than being "missing completely at random" (MCAR). These missing data patterns are referred to as "missing data mechanisms" within the literature that discusses missing data problems. Unfortunately, the nomenclature surrounding missing data mechanisms can be somewhat misleading for the uninitiated. For example, when data is said to be missing at random this means that there is some clearly identifiable *patterns* of "missingness" within the dataset. In other words, the probability of missing values occurring for a particular variable depends on the values of another variable (or on the values of a particular combination of the other variables in the dataset). In fact, the definitions of the MAR and MCAR assumptions have been the cause of some confusion even among statisticians, as Allison (2001) points out,

> *"More generally, researchers have often claimed or assumed that their data are "missing at random" without a clear understanding of what this means. Even statisticians were once vague or equivocal about this notion. However, Rubin (1976) put things on a solid foundation by rigorously defining different assumptions that might plausibly be made about missing data mechanisms"*

Section 1.3.1 gives a formal summary of Rubin's (1976) definitions - as referred to by Allison, above - and clarifies the key concept of "ignorability" for practical purposes. Section 1.3.2 attempts to clarify the concepts of MAR and MCAR using a simple illustrative dataset. Section 1.3.3 describes how missing data mechanisms are defined in practice and explains why it is impossible to prove the MAR and MCAR assumptions. Section 1.3.4 explains why it is essential to assess the feasibility of the imputation process for data that is MAR.

1.3.1 Formal Definitions of Missing Data Mechanisms

The statistical definitions of the MAR and MCAR assumptions were rigorously defined by Rubin (1976). This section gives a summary of Rubin's definitions using a simple illustrative dataset. It can be important to refer to these definitions when discussing missing data mechanisms, to avoid misunderstandings.

$$Y = \begin{array}{c} \begin{array}{ccc} \underline{\textbf{\textit{Sex}}} & \underline{\textbf{\textit{Income}}} & \underline{\textbf{\textit{Age}}} \end{array} \\ \left[\begin{array}{ccc} M & 20000 & ? \\ M & ? & 22 \\ M & ? & ? \\ ? & 38000 & 40 \\ F & 25000 & 38 \\ F & ? & 64 \end{array}\right] \end{array} \qquad M = \begin{array}{c} \begin{array}{ccc} \underline{\textbf{\textit{Sex}}} & \underline{\textbf{\textit{Income}}} & \underline{\textbf{\textit{Age}}} \end{array} \\ \left[\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{array}\right] \end{array}$$

Consider the example given by the matrices above. Let Y be defined as a data matrix with one or more missing values in one or more of its columns, with elements represented by Y_{ij} where the ? symbol represents a missing value. Let M be defined as a matrix of corresponding binary indicators with elements represented by M_{ij} , such that;

$$M_{ij} = 1 \quad \text{if} \quad Y_{ij} \quad \text{is present in } Y$$

$$M_{ij} = 0 \quad \text{if} \quad Y_{ij} \quad \text{is missing in } Y$$

Further, let Y_{obs} represent the subset of all values present in *Y*, and let Y_{mis} represent the subset of all values missing in *Y*, such that $Y = (Y_{obs}, Y_{mis})$ represents the entire dataset in *Y*. Finally, let ϕ represent a set of unknown parameters which describe the distribution in *M*, then;

For MAR $\quad P(M \mid Y_{obs}, Y_{mis}, \phi) = P(M \mid Y_{obs}, \phi) \quad$ for all Y_{mis}, ϕ

For MCAR $\quad P(M \mid Y_{obs}, Y_{mis}, \phi) = P(M \mid \phi) \quad$ for all Y, ϕ

<u>MAR and ignorability are equivalent conditions in practice</u>

The missing data mechanism is said to be *"ignorable"* - as defined by Rubin (1987) - when the data in *Y* is MAR, and when ϕ (the *M* distribution parameter) and θ (the *Y* distribution parameter) are unrelated. However, in practice, it is hard to imagine a situation where ϕ and θ can be related, since knowing ϕ is very unlikely to tell us anything about θ, and vice-versa. In addition, even in the very rare cases where ϕ and θ are related imputation methods should produce the same results. Essentially, this means that we can treat MAR and ignorability as equivalent conditions and that we do not need to create or process the M matrix and ϕ when executing imputation algorithms for data that is MAR (Allison, 2001).

1.3.2 Informal Definitions of the MAR and MCAR Concepts

Generally, when considering missing data mechanisms we are interested in finding the probability that a value will be missing, rather than attempting to impute it. This section discusses missing data mechanisms informally, using the illustrative data matrix shown below - where the missing values are represented by ? symbols.

$$\begin{array}{cc} & \begin{array}{ccc} \underline{\textit{Sex}} & \underline{\textit{Income}} & \underline{\textit{Age}} \end{array} \\ Y \;= & \left[\begin{array}{ccc} M & 20000 & ? \\ M & ? & 22 \\ M & ? & ? \\ ? & 38000 & 40 \\ F & 25000 & 38 \\ F & ? & 64 \end{array}\right] \end{array}$$

Fig 1.3 – Data matrix illustrating the MAR and MCAR missing data mechanisms

The missing values in a particular column are said to be MAR if the probability that they are missing is unrelated to their value, after controlling for values in the other columns.

For example, suppose that 50% of the males described by the data in the Fig. 1.3 matrix failed to report their income, but only 10% of the females failed to report their income. We could then say that the probability of a person's income being missing depended on their sex. In this case the MAR condition would be satisfied if the probability of missing income values occurring in both categories (male and female) was unrelated to the values of the income variables within those categories. Note that, by the MAR definition, the probability of a person's income being missing could also depend on their age, or on any combination of the set of variables used to describe a person.

The missing values in a particular column are said to be MCAR if the probability that they are missing is unrelated to their value or to the values in any other column.

For example, the MCAR condition would be satisfied if the probability that the values in the income column (variable) were missing did not depend on the values in any column in the matrix, including the income column itself. When the MCAR condition is satisfied for every column in the matrix then the subset of matrix rows (observations) that have a *complete* set of known values can be regarded as being a random sample of all rows in the matrix. Note that the MCAR condition allows the missingness patterns in two or more columns to be related. For example, if everyone who failed to report their age also failed to report their income, then the MCAR condition could still be satisfied for the age and income variables.

1.3.3 Considering Missing Data Mechanisms in Practice

When considering alternative solutions to missing data problems it is important to realise that making an incorrect assumption about the missing data mechanism for any particular variable could devalue the results of the data analysis process. This could result in misleading conclusions being drawn, which would exacerbate the missing data problem.

> *"It is clear that if the imputation model is seriously flawed in terms of capturing the missing data mechanism, then so will any analysis based on such imputations. This problem can be avoided by carefully investigating each specific application, and by making best use of knowledge and data."* (Barnard and Meng, 1999)

Knowledge of the missing value dataset defines missing data mechanisms

The imputation of missing data is almost invariably a knowledge intensive process, where each missing data problem has its own unique characteristics. Consequently, the knowledge possessed by the data analyst concerning the properties of the dataset with missing values is the most important tool available when defining missing data mechanisms. For example, the dataset with missing values may have been created using the data taken from a set of returned questionnaires designed as part of a survey. In this case some questionnaires could contain missing data because some respondents failed to answer some of the questions put to them. The knowledge possessed by the designers of the questionnaire is paramount in this case, since they will understand the relationships between the variables, and consequently they will be able to define the missing data mechanism for each variable. For example, if it was *known* that respondents with certain characteristics (such as age or sex) were more likely to answer certain questions, then missing answers to those questions would be known to be MAR. However, if all respondents were considered equally likely to answer every question, then missing answers to *all* questions would be MCAR, (Barnard and Meng, 1999)

When detailed knowledge of the relationships between the variables in the missing value dataset is unavailable it can be extremely difficult to define the mechanisms that lead to missing data using diagnostic procedures alone (Graham et al, 1994). However, regression based diagnostics can sometimes be used to detect non-MCAR patterns in situations where a good linear regression model can be fitted to the data (Simonoff, 1988; Toutenburg and Fieger, 2000)

MAR and MCAR are assumptions which cannot be proved or disproved

It is very important to emphasise that the MAR and MCAR mechanisms are, by definition, *assumptions* which provide a conceptual framework for the analysis of missing data problems and for assessing the applicability of any particular imputation technique, and that the idea of proving or disproving the correctness of these assumptions is not the point. This is perhaps the most difficult idea to comprehend for those who are new to the study of missing data mechanisms, but it is *the key idea* underpinning this area of study. In fact, the MAR or MCAR assumption for a particular variable can never be proved - for the following reason. To prove both the MAR and the MCAR assumptions we need to prove that the probability of missing values occurring for a particular variable does not depend on the values of that variable (see Little and Rubin, 2002, among others). However, we can never prove that this supposition is either true or false, since we cannot compare the pattern in the subset of missing (unknown) values with the pattern in the subset of observed (known) values.

1.3.4 Solving the Missing Data Problem: Deletion or Imputation?

Consideration of the missing data mechanism is very important when deciding how to solve a particular missing data problem. Many statistical software applications offer the option of simply deleting *all rows* that have *any* missing values from the dataset (see for example, Nie et al, 1975). This approach is referred to as “listwise deletion” or “complete case analysis”. This can be a good solution when the missing data is MCAR and when the proportion of deleted rows is small (say up to 10%), because deleting the rows should not seriously bias the remaining data. However, when the values are MAR and the proportion of missing values is large, listwise deletion can seriously bias the remaining data, for the reasons explained below.

Consider the data matrix given in Fig. 1.3, above. Suppose that the matrix contained an equal number of male and female observations (matrix rows), and that 50% of males failed to report their income, but only 10% of females failed to report their income. In this case 30% of the rows in the data matrix would have missing values and would be deleted. But the deleted rows in the male category would represent 25% of the dataset, whereas the deleted rows in the female category would represent only 5% of the dataset. This would bias the remaining data and any subsequent analysis based on this biased data could produce misleading conclusions. In situations of this type imputation is clearly preferable to listwise deletion.

In many cases the solution to the missing data problem comes down to a choice between imputation or listwise deletion. The key question to ask when trying to make this choice is; Is the imputation process feasible? The proposed imputation evaluation method has been devised to answer this question - and it is argued that this method makes a useful contribution to imputation theory, because it can be used to assess the feasibility of imputing missing values in *any numeric multivariate dataset, using any imputation method.*

1.4 Summary of Thesis Chapters and Contribution

An overview of the structure of the thesis is given in Fig. 1.2, above. This section summarises the contents of the following chapters and explains how they contribute to imputation theory.

- ***Chapter 2*** Discusses the theory underpinning maximum likelihood based imputation and shows how this approach can be used to impute missing values in datasets with multivariate missingness patterns. The description of the author's implementation of the expectation-maximisation algorithm, and the experiments that evaluate its performance, contribute to the theory of maximum likelihood based imputation techniques.

- ***Chapter 3*** Explains the ideas underpinning the functionality of a novel, fast, nearest neighbour (NN) imputation algorithm and shows how these ideas can be used to reduce the execution time of the NN imputation process. A description of the experiments that evaluate the performance of the new NN algorithm is given. The ideas and the experimental results given in this chapter contribute to NN imputation theory.

- ***Chapter 4*** Describes the equations and processes which form the basis of the proposed imputation evaluation method and shows how this method can be used to evaluate any imputation technique. The proposed method is compared with the most similar methods found within the literature and it is shown that the proposed method builds on the ideas underpinning these methods, but differs from them in several important respects. The descriptions and explanations given in this chapter form the principal contribution to knowledge made by this thesis.

- ***Chapter 5*** Explains how the proposed method was experimentally evaluated and shows that this method produces reliable and valid estimates of imputation accuracy when it is used to evaluate the imputation techniques described in chapters 2 and 3. A description of the software that implements the method is given and an explanation of how this software can be used to compare the predictive power of candidate imputation methods is provided. This chapter extends the contribution made by chapter 4 by experimentally evaluating the method which forms the principal contribution.

- ***Chapter 6*** Explains how the proposed method was used to address the collaborating company's (TBR's) missing data problem. A description of the experiments that were performed in order to find the most accurate methods for imputing TBR's missing values is given. The experimental results are analysed and conclusions are drawn.

- ***Chapter 7*** Summarises the thesis, draws conclusions and describes how the work described in chapters one to six could be continued.

Chapter Two

Maximum Likelihood Imputation Via the EM Algorithm

2. Maximum Likelihood Imputation Via the EM Algorithm

The proposed imputation evaluation method is general purpose in nature, because it can be used to assess the feasibility of applying any imputation method to any numeric dataset. However, a "first cut" imputation method had to be implemented alongside the proposed evaluation method (in the form of an integrated software application) so that the proposed evaluation method would have an imputation method to evaluate. The imputation method that was chosen had to be general purpose in nature, in that it had to be capable of imputing missing values in any numeric dataset. Allison (2001) argues that there are only two methods of this type worth considering,

> *"Many alternative methods have been proposed ....Unfortunately, most of these methods have little value, and many of them are inferior to listwise deletion. That's the bad news. The good news is that statisticians have developed two novel approaches to handling missing data - maximum likelihood estimation and multiple imputation - that offer substantial improvements over listwise deletion."*

This conclusion seems to be generally accepted among statisticians. For example, Little and Rubin (2002) - who have produced *the* standard reference work on missing data methods - devote the major portion of their book to a discussion of the maximum likelihood estimation (MLE) and multiple imputation (MI) methods.

The imputation method selected for the first tests of the proposed imputation evaluation method was MLE via the expectation-maximisation (EM) algorithm (Dempster, Laird and Rubin, 1977). MLE via EM was chosen in preference to MI because the MI approach can already be used to evaluate the results of the imputation process - i.e. MI is, at least in part, an imputation evaluation method, although it was designed primarily as an imputation technique. In fact, the ideas underpinning the proposed method build on the ideas underpinning MI (the similarities and differences between MI and the proposed method are described in chapter four). The following sections discuss the theory underpinning MLE and the EM algorithm, and explain how these techniques can be implemented in practice.

- **Section 2.1** explains the fundamental concept underpinning MLE and describes how the MLE approach can be used to impute missing values in multivariate datasets.
- **Section 2.2** discusses the history and utility of the EM algorithm and gives an explanation of how EM can be implemented in practice, including a description of how EM can utilise the SWEEP operator to generate regression equations.
- **Section 2.3** describes how the author has implemented the EM algorithm as a software application and explains how the functionality of the new implementation was verified and how its performance was evaluated.

2.1 Maximum Likelihood Estimation

Section 2.1.1 gives an explanation of the fundamental concept underpinning maximum likelihood estimation (MLE), using a simple illustrative example. Section 2.1.2 explains how MLE can be applied for the imputation of missing values in incomplete multivariate datasets.

2.1.1 The Fundamental Concept Underpinning MLE

Suppose we have a biased coin, such that the probability of the coin falling on a head = 0.6, and the probability of it falling on a tail = 0.4. Then suppose that we throw this coin twice - there are four possible outcomes, as follows;

(1) Head, Head **(2)** Head, Tail **(3)** Tail, Head **(4)** Tail, Tail

Intuitively, we can see that, since the coin is biased towards falling on a head, outcome (1) is most probable, outcomes (2) and (3) are next (and equally) probable and outcome (4) is least probable. Stated more formally, we can say that the sequence of coin throws follows the Bernoulli distribution, such that the probability of any specific sequence of heads and tails occurring is given by;

$$P(Y \mid p) = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i} \qquad (2.1)$$

Where Y represents a set of n throws, y_i represents a particular throw within this set, and p gives the probability of a head occurring on any particular throw (which in this case = 0.6). For example, let 1 represent a head being thrown and 0 represent a tail being thrown. Then the probability of the occurrence of the set of throws represented by $Y = \{1, 1\}$ is given by;

$$P(1{,}1 \mid 0.6) = \prod_{i=1}^{2} 0.6^{y_i}(1-0.6)^{1-y_i} = 0.6 \times 0.6 = 0.36$$

Suppose that another biased coin is thrown 5 times and that it falls on a head 4 times, such that $Y = \{1, 1, 1, 1, 0\}$ but this time the probability of the coin falling on a head is *unknown*. It follows that calculations such as the one given above cannot be performed on this set of throws, since the value of p cannot be "plugged in" to the equation. One approach to solving this problem is to find an estimate for the value of p which maximises the likelihood that the set of throws $Y = \{1, 1, 1, 1, 0\}$ will occur. The process of finding the required value of p is referred to as *"maximum likelihood estimation"*. Stated more formally, we need to find the value of p which maximises the likelihood $L(p \mid Y)$, where;

$$L(p \mid Y) = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i} \qquad (2.2)$$

Notice that the right hand side of this equation is identical to the right hand side of equation (2.1) above. However, in equation (2.1) we need to find the probability that the observed data Y will occur for a given value of p. Whereas in equation (2.2) we need to find the value of p which maximises the likelihood that the observed data Y will occur. In this case we can find the required value of p by rearranging equation (2.2) so that p appears on the left hand side (the rearrangement is as given by Dunham, 2003);

$$L(p \mid Y) = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i} = p^{\sum_{i=1}^{n} y_i}(1-p)^{n-\sum_{i=1}^{n} y_i}$$

Taking the log of each side (referred to as the loglikelihood) gives

$$l(p) = \log L(p) = \sum_{i=1}^{n} y_i \log(p) + \left(n - \sum_{i=1}^{n} y_i\right)\log(1-p)$$

Then taking the derivative with respect to p gives

$$\frac{\partial l(p)}{\partial p} = \sum_{i=1}^{n} \frac{y_i}{p} - \frac{n - \sum_{i=1}^{n} y_i}{1-p}$$

Finally, setting the right hand side equal to zero, to find the $l\,(p)$ maximum value, gives

$$p = \frac{\sum_{i=1}^{n} y_i}{n}$$

applying this to the problem described by equation (2.2) gives

$$\hat{p} = \frac{\sum_{i=1}^{5} y_i}{5} = \frac{4}{5} = 0.8$$

Thus, we can say that the value of p that maximizes the likelihood of Y = {1, 1, 1, 1, 0} occurring is 0.8. It is important to emphasise that this value of p is an estimate based on single experiment with a very small sample, and that another such experiment involving 5 throws of the biased coin could easily result in a completely different sequence, such as Y = {0, 0, 0, 0, 1}. However, the same MLE approach could be applied to a much larger experimental sample (say 1000 throws) and the results would be much more reliable, but still not conclusive, since the amount of bias in the coin is unknown, and can only be estimated.

This simple example explains the central concept underpinning the MLE approach. This approach can be applied for the solution of much more complex problems than the one described above, such as the imputation of missing values in incomplete multivariate datasets, as described below.

2.1.2 Applying MLE to Incomplete Multivariate Datasets

Conceptually, the equation which gives the probability of the occurrence of any specific *complete* multivariate numeric dataset is similar to equation (2.1) as given in the previous section. However, the parameters that describe the multivariate distribution are much more complex, as shown in Fig. 2.1, below,

$$P(Y \mid \theta) = \prod_{i=1}^{n} f(y_i \mid \theta) \qquad (2.3)$$

$$\theta = (\mu, \Sigma) = \begin{bmatrix} -1 & \mu_1 & \mu_2 \cdots & \mu_p \\ \mu_1 & \Sigma_{11} & \Sigma_{12} \cdots & \Sigma_{1p} \\ \mu_2 & \Sigma_{12} & \Sigma_{22} \ldots & \Sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_p & \Sigma_{1p} & \Sigma_{2p} & \Sigma_{pp} \end{bmatrix}$$

Where $(\mu_1, \mu_2 \ldots\ldots \mu_p)$ contains the mean values for each Y column.

and $\Sigma = CSSCP / n$ is the Y matrix covariance matrix, where;

$CSSCP$ = Corrected Sums of Squares and Cross Products matrix,
as described in Tabachnick and Fidell (2000)

Fig 2.1 – The parameters that describe a complete multivariate numeric dataset

Where Y is a data matrix with all values present, $\theta = (\mu, \Sigma)$ represents the set of parameters which describe the distribution of the data in Y and $f(y_i \mid \theta)$ gives the probability of the occurrence of each row y_i where i is in the range 1 to n, and n gives the number of rows in Y. Thus, the probability of the occurrence of the complete Y dataset equals the product of the probabilities of the occurrence of each row in Y.

However, we are interested in applying the MLE process to *incomplete* multivariate datasets, such as the Y matrix shown below. To do this we must find an estimate for the value of $\theta = (\mu, \Sigma)$ which maximises the likelihood that the incomplete dataset in Y would occur, as we did in the example in the previous section. The first step is to specify the equation for the likelihood. Conceptually, this equation is similar to equation (2.2) given above - but some explanation of the terminology used is required before it can be presented.

$$Y = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The Y matrix has 6 rows and 5 columns

The *missing* values are represented by a value of 0

The *present* values are represented by a value of 1

Consider an incomplete data matrix such as the one shown above. Let Y_{obs} represent the subset of all values *present* in *Y*, and let Y_{mis} represent the subset of all values *missing* in *Y*, such that $Y = (Y_{obs}, Y_{mis})$ represents the entire dataset in *Y*. Now let,

$$Y_{obs} = (Y_{obs,1}, Y_{obs,2}, \ldots\ldots Y_{obs,i})$$

represent the subset Y_{obs}, where each element in $(Y_{obs,1}, Y_{obs,2}, \ldots\ldots Y_{obs,i})$ represents the set of observed values in the corresponding row in *Y*. Further, let $(\mu_{obs,i}, \Sigma_{obs,i})$ represent the mean and covariance matrix for a particular row *i* in *Y*, (rather than for all rows in *Y*, as given in equation (2.3) above). The loglikelihood based on Y_{obs} is then given by,

$$l(\mu, \Sigma \mid Y_{obs}) = \text{const} - \frac{1}{2}\sum_{i=1}^{n} \ln\left|\Sigma_{obs,i}\right| - \frac{1}{2}\sum_{i=1}^{n} (y_{obs,i} - \mu_{obs,i})^T \Sigma_{obs,i}^{-1} (y_{obs,i} - \mu_{obs,i})$$

As given by Little and Rubin (2002). To impute the missing values in *Y*, we must find an estimate for the value of (μ, Σ) which maximises the likelihood of the occurrence of Y_{obs}. When found, the required value of (μ, Σ) can be used to estimate the missing values in *Y*, thus completing the imputation process by producing a *Y* matrix with all missing values "filled in". However, the process of finding the value of (μ, Σ) which maximises $l(\mu, \Sigma \mid Y_{obs})$ is complex, and the method of simply rearranging the above equation, so that (μ, Σ) appears on the left hand side - as we did for equation (2.2) - cannot be applied in this case. In fact, this maximum likelihood estimation process is so complex that an iterative procedure, such as the EM algorithm, is regarded as the simplest way to proceed (Schafer, 1997; Little and Rubin, 2002). The following sections explain how this can be achieved.

2.2 The Expectation-Maximization Algorithm

Section 2.2.1 summarises the history of the expectation-maximisation (EM) algorithm and discusses its utility. Section 2.2.2 describes the type of dataset that can be processed by the EM algorithm. Section 2.2.3 explains how the EM algorithm can be used to impute missing values in multivariate numeric datasets. Section 2.2.4 explains how the SWEEP operator can be used to generate the regression equations used by the EM algorithm.

2.2.1 History and Utility of the EM Algorithm

The idea of solving complex statistical problems using an iterative MLE based approach goes back as least as far as McKendrick (1926), who discusses the idea with reference to a medical application. Hartley (1958), considers the general case and develops the theory extensively, explaining many of the key concepts underpinning the entire approach. Orchard and Woodbury (1972) go on to discuss the general applicability of the approach referring to it as *"the missing information principle"*. Beale and Little (1975) develop these ideas further for the *"multivariate normal population"* by creating an *"iterated form"* of the method proposed by Buck (1960). The phrase *"EM algorithm"* first appears in the seminal paper by Dempster, Laird and Rubin (1977), who describe the fundamental properties of the EM algorithm, and discuss its general applicability to the problem of *"computing maximum likelihood estimates from incomplete data"*. The concepts presented in that paper sparked a revolution in the analysis of incomplete multivariate data, allowing for the efficient imputation of multivariate missing data using an iterative MLE based approach.

The EM imputation method compares very favourably with other regression based imputation methods, such as the *"singular value decomposition method"* (SVD) proposed by Krzanowski (1988) and the *"principal component method"* (PCM) proposed by Dear (1959). This conclusion has been reached by several researchers. See, for example, the useful comparative analysis of the results produced by EM, SVD and PCM given in Bello (1995), and the discussion of the application of MLE via the EM algorithm given in the standard reference book on imputation methods produced by Little and Rubin (2002).

However, the EM approach can be used for more than just imputation. In fact, the range of problems that can be addressed using EM is wide and varied, including problems which do not usually involve the analysis of missing data, as discussed by Meng and Pellow (1992) and McLachlan and Krishnan (1996), and as succinctly summarised by Schafer (1997)

> *"The influence of EM has been far reaching, not merely as a computational technique, but as a paradigm for approaching difficult statistical problems. There are many statistical problems which, at first glance, may not appear to involve missing data, but which can be reformulated as missing data problems: mixture models, hierarchical or random effects models, experiments with unbalanced data and many more."*

2.2.2 EM Imputation Algorithm Data Assumptions

Consider a multivariate numeric data matrix Y with one or more missing values in one or more of its columns. The EM algorithm can be used to impute the missing values in Y using an iterative, regression based procedure - assuming that the dataset is suited to the EM process. This section describes the type of dataset that can be processed by the EM algorithm and discusses the problems that can arise when incorrect assumptions are made regarding the properties of the dataset to be processed.

The version of the EM algorithm described here will find the most reliable estimates for the missing values when all of the columns in the Y matrix are *perfectly* normally distributed, and when the missing values are all MAR, as described in chapter one. However, in practice, it is very unlikely that any data matrix will be perfectly normally distributed, and it will very rarely happen that the missing values in every column in the matrix are missing at random – so how should we proceed? Should we test the data to see if it is suitable to be processed by the EM algorithm? Or should we assume that the data meets requirements and run the EM algorithm immediately? The answer depends entirely on the nature of the dataset being processed and on the circumstances surrounding the particular missing data problem.

Perhaps the most sensible approach would be to proceed with the imputation process unless our knowledge of the dataset suggests that it is not suitable to be processed by the EM algorithm. For example, suppose our knowledge of the data leads us to believe that the columns in the data matrix are far from being normally distributed. To address this problem, suppose we test every column in the matrix and find that 80% of these are approximately normally distributed, with a small proportion of outliers - but that the distributions in the remaining columns are unacceptable. The decision to be made in this case is whether the 20% departure from normality invalidates the EM process, or whether it can be "worked around" or ignored. Again, everything depends on the circumstances surrounding the missing data problem. For example, the offending columns could be deleted from the matrix, but then the missing data in those columns could not be imputed, and the observed data could not be used to impute missing values in the remaining columns.

However, even in cases where some of the variables are non-normal, the EM algorithm can still produce reliable estimates for missing values (Schafer, 1997). Furthermore, in some cases variables can be transformed to approximate normality prior to imputation - e.g. using the Box-Cox algorithm described in chapter six.

The MCAR assumption complicates the EM imputation process

When values are MCAR the parameters describing the missing data pattern (represented as a binary matrix - see chapter one) must be re-estimated at each iteration of the EM algorithm. And modelling the MCAR missing data pattern is very problematic in most cases and may be impossible for some datasets. Hence, the version of the EM algorithm described below assumes that the data is MAR, since implementing an MCAR version of the EM algorithm would be extremely difficult for the reasons given above, as Allison (2000) points out;

> *"There are often strong reasons to suspect that the data are not MAR. Unfortunately, not much can be done about this. While it's possible to formulate and estimate models for data that are not MAR, such models are complex, untestable, and require specialized software. Hence any general purpose method will necessarily invoke the MAR assumption"*

2.2.3 Functional Outline of the EM Imputation Algorithm

```
function matrix  EM_algorithm_for_missing_data_imputation ( matrix Y, double e )
    matrix $\theta_{new}$ = initial_parameter_estimate_for_matrix (Y)
    repeat
        estimate_missing_values_in_matrix_Y_using ($\theta_{new}$)
        $\theta_{old}$ = $\theta_{new}$
        $\theta_{new}$ = new_parameter_estimate_for_matrix (Y)
    until em_has_converged ( $\theta_{new}$ , $\theta_{old}$ , e )
    impute_missing_values_ in_matrix_Y_using ($\theta_{new}$)
    return Y
end function
```

$$\text{where} \quad \theta_{new} = (\mu, \Sigma) = \begin{bmatrix} -1 & \mu_1 & \mu_2 \cdots & \mu_p \\ \mu_1 & \Sigma_{11} & \Sigma_{12} \cdots & \Sigma_{1p} \\ \mu_2 & \Sigma_{12} & \Sigma_{22} \ldots & \Sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_p & \Sigma_{1p} & \Sigma_{2p} & \Sigma_{pp} \end{bmatrix}$$

Fig 2.2 – Functional outline of the EM algorithm for multivariate imputation

The functional outline given in Fig. 2.2 shows how the EM algorithm can be used to impute the missing values in a multivariate, numeric data matrix Y, which is passed as a parameter to the function, then returned with the missing values "filled in" just before the algorithm terminates. Notice that the initial value of the Y distribution parameter θ_{new} must be estimated before the ***repeat*** loop starts (see Appendix B for an explanation of how this can be achieved in practice). This parameter takes the form of the augmented covariance matrix for the Y matrix, as shown in Fig. 2.2, above.

The function uses successive values of θ_{new} to test for EM "convergence" from one iteration of the ***repeat*** loop to the next. This is achieved by comparing each corresponding pair of elements in θ_{new} and θ_{old} using the function below (which is called from the function above).

```
function boolean em_has_converged ($\theta_{new}$ , $\theta_{old}$ , $e$ )
    int p = num_columns_in ($\theta_{new}$ )
    for int i = 0 to p
        for int j = i to p
            if $(|\theta_{new}(i,j) - \theta_{old}(i,j)| > e|\theta_{new}(i,j)|)$
                return false
        next j
    next i
    return true
end function
```

Where this function only returns a value of *true* if the absolute difference between *every* corresponding pair of elements in θ_{new} and θ_{old} is $\leq e|\theta_{new}(i,j)|$ where $\theta_{new}(i,j)$ represents each θ_{new} element. The execution of this function is equivalent to comparing successive values of the observed data loglikelihood, using;

$$\left|l(\theta_{new}|Y_{obs}) - l(\theta_{old}|Y_{obs})\right| \leq e\left|l(\theta_{new}|Y_{obs})\right|$$

where e is a small value, such as 0.0001. And where the value of $l(\theta_{new}|Y_{obs})$ is *guaranteed not to decrease* (and will normally increase) with each iteration of EM, which makes convergence possible. Although the rate of convergence of EM can differ significantly from dataset to dataset in practice, which was found to be the case when conducting the experiments described in chapters five and six. See Meng (1990), Meng (1994) and Schafer (1997) for detailed discussions of the theory underpinning the rate of EM convergence.

The key concept underpinning the functionality of the EM algorithm outline given in Fig. 2.2 lies in the iterative re-estimation and reuse of θ_{new} . Notice in particular the ***repeat …. until*** loop functionality, where;

1. θ_{new} is used to estimate the missing values in Y
2. Y is used to estimate θ_{new}
3. θ_{new} is used to estimate the missing values in Y
4. Y is used to estimate θ_{new}

…and so on, until the difference between successive values of θ_{new} converge to the value of e (which is passed as a parameter to the EM function). When convergence is achieved the final set of element values in θ_{new} will contain the Y distribution parameters which maximise the likelihood of the occurrence of the dataset in *Y*. And, finally, θ_{new} is used to compute the estimates for the missing values in *Y*, thus completing the imputation process.

The entire procedure is elegant and procedurally efficient and should have a fast execution time if it is well implemented (e.g. using the methods described in section 2.4). In fact, the EM imputation algorithm is generally recognised as having *"elegant statistical properties and sound theoretical justification"* as pointed out by Bello (1995), among others.

2.2.4 Using the SWEEP Operator to Impute Missing Values

The EM imputation algorithm is essentially a regression based procedure which converts known values into predictor values, thus allowing missing values to be estimated using the regression equations so formed. The most complex problem to be solved when implementing EM in practice is to generate regression equations for each missingness pattern in the data matrix. For example, consider the *Y* data matrix given below, where 1 represents a known value and 0 represents a missing value. Rows 1 and 4 have missingness pattern 10101. Rows 2 and 6 have missingness pattern 10011. Row 3 has missingness pattern 00110.

$$Y = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The problem is to find a method to generate regression equations for the imputation of each missing value in each different missingness pattern. Where each term in these equations is formed using the product of one of the known values in the row being imputed and one of the derived regression coefficients for the missingness pattern in question. This can be achieved using the versatile matrix "SWEEP operator" which can be applied for the solution of any sort of iterative, regression based, imputation problem.

The SWEEP operator was first introduced by Beaton in (1964), but the version described here was originally defined by Dempster (1969). One of the earliest and most detailed tutorials on the use of SWEEP operator and related matrix transformations can be found in Goodnight (1979). A detailed and accessible description of the theory underpinning the use of the SWEEP operator with the EM algorithm can be found in Little and Rubin (2002).

Applying the SWEEP operator to impute missing values as part of the EM algorithm

The sections below give a straightforward explanation of how the SWEEP operator can be used to impute missing values in practice. The text includes a pseudo-code function that can be used to implement SWEEP as part of the EM algorithm. This function forms an integral part of the EM functional outline given in Fig. 2.2. That is, it will be called repeatedly from the procedure *estimate_missing_values_in_matrix_Y_using* (θ_{new}) which is called from within the ***repeat …. until*** loop given in Fig. 2.2. The precise nature of this functionality is described in great detail in the complete EM algorithm pseudo-code given in Appendix B.

When the SWEEP operator is applied to the parameter describing the augmented Y covariance matrix θ (as described in the previous section) it converts the known values into predictor values, which allows the missing values to be imputed using the regression equations so formed. For example, suppose a data matrix has four columns labelled Y_1 to Y_4 where the data in a particular row is present for columns Y_1 and Y_2 but is missing for columns Y_3 and Y_4. In this case the repeated and correct use of the SWEEP operator on the θ matrix can be used to generate the regression coefficients of Y_3 on Y_1 and Y_2 and similarly of Y_4 on Y_1 and Y_2. This allows the missing values in columns Y_3 and Y_4 to be imputed using the pseudo-code implementation described below.

Symmetric (p x p) numeric matrices can be “swept” on any row and column **-** i.e. the notation used to represent the SWEEP operation is $SWP[k]T$ where T is the matrix being swept, and k is in the range 1 to p or 0 to p for augmented matrices, such as the one used in the example below. In practice, the SWEEP operator can be considered as a function, where the parameters passed to this function are; **(1)** the row and column to sweep on, in this case denoted by *k* and, **(2)** the matrix to be swept, in this case denoted by *T*. The function adjusts the elements of T by sequentially executing the five steps defined below, then returns the adjusted *T*. For example, the matrix transformation given below shows how the SWEEP operation $\theta = SWP[0]T$ would be applied to a (3 x 3) version of the symmetric matrix *T*.

(1) $\theta_{kk} = -1/t_{kk}$

(2) $\theta_{jk} = t_{jk}/t_{kk}$ ***for*** $j \neq k$

(3) $\theta_{kj} = \theta_{jk}$ ***for*** $j \neq k$

(4) $\theta_{jl} = t_{jl} - t_{jk}t_{kl}/t_{kk}$ ***for*** $j \neq k$ *and* $l \neq k$

(5) $\theta_{lj} = \theta_{jl}$ ***for*** $j \neq k$ *and* $l \neq k$

$$T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{12} & t_{22} & t_{23} \\ t_{13} & t_{23} & t_{33} \end{bmatrix} \qquad \theta = SWP[0]T = \begin{bmatrix} -1/t_{11} & t_{12}/t_{11} & t_{13}/t_{11} \\ t_{12}/t_{11} & t_{22} - t_{12}^2/t_{11} & t_{23} - t_{13}t_{12}/t_{11} \\ t_{13}/t_{11} & t_{23} - t_{13}t_{12}/t_{11} & t_{33} - t_{13}^2/t_{11} \end{bmatrix}$$

Thus, we can say that the matrix T has been “swept on row and column 0”, and that the matrix θ has been set equal to the adjusted version of T created by the SWEEP operation (see Appendix B for an explanation of the purpose and use of the T matrix within the EM algorithm). Notice that, for symmetric matrices (such as those swept by the EM algorithm) the upper right triangle is a mirror image of the lower left triangle - so the top right corner element and the bottom left corner element are equal etc.

A pseudo-code implementation of the SWEEP operator

The pseudo-code given below, which is based on the five steps given above, provides a fast and efficient computational procedure for implementing the SWEEP function. Note that the rows and columns of the (p + 1) x (p + 1) symmetric matrix passed to the function must be indexed from 0 to p, rather than from 1 to (p + 1). This is specifically required when processing the augmented EM matrices θ and T (as explained further in Appendix B).

```
function matrix  sweep_matrix_on ( int k,  matrix g )
    int p = num_columns_in (g)
    matrix s = new matrix ( p, p )
    $s_{kk} = -1/g_{kk}$
    for int $i = 0$ to p && $i \neq k$
        $s_{ik} = -g_{ik} s_{kk}$
        $s_{ki} = s_{ik}$
    next i
    for int $i = 0$ to p && $i \neq k$
        for int $j = 0$ to p && $j \neq k$
            $s_{ij} = g_{ij} - s_{ik} g_{kj}$
            $s_{ji} = s_{ij}$
        next j
    next i
    return s
end function
```

The most important point to note regarding the use of this function by the EM algorithm is that calling it to sweep the matrix θ on an element at row and column k converts that element from a dependant variable to an independent variable in the regression equation formed by its related elements in the swept matrix. Recalling that, for the EM algorithm, θ forms the (p + 1) x (p + 1) augmented covariance matrix - where p is the number of columns in Y, which are indexed from 0 to p. It can be shown that, for the bivariate case, with a (3 x 3) θ matrix, the function call $\theta =$ *sweep_matrix_on* $(1, \theta)$ will yield the regression coefficients of Y_2 on Y_1 as shown below,

$$\theta = \begin{bmatrix} -1 & \mu_1 & \mu_2 \\ \mu_1 & \Sigma_{11} & \Sigma_{12} \\ \mu_2 & \Sigma_{21} & \Sigma_{22} \end{bmatrix}$$

After the sweep operation $\theta = SWP\ [1]\theta$ has been performed;

θ_{02} or θ_{20} will give the intercept of Y_{22} on Y_{11}

θ_{12} or θ_{21} will give the slope of Y_{22} on Y_{11}, and consequently,

$Y_{22} = \theta_{02} + \theta_{12} Y_{11}$ will give the regression equation of Y_{22} on Y_{11}

This important result forms the functional basis of the expectation step of the EM algorithm as applied for the estimation of missing values in a multivariate dataset. And, *essentially* for the EM algorithm, the bivariate case given above can be extended to the multivariate case by performing a set of related consecutive sweeps of the θ matrix - where the set of sweep indexes for a particular multivariate missingness pattern can be represented by;

$SWP\ [k_1,.....\ k_n]\theta$ or by the equivalent notation;

$SWP\ [M\ (k)]\theta$ where $k \in M$ and $M = \{k_1,....\ k_n\}$

In addition - as Little and Rubin (2002) explain - it can be shown algebraically that the SWEEP operator is fully commutative. This means that the final computational result of any given set of consecutive sweeps will be the same regardless of the order in which those sweeps are performed. Stated more formally, in the case where the set $A = \{\ j_1,....\ j_n\}$ can be any permutation of the set $M = \{k_1,....\ k_n\}$ we can say;

In the general case $SWP\ [k_1,.....\ k_n]\theta = SWP\ [\ j_1,.....\ j_n]\theta$

Or equivalently $SWP\ [M\ (k)]\theta = SWP\ [A(\ j)]\theta$

<u>The reverse SWEEP operator is an essential part of the EM algorithm</u>

The reverse SWEEP operator is called repeatedly from the complete EM algorithm pseudo-code given in Appendix B. The reverse SWEEP operation is used to return a swept matrix to its original form using the notation $RSW\ [k]\theta$ where θ is the matrix being swept in reverse, and k is in the range 1 to p, or 0 to p for augmented matrices, such as θ. For example, the following consecutive operations would sweep the θ matrix then return it to its original form;

$$\theta = SWP[k]\theta \quad \text{followed by} \quad \theta = RSW\ [k]\theta$$

The reverse SWEEP operation $\theta = RSW\ [k]\theta$ would be performed by sequentially executing the five steps defined below. Notice that these steps are almost identical to the steps for $\theta = SWP[k]\theta$, the only difference being the presence of the minus sign in step (2).

(1) $\theta_{kk} = -1/\theta_{kk}$ (4) $\theta_{jl} = \theta_{jl} - \theta_{jk}\theta_{kl}/\theta_{kk}$ **for** $j \neq k$ *and* $l \neq k$

(2) $\theta_{jk} = -\theta_{jk}/\theta_{kk}$ **for** $j \neq k$ (5) $\theta_{lj} = \theta_{jl}$ **for** $j \neq k$ *and* $l \neq k$

(3) $\theta_{kj} = \theta_{jk}$ **for** $j \neq k$

The RSW function could be implemented using a slightly adapted version of the pseudo-code for the ***sweep_matrix_on*** () function (given above) where the difference in the step (2) calculation is accounted for. Alternatively, the ***sweep_matrix_on*** () function could be adapted to perform both operations by passing an extra parameter to it, which would be used to determine the correct calculation to perform at step (2). Note that the notation for performing consecutive reverse sweeps is the same as for the sweep operation - i.e. $RSW\ [k_1, \ldots.. k_n]\theta$ or $RSW\ [M\ (k)]\theta$. Also note that RSW is fully commutative as for the SWEEP operation itself.

2.3 A New Implementation of the EM Imputation Algorithm

The functional outline of the EM algorithm and the SWEEP operator pseudo-code given above can be used to implement the EM algorithm in practice. These functional explanations have been expanded to give a complete pseudo-code listing for the implementation of EM for numeric multivariate imputation, which is given in Appendix B.

This pseudo-code has been implemented as a fully reusable software class using the Microsoft C# programming language. The resulting EM class can be instantiated and used by any C# imputation application, regardless of the user interface created to support the imputation process. The user interface that has been developed to support the new EM implementation - and the implementation of the new imputation evaluation method - is described in chapter five. The following section explains how the functionality of the new EM implementation was tested and verified in practice.

2.3.1 Verifying the Functionality of the New EM Implementation

The results produced by the EM implementation described above were verified as being correct by comparing the augmented covariance matrices produced by the new implementation with the covariance matrices produced by the NORM imputation software. Several different versions of the NORM software can be freely downloaded from the Pennsylvania State University website at http://www.stat.psu.edu/~jls/misoftwa.html The most recent NORM application, (Version 2.03 for Microsoft Windows XP) was used to perform the verification experiments.

The NORM application was produced by Professor Joseph L. Schafer of Pennsylvania State University using the S-PLUS programming language. Schafer is one of the world's leading authorities on the analysis of incomplete multivariate data, and the NORM application is very highly regarded academically. For example, Allison (2000) shows that in some cases NORM produces less biased imputed values than the commercial SOLAS (version 1.1) imputation application, which uses the multiple imputation method developed by Lavori et al (1995). And a useful description of how multiple imputation has been implemented (and could be better implemented) in the SOLAS (version 3.0) application and several other commercial imputation applications can be found in Horton and Lipsitz (2001).

NORM operates only on numeric data, and estimates missing values using multiple imputation via the Bayesian data augmentation algorithm (Schafer, 1997). The EM algorithm is used *only* to generate the initial value of the covariance matrix $\theta = (\mu, \Sigma)$, so as to speed up the data augmentation process. Use of the EM algorithm is an optional part of the NORM imputation process, but the resulting covariance matrix is automatically exported to a text file, which proved to be invaluable when testing the new EM implementation. Datasets with missing values must be loaded into the NORM application via text files. When the imputation process is complete NORM automatically copies the completed data matrix (with the missing values "filled in") into another text file, which can then be used to update the missing values in the source dataset.

Verifying the new EM implementation by comparing it with the NORM version of EM

NORM comes with two demonstration datasets that have missing values, which are stored in text files. The first file contains a data matrix with 25 rows and 4 columns, with 27% missing values. The second file contains a data matrix with 279 rows and 12 columns, with just under 8% missing values. These files were used to test the new EM implementation. In both cases all of the augmented covariance matrix element values produced by the new EM implementation and by NORM proved to be the same, to four significant figures. The final test was performed against an independent dataset with 32% missing values - as created by Ryan and Joiner in (1994) and described by Schafer in (1997). Again, all of the covariance matrix elements were found to be same, to four significant figures. These tests strongly suggest that the new EM implementation produces correctly imputed values, since the probability of both applications producing the same covariance matrix in three different datasets by chance was assumed to be very small.

2.4 Decreasing the Execution Time of the EM Imputation Algorithm

The proposed imputation evaluation method requires repeated executions of the imputation method that is used to “put back” the values that are deleted, as explained in chapters one and four. When using the method in practice it could be necessary - for some experiments - to repeat the imputation process tens, or even hundreds, of times (e.g. see chapters five and six). Consequently, it is important that the implementation of the EM algorithm described above should execute as quickly as possible.

However, techniques for decreasing the execution time of imputation algorithms have received very little attention in the literature. By contrast, performance issues (algorithm execution time and so on) have received a great deal of attention in other disciplines. For example, in the field of data mining (Fayyad et al, 1996; Berson and Smith, 1997) a considerable amount of work has been done which focuses primarily on decreasing the execution time of various clustering algorithms. See, for example, the related papers by Ester et al (1996) and Wang and Hamilton (2003). This section redresses the balance somewhat by discussing the issues surrounding the execution speed of the EM algorithm.

2.4.1 Factors Affecting EM Algorithm Execution Time

Generally speaking, the execution time of *any imputation algorithm* will depend primarily on the size of the dataset being processed - e.g. in the case of the EM algorithm, execution time will increase as the size of the *Y* data matrix increases. However, the execution time of the EM algorithm is also closely linked to the number of missingness patterns present in *Y*, since the EM process must, by definition, use different regression equations (the number and size of the coefficients will differ) to impute the missing values in each missingness pattern. The maximum number of patterns that can occur in the *Y* matrix is 2^p, where p is the number of columns (variables) in *Y*. However, each missingness pattern must be contained in at least one *Y* row, therefore the number of possible patterns cannot exceed the number of rows in *Y*. It follows that the execution of the EM algorithm will take longer and longer to achieve as the number of missingness patterns in *Y* approaches the number of rows in *Y*, and that the slowest execution time will occur in the rare cases when *every Y* row has a unique pattern.

It is generally accepted (see for example, Aho et al, 1983; Knuth, 1997) that the execution time of *almost any algorithm* can be reduced by creating “lookup tables” etc. and storing these in RAM - thus removing the need to repeatedly recreate these data structures as they become needed to support the processing. Conversely, RAM storage requirements can usually be reduced by repeatedly recreating smaller data structures as they become needed during the processing. The key question to ask in each specific case is whether the decrease in execution time is sufficient to justify the RAM needed to store the lookup table. The pseudo-code version of the EM algorithm given in Appendix B attempts to optimise performance *and* minimise RAM storage requirements by creating only those lookup tables which prove to be the most beneficial for decreasing EM algorithm execution time.

The fastest possible processing of the EM sufficient statistics matrix T_{obs} will be needed for any efficient EM implementation (see Appendix B for a description of the purpose and use of the T_{obs} matrix). This can be achieved by creating the initial version of the T_{obs} matrix only once (at the start of algorithm) then storing and reusing it repeatedly. This avoids the unnecessary re-calculation of T_{obs} at the start of every EM iteration, which considerably reduces algorithm execution time, since the initial version of T_{obs} must be calculated using all of the rows in the *Y* data matrix, and this calculation will take proportionately longer to execute as the number of rows in Y increases. This approach is also recommended by Schafer (1997) and it has been implemented within the pseudo-code given in Appendix B.

A fast method for generating the regression equations needed to impute the missing values within each missingness pattern is an essential part of any EM implementation. This can be achieved by using the SWEEP operator (see section 2.2.4), and this approach has been used within the pseudo-code given in Appendix B. Use of the SWEEP operator when implementing EM is also is recommended by both Little and Rubin (2002) and Schafer (1997), who are perhaps the worlds leading authorities on the implementation of MLE based imputation methods.

To facilitate EM processing the *Y* data matrix must be sorted into missingness pattern order. In other words, all rows with the same missingness pattern must be adjacent in the *Y* matrix. To achieve faster EM execution times it is essential this sorting process is performed using a sorting algorithm which requires no more than $(n \log n)$ row comparisons, where n is the number of rows in *Y*. More generally, it is important to note that *any* processing of the *Y* matrix requiring n^2 operations will become impractically slow when n becomes sufficiently large, where the value of n that causes unacceptable performance will of course depend on the computer hardware and software configuration used for the EM implementation.

2.4.2 Measuring EM Execution Time Using Large Simulated Datasets

The experiments described below contribute to the theory underpinning maximum likelihood based imputation via the EM algorithm. The experiments were designed with the following three objectives in mind. The first two objectives do not seem to have been discussed anywhere else in the literature. The third objective is specific to the work described in this thesis. An explanation of how these objectives were achieved follows the list.

1. To define a standard method for creating randomly generated missing value datasets that can be used to compare the execution times of various EM algorithm implementations.

2. To establish a set of benchmark EM algorithm execution times and related performance statistics (see table 2.2) against which any EM implementation can be compared.

3. To measure the execution time of the new EM implementation, so as to discover whether this implementation was adequate for the purpose for which it was created.

All of the experiments were performed against simulated data matrices containing 12 columns and between 1 and 5 million rows. The randomly generated numeric values inserted into these matrices were integers between 1 and 100, with values greater than 70 being replaced with missing value indicators - numerically encoded as values of minus 9. Consequently, approximately 30% of the data appeared as missing, with the missing values appearing in randomly created missingness patterns within the rows in each simulated matrix. The EM performance experiments executed against these datasets were all carried out in November 2005, using the computer hardware and software configuration described in Appendix C.

The first five experiments were designed to measure the execution time of the new EM implementation when processing a simulated data matrix containing 1 million rows and 12 columns. Five different randomly generated datasets with missing values were created using the method described above. The results are shown in table 2.1, below. The three rightmost columns of the table show that the method used to generate the experimental datasets produced consistent results across all experiments. It can be seen that EM execution time is similar for each experiment, with (on average) 35% of this time being used to sort the data matrix rows into missingness pattern order (all rows with the same missingness pattern were made to be adjacent in the data matrix).

Table 2.1 - EM algorithm execution times for 5 simulated datasets containing one million rows

Dataset Number	Execution Time (seconds)	Sort Time (seconds)	Number of EM Iterations	Number of Missing Values	Number of Missingness Patterns
1.	214	75	23	3,517,656	4083
2.	233	77	26	3,518,489	4080
3.	208	74	22	3,515,742	4081
4.	206	73	21	3,515,821	4087
5.	209	73	22	3,514,267	4086
Column mean value	**214**	**74**	**23**	**3,516,397**	**4083**

The second set of experiments were designed to measure the execution time of the new EM implementation when processing simulated data matrices containing 12 columns and between 1 and 5 million rows. Five different randomly generated datasets with missing values were created using the method described above. The results are shown in table 2.2, below - with row 1 of the table showing the mean values of the results of the first set of experiments. Again, the three rightmost table columns show that the missing data simulation method produced consistent datasets. It can be seen that the relationship between algorithm execution time and the number of rows in the data matrix is approximately linear, with execution time steadily increasing as the size of the data matrix increases. Column 3 of table 2.2 shows that, on average, 39% of the algorithm's execution time was required for sorting the data matrix rows into missingness pattern order.

Table 2.2 - EM algorithm execution times for simulated datasets containing 1 to 5 million rows

Number of Data Rows	Execution Time (seconds)	Sort Time (seconds)	Number of EM Iterations	Number of Missing Values	Number of Missingness Patterns
1 million	214	74	23	3,516,397	4083
2 million	448	164	24	7,035,515	4092
3 million	685	264	24	10,541,612	4094
4 million	962	365	25	14,058,748	4094
5 million	1143	478	20	17,579,973	4097

The final experiment shows that 60 million simulated data values (including over 17 million missing values) can be processed in just over 19 minutes using the new version of the EM algorithm developed by the author. Consequently, the new imputation evaluation method can be executed 10 times against a dataset containing 60 million values in just over 3 hours (using the hardware and software configuration described in Appendix C). It is therefore argued that the new implementation of the EM algorithm offers execution times which allow the proposed imputation evaluation method to be tested and evaluated against quite large datasets in practice. And that, consequently, the new implementation of EM has been successful, because it has been shown to be more than adequate for the purpose for which it was created.

2.5 Summary

This chapter has explained the fundamental concepts underpinning the implementation of MLE via the EM algorithm, and has shown how this approach can be used to impute missing values in datasets with multivariate missingness patterns. The history and utility of the EM algorithm has been discussed and the type of datasets that can be processed by the EM algorithm have been described.

A description of how the author has implemented the EM algorithm as a software application has been given, including an explanation of how the SWEEP operator was used to the generate the regression equations needed for the execution of the EM imputation process. The experiments that evaluate the performance of the new EM implementation make some contribution to the theory of maximum likelihood imputation via the EM algorithm.

The following chapter describes the second imputation technique that has been implemented and explains why it was chosen. Chapter four goes on to give a formal description of the imputation evaluation method devised by the author, and explains how it can be used to estimate the predictive accuracy of the imputed values generated by the imputation techniques described in this chapter and in chapter three.

Chapter five describes how the two imputation techniques that were chosen have been implemented alongside the proposed imputation evaluation method in the form of an integrated software application, and explains how this application was used to experimentally evaluate the reliability and the validity of the proposed method. Chapter six explains how the software application was used to assess the feasibility of imputing the missing values in the collaborating company’s dataset.

Chapter Three

Nearest Neighbour Imputation

3. Nearest Neighbour Imputation

The EM imputation algorithm was selected for the first tests of the proposed imputation evaluation method for the reasons given in chapter two. This chapter describes the second imputation method that was implemented and evaluated, and explains why it was chosen.

Imputation methods can be broadly categorised into parametric methods (statistical approaches) and non-parametric methods (usually employing data mining techniques). The EM algorithm was selected from among the parametric methods, and so it was decided that a non-parametric method should be implemented next, for reasons of balance. The nearest neighbour (NN) imputation method was selected from among the non-parametric group because it is perhaps the most general purpose of all the non-parametric methods, and because the ideas underlying it can be easily understood, as described below.

Nearest neighbour imputation algorithms replace the missing values within any particular data matrix row (observation) by taking copies of the corresponding known values from the most similar observation found in the dataset. This approach has two principal advantages over parametric methods.

- For NN imputation the distributions of the variables in the missing value dataset are *not* required to conform to any particular model. However, parametric imputation methods can be sensitive to model misspecification, which can result in poorly imputed values (Lazzeroni et al, 1990; Durrant, 2005).

- The imputed values generated by NN algorithms are *copied* directly from real cases (donor observations). In other words, *"they may not be perfect substitutes, but are unlikely to be nonsensical values"* (Chen and Shao, 2000). This avoids the nonsensical errors that are occasionally produced by parametric approaches, such as producing negative estimates for the number of employees in a business organisation etc.

The following sections explain how NN imputation was implemented as part of the integrated application that implements the proposed imputation evaluation method.

- **Section 3.1** explains the ideas underpinning the functionality of a general purpose NN imputation algorithm devised by the author, and explains how similar methods can be evaluated in practice.

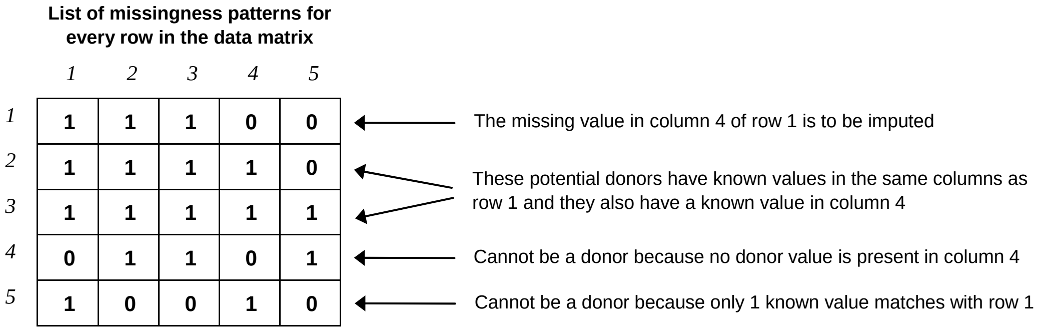
- **Section 3.2** proposes a method for reducing the execution time of any nearest neighbour imputation algorithm and describes how this idea was evaluated experimentally. ***This section forms part of the contribution made by this thesis.***

3.1 Implementation of the NN Imputation Algorithm

Section 3.1.1 explains how the missingness patterns within the dataset rows can be used to find each nearest neighbour. Section 3.1.2 explains how this approach was implemented in the form of a general purpose NN imputation algorithm. Section 3.1.3 proposes some general rules that can be applied for the evaluation of any NN imputation method.

3.1.1 Using the Missingness Pattern Structure to Find Nearest Neighbours

Nearest neighbour algorithms impute a missing value in a particular matrix row (dataset observation) S_m by taking a copy of the known value from the most similar donor row S_i, such that $S_{mc} = S_{ic}$, where c is the matrix column (variable) that has a missing value. And where the most similar donor row is found by comparing S_m with all of the other rows in the matrix, and using the row that returns the smallest value of $d\left(S_m, S_i\right)$ as the donor - i.e. finding the minimum value of the similarity measure $d\left(S_m, S_i\right)$ for all $S_i \in P$, where $P = \{S_1, ... S_n\}$ is the set of all matrix rows and where the variables in S_m and S_i are suitably scaled, so that each variable carries the required weight in the similarity calculations. And where $d\left(S_m, S_i\right)$ can be measured using any similarity function, such as the simple Euclidean distance, or a more complex measure, such as the Hellinger distance (Lee and Shin, 1999) or the Mahalanobis distance (Mahalanobis 1936; Stage and Crookston 2002). However, only some $\left(S_m, S_i\right)$ pairs can be meaningfully compared, as follows;



List of missingness patterns for every row in the data matrix

	1	2	3	4	5
1	1	1	1	0	0
2	1	1	1	1	0
3	1	1	1	1	1
4	0	1	1	0	1
5	1	0	0	1	0

The missing value in column 4 of row 1 is to be imputed

These potential donors have known values in the same columns as row 1 and they also have a known value in column 4

Cannot be a donor because no donor value is present in column 4

Cannot be a donor because only 1 known value matches with row 1

Fig 3.1 - Imputing the missing value in column 4 of row 1 in a data matrix using a NN algorithm

Fig. 3.1 shows the missingness patterns for every row in a data matrix (where 1 represents a known value and 0 represents a missing value). The diagram shows the rows that can be considered as potential donors when imputing the missing value in column 4 of row 1. Notice that row 3 can be a potential donor for any other row, because it has a full set of known values. *The most important point* is that row 2 can be considered as a potential donor, but row 5 cannot, because the similarity between rows 1 and 2 cannot be meaningfully compared with the similarity between rows 1 and 5. For example, a Euclidean distance calculation would produce a smaller value when measuring the similarity between rows 1 and 5, because only one column would be included in the calculation, whereas 3 columns would be included in the calculation when measuring the distance between rows 1 and 2.

However, row 5 could be considered as a potential donor if some form of weighting was included in the similarity calculation to compensate for the reduced number of variables used to measure that similarity. This approach has been tried and tested by Huang and Zhu (2002), who use an experimental *"pseudo-nearest-neighbours"* method to impute missing values in multivariate Gaussian datasets, where the *"pseudo-similarity"* measurement implemented *"is actually a weighted correlation value between the two vectors with partially missing element values"*. A less complex, but somewhat similar, method of weighting was proposed much earlier by Dixon (1979) - and Junninen et al (2004) have evaluated Dixon's method in practice by comparing it to various other imputation techniques. However, these weighting methods are best applied to Gaussian datasets where a good linear regression model can be fitted to the data - i.e. where the correlations between the variables are strong. Consequently, similarity measures that utilise variable weighting schemes were not implemented as part of the first software versions of the new imputation evaluation method, because this approach was not considered to be sufficiently general purpose in nature for the initial experiments. An algorithmic implementation of the imputation method described in Fig. 3.1 is given below.

3.1.2 A General Purpose Nearest Neighbour Imputation Algorithm

```
function matrix  generic_NN_imputation_in_column ( int c,  matrix data )
    dataMatrixRow  missRow, donorRow, closestRow
    removeEmptyRowsIn ( data )

    for m = 1  to num_rows_in ( data )
        missRow = data ( m )
        if ( missRow . patt ( c ) == 0 )
            minDistance = null
            for d = 1  to num_rows_in ( data )
                donorRow = donors ( d )
                if ( donorRow . patt ( c ) == 1 )
                    match = true
                    j = 1
                    while ( j <= num_cols_in ( missRow )  &&  match == true )
                        if ( missRow . patt ( j ) == 1  &&  donorRow . patt ( j ) == 0 )
                            match = false
                        end if
                        j ++
                    end while
                    if ( match == true )
                        distance = euclideanDistanceBetween ( missRow, donorRow )
                        if ( distance < minDistance  ||  minDistance == null )
                            minDistance = distance
                            closestRow = donorRow
                        end if
                    end if
                end  if
            next  d
            if  ( minDistance  !=  null )
                missRow ( c )  =  closestRow ( c )
            end  if
        end  if
    next  m
    return  data
end  function
```

Fig 3.2 – A general purpose nearest neighbour imputation algorithm

The algorithm implements the row comparison method described in Fig. 3.1. The Euclidean distance is used to measure the similarity between observations, but any other similarity measure could be substituted. The following parameters are passed to the algorithm;

(1) *int c* is the *data* matrix column containing the values to be imputed.

(2) *matrix data* is the data matrix, which is passed to the algorithm with missing values in column *c* and returned with imputed values in column *c*,

Where the statement *missRow (c) = closestRow (c)* imputes each missing value. And where each *dataMatrixRow* object contains a binary array *patt( )* which represents its missingness pattern (see chapter one).

3.1.3 Evaluating Nearest Neighbour Imputation Algorithms

The algorithm given in Fig. 3.2 is general purpose in nature, because it can be used to impute missing values in any numeric multivariate dataset. Many other NN imputation algorithms have been devised, but some of these have limited utility, because they were designed to solve a unique missing data problem. In other words, the functionality of these algorithms was customized to suit a specific type of dataset. However, NN algorithms all share the same basic functionality in that they all impute missing values by finding nearest neighbours and then copying values from them. Consequently, it is argued that the following three questions are the most appropriate ones to ask when evaluating *any* NN imputation algorithm.

1. Is the similarity measure used suitable for the solution of the missing data problem? That is, does this measure find the best possible donor rows for imputation purposes?

2. Is the method used to decide which dataset rows should be *considered* as potential donors appropriate? - e.g. if the dataset has been segmented, was this done in such a way that the search for each donor row takes place within the best possible *subset* of rows?

3. Is the method used to decide which rows can be *meaningfully* compared logical, given the nature of the data? This question is discussed further, below.

The first two questions need not be asked for the algorithm given in Fig. 3.2, because it is generic by design. That is, it can employ *any method* for measuring the similarity between observations and it can be executed against *any sort* of segmented dataset, regardless of the classification scheme used. On the other hand, the algorithm does specify which rows can be meaningfully compared, as shown in Fig. 3.1. However, it is argued that this is a good general purpose approach which can be applied for the solution of many missing data problems, for the following reasons;

1. Non-response in surveys is perhaps the most prevalent missing data problem (Rubin, 1996a), and it is often found that several of the variables in a survey dataset - such as a set of questionnaires - have some missing values (Allison, 2001). This creates many different missingness patterns within the dataset. The algorithm given in Fig. 3.2 can be applied to all such datasets (from surveys or otherwise), regardless of the structure and distribution of the missingness patterns they contain.

2. Employing any method that involves comparing rows that do not have a common set of known values requires the use of a similarity measure that can return different values depending on the number of common values in the rows compared. It is argued that this approach should be avoided whenever possible, because it builds an additional level of uncertainty into the NN imputation process.

3.2 Decreasing the Execution Time of NN Imputation Algorithms

The proposed imputation evaluation method requires repeated executions of the imputation process that is used to “put back” the values that are deleted. Consequently, it is important that the implementation of the nearest neighbour algorithm should execute as quickly as possible. ***This section describes a new approach for decreasing the execution time of any NN imputation algorithm which forms part of the contribution.***

A considerable amount of work has been done to evaluate and compare the results produced by various NN imputation algorithms (Wasito and Mirkin 2005/2006; Durrant, 2005; Kalton, 1982) and to analyse the functionality and properties of such algorithms (Chen and Shao, 2000; Fay, 1999; Rancourt et al, 1994). In addition, several methods for measuring the similarity between dataset rows when searching for the nearest neighbour have been proposed (Dixon, 1979; Huang and Zhu 2002; Stage and Crookston 2002; Lee and Shin, 1999). However, the slow execution time and the resulting poor scalability of multivariate NN imputation algorithms has received very little attention - i.e. searching the dataset for each nearest neighbour takes longer and longer as the size of that dataset increases.

Section 3.2.1 addresses this problem by explaining how the execution time of the NN imputation process can be reduced. Sections 3.2.2 and 3.2.3 explain how this approach was implemented in practice. Section 3.2.4 describes how the execution time of the resulting algorithm was evaluated using a set of simulated missing value datasets. Section 3.2.5 describes how the algorithm was evaluated using a real survey dataset and discusses the need for segmentation when performing NN imputation in large datasets.

3.2.1 Using the Missingness Pattern Structure to Decrease Execution Time

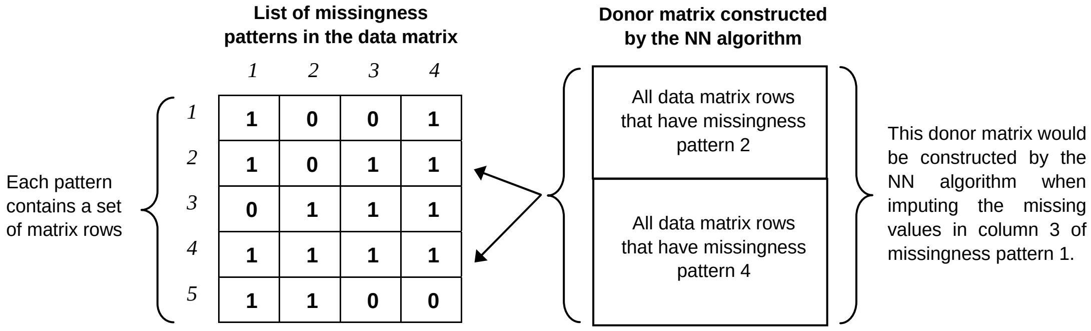
Generally, when NN algorithms are executed against *complete* datasets they compare a particular dataset row with every other row when searching for that rows nearest neighbour (unless the dataset is segmented by class, so that this search can be limited to the subset of rows within a single class, as discussed in section 3.2.5). Consequently, it is hard to see how the execution time of this type of NN algorithm can be decreased at the macro level.

However, when NN algorithms are executed against *incomplete* datasets this is not the case. For example, suppose the dataset has 99% missing values. In this case many of the rows would be empty or they would have very few known values. Therefore, far fewer row comparisons would be required to find any particular nearest neighbour. Extending this idea it can be seen that as the proportion of missing data increases NN algorithm execution time can, in principle, be decreased by a corresponding proportion. This can be achieved by creating an algorithm that makes the best possible use of the information content within the missingness patterns that exist within the dataset - as described in the following section. ***This approach was first described by the author of this thesis in Solomon et al (2007b) - see Appendix E for the full paper.***

3.2.2 Using Donor Matrices to Speed Up NN Imputation Algorithms

It is argued that the functionality of *all NN imputation algorithms* that process datasets with multivariate missingness patterns must be fundamentally similar to the functionality of the algorithm given in Fig. 3.2, because every row *must be* compared with every other row when searching for each nearest neighbour. This is unavoidable, since it will be necessary to test each row in the dataset to discover whether it can be meaningfully compared with the imputed row before measuring the similarity. Where meaningful comparisons are defined in Fig. 3.1 for the algorithms given here, but they could be defined otherwise, as required.

More precisely, we can say that NN algorithms generally have a time complexity of $O(n^2)$, where n is the number of rows in the data matrix (Dunham, 2003; Aho et al, 1983) - i.e. the algorithm execution time $T(n)$ is proportional to the square of the number of rows in the data matrix, such that $T(n) = cn^2$, where *c* is the constant of proportionality. However, the algorithm given in the following section reduces the size of the constant of proportionality for all missing value datasets, which in turn reduces $T(n)$. An explanation of the ideas underpinning the algorithm's functionality is given below.


List of missingness patterns in the data matrix
Donor matrix constructed by the NN algorithm
Each pattern contains a set of matrix rows
All data matrix rows that have missingness pattern 2
All data matrix rows that have missingness pattern 4
This donor matrix would be constructed by the NN algorithm when imputing the missing values in column 3 of missingness pattern 1.

	1	2	3	4
1	1	0	0	1
2	1	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	0	0

Fig 3.3 - Constructing a donor matrix for multivariate NN imputation

The donor matrix shown in Fig. 3.3 would be constructed by the algorithm when imputing the missing values in column 3 of every data matrix row that has missingness pattern 1, where 1 represents a known value and 0 represents a missing value. Only those data matrix rows that have the pattern shown in rows 2 and 4 can be added to this donor matrix, because these are the *only* rows with a known value in column 3 *and* with the same set of known values as the rows that have pattern 1. It can be seen that pattern 5 also has missing values in column 3. However, the rows with pattern 5 cannot use the same donor matrix as the rows with pattern 1, since only those rows with pattern 4 can be used to construct the donor matrix in this case. Notice also that the rows with pattern 4 can be added to *every* donor matrix constructed by the algorithm, because pattern 4 has a full set of known values. This method is similar to, but faster than, the method implemented in the algorithm given in Fig. 3.2, because the search for each nearest neighbour is carried out within the subset of data matrix rows added to each donor matrix (rather than searching all of the rows in the data matrix every time).

The idea of reducing execution time by creating donor matrices can be applied to almost any NN imputation algorithm. The principle underlying this idea is that the number of row comparisons can almost always be reduced by utilising the information content within the missingness patterns. The example given in Fig. 3.3 shows one way of applying this principle, but many other ways of applying it could be used. For example, when similarity measures that employ variable weighting schemes are used (Dixon, 1979; Huang and Zhu, 2002) the minimum number of matching known values in each pair of rows compared could be specified - so as to avoid excessively weighted row comparisons. Or a specific combination of matching known values could be specified as being essential when searching for donor rows for the imputation of any particular variable. An algorithmic implementation of the approach described in Fig. 3.3 is given below. ***This algorithm is novel and it forms part of the contribution to knowledge made by this dissertation.***

3.2.3 A Fast General Purpose Nearest Neighbour Imputation Algorithm

```
function matrix fast_NN_imputation_in_column ( int c, matrix data )
    missPatternRow missPatt, matchPatt
    dataMatrixRow missRow, donorRow, closestRow
    removeEmptyRowsIn ( data )
    vector patterns = missPatternListFor ( data )

    for i = 1 to num_rows_in ( patterns )
        missPatt = patterns ( i )
        if ( missPatt . patt ( c ) == 0 )
            donors = new vector ( )
            for p = 1 to num_rows_in ( patterns )
                matchPatt = patterns ( p )
                if ( matchPatt . patt ( c ) == 1 )
                    match = true
                    j = 1
                    while ( j <= num_cols_in ( missPatt . patt ) && match == true )
                        if ( missPatt . patt ( j ) == 1 && matchPatt . patt ( j ) == 0 )
                            match = false
                        end if
                        j ++
                    end while
                    if ( match == true )
                        for r = matchPatt . pattStartRow to matchPatt . pattEndRow
                            donors . add_to_end ( data ( r ) )
                        next r
                    end if
                end if
            next p

            for m = missPatt . pattStartRow to missPatt . pattEndRow
                missRow = data ( m )
                minDistance = null
                for d = 1 to num_rows_in ( donors )
                    donorRow = donors ( d )
                    distance = euclideanDistanceBetween ( missRow, donorRow )
                    if ( distance < minDistance || minDistance == null )
                        minDistance = distance
                        closestRow = donorRow
                    end if
                next d
                if ( minDistance != null )
                    missRow ( c ) = closestRow ( c )
                end if
            next m
        end if
    next i
    return data
end function
```

Fig 3.4 – A fast general purpose nearest neighbour imputation algorithm

The algorithm uses the Euclidean distance as a NN similarity measure, but any other measure could be substituted. The parameters passed to the algorithm are the same as those passed to the algorithm given in Fig. 3.2. The function *missPatternListFor (data)* creates and returns the *patterns* vector, which contains a list of *missPatternRow* objects, which represent the missingness patterns in the data matrix. Where each *missPatternRow* object has the following attributes;

(1) A binary array *patt()* representing the missingness pattern.

(2) *pattStartRow* which gives the first row number of the pattern in the *data* matrix.

(3) *pattEndRow* which gives the last row number of the pattern in the *data* matrix.

Note that the algorithm requires the *data* matrix to be sorted into missingness pattern order (all rows with the same missingness pattern must be adjacent) so that the *missPatternRow* objects can be utilised during the processing.

3.2.4 Performance Evaluation Using Simulated Missing Value Datasets

It is important to note that the NN algorithms given in Fig. 3.2 and Fig. 3.4 will produce *the same set of imputed values* when they are executed against the same *data* matrix - provided that the rows in this matrix are sorted into the same order (matrix row order affects the values imputed by NN algorithms). In other words, the algorithm given in Fig. 3.4 is simply an enhanced version of the algorithm given in Fig. 3.2.

The pseudo-code versions of the two algorithms have been implemented using the Microsoft C# programming language (see Appendix C). This section explains how the execution times of the C# versions of the algorithms were compared by using them to impute missing values in a set of randomly generated datasets. The results of the experiments are given in table 3.1, below.

<u>The experiments were designed to try to answer the following key questions;</u>

(1) Would the method of creating donor matrices (implemented in the Fig. 3.4 algorithm) decrease NN algorithm execution time?

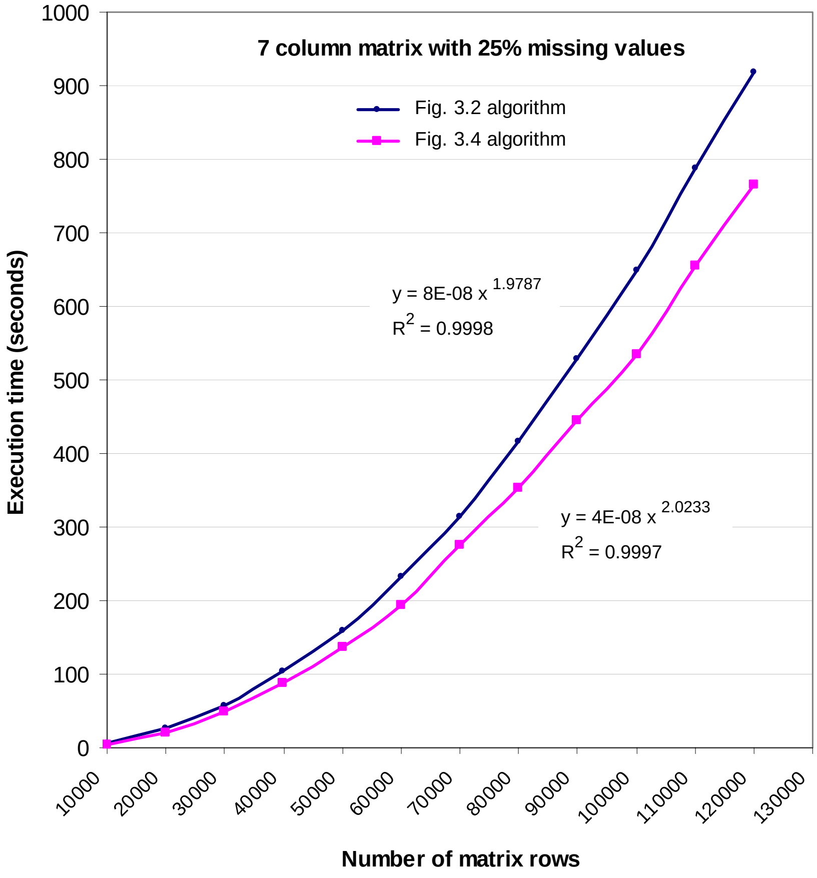
(2) Would the execution times of either of the two algorithms decrease as the proportion of missing data in the matrix was increased?

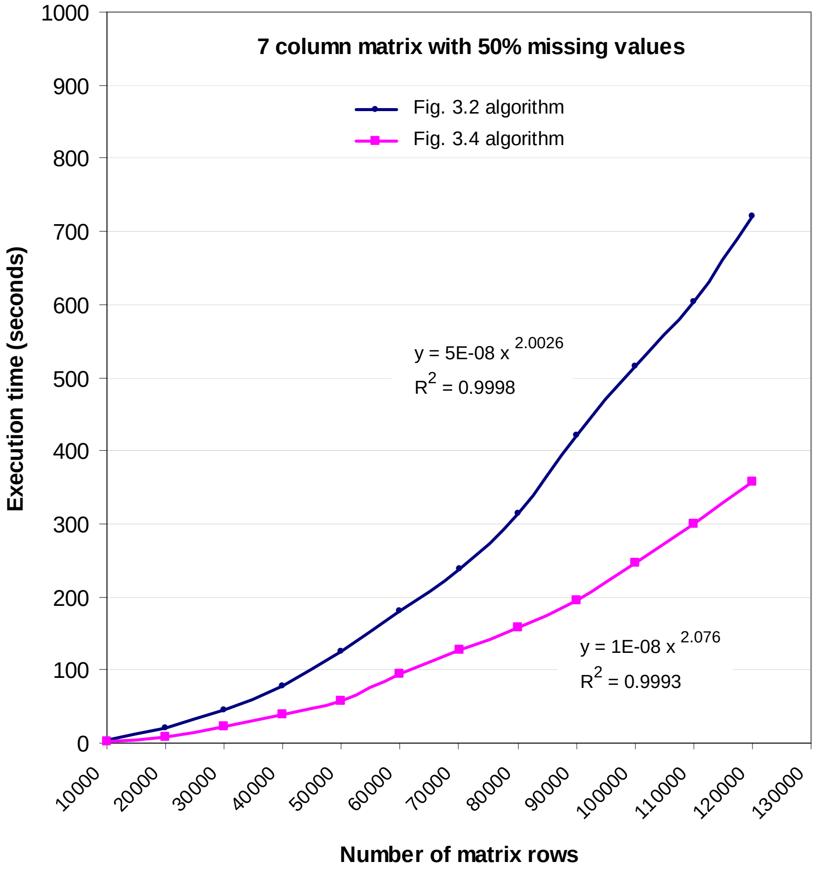
Table 3.1 - Comparison of execution times for the two NN algorithms

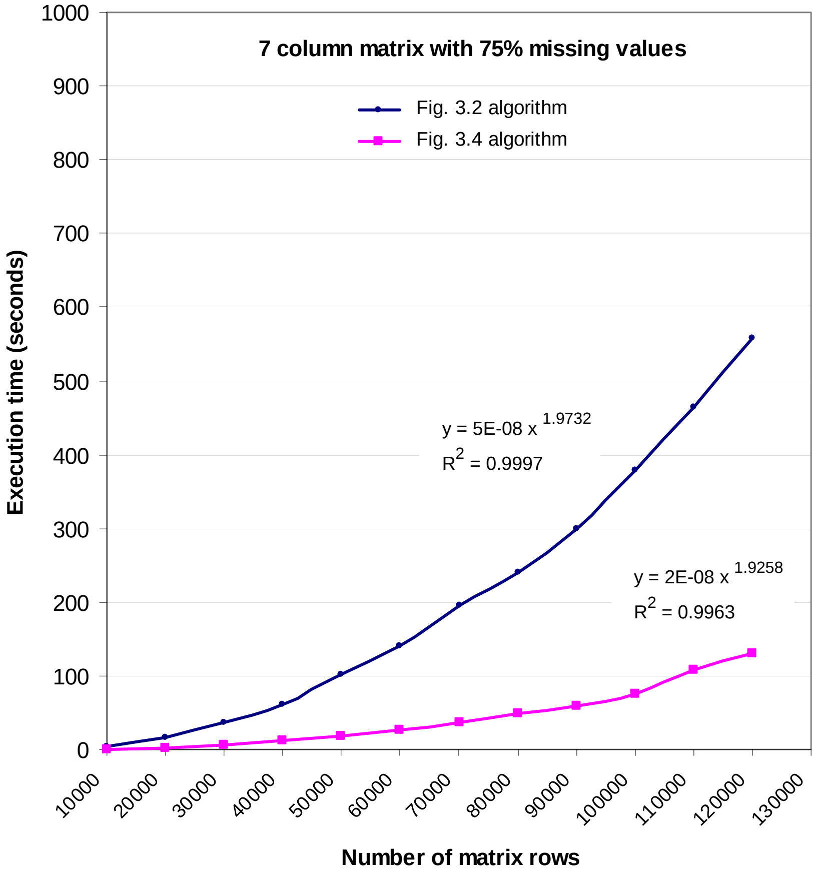
Number of rows in the data matrix	% of missing values in the data matrix	Fig. 3.2 algorithm execution time (seconds)	Fig. 3.4 algorithm execution time (seconds)
10,000	25	7	5
	50	5	2
	75	4	1
20,000	25	26	21
	50	20	9
	75	16	3
30,000	25	58	50
	50	46	22
	75	37	7
40,000	25	104	87
	50	78	38
	75	62	11
50,000	25	159	136
	50	125	58
	75	102	18
60,000	25	233	193
	50	181	94
	75	140	29
70,000	25	315	275
	50	238	127
	75	195	34
80,000	25	417	353
	50	314	159
	75	241	45
90,000	25	529	445
	50	421	195
	75	300	57
100,000	25	649	534
	50	515	247
	75	379	72
110,000	25	787	655
	50	603	300
	75	464	95
120,000	25	918	765
	50	720	357
	75	559	118

The experiments were performed against 12 sets of randomly generated data matrices containing between 10,000 and 120,000 rows, as shown in column 1 of table 3.1. The values inserted into these matrices were randomly generated integers in the range 1 to 100, where 25%, 50% and 75% of these values were randomly deleted in three stages. This process generated 36 different matrices, as shown in column 2 of table 3.1. Each of these matrices contained 7 columns, where the missing values in column 1 were imputed for every experiment.

Each matrix contained the maximum possible number of missingness patterns. Generally, the maximum number of patterns in any matrix equals 2^n, where n is the number of columns in the matrix. In this case each matrix contained 7 columns, therefore 128 patterns were added to each matrix - where these patterns were balanced and evenly distributed across the rows in each matrix, because the missing values were deleted completely at random from across the entire matrix. The algorithms given in Fig. 3.2 and Fig. 3.4 were executed against all 36 of the matrices, thus creating 72 sets of experimental results, as shown in columns 3 and 4 of table 3.1. The results given in table 3.1, above are presented much more clearly in line chart form, below.




7 column matrix with 50% missing values
Fig. 3.2 algorithm
Fig. 3.4 algorithm
y = 5E-08 x 2.0026
R2 = 0.9998
y = 1E-08 x 2.076
R2 = 0.9993
Execution time (seconds)
0
100
200
300
400
500
600
700
800
900
1000
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
130000
Number of matrix rows


7 column matrix with 75% missing values
Fig. 3.2 algorithm
Fig. 3.4 algorithm
y = 5E-08 x 1.9732
R2 = 0.9997
y = 2E-08 x 1.9258
R2 = 0.9963
Execution time (seconds)
0
100
200
300
400
500
600
700
800
900
1000
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
110000
120000
130000
Number of matrix rows

- The line charts given above show the comparative execution times (in seconds) for the algorithms given in Fig. 3.2 and Fig. 3.4. Algorithm execution times are shown on the y axis of each chart. The number of rows in the data matrices are shown on the x axis.

- The charts show that the Fig. 3.4 algorithm executed more quickly than the Fig. 3.2 algorithm for every experiment - i.e. the method of creating donor matrices implemented as part of the Fig. 3.4 algorithm did in fact decrease the execution times. It can be seen that this performance advantage increases as the proportion of missing data increases, with the largest difference between the algorithm execution times occurring for the largest input matrix shown on the final chart.

- The execution time increases for both algorithms as the number of matrix rows increases, as expected. However, execution time also decreases for both algorithms as the proportion of missing data in each matrix increases - i.e. the fastest execution times for both algorithms are shown on the final chart, where each matrix has 75% missing values.

- The regression equations shown on the charts are very close to the expected $T(n) = cn^2$ results (see section 3.2.2), with the very small differences in the expected exponents perhaps explained by the small samples (12 results) used for the regression calculations.

- The very high R^2 values suggested that the regression equations could be used to predict algorithm execution times for larger matrices with some confidence - e.g. the final chart shows that the equation for the Fig. 3.2 algorithm had an R^2 value of 0.9997. Some calculations were performed and the predicted execution times are shown in table 3.2, below. It can be seen that the method of creating donor matrices decreases NN algorithm execution time even for quite large matrices, and that this performance advantage increases markedly as the proportion of missing values in each data matrix is increased.

Table 3.2 - Predicted execution times (to the nearest hour) for the two algorithms

Number of rows in the data matrix	% of missing values in the data matrix	Fig. 3.2 algorithm execution time (nearest hour)	Fig. 3.4 algorithm execution time (nearest hour)
1,000,000	25 50 75	17 14 10	15 8 2
2,000,000	25 50 75	65 58 38	62 33 8
3,000,000	25 50 75	146 130 84	142 78 17
4,000,000	25 50 75	257 231 148	253 141 29

3.2.5 Performance Evaluation Using Two Survey Datasets

The experiments described in the previous section were performed against simulated datasets. However, it is also important to compare the execution times of the Fig. 3.2 and Fig. 3.4 algorithms using real datasets. For example, the simulated datasets all contained a full set of well balanced and evenly distributed missingness patterns, but this can hardly be expected to occur in practice. This point is crucial for the evaluation of the Fig. 3.4 algorithm, since the missingness pattern structure is used to create the donor matrices - i.e. the relative sizes and distributions of the missingness patterns are the key factors affecting the execution time of the Fig. 3.4 algorithm.

The experiments described below compare the execution times of the two algorithms when they are executed against segmented datasets. This is an important consideration, because in practice datasets are often segmented by class, so that the search for each nearest neighbour can be limited to the subset of rows within a single class (Chen and Shao, 2000). The required classes can be created by segmenting the dataset using a fully observed categorical variable (as for the experiments described below) or by dividing the dataset into numeric class intervals using a fully observed numeric variable.

This approach can improve the quality of the imputed values *and* reduce the execution time of the NN algorithm. Execution time is reduced because the search for each nearest neighbour takes place within a smaller number of dataset rows - i.e. the subset of rows contained within a single class. The quality of the imputed values can be improved because they are taken (copied) from a set of donor rows that share the same characteristics - e.g. a set of people within a specified age group etc.

Description of the experimental datasets

Table 3.3, below describes the datasets that were used to perform the experiments. A much more detailed description of the datasets is given in chapter six. Each of the variables described in table 3.3 represents one of the columns inserted into the experimental data matrices. The ten variables taken together represent a data matrix row which describes a Firm (a business enterprise) within the UK. The first four variables listed were fully observed, but the other six (the currency variables) all had large proportions of missing data.

Table 3.3 - Description of the variables (data matrix columns) in the experimental datasets

Variable name	Data type	Variable description
UKSIC_Category	*Integer*	An integer representation of a categorical alphanumeric code which defines the commercial activities carried out by each Firm, such as *"Publishing of software"* etc.
Employees	*Integer*	Specifies the number of people employed by each Firm
Easting **Northing**	*Integer*	Pinpoints the geographical location of each Firm on the UK map, using two UK Ordnance Survey (OS) mapping co-ordinates.
Sales **Payroll** **Depreciation** **DirectorPay** **NetWorth** **PBT (Profit Before Tax)**	*Currency*	Six numeric variables that describe each Firm's financial situation. These variables all had large proportions of missing values. The Payroll variable was imputed for every experiment.

Description of the experimental process

The experiments were performed against two separate datasets, each of which contained the ten variables described in table 3.3 - i.e. both datasets had the same column structure, but the set of rows contained within each dataset differed, as follows. The first (and the largest) dataset described 1,128,463 MICRO Firms, which are defined as those Firms with less than 10 employees. The second (much smaller) dataset described 271,955 SMALL Firms, which are defined as those Firms that have between 10 and 49 employees.

The **UKSIC_Category** variable was used to segment both of the datasets at four different levels of granularity (see chapter six for a detailed description of how this was achieved). Where the lowest level of granularity created the smallest number of segments and the highest level of granularity created the largest number of segments. This process created eight different data matrices, as shown in column 5 of table 3.4, below. The algorithms described in Fig. 3.2 and Fig. 3.4 were executed against each of these matrices, where the missing **Payroll** values were imputed for every experiment. This process created 16 sets of experimental results, as shown in the two rightmost columns of table 3.4.

It is important to note that the Fig. 3.2 and Fig. 3.4 algorithms were amended so that the search for each nearest neighbour could be limited to the subset of rows within a single **UKSIC_Category**. The amendments made to the algorithms were quite simple, but they are not shown in the pseudo-code given in Fig. 3.2 and Fig. 3.4 for reasons of clarity. The experimental results are tabulated below.

The experiments were designed to try to answer the following key questions;

(1) Would the method of creating donor matrices (implemented in the Fig. 3.4 algorithm) decrease algorithm execution time *for large segmented datasets*?

(2) Would dividing the data matrix into an increasingly large number of segments *steadily* decrease the execution times of either of the two algorithms?

These questions need to be addressed because in practice datasets are often segmented by class, so that the search for each nearest neighbour can be limited to the subset of rows within a single class (Chen and Shao, 2000). Therefore, it is important to compare the performance the Fig 3.2 and Fig 3.4 algorithms when they are executed against segmented datasets.

Table 3.4 - Comparison of algorithm execution times using segmented datasets

Description of experimental dataset	Number of rows in the data matrix	% of missing values in the data matrix	Number of missingness patterns	Number of category segments	Fig. 3.2 algorithm execution time (minutes)	Fig. 3.4 algorithm execution time (minutes)
MICRO Firms (less than 10 employees)	1,128,463	61.72%	28	58 203 412 488	312 119 53 47	44 (0.14) 16 (0.13) 9 (0.17) 8 (0.17)
SMALL Firms (10 to 49 employees)	271,955	54.97%	27	57 200 409 485	17 6 3 3	6 (0.35) 2 (0.33) 1 (0.33) 1 (0.33)

The figures given in the three rightmost columns show that dividing the matrix into an increasingly large number of segments steadily decreased the execution time required for both algorithms. This occurred because as the number of segments was increased the search for each nearest neighbour took place within a smaller number of rows. The figures in brackets given in the rightmost column show the improvement in performance offered by the Fig. 3.4 algorithm for both datasets. For example, the first row of figures show that the Fig. 3.4 algorithm executed in just under one seventh of the time (given to the nearest minute) taken by the Fig. 3.2 algorithm. It can be seen that these performance improvements are similar within each dataset, regardless of the number of segments created.

Overall conclusion for the NN algorithm performance evaluation experiments

The method of creating imputation donor matrices implemented in the Fig. 3.4 algorithm decreased the execution time of the Fig. 3.2 algorithm for both simulated and real datasets. This performance advantage increased markedly as the proportion of missing values in the data matrix was increased.

3.3 Summary

This chapter has explained the ideas underpinning the functionality of a general purpose NN imputation algorithm that has been devised by the author, and has shown how these ideas can be used to reduce the execution time of the NN imputation process.

An explanation of how the new algorithm has been implemented in practice has been given and some general rules for the evaluation of any NN imputation procedure have been proposed. A description of the experiments that were performed to evaluate the performance of the new algorithm has also been given, and the experimental results have been presented, analysed and discussed. The ideas and the experimental results presented in this chapter form part of the contribution to knowledge made by this thesis.

This completes the descriptions of the two imputation techniques that have been implemented alongside the proposed imputation evaluation method in the form of an integrated software application. The following chapter gives a detailed description of the proposed evaluation method and explains how it can be used to estimate the predictive accuracy of the imputed values generated by any imputation technique - including the techniques described in this chapter and in chapter two.

Chapter five goes on to describe how the integrated software application was used to experimentally evaluate the reliability and the validity of the proposed imputation evaluation method. Chapter six completes the fulfilment of the project objectives by using the integrated application to assess the feasibility of imputing the missing values in the collaborating company’s dataset.

Chapter Four

A Stochastic Method for Estimating Imputation Accuracy

4. A Stochastic Method for Estimating Imputation Accuracy

This chapter describes the equations and processes which form the basis of the proposed imputation evaluation method and shows how this method can be used to evaluate any imputation technique. The proposed method is compared with the most similar methods found within the literature and it is shown that the proposed method builds on the ideas underpinning the most similar methods, but differs from them in several important respects. ***The functional steps of the proposed method are summarised below,***

1. A small proportion (perhaps up to 5%) of the known values are deleted *at random* from within the variable to be evaluated (which will already have some missing values).
2. Deleted values are recorded just before they are deleted, and a measure of how accurately they have been "put back" is taken when the imputation process is complete.
3. Steps 1 and 2 are repeated several times and the accuracy statistics computed at step 2 are stored after each repetition.
4. The stored statistics are aggregated so that the estimates of imputation accuracy produced will be more statistically reliable.

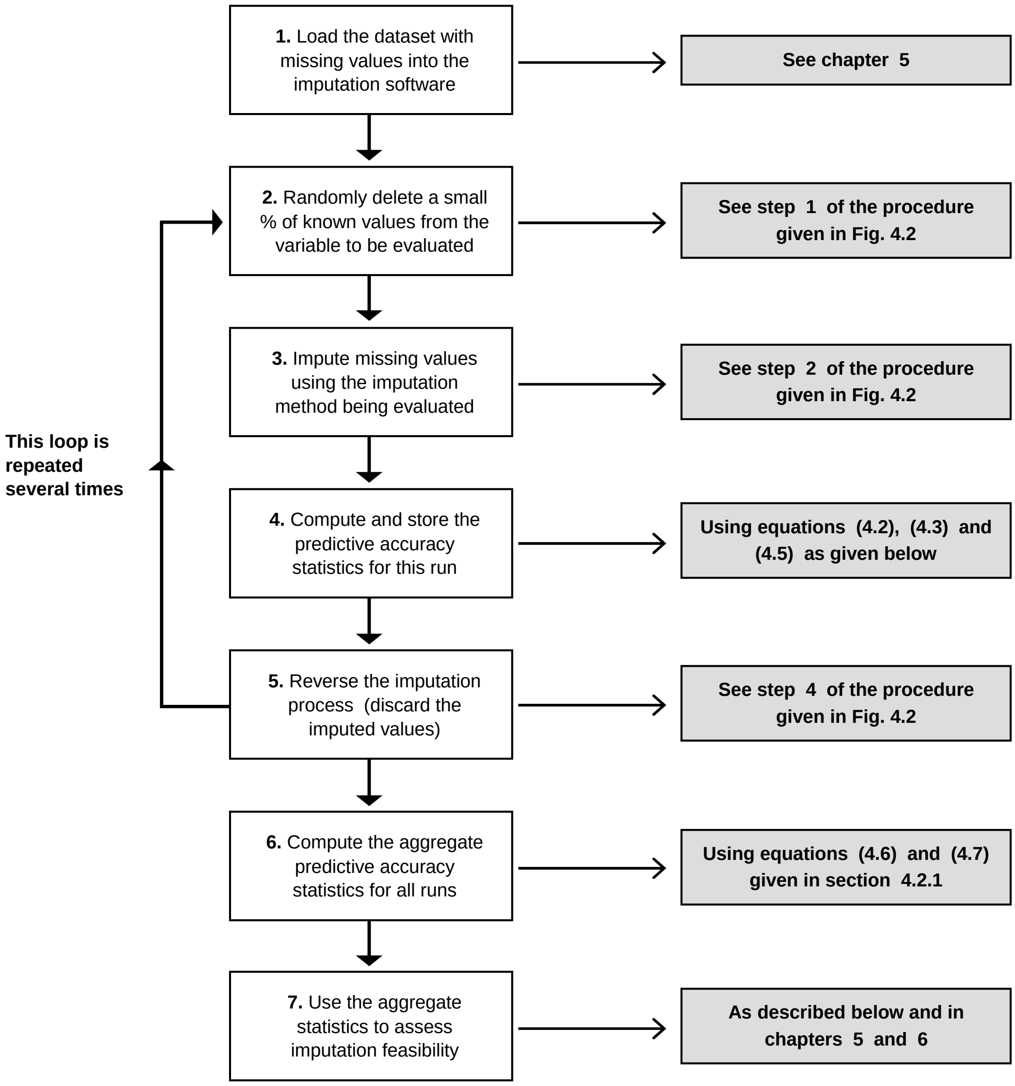
This method can be used to estimate the predictive accuracy of the imputed values for any variable in the missing value dataset, where the required variable is chosen by the user of the software that implements the method. However, the evaluation process can be repeated for all of the variables in the dataset, if required. It is important to emphasise that the method can only *estimate* the accuracy of the imputed values and can never *prove* this accuracy, for the following reason. The true (actual, real) values of the missing data items are by definition, unknown. Therefore, it is impossible to *prove* that any imputation procedure has imputed values accurately, since the true values can never be compared with the imputed values.

The descriptions and explanations given in this chapter form the principal contribution to knowledge made by this thesis. In particular, the equations and procedures described in section 4.2 are novel and the method used to compare the predictive accuracy of imputed values in different data segments is also original.

- **Section 4.1** gives a functional overview of the proposed method with reference to the formal description of the method which follows.
- **Section 4.2** describes the equations and the processes which form the basis of the method and explains how the proposed approach can be applied in practice.
- **Section 4.3** describes the most similar methods found within the literature and compares them with the proposed approach.

4.1 Functional Overview of the Method

The diagram gives a general overview of the sequence of steps that are performed whenever the proposed imputation evaluation method is employed.



1. Load the dataset with missing values into the imputation software
See chapter 5
2. Randomly delete a small % of known values from the variable to be evaluated
See step 1 of the procedure given in Fig. 4.2
3. Impute missing values using the imputation method being evaluated
See step 2 of the procedure given in Fig. 4.2
This loop is repeated several times
4. Compute and store the predictive accuracy statistics for this run
Using equations (4.2), (4.3) and (4.5) as given below
5. Reverse the imputation process (discard the imputed values)
See step 4 of the procedure given in Fig. 4.2
6. Compute the aggregate predictive accuracy statistics for all runs
Using equations (4.6) and (4.7) given in section 4.2.1
7. Use the aggregate statistics to assess imputation feasibility
As described below and in chapters 5 and 6

Fig 4.1 – Functional overview of the proposed imputation evaluation method

Notice in particular the loop shown in steps 2 to 5 - which is given as a procedure in Fig. 4.2. This loop forms a essential part of the method because repeating the stochastic part of the process will produce more statistically reliable estimates of imputation accuracy.

4.2 Description of the Method

Section 4.2.1 gives a formal description of the equations and procedures which form the basis of the proposed method. Section 4.2.2 and 4.2.3 explain how the method can be used to estimate and compare the accuracy of the imputed values in different data segments (sets of related data matrix rows).

4.2.1 Formal Description of the Method: Equations and Procedure

Consider a data matrix Y containing *only* real numbers, such as the matrix shown below. Each column in the matrix stores the values taken by a particular numeric variable. Each row in the matrix stores the values of a set of related numeric variables - such as a statistical observation or a set of values describing the attributes of a particular object.

$$\begin{array}{r|ccccc|}
\boldsymbol{i\ =\ 1} & 1 & 0 & 1 & 0 & 1 \\
\boldsymbol{2} & 1 & 0 & 0 & 1 & 1 \\
\boldsymbol{3} & 0 & 0 & 1 & 1 & 0 \\
\boldsymbol{4} & 1 & 0 & 1 & 0 & 1 \\
\boldsymbol{5} & 1 & 1 & 1 & 1 & 1 \\
\boldsymbol{6} & 1 & 0 & 0 & 1 & 1 \\
 & \boldsymbol{j\ =\ 1} & \boldsymbol{2} & \boldsymbol{3} & \boldsymbol{4} & \boldsymbol{5}
\end{array}$$

The Y matrix has 6 rows and 5 columns

The *known* values are represented by a value of 1

The *missing* values are represented by a value of 0

The rows are indexed as $i = 1 \ to \ n$

The rows are indexed as $j = 1 \ to \ p$

The equations given below can be used to estimate the accuracy of the imputed values in any numeric data matrix of the type shown above when the proposed imputation evaluation method is employed. A description of how these equations can be used to estimate the accuracy of the imputed values generated by any imputation method follows the equations.

***The equations and procedures which follow were devised by the author and they were first described in Solomon et al (2007a) - see Appendix E for the full paper*.**

$$RD_{ij} = \frac{\left| Y_{ij}.trueVal - Y_{ij}.imputedVal \right|}{\left| Y_{ij}.trueVal \right|} \tag{4.1}$$

Where $Y_{ij}.trueVal$ is the true (known) value that was deleted.

and $Y_{ij}.imputedVal$ is the value generated by the imputation process.

$$MRD = \frac{1}{m} \sum_{i \in M} RD_{ij} \tag{4.2}$$

$$SRD = \sqrt{\frac{1}{m} \sum_{i \in M} \left(RD_{ij} - MRD \right)^2} \tag{4.3}$$

$$RZ_{ij} = \frac{RD_{ij} - MRD}{SRD} \tag{4.4}$$

$$MRZ = \frac{1}{z} \sum_{i \in Z} RZ_{ij} \tag{4.5}$$

Where MRD is the acronym for the ***M**ean **R**elative **D**ifference.

and SRD is the acronym for the **S**tandard deviation of the **R**elative **D**ifference.

and RZ is the **R**elative difference ***Z*** score, where the MRZ gives the ***M**ean RZ value.

and j is the *Y* matrix column from which the values were randomly deleted and “put back”

and $M = \{ r_1 ,.... r_m \}$ is the set of rows in Y with a deleted value.

and i indexes the set of rows in M (which will differ for every execution of the method).

and $Z = \{ r_1 ,.... r_z \}$ is the set of rows in Y that have an RD outlier value.

The *RD* gives the relative differences between the deleted (known) values and the imputed values in column j of Y - i.e. if any particular RD_{ij} value equals zero then the deleted value has been "put back" with 100% accuracy. The *MRD* gives the mean *RD* value - where smaller *MRD* values indicate greater accuracy within the imputed values as a whole. The *SRD* gives the standard deviation of the *RD* - where larger *SRD* values indicate greater variability of imputation accuracy. Values of RZ within any required range, such as ± 3 *SRD's* above and below the *MRD*, define *RD* outliers. Essentially, the *RZ* is a measure of the number of *SRD's* by which any particular value of RD deviates from the *MRD* - where the set of *RD* values are assumed to be approximately normally distributed for this purpose. The *MRZ* gives the mean RZ value - where $PZ = z / m$ gives the proportion of RD outliers found within the set M.

The main procedure: Implementing the repetitive, stochastic evaluation process

The imputation method being evaluated is executed T times and the values of equations (4.2), (4.3) and (4.5) are computed after each execution. The aggregate values are then used to estimate imputation accuracy after the loop terminates. This repetitive process should produce more statistically reliable estimates of imputation accuracy, as follows;

```
for t = 1 to T
    1. Randomly delete a small proportion of the known values in column j of Y to create matrix $Y_t$
    2. Impute the missing values in $Y_t$ using the imputation method being evaluated
    3. Compute a set of evaluation statistics $S_t$ using $Y_t$ - i.e. compute values for (4.2), (4.3) and (4.5)
    4. Restore the Y matrix to its original condition (as it was before starting the whole process)
next t
```

Fig 4.2 - Functional outline of the repetitive, stochastic imputation evaluation process

The procedure produces *a set of sets* of imputation accuracy statistics $\{ S_1 \ldots S_T \}$ which describe the values that were deleted from the Y matrix and then "put back" into the matrices $\{ Y_1 \ldots Y_T \}$ by the imputation method being evaluated. The means $\hat{\mu}(k)$ and the standard deviations $\hat{\sigma}(k)$ of the imputation accuracy statistics in each of the sets $\{ S_1 \ldots S_T \}$ can then be used to estimate the accuracy of the imputed values, as follows;

$$\hat{\mu}(k) = \frac{1}{T}\sum_{t=1}^{T} S_{t(k)} \qquad (4.6) \qquad\qquad \hat{\sigma}(k) = \sqrt{\frac{1}{T}\sum_{t=1}^{T} \left(S_{t(k)} - \hat{\mu}(k)\right)^2} \qquad (4.7)$$

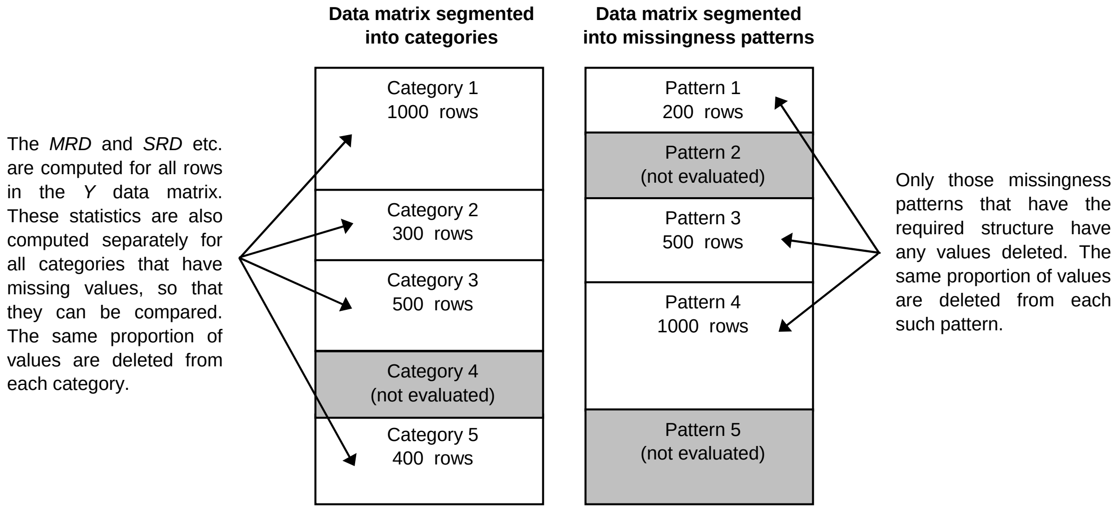
Where k indexes the members of each set S_t - i.e. the values of equations (4.2), (4.3) and (4.5) which are computed at step 3 of the Fig. 4.2 procedure. For example, $\hat{\mu}(1)$ will give the mean of the MRD values that were computed at step 3 of the procedure.

Repetition of the imputation process reflects the uncertainty within the imputed values

Randomly deleting values from column *j* of the *Y* matrix will produce a unique set of imputation accuracy statistics for each iteration of the procedure given in Fig. 4.2. This repetitive, stochastic approach has formed an important part of imputation theory ever since Rubin (1978) first proposed the idea of multiple imputation (see section 4.3.3). The key idea underpinning all such approaches is that the repetitive stochastic nature of the overall process will reflect the uncertainty that exists within the imputed values. And this uncertainty will always be present to some degree, because it is impossible to *prove* that missing values have been imputed accurately. For example, if large, but very similar, values of the *MRD* appeared for many iterations of the loop given in Fig. 4.2, then the imputation process would have high uncertainty, but this uncertainty would not depend on the set of values that were missing.

4.2.2 Estimating the Accuracy of the Imputed Values in Data Segments

Generally, smaller *MRD* values indicate greater accuracy within the imputed values as a whole. However, larger *SRD* values show that this accuracy is highly variable, and therefore it may be localised within one or more clearly defined data segments (sets of related rows) within the variable being evaluated - such as a particular set of missingness patterns or a set of categories with clearly defined boundaries. In these cases it can be useful to discover the distribution of the *RD* values across these data segments, or to discover whether some segments contain higher proportions of *RD* outlier values than others. To achieve this it is essential to delete the *same proportion* of values from each segment before estimating the accuracy of the imputed values, so that each segment can be assessed equally. Deleting the same proportion of values from each segment will also preserve the relative number of missing values in each data segment, which will be particularly important when the data within the *Y* matrix is MAR.


Data matrix segmented into categories
Data matrix segmented into missingness patterns
Category 1 1000 rows
Category 2 300 rows
Category 3 500 rows
Category 4 (not evaluated)
Category 5 400 rows
Pattern 1 200 rows
Pattern 2 (not evaluated)
Pattern 3 500 rows
Pattern 4 1000 rows
Pattern 5 (not evaluated)

The *MRD* and *SRD* etc. are computed for all rows in the *Y* data matrix. These statistics are also computed separately for all categories that have missing values, so that they can be compared. The same proportion of values are deleted from each category.

Only those missingness patterns that have the required structure have any values deleted. The same proportion of values are deleted from each such pattern.

Fig 4.3 - Estimating the accuracy of the imputed values in different data segments

Estimating the accuracy of the imputed values in each data segment separately allows the imputation accuracy statistics for each data segment to be compared and analysed. For example, a comparison of the statistics for each segment could reveal that the poor imputation accuracy in a particular segment has been caused by one or two extreme outlier RD values (see equations (4.4) and (4.5)). In such a case the owner of the missing value dataset might decide to examine the data rows in the offending segment in detail, to discover why this has occurred. For example, the collaborating company (TBR, see chapter 6) considered this type of information to be very useful when they were compiling reports describing the Firms in different UKSIC categories - i.e. they wanted to know whether the imputed values in some UKSIC categories were more accurate than in others (they might decide not to impute the missing values in the least accurately imputed categories under any circumstances).

When the dataset is segmented by category the process of deleting the same proportion of values from each category can be achieved by counting the number of rows in each category - i.e. larger categories will have more values deleted from them. The random deletion process can then be achieved by sorting the data matrix by category (as shown above), then deleting at random from within each category (each set of adjacent rows). However, it should be ensured that the proportion of known values that are retained within each category is sufficient to support the imputation process (where this is required, depending on the imputation method used, and on the proportion of *truly missing values* in each category).

However, randomly deleting the same proportion of values from each of the *missingness patterns* to be evaluated is far from simple, because the algorithm used to perform these deletions must ensure that the process does not create any *new* (and hence artificial) missingness patterns within the data matrix. This process will allow the accuracy of the imputed values in each evaluated pattern to be assessed equally, which will be particularly important when data is MAR. This missingness pattern comparison process could have significant benefits for the evaluation of regression based imputation methods (such as the EM algorithm), because it allows the owners of the missing value dataset to see which missingness patterns produce the best and worst regression equations (and therefore the most and least reliable imputed values) for any particular dataset.

A description of algorithm used to delete the same proportion of values from each missingness pattern (which is the most procedurally complex part of the proposed approach), is given below. The algorithm and the equations and procedure which follow it were first described by the author in Solomon et al (2007a).

```
function  matrix  balanced_ random_deletion_across_all_evaluated_patterns_in_the_data_matrix
            ( matrix  data,  vector  patterns,  int  c,  int  d )
    dataMatrixRow  data_row
    vector  match_rows
    missPatternRow  patt
    integer  rows_to_add,  random_row
    boolean  match

    for  i  =  1  to  num_rows_in ( patterns )
        patt  =  patterns ( i )
        if  ( patt ( c )  ==  missing  &&  some_values_are_present_in ( patt )  ==  true )
            match_rows  =  new  vector ( )
            for  k  =  1  to  num_rows_in ( data )
                data_row  =  data ( k )
                if  ( data_row ( c )  ==  present )
                    match  =  true
                    for  j  =  1  to  num_columns_in ( data )
                        if  ( patt ( j )  ==  present  &&  data_row ( j )  ==  missing )
                            match  =  false
                        end  if
                    next  j
                    if  ( match  ==  true )
                        match_rows . Add_To_End ( k )
                    end  if
                end  if
            next  k

            rows_to_add  =  ( d / 100 )  *  num_rows_in ( patt )
            if  ( num_rows_in ( match_rows )  >  rows_to_add  *  2 )
                for  k  =  1  to  rows_to_add
                    random_row  =  Random ( 1,  num_rows_in ( match_rows ) )
                    data_row  =  data ( match_rows ( random_row ) )
                    match_rows . Remove_Row ( random_row )
                    for  j  =  1  to  num_columns_in ( data )
                        if  ( patt ( j )  ==  missing )
                            data_row ( j )  =  missing
                        end  if
                    next  j
                next  k
            end  if
        end  if
    next  i
    return  data
end  function
```

Fig 4.4 - An algorithm to perform balanced random deletions across missingness patterns

The following parameters are passed to the algorithm; **(1)** *matrix data* is a multivariate data matrix containing several different missingness patterns. **(2)** *vector patterns* contains a list of the missingness patterns in the *data* matrix. **(3)** *int c* is the *data* matrix column from which the values will be deleted. **(4)** *int d* is the proportion of missing values to be added to each of the patterns to be evaluated - i.e. the number of rows in each pattern that has missing values in column *c* will be increased by *d%* (which is equivalent to deleting the values).

Inside the code: An explanation of the functionality of the pseudo-code algorithm

The algorithm increases the number of rows in each of the missingness patterns to be evaluated by the same proportion. This is achieved by transferring *data* matrix rows from one pattern to another. For example, when deleting from *data* column one the algorithm might transfer a *data* row by changing its pattern from "1111" to "0111". However, the *data* rows transferred must have known values in the same columns as the *data* rows in the pattern to be evaluated (the pattern with rows added to it). For example, if the pattern to be evaluated was "0011", then *data* rows with the pattern "1100" could *not be* transferred to that pattern, but *data* rows with the pattern "1111", could be transferred to it.

The final pair of nested *for* loops perform the random row transfers. However, this can only be achieved for a particular pattern if the number of *data* rows available for transfer (as stored in the *match_rows* vector) is more than double the number of rows to be added to the pattern to be evaluated. This ensures the stochastic nature of the row transfer process under repeated executions, which is an essential part of the proposed method. If the number of *data* rows available for transfer is too small, then the accuracy of the imputed values in the pattern to be evaluated cannot be estimated separately. However, this should occur very rarely - i.e. when the proportion of missing values in column *c* is large (perhaps above 80%), or when the number of missingness patterns is a small proportion of the number of possible patterns. In these cases the method of comparing imputation accuracy across a clearly defined set of categories should be preferred.

4.2.3 Comparing the Accuracy of the Imputed Values in Data Segments

The idea of performing balanced random deletions across data segments (the set of categories or missingness patterns in the data matrix) allows the accuracy of the imputed values in each data segment to be assessed equally - i.e. the imputation accuracy statistics returned by equations (4.6) and (4.7) can be computed correctly for each different segment. To achieve this the procedure given in Fig. 4.2 needs to be modified as follows;

```
for  t = 1 to T
    1. Randomly delete the same proportion of values from each data segment in Y to create matrix $Y_t$
    2. Impute the missing values in $Y_t$ using the imputation method being evaluated
    3. Compute a set of evaluation statistics $S_t$ using $Y_t$ - i.e. compute values for (4.2), (4.3) and (4.5)
    4. for each data segment d contained in Y
            Compute a set of evaluation statistics $S_{td}$ - i.e. compute values for (4.2), (4.3) and (4.5)
       next  d
    5. Restore the Y matrix to its original condition (as it was before starting the whole process)
next  t
```

Fig 4.5 - Adjusted functional outline for the proposed imputation evaluation method

Steps 2, 3 and 5 are identical to those given in the Fig. 4.2 procedure. Step 3 is retained so that equations (4.6) and (4.7) can still be used to estimate the accuracy of the imputed values within the data matrix *as a whole*. Generally, the procedure provides the same imputation evaluation functionality as the procedure given in Fig. 4.2, while adding the following improvements;

- ***Step 1 has been enhanced*** - So that it will execute the procedures described in the preceding section - i.e. Step 1 will now delete the *same proportion* of values from each data segment. This will allow the accuracy of the imputed values in each data segment to be assessed equally.

- ***Step 4 has been inserted*** - This will produce a separate set of imputation accuracy statistics for each data segment - i.e. equations (4.2), (4.3) and (4.5) are computed *for each data segment d contained in Y*. The following equations can then be used to estimate the accuracy of the imputed values in each data segment;

$$\hat{\mu}_d(k) = \frac{1}{T}\sum_{t=1}^{T} S_{td}(k) \qquad (4.8) \qquad\qquad \hat{\sigma}_d(k) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(S_{td}(k) - \hat{\mu}_d(k)\right)^2} \qquad (4.9)$$

These equations are very similar to equations (4.6) and (4.7). The only difference being the addition of the qualifier d which specifies the data segment that $\hat{\mu}_d(k)$ and $\hat{\sigma}_d(k)$ refer to. These statistics can be computed for all segments in Y using a loop similar to the one given in Step 4 of the above procedure.

4.3 Comparative Evaluation of Similar Methods

This section compares the proposed method with the most similar methods found within the literature. Section 4.3.1 describes how the general idea of estimating imputation accuracy has been applied by other researchers. Section 4.3.2 and 4.3.3 describe the methods that are most similar to the proposed method - i.e. uncertainty estimation methods. Section 4.3.4 discusses the limitations of uncertainty estimation methods. Section 4.3.5 compares uncertainty estimation methods with the proposed method and discusses the similarities and the differences between these approaches.

4.3.1 Similar Approaches Used by Other Researchers

The general idea of evaluating imputation methods by measuring how accurately a set of deleted values have been "put back" has been frequently employed to evaluate the success of various new, and existing, imputation methods. However, researchers who employ this approach generally; **(1)** Create *simulated* datasets that have no missing values. **(2)** Delete values at random from these datasets, so that replacement accuracy can be measured. The following examples are typical of the approaches used by other researchers.

- Tseng et al (2003) use two different methods of random data generation to create simulated datasets that have no missing values. Values are then deleted at random in increasingly large proportions from a single variable within these datasets. A new imputation method is then employed, and replacement accuracy is used to demonstrate that the new method performs better than some other methods, when it is applied to a specific type of dataset.

- Wasito and Mirkin (2005) use a very similar approach to Tseng et al (2003). The principal difference being that values are deleted at random across the whole data matrix, rather than from just one variable. Again, the idea is to demonstrate that the new imputation method devised by the authors performs better than some other methods when applied to a specific type of randomly generated dataset.

- Starick and Watson (2006) employ a more sophisticated method of generating the simulated dataset. Firstly, a sample of complete observations are taken from a real (not simulated) dataset - i.e. those dataset rows that have no missing values are copied. Secondly, values are deleted from the newly created dataset in such a way that the missing data mechanism (assumed to be MAR) within the source dataset is modelled. Various imputation methods are then executed and the replacement accuracy achieved by each method is compared using various statistical techniques - which are partly based on the evaluation criteria proposed by Chambers (2001).

The proposed method has some similarity to the approaches described above. However, these approaches also differ in several important respects. Firstly, they are usually applied to simulated, rather than real, datasets. Secondly, they are not devised to be general purpose imputation evaluation techniques - i.e. they are usually designed for a specific purpose (as an incidental part of a larger project), or to be applied to a specific type of dataset. Thirdly, the idea of measuring replacement accuracy using the statistics generated via a repetitive stochastic algorithm is not used. And finally, the idea of comparing the accuracy of the imputed values in different data segments is not used (this idea was first proposed by the author in Solomon et al (2007a)).

The proposed method is most similar to uncertainty estimation methods

The proposed method is most similar to those imputation evaluation methods that estimate the uncertainty inherent within the imputed values. Several such methods have been proposed, and a good general overview of these can be found in Little and Rubin (2002), with more detailed discussions given in Lee et al (2002) and Shao (2002). The following sections give a compact and straightforward summary of the functionality of the uncertainty estimation methods that are most similar to the proposed method. And it can be seen that the procedures described below have some similarity with the procedure given in Fig. 4.2, above.

4.3.2 Bootstrap and Jackknife Uncertainty Estimation

Consider a variable $Y = (y_1, \dots y_n)$ where some of the values are missing and where *Y* can be any of the variables in a multivariate dataset containing different missingness patterns. The Bootstrap variance estimation method (Efron, 1994; Shao and Sitter, 1996) and the Jackknife variance estimation method (Rao and Shao, 1992; Rao, 1996a; Chen and Shao, 2001) can be used to estimate the uncertainty created by imputing the missing values in *Y*. Where uncertainty is estimated by computing the variance of a set of parameter point estimates (such as the mean), which describe a set of samples taken from *Y*, as follows;

```
for  b = 1 to B
    1. Create a new bootstrap sample $Y_b$ by randomly selecting some rows (with replacement) from Y
    2. Impute the missing values in $Y_b$ using the imputation method being evaluated
    3. Compute a parameter point estimate $\hat{\theta}_b$ which describes the values in $Y_b$
    4. Restore the matrix that contains Y to its original condition (as it was before starting this process)
next  b
```

Fig 4.6 - Estimating imputation uncertainty using the Bootstrap method

The procedure given in Fig. 4.6, above produces a set of estimates $\{\hat{\theta}_1 \ldots \hat{\theta}_B\}$ which describe the Bootstrap samples $\{Y_1 \ldots Y_B\}$. The Bootstrap estimate of the variance $\hat{V}_{boot}$ can then be used to estimate imputation uncertainty, as follows;

$$\hat{V}_{boot} = \frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\theta}_b - \hat{\theta}_{boot}\right)^2 \qquad \text{where} \qquad \hat{\theta}_{boot} = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}_b$$

The Fig. 4.6 procedure imputes the missing Y_b values B times, then computes the variance of the resulting set of $\hat{\theta}_b$ estimates. The Jackknife variance estimation method is quite similar. The difference lies in the method used to create the set of samples, which in turn requires a more complex method of computing the variance, as described below;

```
1. Impute the missing values in $Y$ using the imputation method being evaluated
2. Compute a parameter point estimate $\hat{\theta}$ which describes the values in $Y$
for $j = 1$ to $n$
    1. Delete value $j$ (matrix row $j$) from $Y$ to create a new jackknife sample $Y_{(\backslash j)}$
    2. Impute the missing values in $Y_{(\backslash j)}$ using the imputation method being evaluated
    3. Compute the same parameter estimate as above $\hat{\theta}_{(\backslash j)}$ which describes the values in $Y_{(\backslash j)}$
    4. Restore the matrix that contains $Y$ to its original condition (as it was before starting this process)
next $j$
```

Fig 4.7 - Estimating imputation uncertainty using the Jackknife method

The procedure given in Fig. 4.7 produces a set of estimates $\{\hat{\theta}_{(\backslash 1)} \ldots \hat{\theta}_{(\backslash n)}\}$ which describe the Jackknife samples $\{Y_{(\backslash 1)} \ldots Y_{(\backslash n)}\}$. Where n is the number of values in $Y = (y_1, \ldots y_n)$, i.e. the number of rows in the multivariate data matrix which contains the variable Y. The Jackknife estimate of the variance $\hat{V}_{jack}$ can then be used to estimate imputation uncertainty, as follows;

$$\hat{V}_{jack} = \frac{1}{n(n-1)}\sum_{j=1}^{n}\left(\tilde{\theta}_j - \hat{\theta}_{jack}\right)^2$$

$$\text{where} \qquad \tilde{\theta}_j = n\hat{\theta} - (n-1)\,\hat{\theta}_{(\backslash j)} \qquad \text{and} \qquad \hat{\theta}_{jack} = \frac{1}{n}\sum_{j=1}^{n}\tilde{\theta}_j$$

The Jackknife method can be much more computationally intensive than the Bootstrap method when n is large, since the imputation process must be repeated n times. However, in these cases the execution time of the Jackknife procedure can be decreased by deleting a *set of values* (rather than just one) at each iteration of the loop. Although this would dilute the fundamental idea underpinning the entire Jackknife approach, so perhaps the Bootstrap method should be preferred in such cases.

4.3.3 Multiple Imputation

Consider a variable $Y = \left(y_1, \dots y_n\right)$ where some of the values are missing and where Y can be any of the variables in a multivariate dataset containing different missingness patterns. Multiple imputation (MI) (Rubin, 1978; Rubin, 1987; Rubin and Schenker, 1986, Rubin, 1996a) can be used to estimate the uncertainty created by imputing the missing values in Y, as follows;

```
for  $d = 1$ to $D$
    1. Impute the missing values in $Y$ using a stochastic method to create a unique imputed dataset $Y_d$
    2. Compute a parameter point estimate $\hat{\theta}_d$ which describes the values in $Y_d$
    3. Compute the variance $V_d$ associated with $\hat{\theta}_d$
    4. Restore the matrix that contains $Y$ to its original condition (as it was before starting this process)
next  $d$
```

Fig 4.8 - Estimating imputation uncertainty using multiple imputation

The procedure given in Fig. 4.8 produces a set of estimates $\{\hat{\theta}_1 \dots \hat{\theta}_D\}$ and a set of associated variances $\{V_1 \dots V_D\}$ which describe the imputed datasets $\{Y_1 \dots Y_D\}$. The combined MI complete-data parameter point estimate for $\{Y_1 \dots Y_D\}$ is then given by;

$$\bar{\theta}_D = \frac{1}{D}\sum_{d=1}^{D}\hat{\theta}_d$$

The total variability T_D, associated with $\bar{\theta}_D$, can then be used to estimate imputation uncertainty, as follows;

$$T_D = \frac{1}{D}\sum_{d=1}^{D} V_d + \frac{D+1}{D}\left(\frac{1}{D-1}\sum_{d=1}^{D}\left(\hat{\theta}_d - \bar{\theta}_D\right)^2\right) \qquad (4.10)$$

It is important to emphasise that MI is primarily an imputation method, rather than a technique designed for the estimation of imputation uncertainty. However,

> *"When the D sets of imputations are repeated random draws from the predictive distribution of the missing values under a particular model for nonresponse, the D complete-data inferences can be combined to form one inference that properly reflects uncertainty due to nonresponse under that model"*

As Little and Rubin (2002) succinctly explain.

4.3.4 Limitations of Uncertainty Estimation Methods

The uncertainty estimation methods described in the two preceding sections have their limitations, and they make certain assumptions about the nature of the missing value dataset. These issues have been the subject of considerable debate among statisticians. The main points for discussion are summarised below.

1. All of the uncertainty estimation methods described above assume (somewhat optimistically) that the imputation process has removed the bias within the dataset that was caused by the missing values (Little and Rubin, 2002).

2. The resampling methods described in section 4.3.2 are based on large-sample theory - i.e. they will return more reliable estimates of the variance for larger samples. However, *"The theory underlying MI is Bayesian and can provide useful inferences in small samples"* (Little and Rubin, 2002).

3. The MI method assumes that the model describing the missing value dataset has been correctly specified - i.e. the reliability of the variance estimates returned by the MI method is sensitive to model misspecification. However, the resampling methods return consistent variance estimates with minimal modelling assumptions, so they are more robust to model misspecification (Lazzeroni et al, 1990; Fay, 1996a).

4. Resampling methods usually require several hundred executions of the imputation process, performed against an equal number of samples drawn from the missing value dataset. This can be impractical in some situations. However, MI is less computationally intensive, since it allows good inferences to be drawn for a wide range of estimands, using perhaps 10 (or less) imputed datasets (Ezzati-Rice et al, 1995).

5. The problem of obtaining reliable variance estimates using the Jackknife method after executing single imputation methods has sparked some debate. In particular, six articles in the Journal of the American Statistical Association, Vol. 91 (434) were devoted to a discussion of a paper on this topic by Rao (1996a). Including comments by Judkins (1996), Binder (1996) and Eltinge (1996). followed by rejoinders from Rubin, (1996b) and Fay (1996b) - and finally, a reply by the author of the debated paper in Rao (1996b). Various opinions were expressed but no single point of view was conclusively demonstrated as being correct.

4.3.5 Comparing the Proposed Method with Uncertainty Estimation Methods

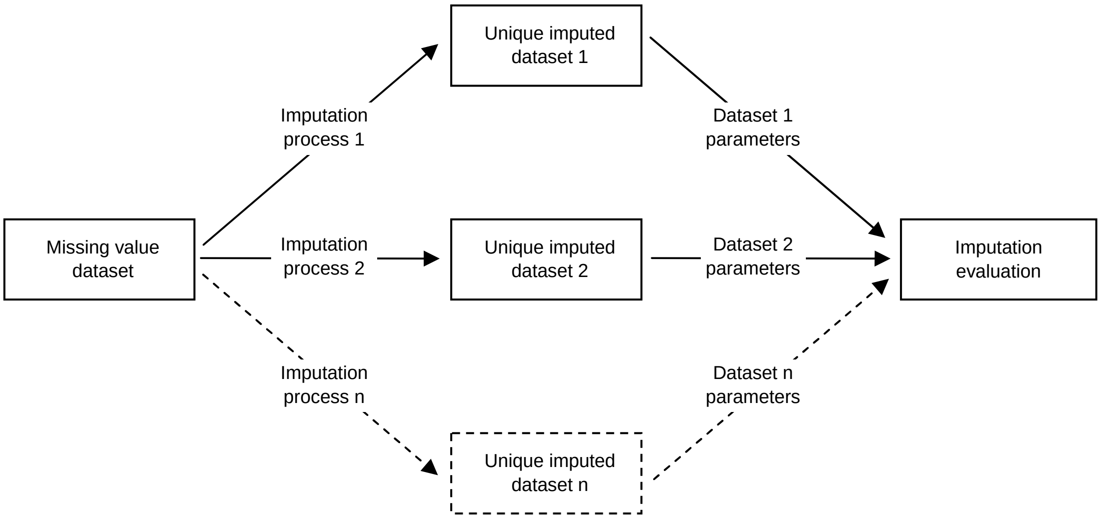


Fig 4.9 – Common features of Bootstrap, Jackknife, MI and the proposed method

The proposed method and the uncertainty estimation methods described in the two preceding sections are similar in that they all execute the imputation method (the method being evaluated) *repeatedly* against the same missing value dataset. This process produces a set of unique imputed datasets, as shown in Fig. 4.9 (although different techniques are used to create these datasets, as explained above). All four of the methods then go on to use the set of parameters that describe the unique datasets to evaluate the results of the imputation process. ***However, the four methods also differ in several important respects, as described below.***

In its purest form, the Jackknife method requires n repetitions of the imputation method, where n is number of rows in the dataset (see section 4.3.2). This can be impractical for large datasets (e.g. datasets containing millions of rows), particularly since the imputation method itself will take longer to execute as the size of the dataset increases. In addition, the Jackknife differs from the other methods in that the *overall process* is deterministic - i.e. a repeated execution of the overall Jackknife process (all iterations taken together) against the same dataset will always produce the same uncertainty estimate.

The Bootstrap method introduces a stochastic element into the overall evaluation process by taking random samples from the dataset. In effect, this sampling process is equivalent to a repetitive randomised form of the listwise deletion process described in chapter one. However, there is no way of knowing whether any particular sample will fully reflect the missing data mechanism within the dataset as a whole. This could bias the uncertainty estimate produced by the Bootstrap when the data is MAR. It could be argued that this hardly matters, since the repetitive process should remove this bias - but it is unclear how many Bootstrap iterations would be needed to achieve this for any particular MAR configuration.

Multiple imputation builds on the ideas underpinning the resampling methods, but differs from them in that it integrates the repetitive stochastic part of the imputation evaluation process *with the imputation procedure* - so that repeated executions of that procedure will *"reflect variation within an imputation model and sensitivity to different imputation models"* (Rubin, 1978). However, unlike the resampling methods, MI achieves this while retaining all of the rows in the dataset. The proposed method also retains all of the rows in the dataset, so it has more in common with MI than either of the resampling methods in this respect.

The proposed method differs from the other methods by estimating the accuracy of the imputed values - i.e. the other methods do not record deleted values and then measure how accurately they have been "put back" by the imputation process. The Bootstrap, MI and the proposed method are similar in that they all employ stochastic procedures, but the Jackknife method does not share this characteristic. However, the proposed method and the Bootstrap differ from MI in that the stochastic part of these methods is performed *before* the imputation process starts, rather than being integrated with that process.

<u>The problem of defining a "proper" multiple imputation method in practice</u>

The Bootstrap/Jackknife methods and the proposed method can all be used to evaluate *any* imputation technique, whereas the MI approach can only be *confidently* used for evaluation purposes when we are sure that the MI method employed is *"proper"*, in the sense defined by Rubin (1987, pp. 118-119) and further summarised by Rubin (1996a). One of the clearest defintions of a proper MI method is given by Durrant (2005), who explains that for a proper MI method, equation (4.10) - given in section 4.3.3 - *"is indeed a valid formula, providing an approximately unbiased estimator of the variance"*. But it is very hard to verify the truth of this statement in practice, as Schafer (1997) points out;

> *"Except in trivial cases (e.g. univariate data missing completely at random), it can be extremely difficult to determine whether a multiple-imputation method is proper"*

Binder and Sun (1996) shed some light on this problem by discussing several complex examples, but these only cover a small proportion of the imputation problems that can occur. In practice, proper MI methods generally employ Bayesian imputation algorithms, even though *theoretically* this is not deemed to be essential. For example, Schafer's (1997) implementation of MI employs the Markov chain Monte Carlo method (Tanner, 2005; Gilks et al, 1996) via the Bayesian data augmentation algorithm (Tanner and Wong, 1987). To summarise, we can say that the evaluation of imputation methods via MI uncertainty estimation can only be confidently applied when we are sure that the MI method used is proper - e.g. when that method employs a Bayesian imputation algorithm. However, the other three methods do not have this limitation.

4.4 Summary

This chapter has described the imputation evaluation method devised by the author and has shown how this method can be used to estimate the predictive accuracy of the imputed values generated by any imputation technique.

A functional overview of the proposed method has been given and the equations and procedures which form the basis of that method have been described in detail. An explanation of how the method can be used to compare the accuracy of the imputed values in different data segments has been given. These descriptions and explanations form the principal contribution to knowledge made by this thesis.

A description of how the general idea of estimating imputation accuracy has been applied by other researchers has been given and it has been shown that the proposed method differs from these approaches in several important respects. The functionality of the most similar methods found within the literature (uncertainty estimation methods) has been described and the limitations of these methods have been discussed. The similarities and the differences between the proposed method and uncertainty estimation methods have been discussed, and it has been shown that the proposed method builds on the ideas underpinning uncertainty estimation methods, but differs from them in several important respects.

The proposed method has been implemented alongside the imputation techniques described in chapters two and three in the form of an integrated software application. The following chapter describes how this application was used to experimentally evaluate the reliability and the validity of the proposed method. Chapter six goes on to explain how the integrated application was used to assess the feasibility of imputing the missing values in the collaborating company’s dataset, thus fulfilling the project objectives.

Chapter Five

Experimental Evaluation of the Method

5. Experimental Evaluation of the Method

This chapter explains how the reliability and validity of the proposed imputation evaluation method was experimentally evaluated. This was essential for the following reasons.

The scientific method requires that experiments should produce consistent and reliable results when they are repeated. The reliability of the results produced when performing experiments using stochastic processes - such as the proposed method - can be difficult to demonstrate, because such experiments will produce different results when they are repeated. However, when the results from several identical stochastic experiments are analysed using statistical methods it should be evident that the stochastic procedures employed have produced *similar* results, where this is required. For example, if the proposed method produced *very different* sets of imputation accuracy statistics when the same experiment was repeated, then how could we know which sets of statistics gave the most useful results?

It is also important to demonstrate that the proposed imputation evaluation method is valid. That is, to show that the imputation accuracy statistics produced by the method can be used assess the feasibility of imputing missing values using the required imputation technique. The explanations in the following sections show how the experimental results (estimates of imputation accuracy) given in this chapter can be used to assess the feasibility of imputing missing values using the imputation techniques described in chapters two and three.

- **Section 5.1** explains how the reliability and validity of the proposed method was experimentally evaluated and shows how the software that implements the method can be used to perform imputation evaluation experiments.
- **Section 5.2** explains how the method can be used to compare the predictive power of candidate imputation methods when they are used to impute the same set of missing values and introduces the idea of "least distortion" imputation evaluation (Pyle, 1999).

5.1 Assessing the Reliability and Validity of the Proposed Method

Section 5.1.1 describes the dataset that was used to perform all of the experiments described in this chapter. Section 5.1.2 describes the top level graphical user interface (GUI) of the software application that implements the proposed method. Section 5.1.3 describes the method reliability experiments and shows how the software's GUI provides access to the equations and processes that form the basis of the proposed method. Section 5.1.4 describes the method validity experiments and shows how the software can be used to estimate the accuracy of the imputed values generated by the EM imputation algorithm.

5.1.1 Description of the Dataset Used for the Experiments

The dataset used to perform the experiments can be freely downloaded from the Statistical Society of Canada (SSC) web page created by Bernier et al. (2002) - which is available at http://www.ssc.ca/documents/case_studies/2002/missing_e.html (last accessed on 17th July 2007). The dataset comes in the form of a Microsoft Excel file containing 2389 data rows and 11 columns, with 2 of these columns having 29.22% missing values. A detailed analysis of the dataset, and a description of the associated imputation case study, can be found in Aguirre and Sun (2003). General information about the SSC can be found on their website - available at http://www.ssc.ca/main/new_e.html The following description of the health survey questionnaire used to create the dataset, and the table containing the descriptions of the dataset variables have also been taken from the web page created by Bernier et al (2002)

> *"This case study on missing data uses a sub-sample of the 1994 National Population Health Survey…..The data represent persons, aged 20-65, living in a private household in the prairie provinces..... The questionnaires include content related to health status, use of health services, determinants of health, a health index, chronic conditions and activity restrictions. The use of health services is probed through visits to health care providers, both traditional and non-traditional, and the use of drugs and other medications. Health determinants include smoking, alcohol use and physical activity. As well, a section on self-care has also been included this cycle. The socio-demographic information includes age, sex, education, ethnicity, household income and labour force status".*

Table 5.1 - Description of the variables in the Canadian SSC health survey dataset

Column (variable) name	**Variables with 29.22% missing values**
GH_Q1	Numerical representation of the answer to "In general, how would you describe your health?"
DVHST94	Derived Health Status Index (3 decimal places) - HUI provisional score
Column (variable) name	**Variables with 100% data present**
AGEGRP	Grouped age cohorts
SEX	Respondent's sex
DVHHIN94	Derived total household income from all sources in the past 12 months
DVBMI94	Derived Body Mass Index (1 decimal place)
DVSMKT94	Derived type of smoker
DVPP94	Derived depression variable - predicted probability (2 decimal points)
NUMCHRON	Numerical sum representing the answers to several health status questions
VISITS	Numerical sum representing the answers to several questions about visits to doctors and clinics.
WT6	Survey weights

5.1.2 Loading the Dataset and Analysing the Variables

This section describes the top level GUI of the software application that implements the proposed imputation evaluation method - focusing on a description of the data loading and variable analysis functionality provided. A description of how the software can be used to evaluate imputation methods is given in the following sections.

The software has been developed using the C# programming language, utilising the computer hardware and software development environment described in Appendix C. The screenshots on the following page show how the software was used to load and analyse the dataset described in the previous section, and it can be seen that the variables shown in the screenshot grids are the same as those given in table 5.1, above. A description of the GUI features shown in the screenshots is given below.

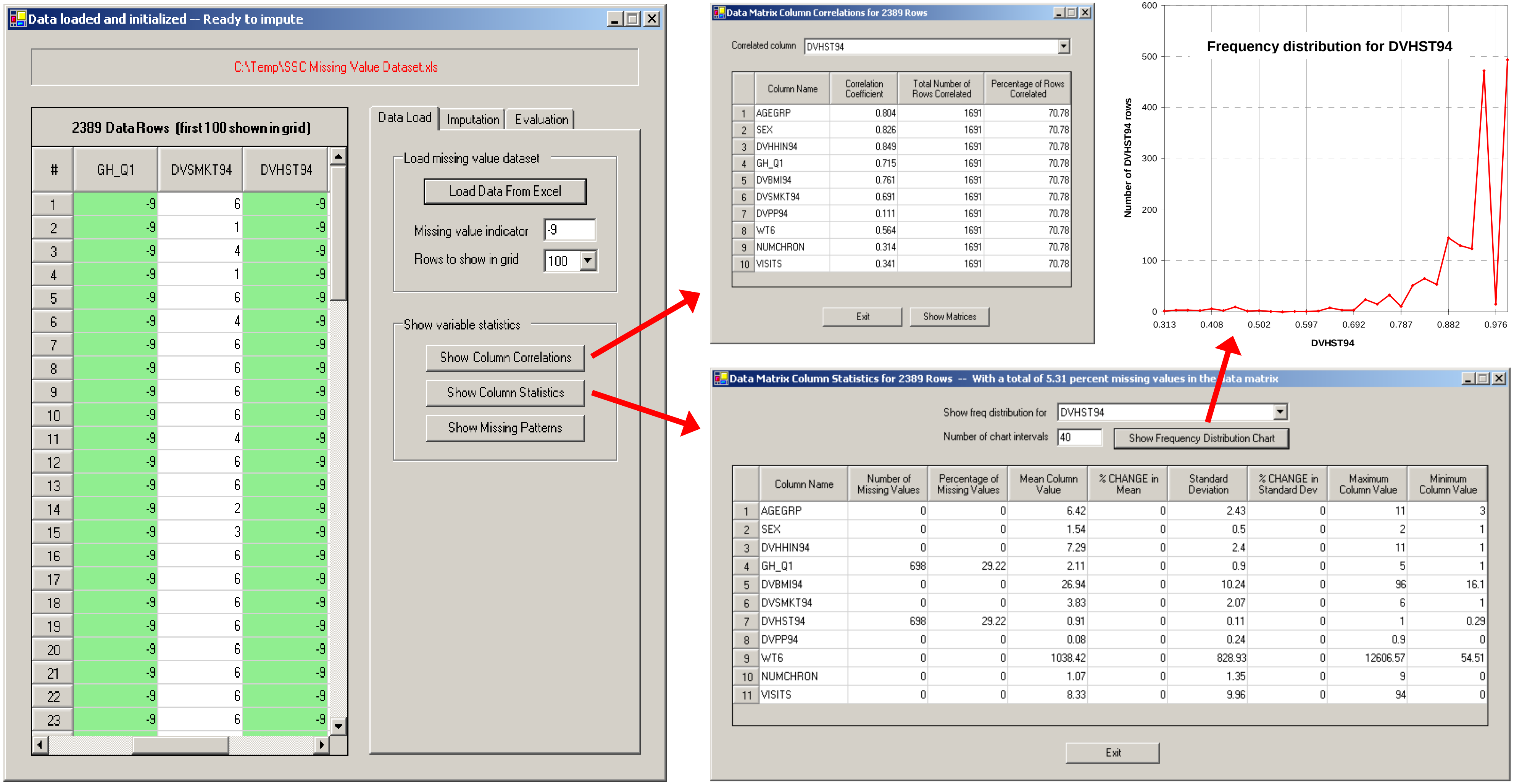
- Datasets can be loaded from Excel files by clicking on the Load Data From Excel button. The ".xls" file is selected using a standard Microsoft file selector window. The selected path/file name is displayed above the on-screen grid. Excel worksheet header row cells are automatically inserted as grid column headers (if any are found). Non-numeric cells found in the Excel worksheet are error trapped, and the user is informed of the problem

- The numeric code used to represent missing values in the Excel worksheet cells *must be* entered by the user before loading the dataset (the default value of -9 is shown in the screenshot). Grid cells with missing values are colour coded for easy identification of missingness patterns - both before and after the imputation process is performed.

- A description of the variables in the loaded dataset can be shown in a sub-window by clicking on the Show Column Statistics button. The maximum, minimum, mean and standard deviation for each variable is shown, as well as the number and proportion of missing values for each variable. The distribution line chart for any variable can also be shown (see the following page). The distortion of the mean and standard deviation caused by the imputation process is also given (the utility of this feature is discussed in section 5.2.3). All variable descriptions are fully updated after imputation is completed.

- A description of the correlations between the dataset variables can be shown in a sub-window by clicking on the Show Column Correlations button. The number and proportion of data matrix rows used to calculate the correlation between each pair of variables is shown. This is required, since some matrix rows will *not have* known values for every possible pair of variables (depending on the set of missingness patterns in the matrix).

- A description of the relative size and distribution of the missingness patterns in the data matrix can be shown in a sub-window by clicking on the Show Missing Patterns button. The importance of missingness pattern analysis is discussed in chapter six.



Data loaded and initialized -- Ready to impute

C:\Temp\SSC Missing Value Dataset.xls

2389 Data Rows (first 100 shown in grid)

#	GH_Q1	DVSMKT94	DVHST94
1	-9	6	-9
2	-9	1	-9
3	-9	4	-9
4	-9	1	-9
5	-9	6	-9
6	-9	4	-9
7	-9	6	-9
8	-9	6	-9
9	-9	6	-9
10	-9	6	-9
11	-9	4	-9
12	-9	6	-9
13	-9	6	-9
14	-9	2	-9
15	-9	3	-9
16	-9	6	-9
17	-9	6	-9
18	-9	6	-9
19	-9	6	-9
20	-9	6	-9
21	-9	6	-9
22	-9	6	-9
23	-9	6	-9

Missing value indicator: -9

Rows to show in grid: 100

Data Matrix Column Correlations for 2389 Rows

Correlated column: DVHST94

	Column Name	Correlation Coefficient	Total Number of Rows Correlated	Percentage of Rows Correlated
1	AGEGRP	0.804	1691	70.78
2	SEX	0.826	1691	70.78
3	DVHHIN94	0.849	1691	70.78
4	GH_Q1	0.715	1691	70.78
5	DVBMI94	0.761	1691	70.78
6	DVSMKT94	0.691	1691	70.78
7	DVPP94	0.111	1691	70.78
8	WT6	0.564	1691	70.78
9	NUMCHRON	0.314	1691	70.78
10	VISITS	0.341	1691	70.78

Data Matrix Column Statistics for 2389 Rows -- With a total of 5.31 percent missing values in the data matrix

Show freq distribution for: DVHST94

Number of chart intervals: 40

	Column Name	Number of Missing Values	Percentage of Missing Values	Mean Column Value	% CHANGE in Mean	Standard Deviation	% CHANGE in Standard Dev	Maximum Column Value	Minimum Column Value
1	AGEGRP	0	0	6.42	0	2.43	0	11	3
2	SEX	0	0	1.54	0	0.5	0	2	1
3	DVHHIN94	0	0	7.29	0	2.4	0	11	1
4	GH_Q1	698	29.22	2.11	0	0.9	0	5	1
5	DVBMI94	0	0	26.94	0	10.24	0	96	16.1
6	DVSMKT94	0	0	3.83	0	2.07	0	6	1
7	DVHST94	698	29.22	0.91	0	0.11	0	1	0.29
8	DVPP94	0	0	0.08	0	0.24	0	0.9	0
9	WT6	0	0	1038.42	0	828.93	0	12606.57	54.51
10	NUMCHRON	0	0	1.07	0	1.35	0	9	0
11	VISITS	0	0	8.33	0	9.96	0	94	0

Fig 5.1 – Implementation of the method: Data loading and variable analysis graphical user interface

5.1.3 Assessing the Reliability of the Method

The previous section explained how the SSC dataset (as described in section 5.1.1) was loaded into the software application that implements the proposed method. This section describes how the dataset was used after it was loaded - i.e. the missing DVHST94 values were imputed and the results were used to assess the reliability and the validity of the proposed method. The experimental process is described in table 5.2, below. This remainder of this section attempts to answer experimental question 1 and the following section attempts to answer experimental question 2.

Table 5.2 - Description of SSC dataset imputation evaluation experiments 1 to 8
Imputation of DVHST94 values using 50 executions of the EM algorithm

<table>
<tr><td colspan="2"><b><u>EXPERIMENTAL QUESTIONS</u></b>
1. Does the evaluation method produce reliable results when it is executed repeatedly against the same dataset?
2. Can the proposed imputation evaluation method be used to assess the feasibility of imputing DVHST94 values?</td></tr>
<tr><td><b>Description of the missing value dataset</b></td><td>● The SSC dataset containing 11 columns and 2389 rows - as described in section 5.1.1</td></tr>
<tr><td><b>Variable to be imputed and evaluated</b></td><td>● The variable to be imputed and evaluated was DVHST94, which had a range of 0.290 to 1.000 (710 possible values, specified to 3 decimal places)
● DVHST94 had 698 missing values - i.e. 29.22% of the 2389 data matrix rows had missing values.</td></tr>
<tr><td><b>Imputation method used for the experiment</b></td><td>● Imputation was performed using the EM algorithm
● The EM algorithm convergence value was 0.0001
● No Box-Cox power transformations were performed for any variable.
● The initial covariance matrix was created using all data matrix rows with a full set of known values.
● All imputed values were rounded to 3 decimal places and negative imputed values were discarded.</td></tr>
<tr><td><b>Imputation evaluation method</b></td><td>● 50 executions of the EM imputation algorithm were performed (using the options described above).
● The number of missing DVHST94 values was increased by 10% for each execution of EM. That is, 4.14% of the known DVHST94 values were randomly deleted and “put back” for each EM execution, using the Fig.4.5 algorithm. With balanced random deletion across missingness patterns.</td></tr>
</table>

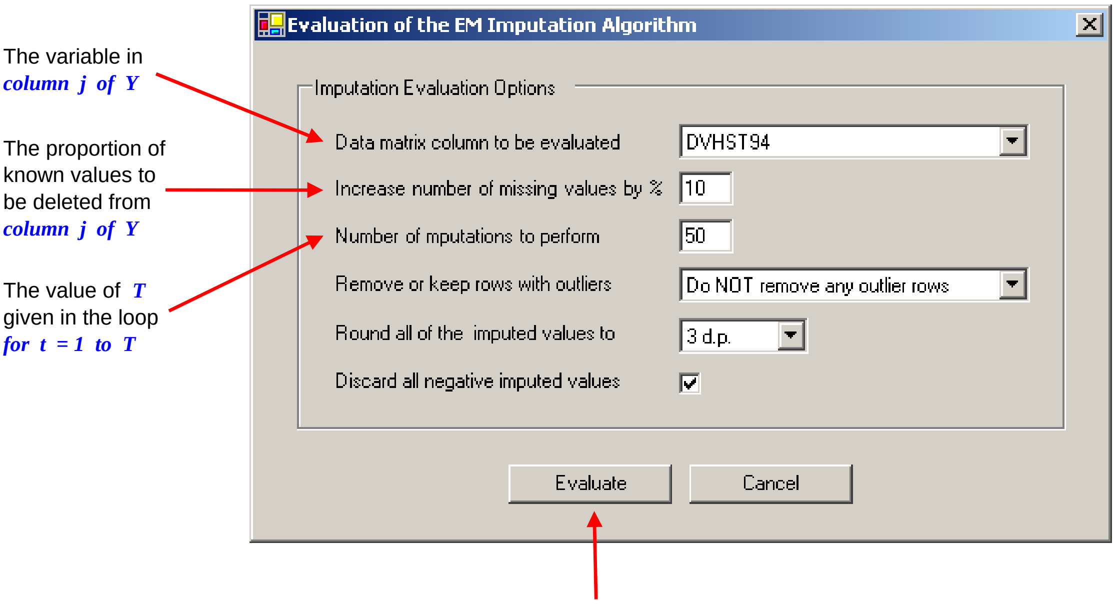
The tabular format shown above was designed to be used as a pro-forma which supports the use of the proposed method - and the same tabular format will be used to define all of the experiments described in this thesis.

Using a standard pro-forma to specify the experimental questions, describe the dataset and define the experimental process allows the experiments to be precisely replicated - which is an essential part of the proposed approach.

The software implements the formal description of the method given in section 4.2.1

The screenshots on the following page show how the experiments described in table 5.2 can be performed using the software. It is important to emphasise that the process described on the following page implements the imputation evaluation method described in section 4.2.1. The relationship between the formal description of the method and its software implementation is shown below (the Fig. 4.2 procedure is repeated here for clarity).

```
for $t = 1$ to $T$
    1. Randomly delete a small proportion of the known values in column $j$ of $Y$ to create matrix $Y_t$
    2. Impute the missing values in $Y_t$ using the imputation method being evaluated
    3. Compute a set of evaluation statistics $S_t$ using $Y_t$ - i.e. compute values for (2.2), (2.3) and (2.5)
    4. Restore the $Y$ matrix to its original condition (as it was before starting the whole process)
next $t$
```



The variable in *column j of Y*

The proportion of known values to be deleted from *column j of Y*

The value of *T* given in the loop *for t = 1 to T*

Evaluation of the EM Imputation Algorithm

Imputation Evaluation Options

Data matrix column to be evaluated: DVHST94

Increase number of missing values by %: 10

Number of mputations to perform: 50

Remove or keep rows with outliers: Do NOT remove any outlier rows

Round all of the imputed values to: 3 d.p.

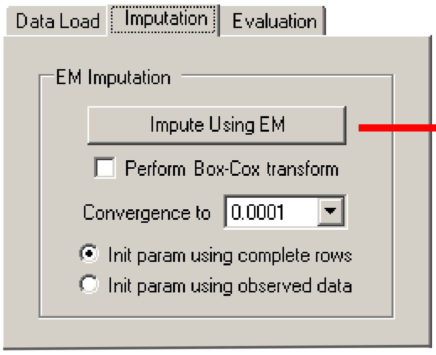
Discard all negative imputed values: ☑

When the user clicks on this button the algorithm shown above is executed and the estimates of imputation accuracy given by equations (4.6) and (4.7) are calculated and displayed in a sub-window - see section 4.2.1 for more details.

Fig 5.2 – Relationship between the method's algorithm and its software implementation

Performing the imputation evaluation experiments described in table 5.2

The following diagram shows how the entire imputation evaluation process can be performed using the software implementation of the method. The application provides a simple "point-and-click" graphical user interface that is easy to use and understand. When the evaluation process is complete the user can view the estimates of imputation accuracy produced by the software - using various tables, line charts and histograms. These sub-windows and charts are automatically generated and displayed by the software. Some examples follow the diagram.



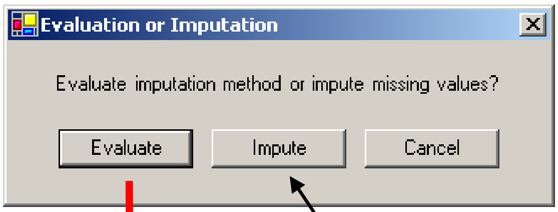
☐ Perform Box-Cox transform

Convergence to 0.0001

● Init param using complete rows

○ Init param using observed data

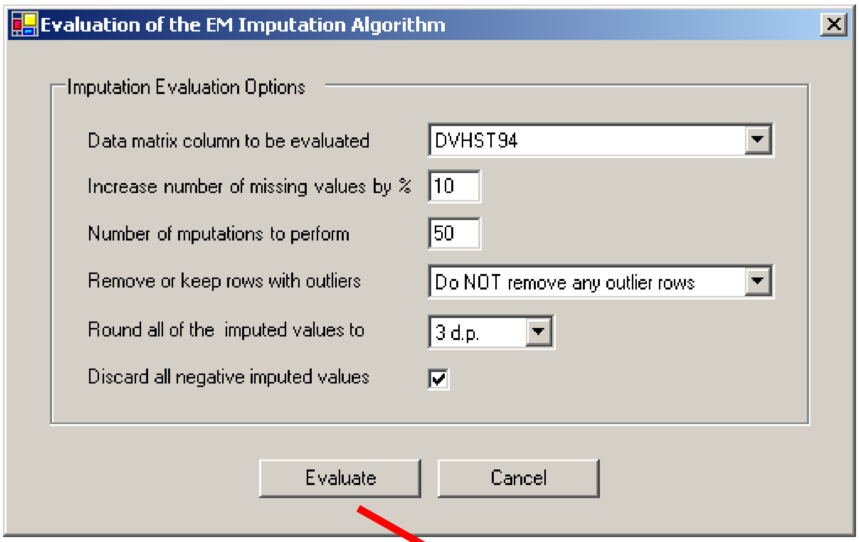
EM imputation algorithm parameters



Evaluate imputation method or impute missing values?

Imputes missing values *without* performing the imputation evaluation process

i.e. The known values *are not* deleted and then "put back"


Evaluation of the EM Imputation Algorithm
Imputation Evaluation Options

Data matrix column to be evaluated DVHST94

Increase number of missing values by % 10

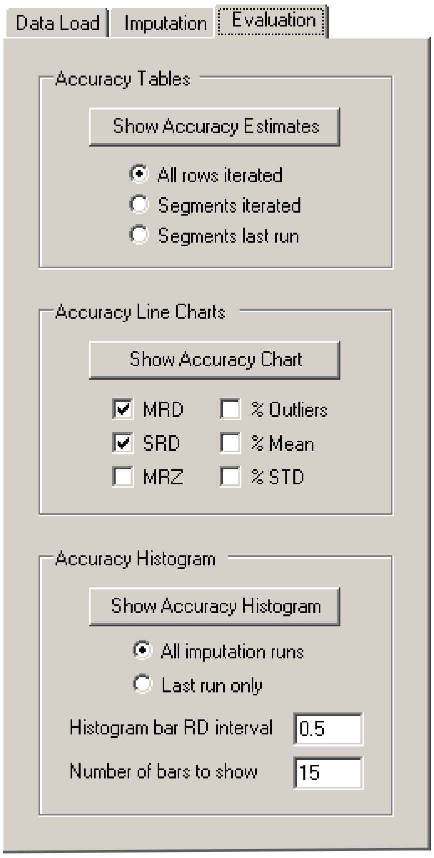
Number of mputations to perform 50

Remove or keep rows with outliers Do NOT remove any outlier rows

Round all of the imputed values to 3 d.p.

Discard all negative imputed values ☑

Imputation evaluation options



● All rows iterated

○ Segments iterated

○ Segments last run

☑ MRD ☐ % Outliers

☑ SRD ☐ % Mean

☐ MRZ ☐ % STD

● All imputation runs

○ Last run only

Histogram bar RD interval 0.5

Number of bars to show 15

Options for display of the three imputation accuracy tables (see the following page for an example)

Options for display of various imputation accuracy line charts (see the following pages for examples)

Options for display of the imputation accuracy histogram (see section 5.1.4 for an example)

Fig 5.3 - Performing imputation evaluation experiments

The diagram shows how the software application developed by the author can be used to perform imputation evaluation experiments.

The software provides a simple "point-and-click" interface that can help researchers to decide whether imputation is an appropriate solution to their missing data problem - as follows;

STEP ONE

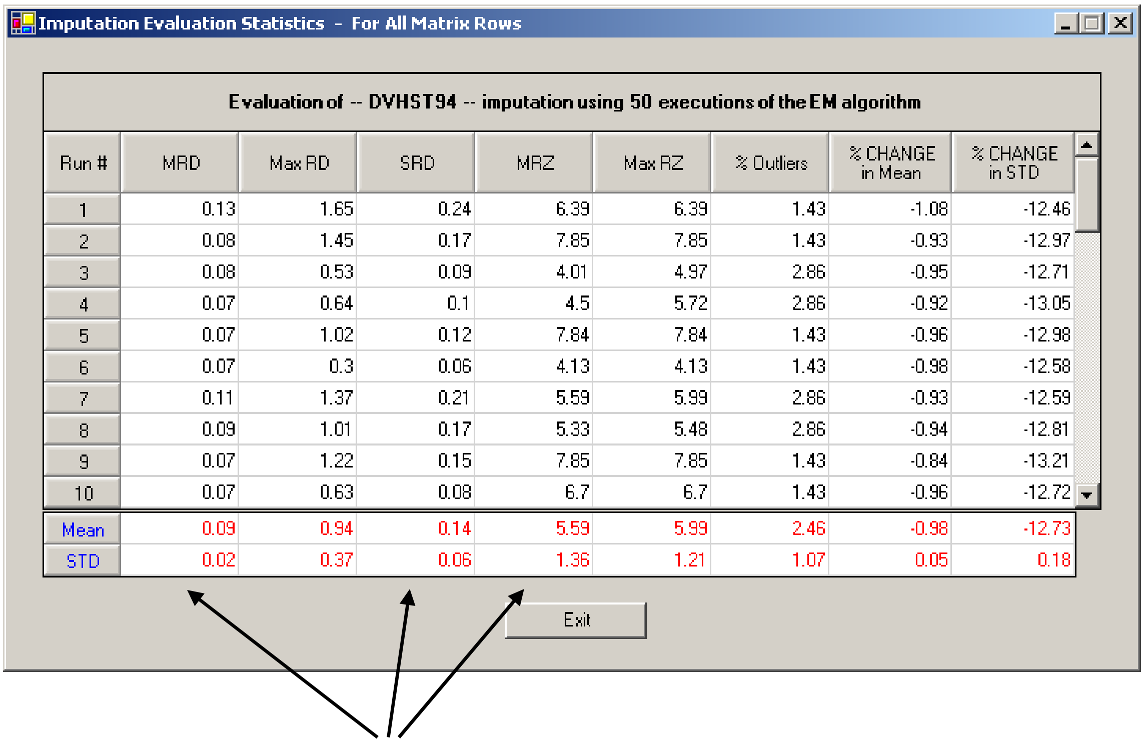
Select the imputation method and set its parameters The default options for the EM algorithm (see above) are suitable for most datasets.

STEP TWO

Choose the imputation evaluation options. For example, enter the number of executions of the EM algorithm to be performed.

STEP THREE

View the imputation accuracy tables, line charts and histogram - using the options shown opposite.



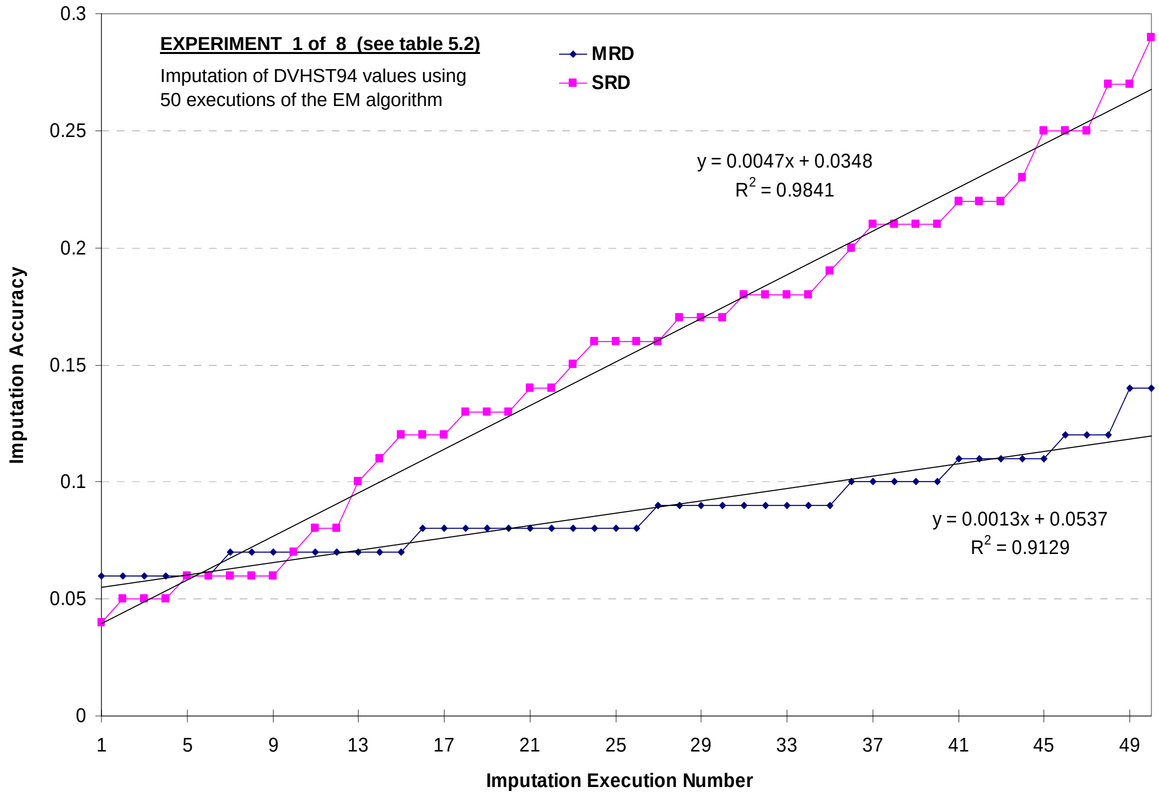
Imputation Evaluation Statistics - For All Matrix Rows

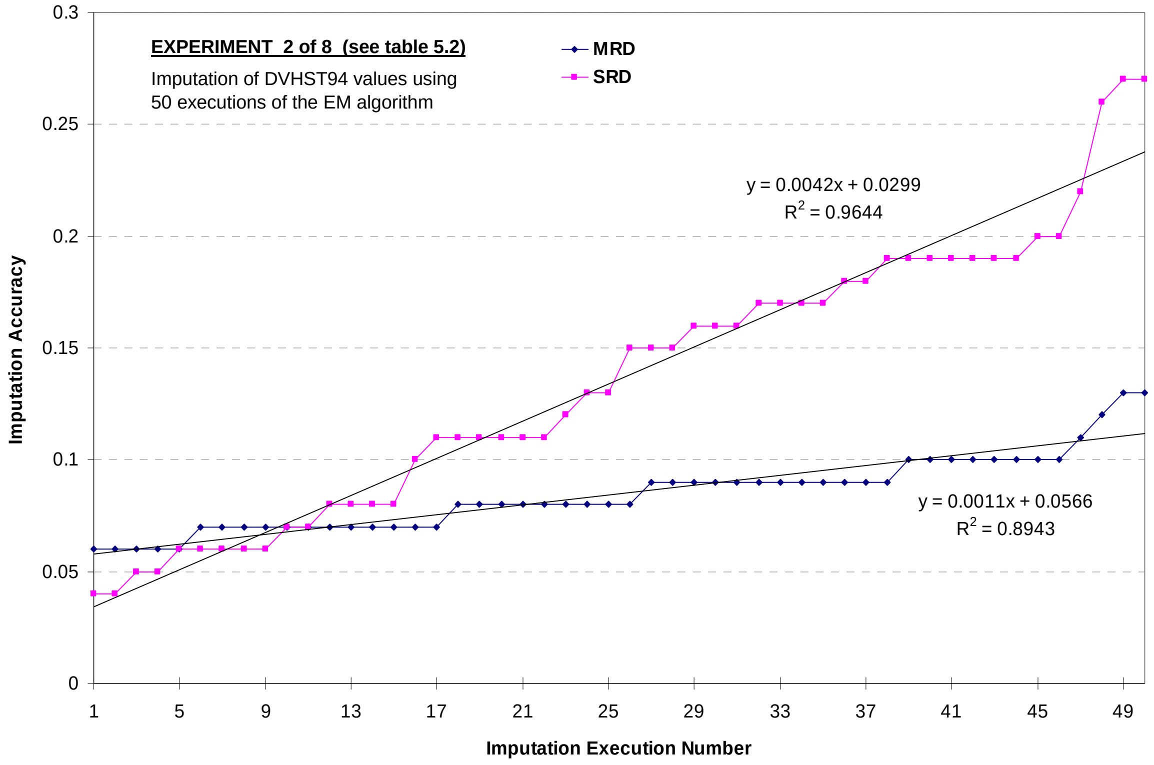
Evaluation of -- DVHST94 -- imputation using 50 executions of the EM algorithm

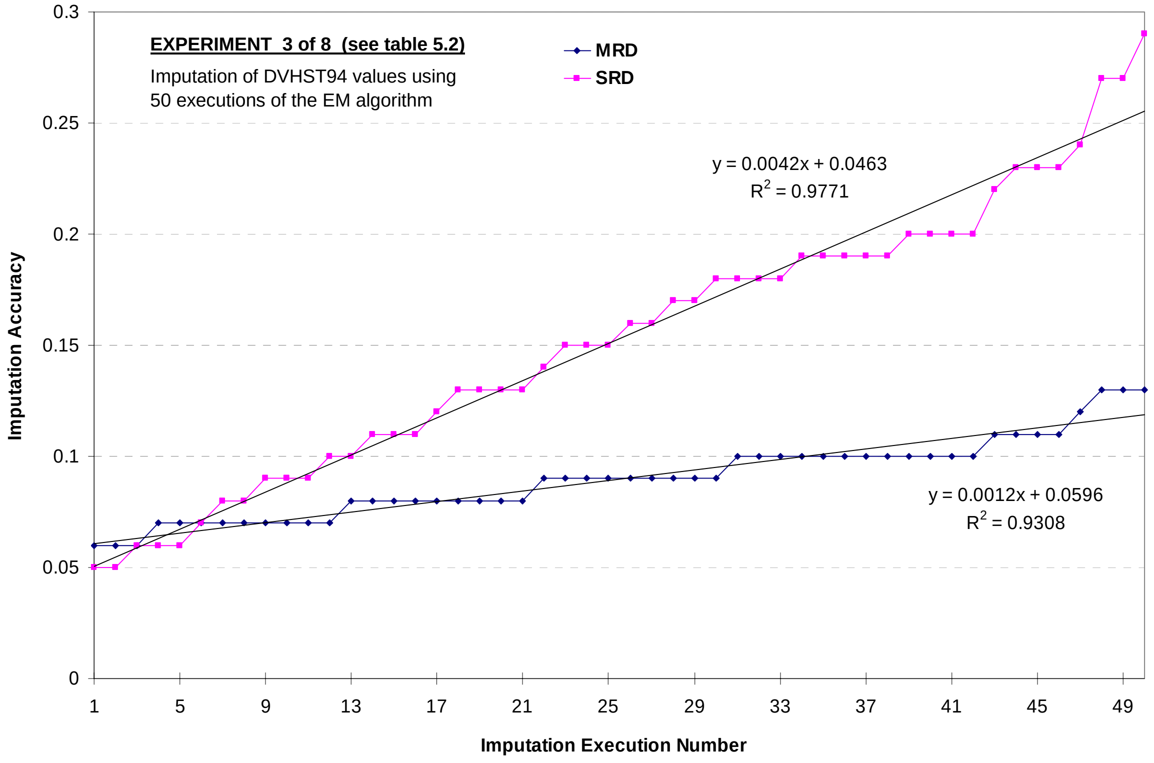
Run #	MRD	Max RD	SRD	MRZ	Max RZ	% Outliers	% CHANGE in Mean	% CHANGE in STD
1	0.13	1.65	0.24	6.39	6.39	1.43	-1.08	-12.46
2	0.08	1.45	0.17	7.85	7.85	1.43	-0.93	-12.97
3	0.08	0.53	0.09	4.01	4.97	2.86	-0.95	-12.71
4	0.07	0.64	0.1	4.5	5.72	2.86	-0.92	-13.05
5	0.07	1.02	0.12	7.84	7.84	1.43	-0.96	-12.98
6	0.07	0.3	0.06	4.13	4.13	1.43	-0.98	-12.58
7	0.11	1.37	0.21	5.59	5.99	2.86	-0.93	-12.59
8	0.09	1.01	0.17	5.33	5.48	2.86	-0.94	-12.81
9	0.07	1.22	0.15	7.85	7.85	1.43	-0.84	-13.21
10	0.07	0.63	0.08	6.7	6.7	1.43	-0.96	-12.72
Mean	0.09	0.94	0.14	5.59	5.99	2.46	-0.98	-12.73
STD	0.02	0.37	0.06	1.36	1.21	1.07	0.05	0.18

The **Mean** values for these 3 columns give the results for equation (4.6), as given in section 4.2.1
The **STD** values give the results for equation (4.7), also given in section 4.2.1.
The number of EM executions **= *T*** as given in the loop ***for t = 1 to T*** in the Fig. 4.2 procedure

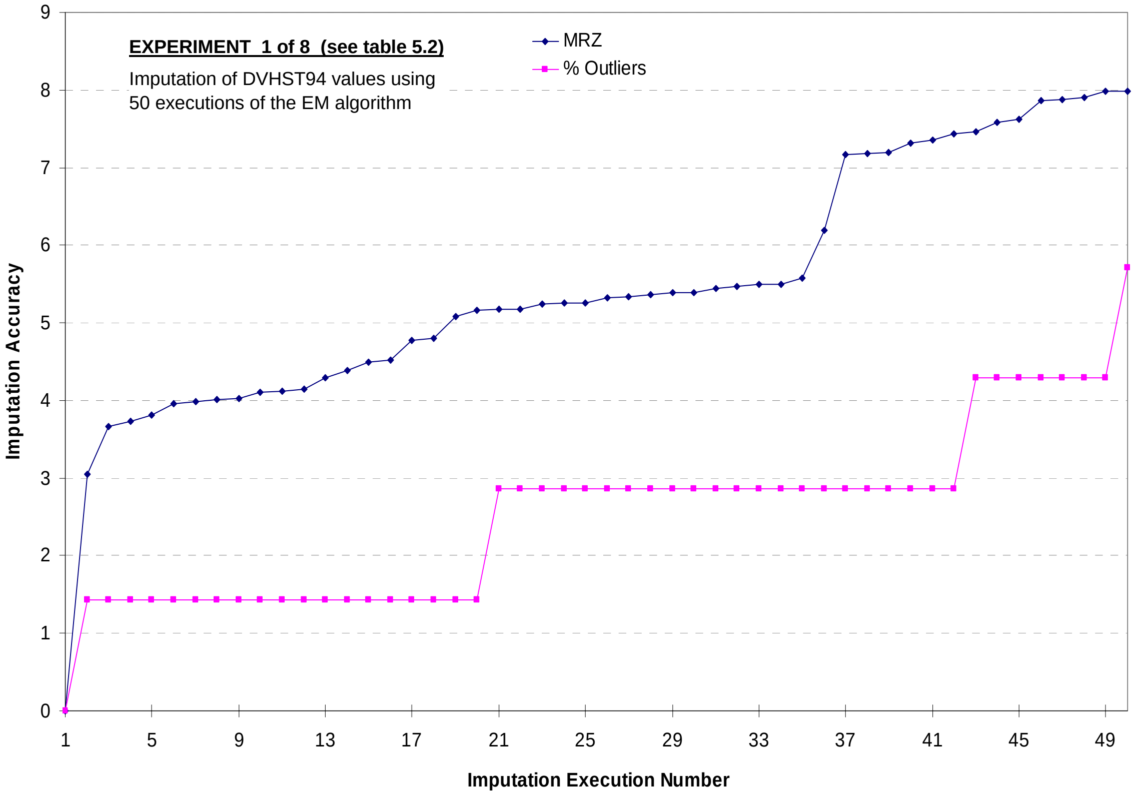
The grid shown in the screenshot above gives the results of one of the experiments described in table 5.2. The grid was obtained by clicking on the Show Accuracy Estimates button shown on the previous page. The line charts below show the results of three more of these experiments, where the values in the MRD and SRD grid columns are plotted on the charts.

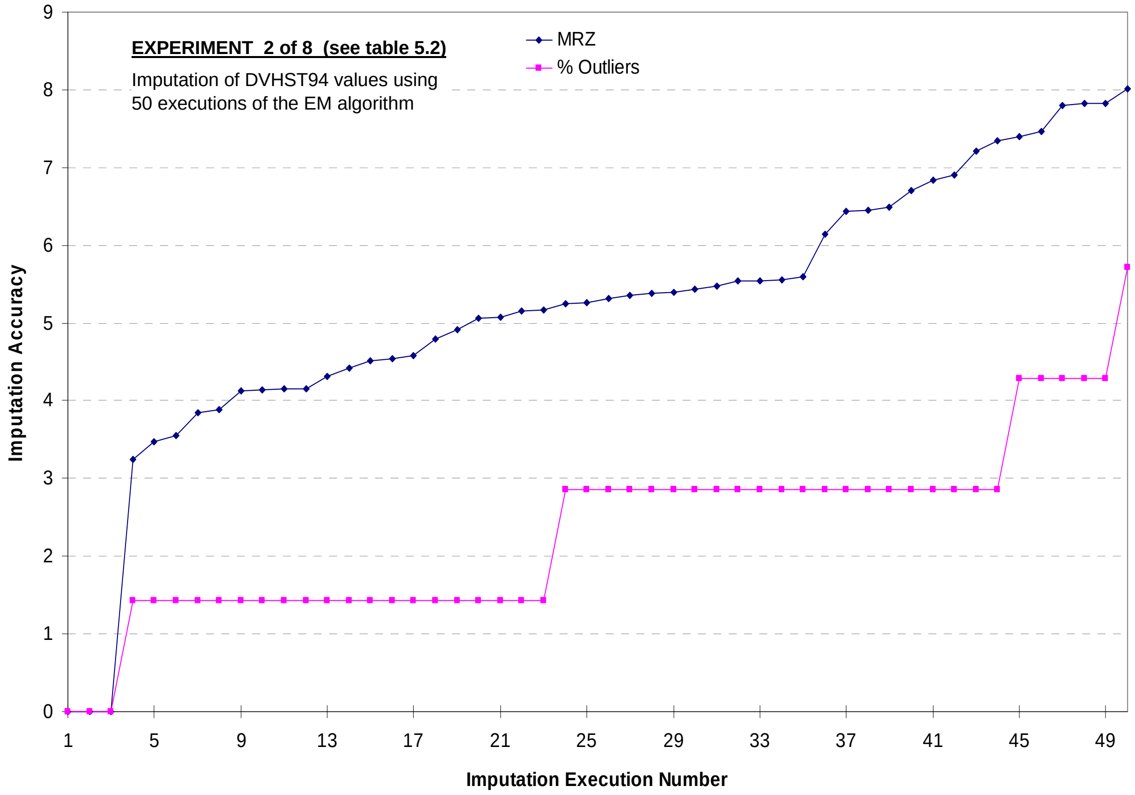


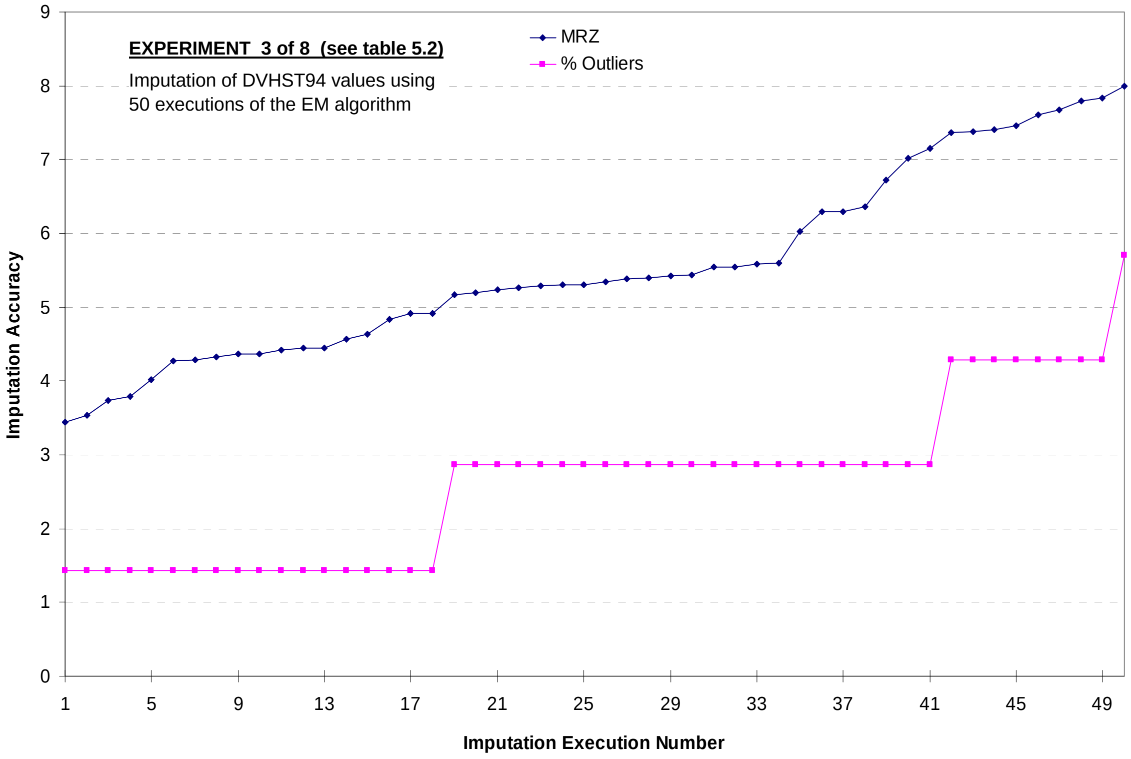




The two sets of data shown on each chart were sorted independently - i.e. the *MRD* and *SRD* figures for each EM execution *do not match*. This method of visual presentation enables the statistics produced for each experiment to be more easily compared and evaluated. ***The idea of the charts is to demonstrate that the method produces consistent results when it is repeated*** - by showing that three identical, consecutive experiments produced similar sets of evaluation statistics. This conclusion is supported by the regression equations (which are shown on the charts *only* for this purpose) and it can be seen that these equations are similar across all three charts. The MRZ and % Outlier charts for the same three experiments are shown below (all figures are rounded to 2 decimal places).







The two sets of data shown on each chart were sorted independently - i.e. the *MRZ* and *% Outliers* figures for each EM execution number *do not match*. It can be seen that the MRZ and % Outliers for each of the three experiments are similar. Five more identical, consecutive experiments were performed (see table 5.2) and the resulting charts were all found to be very similar to those given above. The results for all eight experiments (including the three described in the above charts) are shown below. The means $\hat{\mu}$ and the standard deviations $\hat{\sigma}$ were computed using equations (4.6) and (4.7), as described in section 4.2.1. The mean value for each column in table 5.3, below is given in the bottom row of that table.

Table 5.3 - Aggregated estimates of imputation accuracy for the SSC dataset experiments (imputation of DVHST94 values using 50 executions of the EM algorithm)

Experiment Number	MRD		SRD		MRZ		
	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	% Outliers
1	0.09	0.02	0.15	0.07	5.45	1.60	2.52
2	0.08	0.02	0.14	0.06	5.18	1.80	2.32
3	0.09	0.02	0.15	0.06	5.56	1.26	2.63
4	0.09	0.02	0.16	0.06	5.12	1.72	2.53
5	0.09	0.02	0.15	0.07	5.19	1.34	2.68
6	0.08	0.02	0.16	0.06	5.80	1.23	2.55
7	0.09	0.02	0.14	0.06	5.45	1.91	2.23
8	0.09	0.02	0.15	0.06	5.62	1.33	2.32
Column Mean	**0.09**	**0.02**	**0.15**	**0.06**	**5.42**	**1.52**	**2.47**

Overall conclusion for the method reliability experiments

The six line charts shown above - and the figures given in table 5.3, above - show that the proposed imputation evaluation method produced very similar results when eight identical, consecutive EM imputation experiments were performed against the SSC dataset. This shows that the proposed method produces reliable results when it is used to evaluate regression based imputation methods against numeric datasets of a certain type - i.e. datasets that are similar to the SSC dataset described above.

5.1.4 Assessing the Validity of the Method

This section attempts to answer experimental question 2, given in table 5.2, above - i.e. Can the proposed method be used to assess the feasibility of imputing DVHST94 values? This question can be partially answered by considering the experimental results given above. ***However, the method can also be used to assess imputation accuracy variance, as follows.***

A reasonable benchmark value for *any imputation algorithm* would be an MRD $\hat{\mu}$ value of 0.2 - e.g. the imputation process described in the previous section should "put back" the deleted DVHST94 values to within 20% of their true values (on average). The results in table 5.3, above show that the deleted DVHST94 values were "put back" to within 9% of their true values (on average) for every experiment performed. However, the variance of the RD values also needs to be considered before any conclusion regarding the feasibility of DVHST94 imputation can be drawn. This can be achieved using the approach described below.

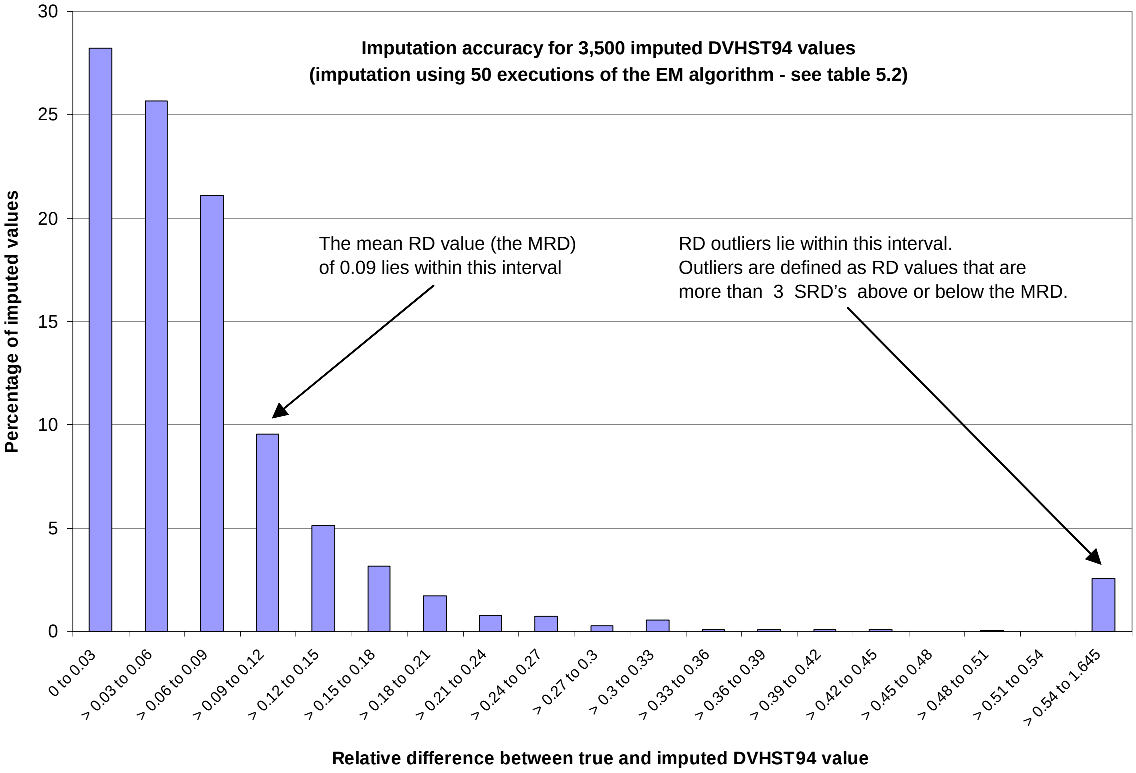
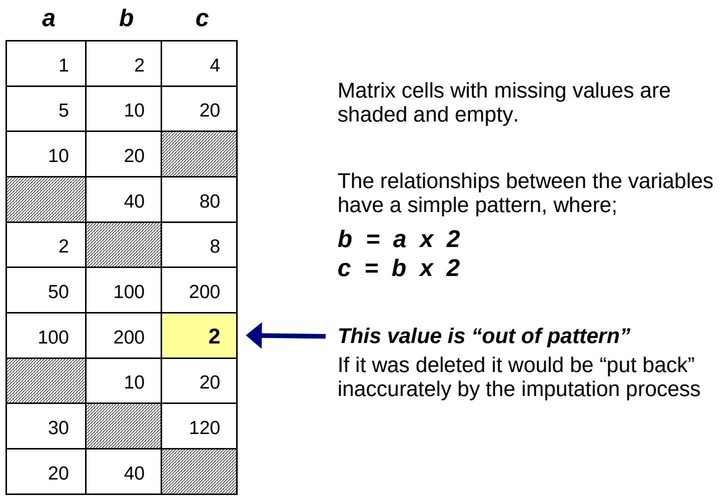


Fig 5.4 - Distribution of the RD values after imputation of DVHST94 using the EM algorithm

The software application that implements the method automatically produces and displays RD histograms such as the one shown above (where the RD values are computed using equation (4.1), given in section 4.2.1). The automatic display of these histograms is one of the most important features of the software, because this helps users to make informed decisions regarding the feasibility of the imputation process. For example, the histogram shown above could be used to assess the feasibility of imputing the missing DVHST94 values (using the experimental process described in table 5.2, above) by using the following reasoning.

The histogram shows that approximately 75% of the imputed DVHST94 values (represented by the three leftmost histogram bars) had an RD value of between zero and 0.09 - i.e. 75% of the values were imputed with an accuracy of 91% or above, with 28% imputed with more than 97% accuracy. The histogram also shows that only 16% of the DVHST94 values were imputed with less than 88% accuracy. However, approximately 2.5% of the imputed values were RD outliers - i.e. the rightmost histogram bar shows that 2.5% of the 3,500 values were imputed with less than 46% accuracy. This raises the following questions; **(1)** Does this small proportion of inaccurately imputed values invalidate the DVHST94 imputation process? **(2)** Why were these values imputed so inaccurately? The answers to these questions are discussed below - with reference to the following, theoretical, illustrative dataset.



a	***b***	***c***
1	2	4
5	10	20
10	20	▨
▨	40	80
2	▨	8
50	100	200
100	200	**2**
▨	10	20
30	▨	120
20	40	▨

Matrix cells with missing values are shaded and empty.

The relationships between the variables have a simple pattern, where;

b = a x 2
c = b x 2

This value is "out of pattern"
If it was deleted it would be "put back" inaccurately by the imputation process

Fig 5.5 - Deleting an "out of pattern" value - so that the imputation process can "put it back"

The relationships between the variables ***a, b*** and ***c*** are very strong, with the exception of the single "out of pattern" value. Any good imputation procedure should produce reasonably accurate estimates for the missing values, because the patterns within a large majority of the known values are so strong. Suppose that the accuracy of the imputed values in column ***c*** was being evaluated using the proposed method. Further, suppose that the random deletion process employed by the method removed the single "out of pattern" value - so that a measure of how accurately it was "put back" could be taken. Most imputation methods (particularly regression based techniques) would then impute the "out of pattern" value very inaccurately ***because these methods would force the imputed value to fall into the patterns that exist within the majority of the known values***. Therefore, when a measure of how accurately the deleted value was "put back" was taken, an RD outlier value would be created. For example, suppose that the imputation method produced a set of imputed values that all fell *perfectly* into the patterns among the known values shown in Fig. 5.5. The following RD value would then be calculated for the "out of pattern" value;

$$RD_{ij} = \frac{\left| Y_{ij}.trueVal - Y_{ij}.imputedVal \right|}{\left| Y_{ij}.trueVal \right|}$$

Where $Y_{ij}.trueVal$ is the true (known) value that was deleted.

and $Y_{ij}.imputedVal$ is the value generated by the imputation process.

$$RD_{ij} = \frac{|2 - 400|}{|2|} = \mathbf{199}$$

However, the process of random deletion and "put back" employed by the proposed method is repeated many times, and consequently many other RD values would be calculated for the dataset shown in Fig. 5.5. Most of these RD values would equal zero, indicating 100% imputation accuracy. Nevertheless, the RD outlier value would inflate the MRD (the mean RD value). And if the MRD was considered in isolation - without taking the distribution of the RD values into account - then the user could draw an invalid conclusion regarding the feasibility of the imputation process. This is the principal reason why the automatic creation and display of RD histograms such as the one shown above is such an important part of the software implementation of the method - i.e. it allows the owner of the missing value dataset to consider the distribution of the RD values, so that the effects of extreme RD values can be put into perspective.

For example, if the proportion of RD outliers is large then the proportion of "out of pattern" values within the dataset will also be large. Consequently, the patterns within the known values in the dataset may not be strong enough to support the imputation method being evaluated. Therefore, the feasibility of employing that method to impute the missing values is questionable. But what proportion of RD outliers will invalidate the imputation process for any particular missing value dataset? The answer to this question can only be provided by the owner of that dataset. This point is discussed further in the overall conclusions for the method validity experiments, as given below.

Choosing the parameters that control the imputation evaluation process

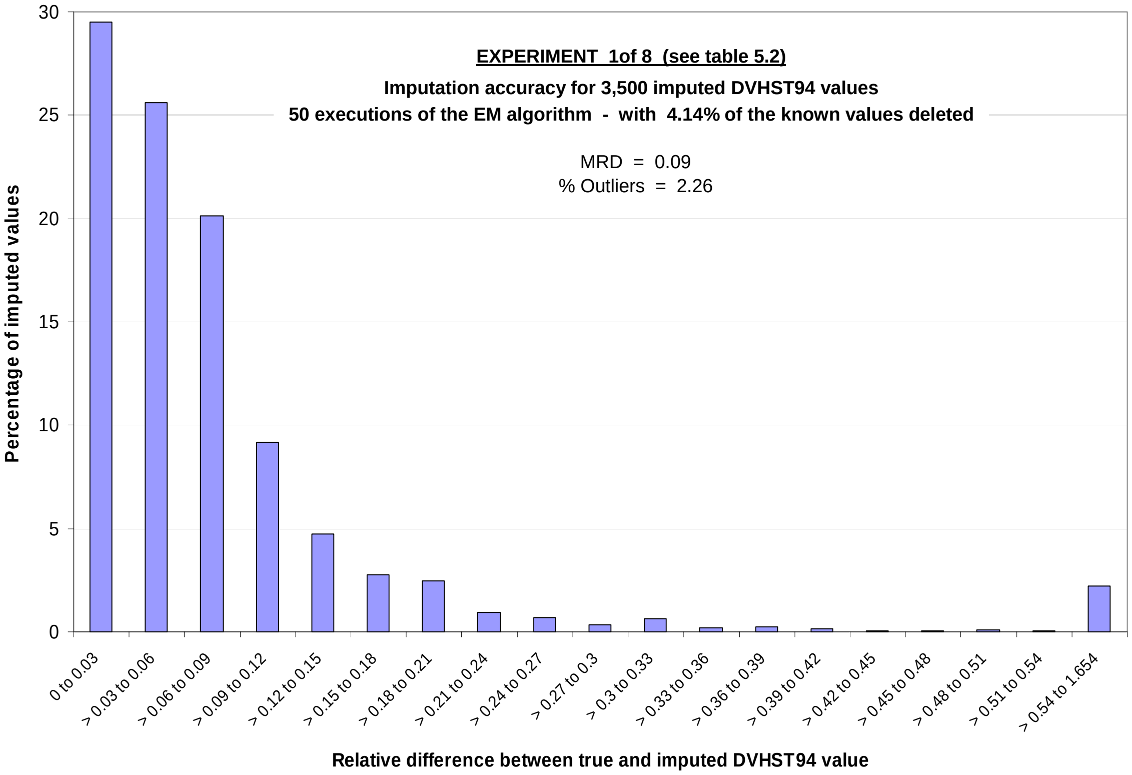
The parameters that control the execution of the imputation evaluation process are shown in Fig. 5.2. These are as follows;

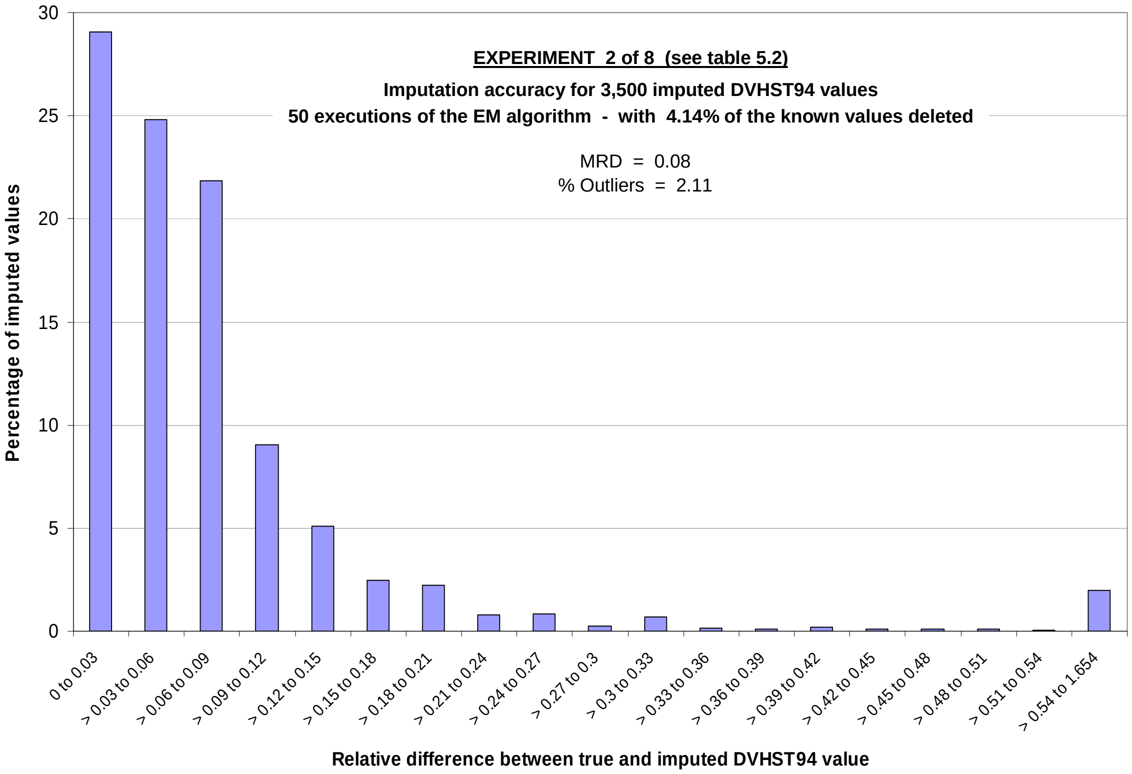
- The first parameter specifies the variable being evaluated for imputation accuracy - e.g. this was the DVHST94 variable for the experiments described above. This parameter is simply a matter of user choice. However, the imputation evaluation process can be repeated for all of the variables that have imputed values, if this is required.

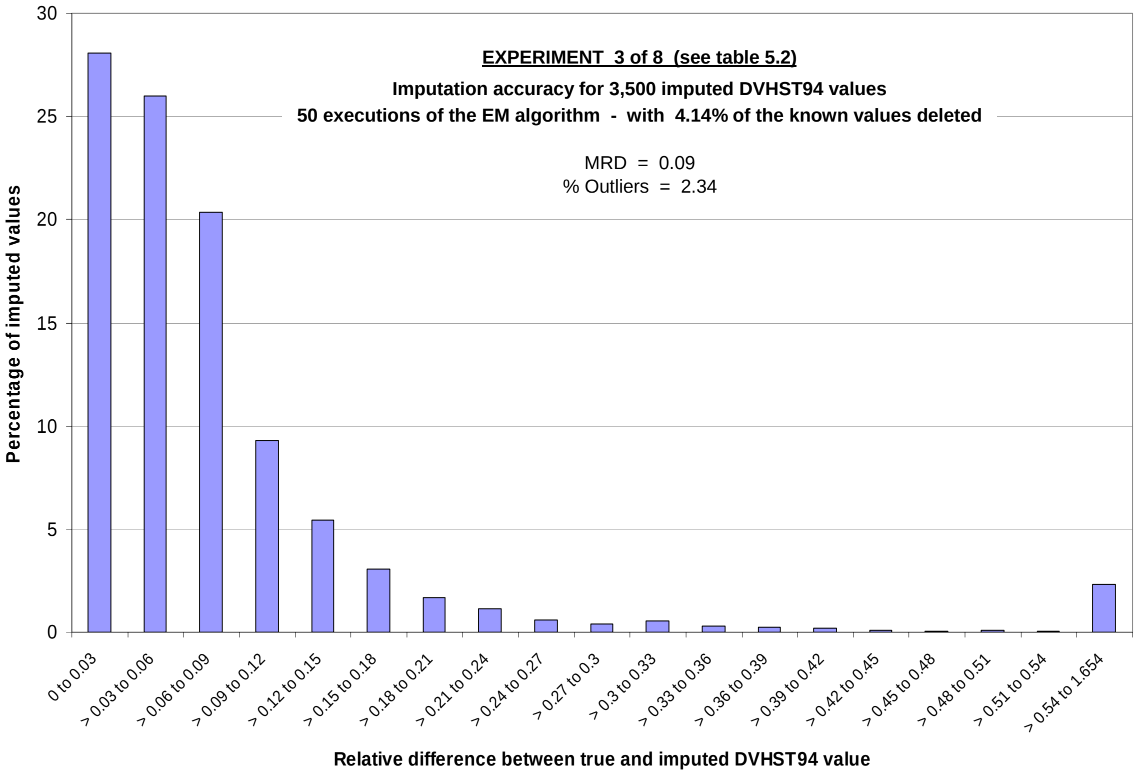
- The second parameter specifies the proportion of known values to be deleted and "put back" by the imputation process - e.g. this was 4.14% for the above experiments (the number of missing values was increased by 10%). Generally, it has been found that deleting between 3% and 5% of the known values produced reliable results for all of the experimental datasets used to date.

- The third parameter specifies the number of executions of the imputation method (the method being evaluated) to be performed - e.g. this was 50 executions of the EM algorithm for the above experiments. Generally, it has been found that at least 10 executions are required for most datasets - but none of the experiments performed to date have required more than 50 executions. However, it could be necessary to repeat the imputation process many more times for some datasets. Consequently, it is important that the implementations of the imputation methods used should execute as quickly as possible.

Some preliminary experimentation may be needed to find the optimum values for the second and third parameters. The simplest way to achieve this is to ensure that the RD histogram displayed by the software gives similar results when the evaluation process is repeated. For example, the three histograms given below show that the parameters chosen for the experiments described in table 5.2 produced very similar imputation accuracy statistics.


EXPERIMENT 1of 8 (see table 5.2)
Imputation accuracy for 3,500 imputed DVHST94 values
50 executions of the EM algorithm - with 4.14% of the known values deleted
MRD = 0.09
% Outliers = 2.26
Percentage of imputed values
30
25
20
15
10
5
0
0 to 0.03
> 0.03 to 0.06
> 0.06 to 0.09
> 0.09 to 0.12
> 0.12 to 0.15
> 0.15 to 0.18
> 0.18 to 0.21
> 0.21 to 0.24
> 0.24 to 0.27
> 0.27 to 0.3
> 0.3 to 0.33
> 0.33 to 0.36
> 0.36 to 0.39
> 0.39 to 0.42
> 0.42 to 0.45
> 0.45 to 0.48
> 0.48 to 0.51
> 0.51 to 0.54
> 0.54 to 1.654
Relative difference between true and imputed DVHST94 value


EXPERIMENT 2 of 8 (see table 5.2)
Imputation accuracy for 3,500 imputed DVHST94 values
50 executions of the EM algorithm - with 4.14% of the known values deleted
MRD = 0.08
% Outliers = 2.11
Percentage of imputed values
30
25
20
15
10
5
0
0 to 0.03
> 0.03 to 0.06
> 0.06 to 0.09
> 0.09 to 0.12
> 0.12 to 0.15
> 0.15 to 0.18
> 0.18 to 0.21
> 0.21 to 0.24
> 0.24 to 0.27
> 0.27 to 0.3
> 0.3 to 0.33
> 0.33 to 0.36
> 0.36 to 0.39
> 0.39 to 0.42
> 0.42 to 0.45
> 0.45 to 0.48
> 0.48 to 0.51
> 0.51 to 0.54
> 0.54 to 1.654
Relative difference between true and imputed DVHST94 value



Overall conclusions for the method validity experiments

The explanations and experimental results given above have shown how the imputation accuracy statistics produced by the proposed method can be used to assess the feasibility of imputing missing values. The experiments have also shown how the variance of the imputation accuracy statistics can be taken into account when considering imputation feasibility. Finally, it has been shown how the parameters that control the execution of the method can be chosen so as to ensure the reliability of the imputation accuracy statistics produced across repeated executions of the method.

However, the final decision regarding the feasibility of the imputation process *for any variable within any dataset* can only be taken by the user of the software. Although it is important to emphasise that this will be a largely subjective decision, based on the user's knowledge and perspective. And of course different users could draw different conclusions regarding the feasibility of the same imputation process. These conclusions could be based on many factors - such as a consideration of how the imputed dataset will be used in practice, or a detailed knowledge of the missing data mechanism.

It is argued that every missing data problem will be unique in some respects and that therefore the evaluation of the results of the imputation process must be based primarily on the knowledge and understanding of the user. Consequently, it is argued that there is no substitute for human judgment when considering these matters, and that the proposed approach simply facilitates the decision making process - by automating the calculation and display of various imputation accuracy statistics.

5.2 Comparing the Predictive Power of Candidate Imputation Methods

This section shows how the proposed method can be used to compare the accuracy of the imputed values generated by the EM and nearest neighbour (NN) algorithms when they are used to impute the missing DVHST94 values in the SSC dataset. The comparison process also provides further evidence of the reliability and the validity of the proposed method.

Section 5.2.1 assesses the feasibility of imputing DVHST94 values using the NN algorithm (see section 3.2.3). Section 5.2.2 explains how the proposed method was used to compare the accuracy of the imputed values generated by the EM and NN algorithms. Sections 5.2.3 and 5.2.4 introduce the ideas underpinning "least distortion" imputation evaluation and show how this technique was used to compare the parameter distortions caused by the two algorithms.

5.2.1 Performing the Nearest Neighbour Imputation Experiments

The previous section explained how the proposed method was used to assess the feasibility of imputing missing DVHST94 values using the EM algorithm. This section explains how the feasibility of imputing DVHST94 values using the NN algorithm was assessed. This allows the predictive power of the two methods to be compared by considering the imputation accuracy statistics generated by both methods. The NN experimental process is described in table 5.4, below. This remainder of this section attempts to answer experimental question 1 and the following two sections attempt to answer experimental question 2.

Table 5.4 - Description of SSC dataset imputation evaluation experiments 9 to 16
Imputation of DVHST94 values using 50 executions of the NN algorithm

EXPERIMENTAL QUESTIONS 1. **Is imputation of the missing DVHST94 values feasible using the nearest neighbour process defined below?** 2. **How does the accuracy of the NN imputed values compare with the accuracy of the EM imputed values?**	
Description of the missing value dataset	• The SSC dataset containing 11 columns and 2389 rows - as described in section 5.1.1
Variable to be imputed and evaluated	• The variable to be imputed and evaluated was DVHST94, which had a range of 0.290 to 1.000 (710 possible values, specified to 3 decimal places) • DVHST94 had 698 missing values - i.e. 29.22% of the 2389 data matrix rows had missing values.
Imputation method used for the experiment	• Imputation was performed using the nearest neighbour algorithm described in section 3.2.3 • The Euclidean distance was used to measure the similarity between observations (matrix rows) • All variables were transformed to standard *Z* scores prior to imputation - so that each variable would carry equal weight in the Euclidean distance calculations.
Imputation evaluation method	• 50 executions of the NN imputation algorithm were performed (using the method described above). • The number of missing DVHST94 values was increased by 10% for each execution of NN. That is, 4.14% of the known DVHST94 values were randomly deleted and "put back" for each NN execution, using the Fig.4.5 algorithm. With balanced random deletion across missingness patterns.

Eight identical, consecutive NN imputation experiments were performed using the method described in table 5.4, above. The results for all eight experiments are shown in table 5.5, below. The means $\hat{\mu}$ and the standard deviations $\hat{\sigma}$ were computed using equations (4.6) and (4.7), as described in chapter 4. The mean value for each column in table 5.5 is given in the bottom row of that table.

Table 5.5 - Aggregated estimates of imputation accuracy for the SSC dataset experiments (imputation of DVHST94 values using 50 executions of the NN algorithm)

Experiment Number	MRD		SRD		MRZ		
	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	% Outliers
9	0.12	0.02	0.18	0.06	5.05	1.44	2.49
10	0.12	0.02	0.19	0.08	4.74	1.53	2.52
11	0.11	0.02	0.17	0.06	4.72	1.19	2.60
12	0.11	0.02	0.17	0.06	5.12	1.07	2.52
13	0.12	0.02	0.18	0.06	4.94	1.16	2.46
14	0.12	0.02	0.18	0.05	5.02	1.51	2.57
15	0.11	0.02	0.17	0.06	5.21	1.49	2.29
16	0.11	0.02	0.17	0.06	5.06	1.16	2.57
Column Mean	**0.12**	**0.02**	**0.18**	**0.06**	**4.98**	**1.32**	**2.50**

Table 5.5 shows that the results for the NN imputation experiments were very similar. It was also found that the eight sets of associated imputation accuracy charts (such as the MRD /SRD charts shown in section 5.1.3) produced by the software were also very similar. ***This provides further evidence of the reliability of the proposed method* -** i.e. it shows that the method produces consistent results when it is used to evaluate nearest neighbour imputation algorithms that are executed against a specific type of numeric dataset.

<u>Assessing the feasibility of imputing DVHST94 values using the NN algorithm</u>

The results given in table 5.5 show that the NN imputation process “put back” the deleted DVHST94 values to within at least 12% of their true values (on average) for every experiment performed. This suggests that imputation of the missing DVHST94 values via NN may be feasible. However, the variance of the RD values also needs to be considered before any conclusion regarding the feasibility of NN imputation can be drawn. This can be achieved by considering the variance of the RD values, as follows.

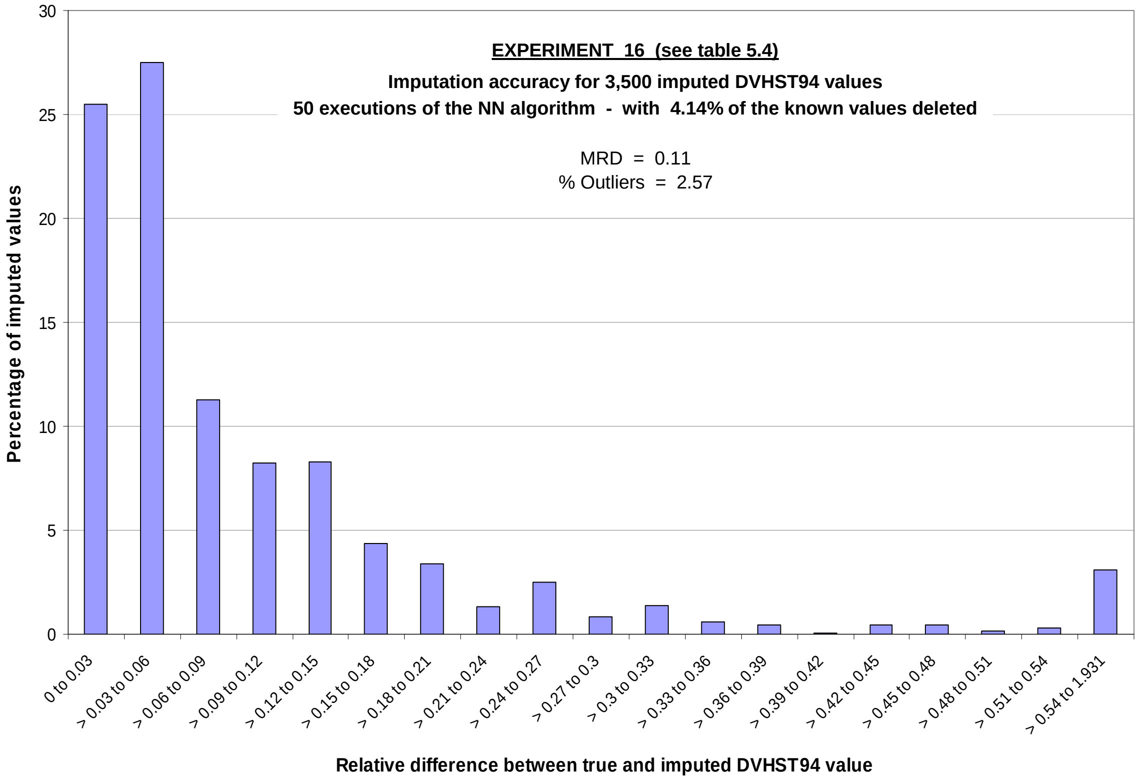


Fig 5.6 - Distribution of the RD values after imputation of DVHST94 using the NN algorithm

The histogram shows that approximately 53% of the imputed DVHST94 values (represented by the two leftmost histogram bars) had an RD value of between zero and 0.06 - i.e. 53% of the values were imputed with an accuracy of 94% or above, with 26% imputed with more than 97% accuracy. The histogram also shows that less than 15% of the DVHST94 values were imputed with less than 82% accuracy. Approximately 3% of the imputed values were RD outliers - i.e. the rightmost histogram bar shows that 3% of the 3,500 values were imputed with less than 46% accuracy.

It was therefore concluded that imputation of the missing DVHST94 values using the NN algorithm is feasible. Although it is important to emphasise that this is - at least in part - a subjective conclusion, which is based on the authors knowledge and perspective (see the discussion of this issue in the conclusions for the previous section).

5.2.2 Choosing the Most Accurate Imputation Method

This section explains how the imputation accuracy statistics generated when executing the EM and NN algorithms can be compared, so that the algorithm that imputes the missing DVHST94 values most accurately can be chosen. This can be achieved as follows.

- The RD distribution histograms shown in Figures 5.4 and 5.6 (see above) can be compared - i.e. the histograms and the analysis given immediately below each histogram can be compared.

- The mean values of the imputation accuracy statistics for the EM algorithm imputation experiments can be compared with the statistics for the NN algorithm experiments, as shown in table 5.6, below (these figures are also given in tables 5.3 and 5.5, above);

Table 5.6 - Comparison of the imputation accuracy produced by the EM and NN algorithms (mean values for DVHST94 imputation across 8 sets of 50 executions of both methods)

Imputation method evaluated	MRD		SRD		MRZ		
	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	% Outliers
EM algorithm	0.09	0.02	0.15	0.06	5.42	1.52	2.47
NN algorithm	0.12	0.02	0.18	0.06	4.98	1.32	2.50

- The $\hat{\mu}$ values for the MRD show that the EM algorithm imputed the missing DVHST94 values more accurately than the NN algorithm - i.e. EM "put back" the deleted values with 91% accuracy, whereas NN achieved 88% accuracy. The MRD standard deviation $\hat{\sigma}$ was the same for both algorithms.

- The $\hat{\mu}$ values for the SRD shows that the accuracy with which the deleted DVHST94 values were "put back" was slightly more variable for the NN algorithm. The SRD standard deviation $\hat{\sigma}$ was the same for both algorithms.

- The % Outliers values show that the EM algorithm produced fewer RD outliers than the NN algorithm. However, the $\hat{\mu}$ and the $\hat{\sigma}$ for the MRZ show that the EM outliers were more extreme, and they had greater variability.

It was therefore concluded that imputation via the EM algorithm should be preferred. However, the differences between the evaluation statistics produced by the two algorithms were quite small, so it could be beneficial to consider some further evidence before deciding which of the algorithms should be employed in practice. The following section describes one way of achieving this.

5.2.3 Least Distortion Evaluation

In many cases the criteria used to evaluate the results of the imputation process will depend primarily on how the imputed dataset will be used in practice. For example, in some cases it will be of primary importance that the mean value for a particular variable is not badly "damaged" by the imputation process - i.e. the mean value measured prior to imputation should not be significantly different to the mean value measured after imputation. Avoiding "damaging" the data can be particularly important when data mining algorithms are to be executed against the data, as Pyle (1999) explains;

> *“Even replacing the values at all has its dangers unless it is carefully done so as to cause the least damage to the data. It is every bit as important to avoid adding bias and distortion to the data as it is to make the information that is present available to the mining tool”*

Consequently, it was considered to be important that the proposed imputation evaluation method should measure the distortion caused by the imputation process. The distortion of the parameters that describe the relationships *between variables* (such as correlations and regression coefficients) can be measured. And the distortion of the parameters that describe *each variable* (such as the mean and standard deviation) can also be measured. This approach has been implemented as part of the software application developed by the author, using the process shown below;

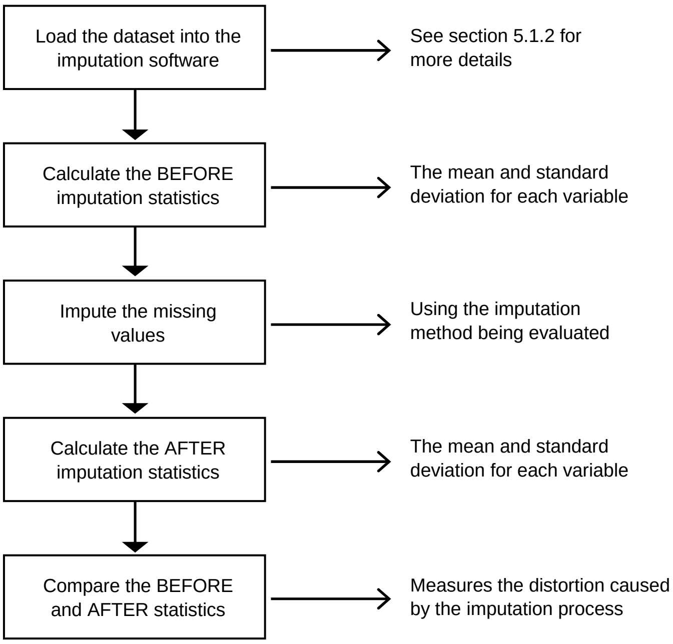

Load the dataset into the imputation software
See section 5.1.2 for more details
Calculate the BEFORE imputation statistics
The mean and standard deviation for each variable
Impute the missing values
Using the imputation method being evaluated
Calculate the AFTER imputation statistics
The mean and standard deviation for each variable
Compare the BEFORE and AFTER statistics
Measures the distortion caused by the imputation process

Fig 5.7 - Implementation of the least distortion evaluation process

The distortions caused by imputation are calculated using the following general formula;

$$\textit{Distortion} = \frac{\textit{After Imputation Statistic} - \textit{Before Imputation Statistic}}{\textit{Before Imputation Statistic}}$$

For example, the post-imputation distortion in the mean might be calculated as follows,

$$\begin{aligned}\textit{Mean before imputation} &= 10\\ \textit{Mean after imputation} &= 12\end{aligned} \qquad \textit{Distortion} = \frac{12 - 10}{10} = 0.2 = 20\ \%$$

However, it is important to note that it will usually be impossible to generate imputed values that minimise the distortion of more than one parameter. For example, consider the following simple univariate example, given by Pyle (1999);

> *"For instance, consider the numbers 1, 2, 3, x, 5, where 'x' represents a missing value. What number should be plugged in as an unbiased estimate of the missing value? Ideally, a value is needed that will at least do no harm to the existing data. And here is a critical point – what does 'least harm' mean exactly? If the mean is to be unbiased, the missing value needs to be 2.75. If the standard deviation is to be unbiased, the missing value needs to be about 4.659"*

In such a situation the user of the imputation software would need to decide whether it was more important to minimise the distortion of the mean, or to minimise the distortion of the standard deviation - since it impossible to do both (as Pyle shows).

Imputation methods usually reduce the variance within imputed variables

In practice, the most common data distortion problem caused by imputation is reduced variance within the imputed variables. For example, regression based imputation methods almost always reduce the variance, because they use the patterns within the *known values* to generate regression equations - which are then used to estimate the missing values. Therefore, they must (by their very nature) strengthen the patterns that existed before imputation was performed - e.g. see the discussion of the dataset given in Fig. 5.5.

The multiple imputation (MI) method (described in section 4.3.3) was devised, at least in part, to solve the problem of variance distortion caused by imputation (Rubin, 1987). The idea of MI is to generate several different estimates for each missing value, using *"repeated random draws from the predictive distribution of the missing values"* (Little and Rubin, 2002) - i.e. stochastic (usually Bayesian) techniques are employed to generate the estimates. The set of estimates generated for each missing value are then "combined" to form a single estimate (e.g. the mean is usually taken). This process will increase the variance within the imputed values in most cases, and hence the distortion of the variance caused by the imputation process should be less severe.

The proponents of MI argue that the MI process allows *"statistically valid"* (Rubin, 1996a) inferences (such as estimates of the variance) to be drawn when analysing the variables in the imputed dataset. However, some doubt has been cast upon the general truth of this assertion. For example, Binder (1996) points out that this depends on the properties of the dataset and on the nature of the missing data problem. Binder makes particular reference to the related papers on this topic by Fay (1991) and Fay (1992), arguing that, *"Fay (1991, 1992) has described what I consider to be a scientific gem. He presented a simple situation where multiple imputation (MI) is not proper, even though one might expect it to be"*. In other words, Fay presents some examples of the MI process which show that equation (4.10), given in section 4.3.3 - does *not yield* an unbiased estimate of the variance - i.e. the MI process used is not *"proper"*, in the sense defined by Rubin (1987).

5.2.4 Comparing the Distortions Caused by the EM and NN Algorithms

Section 5.2.2 concluded that the EM algorithm should be preferred over the NN algorithm for imputation of the DVHST94 values. However, the differences between the imputation accuracy statistics produced by the two algorithms were quite small. This section compares the distortions of the DVHST94 mean and standard deviation caused by both algorithms.

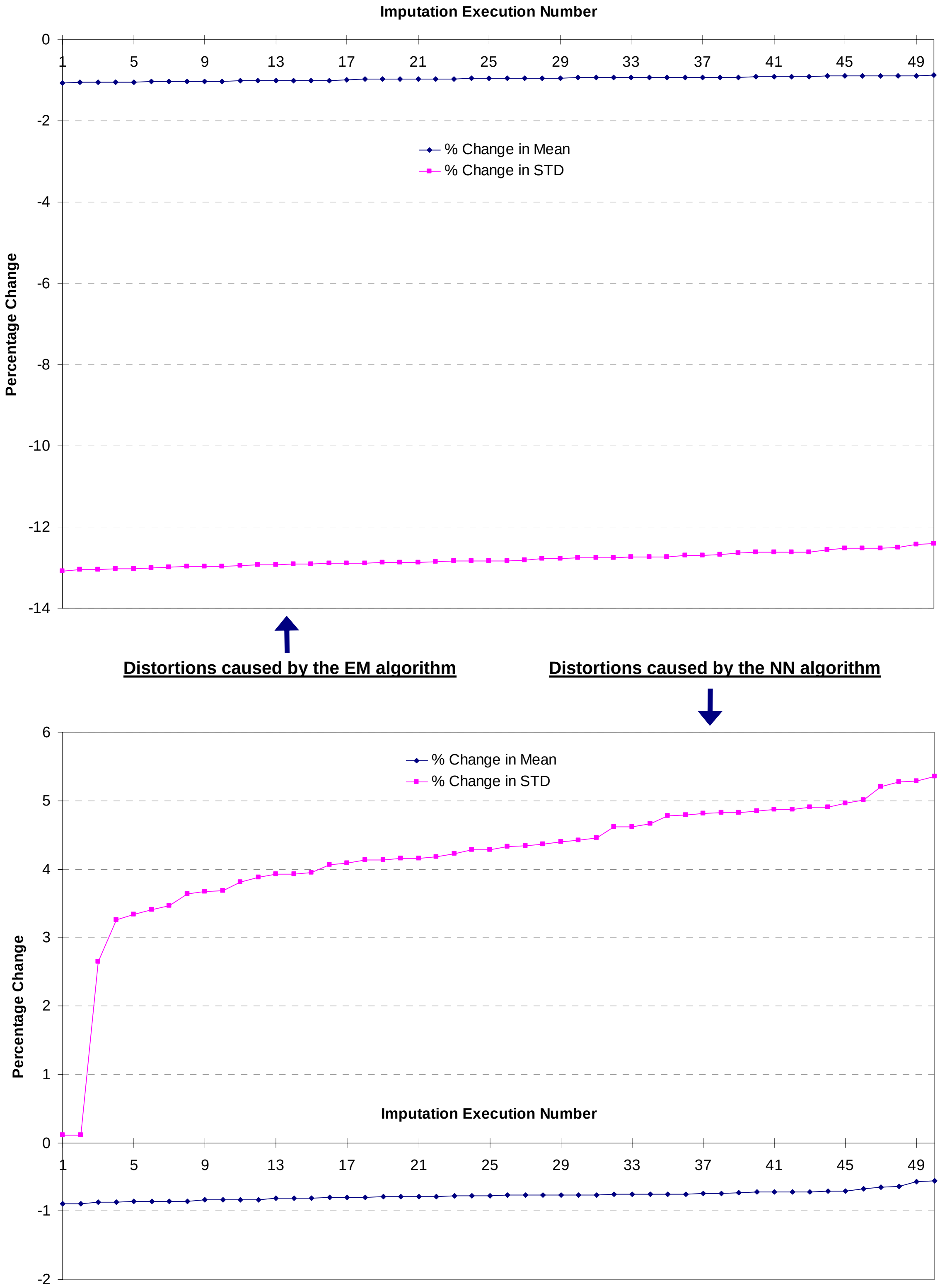


Fig 5.8 – Comparison of DVHST94 parameter distortions caused by the EM and NN algorithms

The software application that implements the proposed method automatically produces and displays line charts such as the ones shown in Fig. 5.8 (see Fig. 5.3 for an explanation of how this is achieved). The two sets of data shown on both charts were sorted independently - i.e. the *% Change in Mean* and the *% Change in STD* figures for each execution number *do not match*. This method of visual presentation enables the parameter distortion statistics produced by the EM and NN imputation algorithms to be more easily compared and evaluated.

The charts show that the distortion of the mean caused by both algorithms was very similar i.e. all 50 executions of both algorithms *reduced* the DVHST94 mean value by between zero and one percent. Therefore, if the distortion of the mean was an important criteria for the comparative evaluation of the imputation methods, then the EM algorithm *should still be preferred* over the NN algorithm, because the distortion produced by both algorithms hardly differs.

However, the distortion of the standard deviation (and hence the variance) caused by the two algorithms differs considerably. The EM imputation process *reduced* the variance of the DVHST94 values by between 12 and 13 percent across all 50 executions charted. This is a typical result for EM, since regression based imputation methods almost always decrease the variance, for the reasons given in the previous section. But the NN imputation process *increased* the variance by between zero and just over five percent across the 50 executions. This is an unusual result, since reduced variance is often caused by over frequent use of the same donor cases when using NN algorithms to generate imputed values (Durrant, 2005).

To conclude, we can say that if the distortion of the variance was an important criteria for the comparative evaluation of the imputation methods, then the user would need to consider the above charts very carefully before deciding whether the EM or the NN algorithm should be used to impute the missing DVHST94 values in practice.

5.3 Summary

This chapter has explained how the proposed imputation evaluation method was experimentally evaluated and has shown that the method produces reliable and valid estimates of imputation accuracy when it is used to evaluate the imputed values generated by the EM and NN imputation techniques.

An explanation of how the software that implements the method can be used to load a missing value dataset, analyse the variables it contains and assess the feasibility of imputing its missing values has been given. A description of how the proposed method can be used to compare the predictive power of candidate imputation methods has been provided. Finally, an explanation of how "least distortion" evaluation can be used to compare the parameter distortions caused by the candidate methods has been given.

The following chapter explains how the proposed method was used to assess the feasibility of imputing the missing values in the collaborating company's dataset. Finally, chapter seven presents the conclusions that have been drawn and describes how the work described in chapters one to six could be continued.

Chapter Six

Applying the Method in Practice: A Case Study

6. Applying the Method in Practice: A Case Study

The work described in this thesis was funded by the EPSRC under the CASE scheme, as described in chapter one. This scheme allows students to collaborate with commercial organisations, so that the results of the work will benefit everyone concerned. For example, to research a topic of common interest, or to find the best way of solving a particular problem. In this case the collaborating company were Trends Business Research (TBR), and TBR's missing data problem forms the case study described in this chapter.

A description of how the collaboration with TBR led to the formulation of the project objectives is given in chapter one. The following sections describe how the proposed imputation evaluation method was used to achieve the first of these objectives. That is, to discover whether imputation of the missing values in TBR's database was feasible, given the large proportions of missing data that it contained.

- **Section 6.1** describes the variables in TBR's missing value dataset.
- **Section 6.2** gives a detailed description of TBR's missing data problem.
- **Section 6.3** explains how the problems caused by the extreme outlier values in the TBR dataset were addressed.
- **Section 6.4** explains how the EM and NN algorithms (see chapters two and three) were customised so that they could be used to impute the missing values in TBR's dataset.
- **Section 6.5** describes the experiments that were performed in order to find the most accurate methods for imputing TBR's missing values.
- **Section 6.6** analyses the experimental results obtained and draws conclusions.

6.1 Description of the Missing Value Dataset

TBR's missing value dataset is stored in a Microsoft SQL Server 2000® database (Vieira, 2003) - referred to as the Trends Central Database (TCD). The records in this database describe approximately 1.48 million UK business organisations - referred to as Firms - ranging from sole traders to conglomerates. The TCD tables contain descriptions of each Firm, including its financial situation, number of employees, business activities and geographical location. The variables used to support the imputation process were extracted from the TCD using SQL, so that they could be loaded into the software application that implements the proposed imputation evaluation method. These variables are described table 6.1, below.

Table 6.1 - Description of the variables in TBR's missing value dataset

Variable name	Data type	Variable description
UKSIC_Category	*Integer*	An integer representation of a categorical alphanumeric code which defines the commercial activities carried out by each Firm, such as *"Publishing of software"* etc.
Employees	*Integer*	Specifies the number of people employed by each Firm
Easting **Northing**	*Integer*	Pinpoints the geographical location of each Firm on the UK map, using two UK Ordnance Survey (OS) mapping co-ordinates.
Sales **Payroll** **Depreciation** **DirectorPay** **NetWorth** **PBT (Profit Before Tax)**	*Currency*	Six numeric variables that describe each Firm's financial situation. These variables all had large proportions of missing values.

Each of the ten variables described in table 6.1 represents one of the columns that was loaded into the data matrix that was used to perform the imputation process. The ten variables taken together represent a data matrix row, which describes a particular Firm.

The first four variables listed in table 6.1 were fully observed, but the other six (the financial variables) all had large proportions of missing data. The imputation of the missing financial values was the problem to be solved.

6.2 Missingness Pattern Analysis: Defining the Missing Data Problem

An analysis of the missingness patterns within the dataset is essential when attempting to solve any missing data problem. This proved to be especially important for the TCD dataset, which was known at the outset to have *very serious* missing financial data problems, as described in the following two sections.

6.2.1 Large Proportions of Missing Data

An analysis of the proportions of missing values for each of the six TCD financial variables (as described in table 6.1, above) revealed the following. Firstly, the proportions of missing values for each variable are unusually large - i.e. they range from 27 to 96 percent, depending on the variable. Secondly, 71 percent of the Firms in the TCD have no known financial figures whatsoever. The details are shown in the associated histogram and table below.

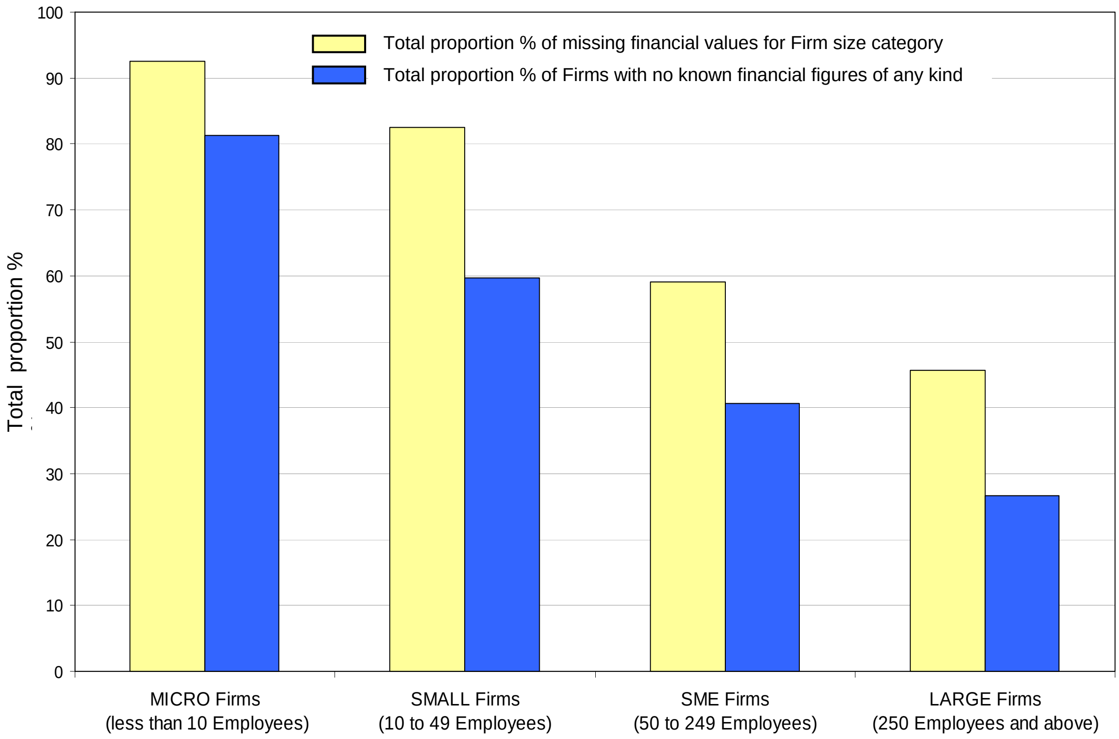


Table 6.2 - Breakdown of proportions of missing financial data for each Firm size category

Financial variable with missing values	Proportions % of missing financial values for each Firm size category			
	1,128,463 MICRO Firms	271,955 SMALL Firms	61,389 SME Firms	18,770 LARGE Firms
Sales	94.52	87.91	67.50	50.49
Payroll	96.15	87.95	63.08	53.39
Depreciation	95.70	87.76	63.90	55.03
DirectorPay	94.00	85.90	59.40	44.49
NetWorth	81.27	59.67	40.69	26.69
PBT	93.89	85.55	58.16	42.58

A commonly recognised Firm size categorization scheme is shown in the associated histogram and table above, where the Firms are divided into four distinct Firm size categories. These categories are frequently used by TBR when they are compiling reports. For example, a report describing the current financial state of small to medium sized (SME) manufacturing companies in a specific geographical region might be required.

Consequently, the missing data problems within each category can be addressed separately, i.e. the TCD dataset can be treated as if it were four separate datasets for the purposes of data analysis. This alleviates TBR's missing data problems, because larger Firms (those with more employees) generally have smaller proportions of missing financial data, as table 6.2 shows. That is, the missing values for all six of the TCD financial variables are MAR, in that the probability of a Firm's financial figures being missing decreases as the Firm's size increases

6.2.2 Unbalanced Missingness Patterns

An analysis of the missingness pattern structure for the TCD financial variables revealed that these patterns are extremely unbalanced. This exacerbates the problems caused by the large proportions of missing financial data described in the preceding section. The histogram below shows that the financial missingness patterns are dominated by the pattern where no financial figures are known, and the pattern where only the NetWorth figure is known. However, some of the Firms in the TCD have a complete set of Financial figures, as shown below.

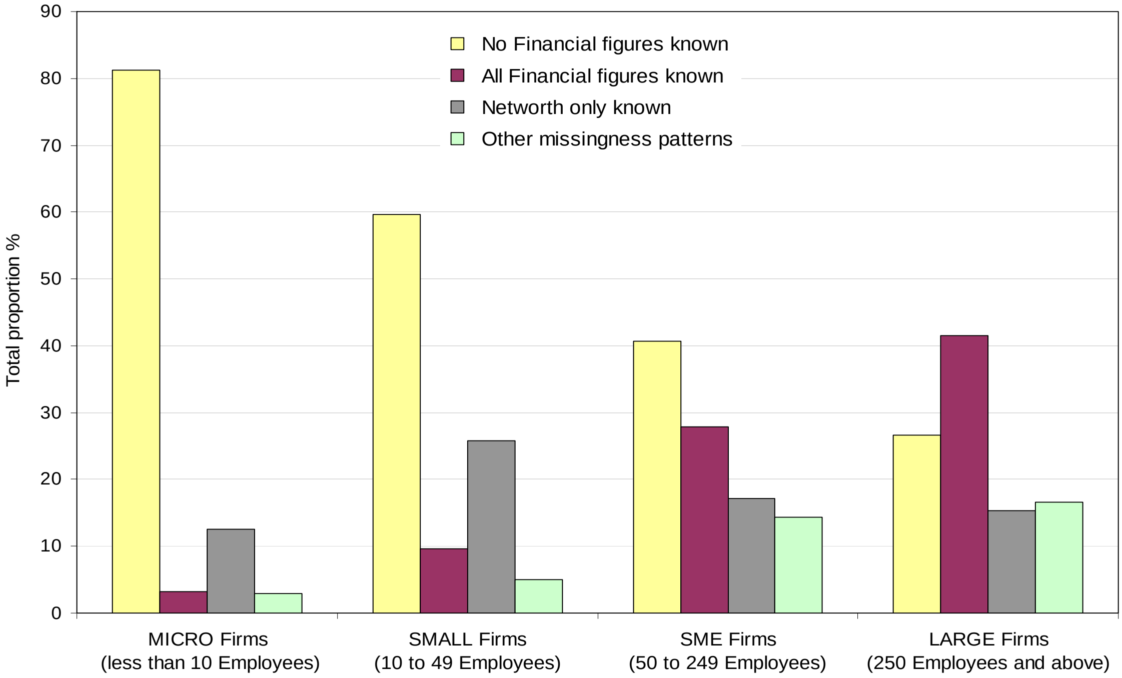


Table 6.3 - Relative sizes of missingness patterns for each Firm size category

Missingness pattern type	MICRO Firms %	SMALL Firms %	SME Firms %	LARGE Firms %
No Financial figures known	81.27	59.67	40.68	26.62
All Financial figures known	3.22	9.56	27.83	41.46
Networth only known	12.57	25.78	17.11	15.28
Other missingness patterns	2.94	4.99	14.38	16.64

The SME and LARGE Firm categories both have a significant proportion of Firms with a *complete set* of Financial figures. Further, the SME and LARGE Firm categories both have a reasonable proportion of Firms that have at least one known Financial value (other than NetWorth), as represented by the *"Other missingness patterns"* histogram bar.

Preliminary experiments revealed that the very large proportions of missing financial data and the unbalanced missingness patterns produced very inaccurate imputed financial values for the MICRO and SMALL Firm categories. It was therefore decided that all further efforts would be concentrated on the imputation of the missing financial values within the SME and LARGE Firm categories.

6.3 Addressing the Problem of Extreme Outlier Values

Preliminary investigations revealed that a small proportion of *very extreme* outlier values were adding long tails of very low frequency intervals to the distributions of the TCD financial variables. The hypothesis was that these distorted distributions were reducing the accuracy of the imputed values. Therefore, an attempt was made to detect and remove those Firms that contained one or more outlier values in an effort to discover whether this would improve imputation accuracy.

Several methods for the detection of outlier values have been proposed. See for example Iglewicz and Hoaglin (1993) and Rousseeuw and Leroy (1987). The outlier detection method proposed by the RSC (2001) was found to be an appropriate choice for the purpose of detecting the long tails of TCD outliers, for the reasons discussed below. The RSC (2001) method is simply a robust version of the outlier detection method that can be used when the variables of interest are normally distributed, as shown below.

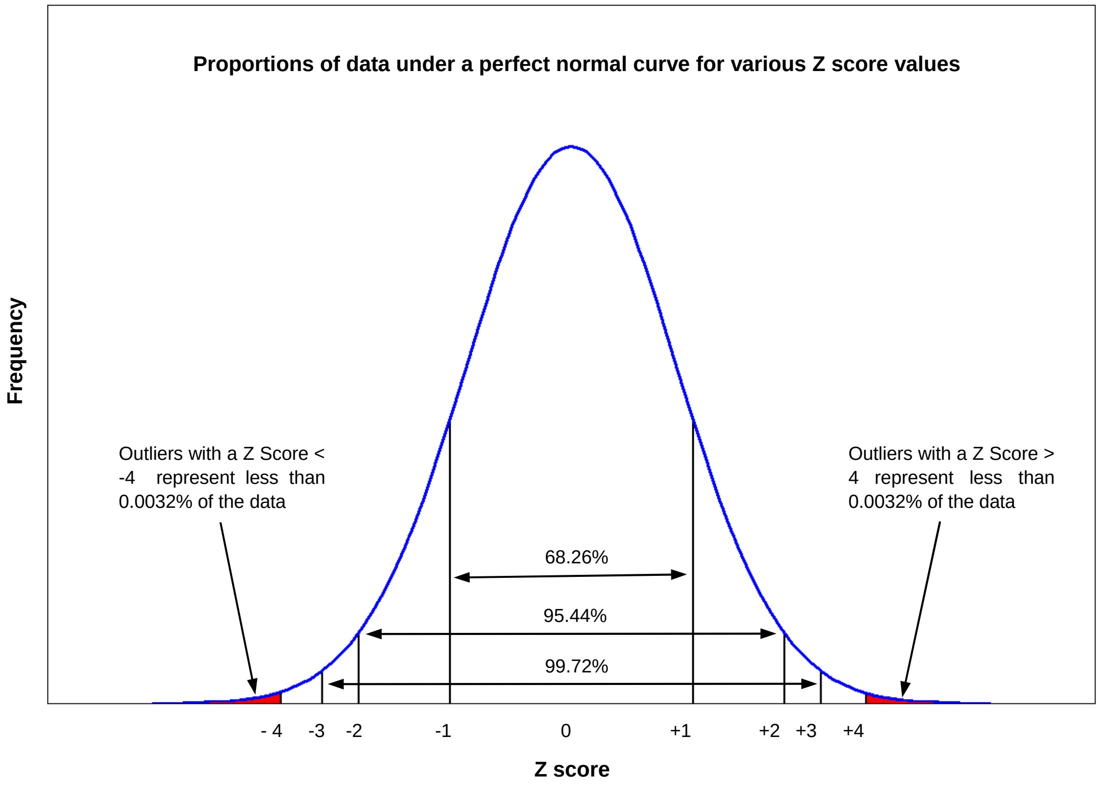


Fig 6.1 - Detecting outlier values in a perfectly normally distributed variable using Z scores

Fig. 6.1 shows how Z scores can be used to detect outlier values in a perfectly normally distributed variable. Where the Z score for any particular value y_i of the variable $y = (y_1, y_2 \ldots\ldots y_n)$ with mean μ and standard deviation σ is given by;

$$Z = \frac{y_i - \mu}{\sigma}$$

Essentially, the *Z* score is a measure of the number of standard deviations by which any particular value of a variable deviates from the mean. Fig. 6.1 shows that for a perfectly normally distributed variable, 68.26% of the data lies within ± 1 standard deviations of the mean, and 99.72% of the data lies within ± 3 standard deviations of the mean. Therefore, when the data is normally distributed, outlier values can be both defined and detected using any required *Z* score range. For example, Fig. 6.1 shows that the range ± 4 would mark approximately 0.0064% of the data values as outliers.

However, this method of detecting outliers failed to produce the desired results for the TCD financial variables because the μ and σ for each of these variables was massively inflated by the small proportions of *very extreme* outlier values. In other words, the outlier values *themselves* were having such a disproportionate effect on the *Z* score calculations that very few outlier values were being detected. Nevertheless, it was found that the *Z* score method could be used to effectively detect TCD financial outliers by using robust, outlier resistant estimates of μ and σ to calculate the Z scores, as follows

$$\hat{Z} = \frac{y_i - med\ (y)}{MAD\ (y)}$$

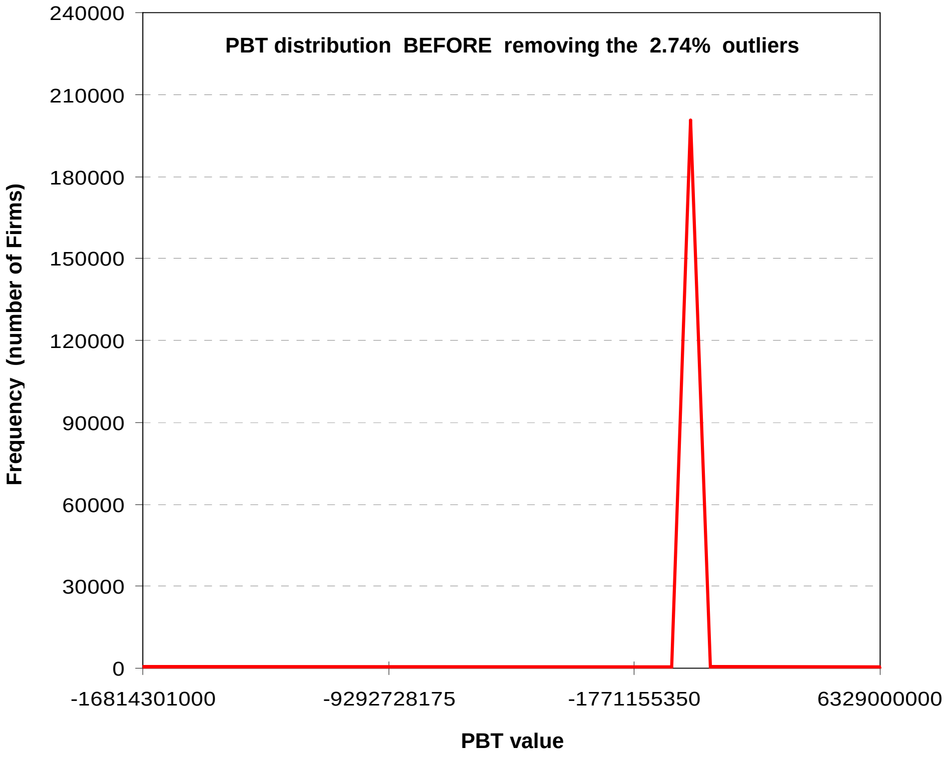
Where $\hat{Z}$ is a median based estimate of the standard Z score and $med(y)$ gives the median value for the variable $y = (y_1, y_2 \ldots\ldots y_n)$ and $MAD(y)$ gives the *Median Absolute Difference* for $y = (y_1, y_2 \ldots\ldots y_n)$, which is computed as follows,

$$MAD(y) = med(|y_1 - med(y)|, |y_2 - med(y)| \ldots\ldots |y_n - med(y)|)$$

The median is more robust measure of central tendency than the mean for distributions with long tails of low frequency extreme outliers (such as the TCD financial variables), because its value is not affected by the outliers *themselves*. The *MAD* is a more robust measure of variability than the standard deviation for distributions with long tails of low frequency extreme outliers, because its value is not affected by the outliers, since it is calculated using only the median - as explained by the RSC (2001).

6.3.1 Detecting TCD Financial Outlier Values Using Robust Z Scores

The outlier detection method described in the previous section has been implemented as part of the software that implements the proposed imputation evaluation method. This functionality was used to detect the outlier values for all of the financial variables in the TCD. This proved to be very effective, as the example below shows.



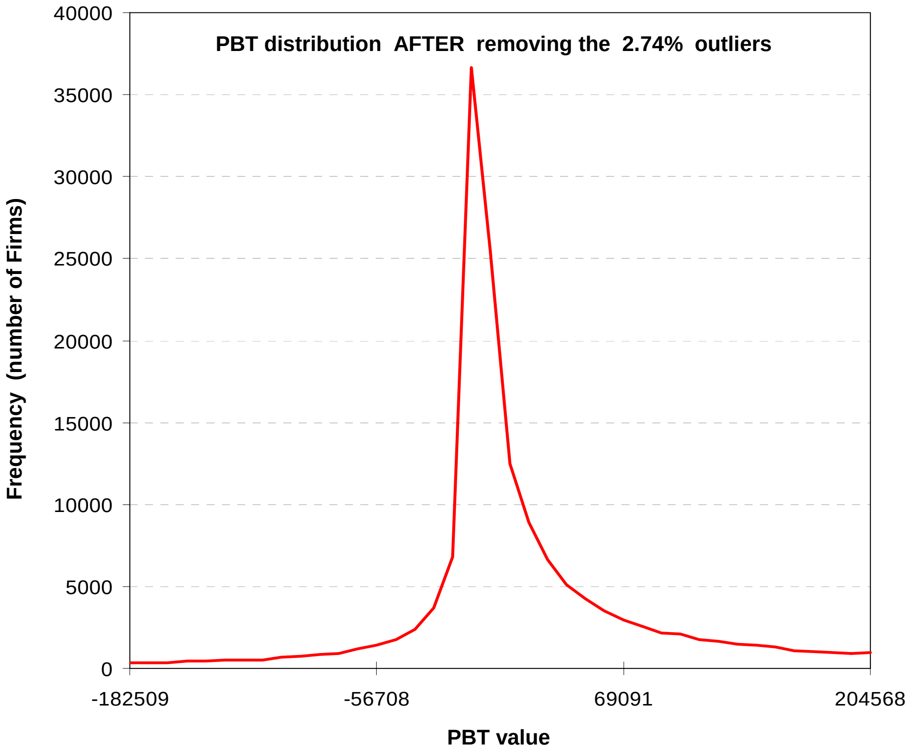


Fig 6.2 - Removing outlier values from the PBT distribution for approximately 1.48 million Firms

Fig. 6.2 shows the results of removing outliers with a robust $\hat{Z}$ score of more than ± 4 from the PBT variable's distribution. The detail in the tails of the distribution shown in the first chart is obscured, because the number of Firms in the outlier intervals is generally less than six, which is a very small proportion of the scale on the chart's y axis. The charts show that removing the small proportion (2.74%) of outlier values has dramatically rescaled the PBT maximum and minimum values. The removal process has also revealed - and centred - the hidden PBT approximation of normality shown in the second chart. The results of applying the same outlier detection process to all of the TCD financial variables are summarised in table 6.4, below.

Table 6.4 - Description of financial outlier values with a robust Z score of more than ± 4

Description of statistic	Sales	Payroll	Depreciation	DirectorPay	NetWorth	PBT
Proportion of outliers with a Z score of more than ± 4	2.41%	2.02%	1.90%	3.35%	7.61%	2.74%
Mean outlier Z score for the financial variable	135	79	254	39	380	219
Maximum outlier Z score for the financial variable	60,268	247,364	923,849	29,712	1,748,221	347,476
(Stan dev / mean) ratio BEFORE removing the outliers	15	33	48	9	57	230
(Stan dev / mean) ratio AFTER removing the outliers	1.39	1.46	1.37	1.83	2.4	2.86

Evidence of the *very extreme* nature of the financial outlier values is shown in the mean and maximum values of the $\hat{Z}$ scores found. In the most extreme case, the NetWorth maximum $\hat{Z}$ score was found to be 1,748,221. Table 6.4 also shows that before removing the outliers the standard deviation was very large compared to the mean, indicating the extremely high variance caused by the outliers. In the most extreme case, the PBT standard deviation was found to be 230 times larger than the mean. However, after removing the outliers it can be seen that the ratio of the standard deviation to the mean decreases substantially.

6.4 Imputation Methods Used for the TCD Experiments

This section describes how the EM and NN algorithms (described in chapters two and three) were customised, so that they could be used to impute the missing values in TBR's dataset. Section 6.4.1 explains how the variables used by the EM algorithm were chosen and transformed, so as to get the best possible results from the EM imputation process. Section 6.4.2 explains how the NN imputation process was customised, so that the missing TCD financial values would be imputed with the greatest possible accuracy.

6.4.1 Using the EM Algorithm to Impute TCD Financial Values

The EM algorithm implementation described in chapter two was used impute the missing TCD financial values. The main questions to be answered when using the EM algorithm for this purpose were;

1. Which of the variables in the TCD dataset should be included in the data matrix used by the EM algorithm?
2. How could the variables in the EM data matrix be transformed, so as to get the best possible results from the EM imputation process?

1. <u>Finding out which variables should be included in the EM data matrix</u>

The Easting and Northing variables pinpoint the geographical location of each Firm on the UK map (see the variable descriptions given in table 6.1). It is often argued that commercial enterprises in the south of the UK are more profitable than their counterparts in the north and that employees in the south earn more. Therefore, it was possible that adding these two variables to the data matrix would increase the predictive power of the regression equations generated by the EM imputation process. However, preliminary experiments revealed that the inclusion of these two variables slightly reduced the accuracy of the imputed values, so they were excluded from the EM data matrix.

It seemed unlikely that adding the UKSIC_Category variable to the EM data matrix would increase the predictive power of the EM regression equations, since this variable is simply an integer representation of an alphanumeric code (see the following section for more details). However, this was tried, but it was found that the accuracy of the imputed financial values decreased noticeably, so the UKSIC_Category was excluded from the EM data matrix.

The other variables in the dataset were the Employees variable and the six financial variables. These variables were added to the data matrix in various combinations and it was found that the most accurate imputed values were produced when they were all included. And of course this was necessary, since it was the missing financial values that were to be imputed.

2. Transforming the variables to get the best results from the EM process

It is well known that the EM algorithm performs at its best when the missing value dataset contains normally distributed variables (see Schafer, 1997, among others). However, some of the TCD financial variables were far from being normally distributed, even after the extreme outlier values were removed (using the method described in the previous section). Therefore, an attempt was made to transform the financial distributions to approximate normality using the equations given in the seminal paper by Box and Cox (1964), and as described below;

The transformation $y^{(\lambda)} = (y_1^{(\lambda)}, y_2^{(\lambda)} \ldots\ldots y_n^{(\lambda)})$ of $y = (y_1, y_2 \ldots\ldots y_n)$ is given by;

$$y^{(\lambda)} = \begin{cases} ((y+c)^{\lambda} - 1)/\lambda & \text{for } \lambda \neq 0 \\ \ln(y+c) & \text{for } \lambda = 0 \end{cases} \qquad (6.1)$$

Where c is a shift parameter chosen large enough to ensure that $(y+c) > 0$, for all values in the matrix column containing y. The use of c is standard procedure, and it was required for the TCD variables, since Box-Cox transformations can only be applied to positive variables, and some of the TCD financial variables (such as PBT) can take negative values.

The main problem to be solved when applying equation (6.1) to any particular variable is to find the value of the λ parameter which will transform $y = (y_1, y_2 \ldots\ldots y_n)$ to the best possible approximation of a normal distribution. Several algorithms have been developed to find the optimal value of λ. See for example Ogwang and Rao (1997) and Press et al. (1992). For imputation purposes the fast and efficient algorithm proposed by Coleman (2004) seemed to be the best choice. This algorithm uses an expectation-maximisation approach (in the spirit of the EM algorithm) to find the optimal value of λ. This is achieved by using an iterative procedure to maximise the following log-likelihood function;

$$\ell(\lambda) = -\frac{n}{2}\ln\left[\frac{1}{n}\sum_{i=1}^{n}\left(y_i^{(\lambda)} - \overline{y^{(\lambda)}}\right)^2\right] + (\lambda - 1)\sum_{i=1}^{n}\ln(y_i) \qquad (6.2)$$

Where i indexes $y^{(\lambda)} = (y_1^{(\lambda)}, y_2^{(\lambda)} \ldots\ldots y_n^{(\lambda)})$ and $y = (y_1, y_2 \ldots\ldots y_n)$, and where,

$$\overline{y^{(\lambda)}} = \frac{1}{n}\sum_{i=1}^{n} y_i^{(\lambda)}$$ in equation (6.2) gives the mean value of $y_i^{(\lambda)}$

Coleman's (2004) algorithm was implemented as part of the software application that implements the proposed imputation evaluation method and it was found to be very effective. The results of applying this algorithm to the SME Payroll variable are shown below.

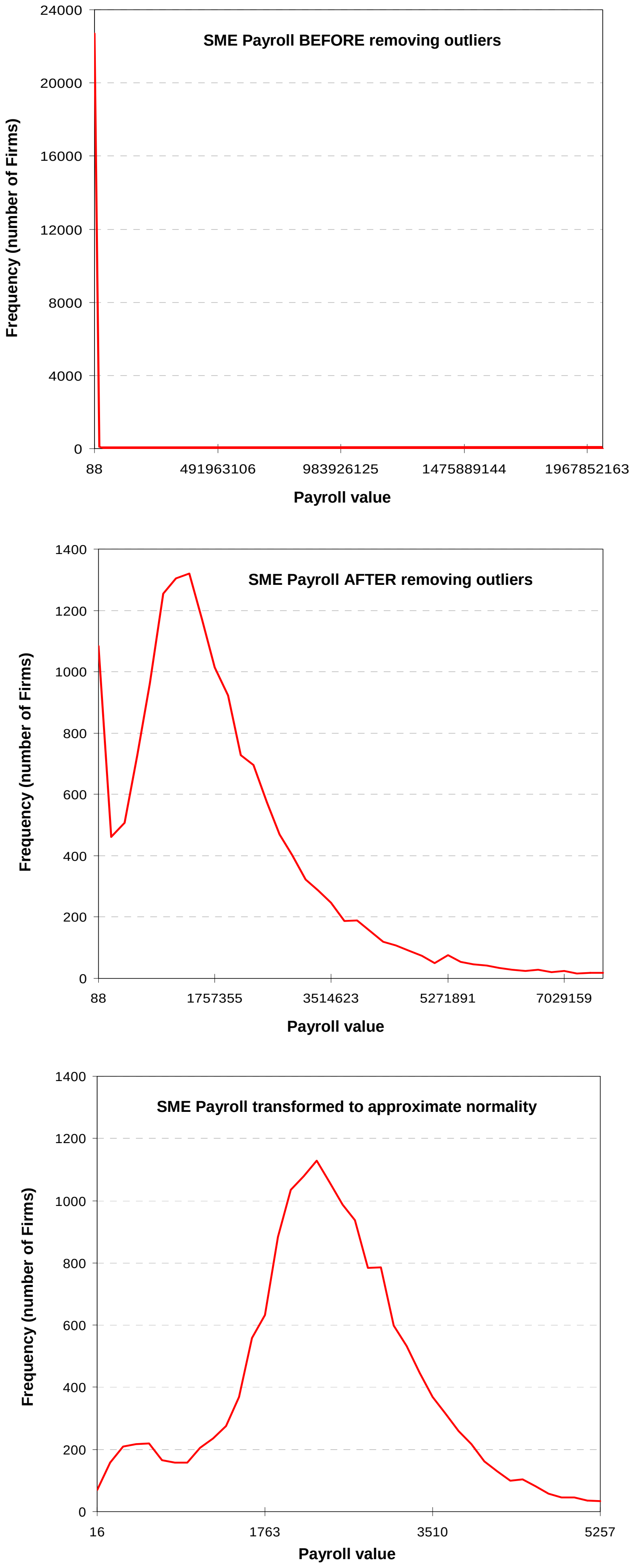


Fig 6.3 - Transforming the SME Payroll variable to an approximate normal distribution

The two stage transformation process shown in Fig. 6.3 was applied to all of the variables in the data matrix used by the EM algorithm and the distributions were transformed to approximate normality in every case. When the imputation process was complete the variables in the matrix were restored to their original distributions, using the reverse Box-Cox transformations given by,

$$y = \begin{cases} (\lambda y^{(\lambda)} + 1)^{1/\lambda} - c & \text{for } \lambda \neq 0 \\ \exp(y^{(\lambda)}) - c & \text{for } \lambda = 0 \end{cases}$$

Where c is the same shift parameter used in equation (6.1), given above. The experimental results given in Appendix A and in section 6.6 show that applying this two stage transformation process to the variables increased the accuracy of the imputed financial values. In addition, it was found that the application of this process increased the execution speed of the EM algorithm considerably - i.e. the number of iterations required for EM to converge (see section 2.2.3 for a discussion of EM convergence) decreased by about 30% (and in some cases much more) for every experiment.

6.4.2 Using the Nearest Neighbour Algorithm to Impute TCD Financial Values

The implementation of the EM algorithm described above *does not* utilise the information content within the fully observed UKSIC_Category, Easting and Northing variables (for the reasons given in the preceding section). However, preliminary experiments revealed that the accuracy of the imputed values increased when these three variables were utilised by the NN algorithm. The question was: would the NN algorithm produce more accurately imputed financial values than the EM algorithm because it utilised this additional information?

The NN algorithm given in Fig. 3.4 was used to perform the imputation process. This algorithm imputes the missing financial value in a particular Firm (matrix row) F_m by taking a copy of the known value from the closest donor Firm F_d , such that;

$$\text{Imputed financial value} = F_{mc} = F_{dc}$$

Where c is the matrix column with the missing financial value (to be imputed).
And where m is the matrix row index of the Firm with the missing value.
And where d is the matrix row index of the Firm containing the donor value.

Where the closest donor Firm F_d is found by comparing F_m with all of the other Firms in the same UKSIC_Category as F_m , and using the Firm that returns the smallest value of the multivariate Euclidean distance function $dist\left(F_m, F_d\right)$ as the donor - i.e. Finding the minimum value of $dist\left(F_m, F_d\right)$ for all $F_d \in R$, where;

$$dist\ (F_m\, , F_d\,) \;=\; \sqrt{\sum_{j \in S} (F_{dj} \;-\; F_{mj}\,)^2} \quad \text{for all} \quad F_d \in R \qquad (6.3)$$

Where $R = \{ F_1 ,....\ F_k \}$ is the set of all Firms in the same UKSIC_Category as F_m

And where $d = 1\ \ to\ \ k\ \ (d \neq m)$ indexes the Firms (matrix rows) in the set R

And where $j \in S$ indexes the matrix columns that have *known* values in both F_m and F_d

i.e. using the matrix row comparison method defined in Fig. 3.1.

Searching for donor Firms within the most suitable UKSIC categories

The method used to decide whether a set of Firms were in the same UKSIC category (deciding which $F_d \in R$) requires further explanation, because the Firms within the TCD dataset can be segmented at five different levels of UKSIC granularity. The lowest level of granularity (level 1) creates the smallest number of segments and the highest level (level 5) creates the largest number of segments, as shown in table 6.5, below.

Table 6.5 - Representation of the Education / Health & Social Work UKSIC categories in the TCD

UKSIC segmentation levels 1 to 5 (Number of segments created)					UKSIC category details as stored in the TCD database		
1 (1)	2 (2)	3 (7)	4 (12)	5 (15)	UKSIC code	UKSIC category description	Number of Firms
8	0	0	0	0	80000	*Education*	58
8	0	2	1	0	80210	*General secondary education*	27,997
8	0	2	2	0	80220	*Technical and vocational secondary education*	183
8	0	3	0	1	80301	*Sub-degree level higher education*	76
8	0	3	0	2	80302	*First-degree level higher education*	4,563
8	0	4	2	1	80421	*Activities of private training providers*	11,359
8	0	4	2	9	80429	*Other adult and other education not elsewhere classified*	25,691
8	5	1	1	0	85110	*Hospital activities*	5,287
8	5	1	1	3	85113	*Nursing home activities*	3,003
8	5	1	2	0	85120	*Medical practice activities*	17,418
8	5	1	3	0	85130	*Dental practice activities*	11,308
8	5	1	4	0	85140	*Other human health activities*	30,574
8	5	2	0	0	85200	*Veterinary activities*	4,367
8	5	3	1	2	85312	*Non-charitable social work activities with accommodation*	19,034
8	5	3	2	2	85322	*Non-charitable social work activities without accommodation*	31,467

The TCD dataset contains ten level 1 UKSIC segments, numbered 0 to 9, and all of the UKSIC categories in segment number 8 are shown in table 6.5, above. Hence, if the dataset was segmented at level 1, then ten segments would be created and the search for each donor Firm would take place within the largest possible number of UKSIC categories. It follows that segmenting the Firms at level 5 should produce the most accurately imputed values, because the search for each donor Firm would then take place within a single UKSIC category. However, preliminary experiments revealed that segmenting at level 3 produced the best results, because segmenting at levels 4 and 5 created several segments with 100% missing values, which meant that no donor Firms could be found for many of the missing values.

However, the benefits of searching for donor Firms within the best UKSIC segments were somewhat reduced, for the following reasons. Firstly, some UKSIC categories had much larger proportions of missing data than others. Secondly, some of the Firms in the TCD had been placed in the wrong UKSIC categories by mistake. This was partly caused by placing Firms which could *not be easily categorised* into "catch all" UKSIC categories, such as the *"Other adult and other education not elsewhere classified"* category, shown in table 6.5.

Scaling the variables used for the Euclidean distance calculations

Equation (6.3) repeatedly measures the distance between two Firms in nine dimensional Euclidean space, because nine of the ten variables given in table 6.1 are included in the $dist\left(F_m, F_d\right)$ computations (the UKSIC_Category is excluded). To clarify, if *only* the Easting and Northing variables were included in the computation then equation (6.3) would find the *geographically closest* donor Firm in the same UKSIC_Category as the Firm with the missing value - i.e. by comparing all of the two dimensional Euclidean distances, which can be easily visualised.

However, some of the nine variables included in the computation had much larger values than others (such as NetWorth) and these variables were having a disproportionate effect on the $dist\left(F_m, F_d\right)$ values. In particular, the Employees variable was being "swamped" by the (much larger) financial variables, so that the number of Employees was having very little effect on the $dist\left(F_m, F_d\right)$ results. This problem was solved as described below.

Firstly, the Employees variable and the six financial variables were scaled, so that they all carried the same weight in the distance calculations. This was achieved by transforming the variable values to their Z scores prior to executing the NN algorithm, as suggested by Manly (1986). This simple process noticeably improved the accuracy of the imputed financial values. Secondly, the Easting and Northing variables were divided by 100,000 just after they were loaded into the data matrix. This gave these variables about one tenth of the weight of the other variables, which proved to be very effective. That is, various weighting schemes were tried for the Easting and Northing variables using a trial and error approach and dividing by 100,000 seemed to produce the most accurately imputed financial values.

6.5 SME and LARGE Firm Financial Imputation Experiments

The missingness pattern analysis process described in section 6.2 defined the TCD missing data problem. This led to the conclusion that all further efforts should be concentrated on the imputation of missing financial values within the SME and LARGE Firm categories. Following this, the investigations described in section 6.3 led to the formulation of the hypothesis that removing all Firms containing one or more financial outlier values might improve imputation accuracy. This led to the implementation of a method for removing the outlier Firms, which proved to be very effective. Following this, the work described in section 6.4 led to the implementation of the EM and NN algorithm modifications needed for the imputation of the missing TCD financial values.

Thus, the stage has been set for the description of the experiments which were designed to find the most accurate methods for imputing the missing financial values. The following two sections give a detailed description of these experiments.

<u>Description of the 48 financial variable imputation experiments</u>

Twelve sets of four imputation experiments were performed. The same TCD financial variable was imputed for each set of four experiments. For example, the list below describes the set of experiments that were performed for the SME Firm Payroll variable.

1. Imputation using the *EM algorithm* with outlier Firms *retained.*
2. Imputation using the *EM algorithm* with outlier Firms *deleted.*
3. Imputation using the *NN algorithm* with outlier Firms *retained.*
4. Imputation using the *NN algorithm* with outlier Firms *deleted.*

The objective was to discover which of these four imputation methods would produce the most accurately imputed SME Payroll values. The same set of four experiments were performed for each of the following twelve variables;

- **<u>SME Firm variables</u>** Sales, Payroll, Depreciation, DirectorPay, NetWorth, PBT
- **<u>LARGE Firm variables</u>** Sales, Payroll, Depreciation, DirectorPay, NetWorth, PBT

Thus, 48 experiments were performed in total. Fifty consecutive executions of the required imputation algorithm (EM or NN) were performed for each of the 48 experiments. Hence, 2,400 executions of the imputation algorithms (1,200 for EM and 1,200 for NN) were performed. A small proportion of the known values were randomly deleted and “put back” for each execution of the EM and NN algorithms, using the procedure given in Fig. 4.5. That is, the proposed imputation evaluation method was executed 48 times, using 50 iterations per execution. The following two sections describe the 48 experiments in more detail, using the pro-forma that has been designed to support the proposed method.

6.5.1 Definition of the EM Imputation Experiments

Tables 6.6 and 6.7 describe the EM algorithm imputation evaluation experiments that were performed for the SME Payroll variable. The same pair of experiments were repeated for all 12 of the SME and LARGE Firm financial variables. That is, the descriptions given in tables 6.6 and 6.7 hold for all of the TCD financial variables, with the only difference being the proportion of missing values for each variable, as given in table 6.2, above.

Table 6.6 - Description of TCD imputation evaluation experiment 1 (EM retaining outlier Firms)
Imputation of SME Payroll values using 50 executions of the EM algorithm

EXPERIMENTAL QUESTION **Can the missing SME Payroll figures be accurately imputed using the EM imputation process described below?**	
Description of the missing value dataset	• All 61389 SME Firms were loaded into the data matrix from the TCD database, using SQL. • The TCD columns loaded into the matrix were: Sales, Payroll, Depreciation, DirectorPay, NetWorth, PBT and Employees. All columns contained integer values only.
Variable to be imputed and evaluated	• The variable to be imputed and evaluated was Payroll. • Payroll had 38724 missing values - i.e. 63.08% of the 61389 data matrix rows had missing values.
Imputation method used for the experiment	• Imputation was performed using the EM algorithm • The EM algorithm convergence value was 0.0001 • Box-Cox power transformations were performed for all variables. • The initial covariance matrix was created using all data matrix rows with a full set of known values. • All imputed values were rounded to the nearest integer before estimating the predictive accuracy of the imputed values.
Imputation evaluation method	• 50 executions of the EM imputation algorithm were performed (using the options described above). • No outlier Firms were removed from the matrix. • 4.16% of the known Payroll values were randomly deleted and “put back” for each EM execution, using the Fig.4.5 algorithm. With balanced random deletion across all missingness patterns.

Table 6.7 - Description of TCD imputation evaluation experiment 2 (EM deleting outlier Firms)
Imputation of SME Payroll values using 50 executions of the EM algorithm

EXPERIMENTAL QUESTION **How would deleting outlier Firms from the data matrix affect EM imputation of the missing SME Payroll figures?**
This experiment was identical to the experiment described in table 6.6, except that all Firms (matrix rows) that contained *any financial variable* with a robust Z score of more than $\pm$ 4 were deleted from the data matrix. That is, 8251 of the 61389 Firms were deleted from the matrix prior to the first execution of the EM imputation process

6.5.2 Definition of the Nearest Neighbour Imputation Experiments

Tables 6.8 and 6.9 describe the NN algorithm imputation evaluation experiments that were performed for the SME Payroll variable. The same pair of experiments were repeated for all 12 of the SME and LARGE Firm financial variables. That is, the descriptions given in tables 6.8 and 6.9 hold for all of the TCD financial variables, with the only difference being the proportion of missing values for each variable, as given in table 6.2, above.

Table 6.8 - Description of TCD imputation evaluation experiment 3 (NN retaining outlier Firms)
Imputation of SME Payroll values using 50 executions of the NN algorithm

EXPERIMENTAL QUESTION **Can the missing SME Payroll figures be accurately imputed using the NN imputation process described below?**	
Description of the missing value dataset	• All 61389 SME Firms were loaded into the data matrix from the TCD database, using SQL. • The TCD columns loaded into the matrix were: Sales, Payroll, Depreciation, DirectorPay, NetWorth, PBT, Employees, Easting, Northing and UKSIC_Category. All columns contained integer values (the UKSIC_Category contained integer representations of alphanumeric codes).
Variable to be imputed and evaluated	• The variable to be imputed and evaluated was Payroll. • Payroll had 38724 missing values - i.e. 63.08% of the 61389 data matrix rows had missing values.
Imputation method used for the experiment	• Imputation was performed using the nearest neighbour algorithm described in section 3.2.3 • The Euclidean distance was used to measure the similarity between Firms (data matrix rows). • All variables except Easting and Northing were transformed to standard *Z* scores prior to imputation - so that each variable would carry equal weight in the Euclidean distance calculations. • The Easting and Northing variables were divided by 100,000 just after they were loaded into the data matrix (see the explanation for this given in section 6.4.2). • The search for each nearest neighbour was carried out within the UKSIC segment to which the recipient Firm (the Firm with a missing value) belonged - i.e. only those Firms in the same UKSIC segment as the recipient Firm were considered as potential donors (as explained in section 6.4.2).
Imputation evaluation method	• 50 executions of the NN imputation algorithm were performed (using the options described above). • No outlier Firms were removed from the matrix. • 4.16% of the known Payroll values were randomly deleted and "put back" for each NN execution, using the Fig.4.5 algorithm. With balanced random deletion across all UKSIC segments.

Table 6.9 - Description of TCD imputation evaluation experiment 4 (NN deleting outlier Firms)
Imputation of SME Payroll values using 50 executions of the NN algorithm

EXPERIMENTAL QUESTION **How would deleting outlier Firms from the data matrix affect NN imputation of the missing SME Payroll figures?**
This experiment was identical to the experiment described in table 6.8, except that all Firms (matrix rows) that contained *any financial variable* with a robust *Z* score of more than ± 4 were deleted from the data matrix. That is, 8251 of the 61389 Firms were deleted from the matrix prior to the first execution of the NN imputation process

6.6 Experimental Results: Estimating Imputation Accuracy

A comprehensive set of experimental results for all 48 experiments is given in Appendix A. This comprises 13 pages of imputation accuracy statistics. Including two overall results tables (one for SME Firm imputation and one for LARGE Firm imputation). And twelve associated pairs of RD histograms and segment analysis tables (one pair for each of the methods shown in table 6.10, below). This section summarises these statistics and draws conclusions.

6.6.1 Estimating the Accuracy of the Imputed Values

The most accurate imputation method found for each of the twelve sets of four experiments described in the preceding section is shown in table 6.10, below. These twelve methods are also highlighted in tables A.1 and A.4 (see Appendix A). The RD histogram and segment analysis table associated with each of these methods is also given in Appendix A. The bullet points following the table summarise the conclusions that were drawn.

Table 6.10 - The twelve most accurate imputation methods found for the TCD financial variables

Financial variable imputed and evaluated	Most accurate imputation found method for variable	Proportion % of missing values imputed
SME Firm Sales	NN with outlier Firms deleted	96.92
SME Firm Payroll	NN with outlier Firms deleted	96.55
SME Firm Depreciation	NN with outlier Firms deleted	95.85
SME Firm DirectorPay	EM with outlier Firms deleted	97.93
SME Firm NetWorth	EM with outlier Firms deleted	99.98
SME Firm PBT	EM with outlier Firms deleted	98.81
LARGE Firm Sales	NN with outlier Firms deleted	90.24
LARGE Firm Payroll	NN with outlier Firms deleted	89.86
LARGE Firm Depreciation	EM with outlier Firms deleted	89.90
LARGE Firm DirectorPay	EM with outlier Firms deleted	69.59
LARGE Firm NetWorth	EM with outlier Firms deleted	99.88
LARGE Firm PBT	EM with outlier Firms deleted	97.82

- Deleting outlier Firms from the data matrix produced more accurately imputed values in 22 out of 24 cases (see tables A.1 and A.4). The disadvantage was that the missing values in the deleted outlier Firms could not be imputed. However, the proportion of imputed values lost because of this was generally quite small, as table 6.10 shows.

- A single anomalous result occurred for the EM algorithm, in that 26.39% of the LARGE Firm DirectorPay imputed values were discarded because they were wrongly imputed as negative numbers. For NN imputation a very small proportion (at most 0.14%) of the missing values were not imputed for some Firms, because the UKSIC segments these Firms belonged to had 100% missing values.

- The RD histograms for the twelve most accurate imputation methods (see sections A.2 and A.5) show that at least 5.22% (and in most cases far more) of the deleted values were "put back" *very inaccurately* for every experiment. That is, the patterns within the known values were not strong enough to support the EM and NN imputation methods. Therefore, the feasibility of using these methods to impute the missing values is questionable. See sections 1.3.2 and 5.1.4 for detailed discussions of this idea.

- The Sales, Payroll and Depreciation variables were imputed *far more accurately* than the other three financial variables for both the SME and LARGE Firm categories. The NN algorithm produced the most accurate imputed values for these variables for five out of six of the results shown in table 6.10, above.

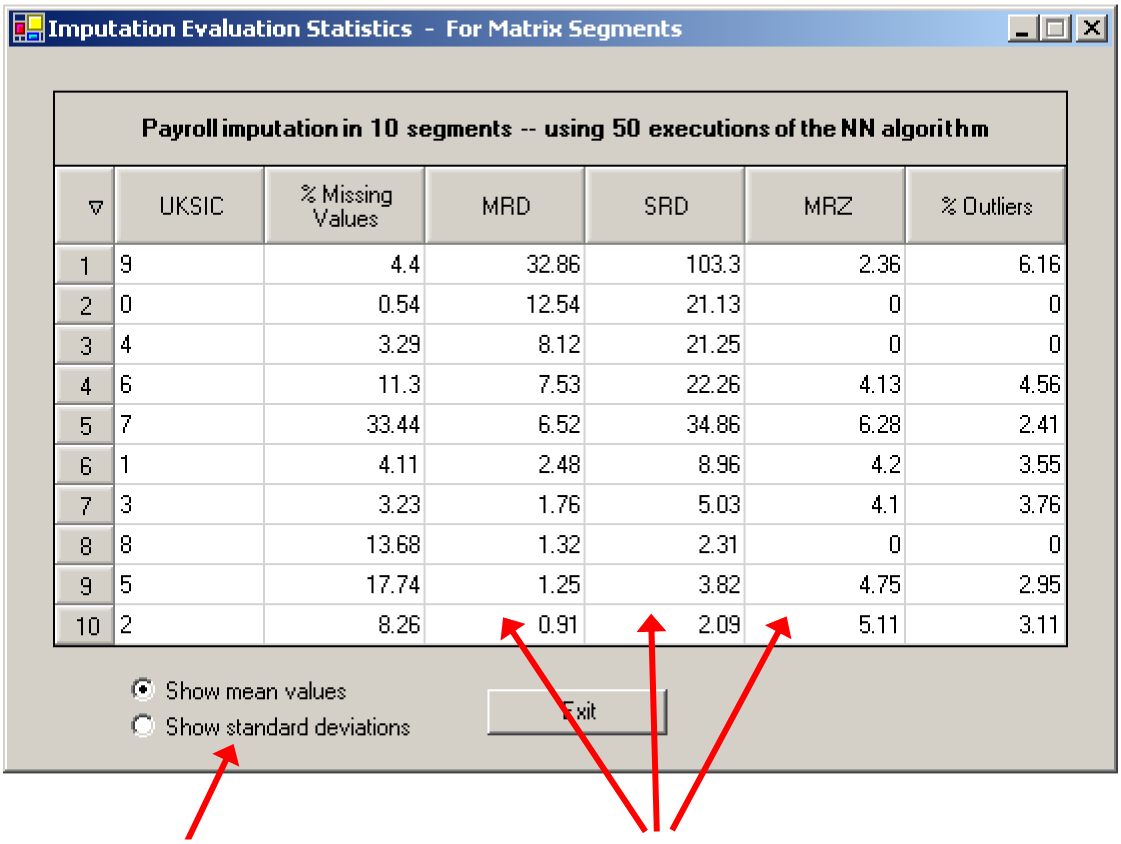
- The accuracy of the imputed DirectorPay, NetWorth and PBT values was *extremely poor*, with huge MRD values being produced across the board. That is, the patterns within the known values for these variables were so weak that it is hard to see how the missing values could be accurately imputed using any imputation method.

- The statistics given in tables A.1 and A.4 show that the changes in the mean and standard deviation (STD) caused by the imputation process were generally quite large. That is, the mean and STD were generally *reduced* by about 20% to 40%, with a few notable exceptions. The NN algorithm caused smaller changes than the EM algorithm in almost every case, with the differences being quite marked for some experiments.

6.6.2 Estimating Imputation Accuracy in Data Segments

The proposed imputation evaluation method allows the accuracy of the imputed values in different data segments to be estimated separately (see sections 4.2.2 and 4.2.3 for full details). This allows the imputation accuracy statistics produced for each segment to be analysed and compared. The procedure used to calculate these statistics is given in Fig. 4.5. The execution of this procedure proved to be quite useful for the TCD dataset, for the reasons given below.

The screenshot below shows how the segment analysis functionality has been implemented within the software. The imputation accuracy statistics shown in the grid were aggregated over 50 executions of the NN algorithm when imputing the missing LARGE Firm Payroll values. The statistics shown in the screenshot are also given in table A.4.2 in section A.6. The segment analysis results tables for the other eleven most accurate methods (see table 6.10, above) are given in sections A.3 and A.6.



Imputation Evaluation Statistics - For Matrix Segments

Payroll imputation in 10 segments -- using 50 executions of the NN algorithm

▽	UKSIC	% Missing Values	MRD	SRD	MRZ	% Outliers
1	9	4.4	32.86	103.3	2.36	6.16
2	0	0.54	12.54	21.13	0	0
3	4	3.29	8.12	21.25	0	0
4	6	11.3	7.53	22.26	4.13	4.56
5	7	33.44	6.52	34.86	6.28	2.41
6	1	4.11	2.48	8.96	4.2	3.55
7	3	3.23	1.76	5.03	4.1	3.76
8	8	13.68	1.32	2.31	0	0
9	5	17.74	1.25	3.82	4.75	2.95
10	2	8.26	0.91	2.09	5.11	3.11

● Show mean values
○ Show standard deviations

These options toggle the grid column values between equations (4.8) and (4.9)

The values shown in these columns are given by equation (4.8) - see section 4.2.3

Fig 6.4 - Implementation of the data segment analysis graphical user interface

The leftmost column shows the UKSIC segment being evaluated - with segmentation at UKSIC level one (see table 6.5, above). Column two shows the proportion of the *total number* of missing values (across the entire dataset) in each segment - i.e. the values given in column two sum to 100%. The four rightmost columns show the imputation accuracy statistics produced for each segment (using equations (4.8) and (4.9), given in chapter four). The grid rows are sorted by MRD descending - i.e. the segment with the *least accurate* imputed values is shown in the top row of the grid.

The screenshot shows that segment 9 (which contains all Firms in the "*Other Community, Social and Personal Service Activities*" UKSIC categories) has the least accurate imputed values. This type of information would be useful to TBR when they were compiling reports describing the Firms in one or more UKSIC categories. For example, it would be known that the imputed values in some UKSIC categories were more accurate than in others, and therefore TBR might decide *not to impute the missing values* in the least accurate categories under any circumstances. ***This approach is unique to the imputation evaluation method described in this dissertation and it is one of the strongest features of that method.***

The proposed method also allows the accuracy of the imputed values in different *missingness patterns* to be estimated separately. For example, the tables given in sections A.3 and A.6 show that the missingness patterns containing the smallest number of known values imputed with the least accuracy in most cases, as one might expect. In particular, the missing values within the pattern with no known financial values were imputed *very inaccurately* across the board. Unfortunately, this pattern contained most of the missing values in the TCD dataset, so discarding the imputed values in this pattern would mean that very few missing values would be imputed for any TCD variable.

However, the histogram given in section 6.2.2 shows that the missingness patterns within the TCD dataset were *extremely unbalanced*. Consequently, the advantages of comparing imputation accuracy across missingness patterns were reduced for the TCD. Nevertheless, the missingness pattern comparison process has significant benefits for the evaluation of regression based imputation methods (such as the EM algorithm), because it allows users to see which missingness patterns produce the best and worst regression equations for any particular dataset.

6.6.3 TCD Imputation Conclusions

The feasibility of imputing the missing TCD financial values is questionable, because the accuracy of the imputed values has been shown to be very poor across the board. The reasons for this are summarised below.

- The large proportions of missing financial values (see section 6.2.1).
- The unbalanced financial variable missingness patterns (see section 6.2.2).
- Some UKSIC categories have much larger proportions of missing data than others.
- Some of the Firms in the TCD have been placed in the wrong UKSIC categories.
- The patterns within the known financial values are generally very weak.
- The financial variables all have small proportions of extreme outlier values.

These problems are embedded within the data itself, and so it is hard to see how they could be solved using any imputation method. However, it is important to emphasise that the final decision regarding the feasibility of imputing the missing financial values can only be taken by the staff at TBR. Consequently, it is argued that there is no substitute for human judgment when considering these matters, and that the proposed imputation evaluation method can only facilitate the decision making process by automating the calculation and display of various imputation accuracy statistics.

6.7 Summary

This chapter has described how the proposed imputation evaluation method was used to address the collaborating company's (TBR's) missing data problem. That is, to discover whether imputation of the missing values in TBR's database (the TCD) was feasible, given the overall poor quality of the data.

The variables in the TCD dataset have been described and a detailed description of TBR's missing data problem has been given. An explanation of how the problems caused by the extreme outlier values in the TCD dataset were addressed has been provided. A description of how the EM and NN algorithms (described in chapters two and three) were customised to suit the TCD dataset has been given. A description of the experiments that were performed in order to find the most accurate methods of imputing the missing TCD values has been provided. And finally, the experimental results have been summarised and conclusions have been drawn.

The following chapter summarises the thesis, draws conclusions and describes how the work described in chapters one to six could be continued.

Chapter Seven

Conclusions and Further Work

7. Conclusions and Further Work

This thesis has described a novel imputation evaluation method and has shown how this method can be used to estimate the accuracy of the imputed values generated by any imputation technique. The work was funded by the EPSRC under the CASE scheme and the resulting collaboration with the partner company led to the formulation of the project objectives, which are described in section 1.1.2. ***These objectives were achieved and the contributions to knowledge made by this work were;***

1. The new imputation evaluation method described in chapter four. The equations and procedures described in this chapter are novel and the method used to compare the accuracy of imputed values in different data segments is also original.
2. The description and experimental evaluation of a novel general purpose NN imputation algorithm given in chapter three.

These contributions are also described in the two published papers given in Appendix E. The following sections summarise the thesis chapters and discuss the conclusions that were drawn for each chapter.

7.1 Theory and Implementation of Imputation Methods

Chapter two summary

Chapter two explained the fundamental concepts underpinning the implementation of MLE based imputation via the EM algorithm and showed how this approach could be used to impute missing values in numeric datasets with multivariate missingness patterns. The history and utility of the EM algorithm was discussed and the type of datasets that can be processed by the EM algorithm were described. A description of how the author has implemented the EM algorithm as a software application was given.

Chapter two conclusions

Techniques for decreasing the execution time of imputation algorithms have received very little attention in the literature. The experiments that evaluate the performance of the author's implementation of the EM algorithm address this problem and make some contribution to the theory of maximum likelihood based imputation. The key findings were as follows;

- It is essential to employ the fastest possible method to generate the regression equations used to impute the missing values in each missingness pattern. This can be achieved by using the SWEEP operator, as described in chapter two and in Appendix B.

- EM execution time can be substantially decreased by sorting the data matrix into missingness pattern order. It is essential this sorting process is performed using an algorithm which requires no more than $(n \log n)$ data matrix row comparisons,

- The fastest possible method of processing the EM sufficient statistics matrix should be employed. This can be achieved by creating the initial version of this matrix only once, then storing and reusing it repeatedly, using the approach described in Appendix B.

Chapter three summary

Chapter three explained the ideas underpinning the development of a novel, general purpose NN imputation algorithm and showed how these ideas could be used to reduce the execution time of the NN imputation process. A description of the experiments that evaluated the performance of the new algorithm was given, and the experimental results were presented, analysed and discussed. The ideas and the experimental results presented in this chapter form part of the contribution to knowledge made by this dissertation.

Chapter three conclusions

As the proportion of missing values in the data matrix increases the execution time of NN imputation algorithms can be decreased by a corresponding proportion. The principle underlying this idea is that the number of row comparisons required to find any particular nearest neighbour can be reduced by utilising the information content within the missingness patterns in the dataset. The algorithm described in chapter three implements this principle and the experimental results show that this approach decreased the execution time of the NN imputation process for both simulated and real datasets. The algorithm’s execution time was found to steadily decrease as the proportion of missing values in the dataset was increased.

7.2 The Proposed Imputation Evaluation Method

Chapter four summary

Chapter four described the proposed imputation evaluation method and showed how this method could be used to estimate the accuracy of the imputed values generated by any imputation technique. A functional overview of the method was given and the equations and procedures which form the basis of the method were described. The proposed method was compared with the most similar methods found within the literature and it was shown that the method builds on the ideas underpinning these methods, but differs from them in several important respects. This ideas presented in this chapter form the principal contribution to knowledge made by this dissertation.

Chapter four conclusions

The general idea of evaluating imputation methods by measuring how accurately a set of deleted values have been “put back” has been frequently employed by other researchers. However, these approaches differ from the proposed method in several important respects. Firstly, they are usually applied to simulated, rather than real, datasets. Secondly, they are not devised to be general purpose imputation evaluation techniques - i.e. they are usually designed for a specific purpose (as an incidental part of a larger project), or to be applied to a specific type of dataset. Thirdly, the idea of measuring imputation accuracy using the statistics generated via a repetitive stochastic algorithm is not used. (see section 4.3.5 for further details). Finally, the idea of comparing the accuracy of the imputed values in different data segments is not used.

The most similar methods found within the literature were uncertainty estimation methods. These methods are similar because they execute the imputation method being evaluated *repeatedly* against the missing value dataset, then go on to use the parameters that describe the resulting set of unique imputed datasets to evaluate the imputation process (see Fig. 4.9 for a diagrammatic representation of this process).

However, the proposed method differs from uncertainty estimation methods by estimating the *accuracy* of the imputed values, rather than estimating the *uncertainty* inherent within those values - i.e. uncertainty estimation methods *do not record deleted values* and then measure how accurately they have been “put back” by the imputation process. In addition, uncertainty estimation methods are not generally used to compare the uncertainty within imputed values in different data segments, although there seems to be no reason why they could not be adapted for this purpose.

Chapter five summary

Chapter five explained how the reliability and validity of the proposed method was experimentally evaluated. A description of the software that implements the method was given, including some screenshots of the graphical user interface that showed how the software can be used in practice. An explanation of how the parameters that control the execution of the method can be chosen was given. Finally, a description of how the method can be used to compare the effectiveness of candidate imputation methods was provided.

Chapter five conclusions

The experimental results given in chapter five show that the method produces reliable and valid estimates of imputation accuracy when it is used to evaluate the imputed values generated by the EM and NN imputation techniques. However, the final decision regarding the feasibility of the imputation process can only be taken by the owner of the missing value dataset, and the proposed method can only facilitate the decision making process by automating the calculation and display of various imputation accuracy statistics.

Chapter six summary

Chapter six described how the method was used to discover whether imputation of the missing values in TBR's database (the TCD) was feasible. The variables in the TCD dataset were described and a detailed description of TBR's missing data problem was given. A description of the experiments that were performed in order to find the most accurate methods of imputing the missing TCD values was given. Finally, the experimental results (given in Appendix A) were summarised and conclusions were drawn.

Chapter six conclusions

The feasibility of imputing the missing values in the TCD dataset is questionable, because the accuracy of the imputed values was shown to be very poor across the board. This occurred because of the extremely poor quality of the dataset. That is, the problems that caused the poor imputation accuracy are embedded within the data itself, so it is hard to see how they could be solved using *any imputation method*. However, it is important to emphasise that the final decision regarding the feasibility of imputing the missing TCD values can only be taken by the staff at TBR.

7.3 Overall Conclusions and Further Work

All imputation methods have the same basic objective. That is, they try to make the best possible use of the information content (the patterns and so on) within the *known* values in a particular dataset, to generate the best possible estimates for the *missing* values in that dataset. And of course, this is the only possible approach, since the information content within the *known values* is the only thing we have to "go on". It follows that imputation evaluation methods should also make the best possible use of the information content within the *known values*, and the method proposed in this thesis does just this.

The idea of evaluating imputation techniques by measuring how accurately they can "put back" a set of deleted values is a simple and intuitive approach to imputation evaluation, which can be easily understood by the owner of any missing value dataset. And it is difficult to discount the results produced by this method, since it would be very hard to deny the success of any imputation technique which can be shown to have *repeatedly* "put back" a set of randomly deleted values with a high degree of accuracy.

Despite these (seemingly obvious) conclusions, this dissertation describes the first attempt to develop the "delete and put back" approach into a general purpose imputation evaluation method. However, the implementation of the method described here could be developed further, as follows,

- The method could be tested against many different *types* of dataset and conclusions regarding its utility for those *types* of dataset could be drawn. It is important to emphasise the idea of dataset *types* in this respect, because every missing value dataset will be unique in some respects and consequently every missing data problem will also be unique. Therefore, the only way to *properly* evaluate the utility of the proposed method for any *particular* dataset is to execute it against that dataset and then consider the imputation accuracy statistics that are produced.

- Chambers (2001) lists five criteria that can be used to evaluate the performance of any imputation technique (see chapter one). The "delete and put back" method described in this thesis evaluates performance using the first of these criteria (predictive accuracy), but it could be adapted to evaluate performance using the other four. This could be achieved by comparing the properties of the deleted (known) values and the values that are "put back" (the imputed values) using techniques that have *not been applied* in this thesis. For example, the correlations between these two sets of variables could be measured, or their distributions could be compared (using line charts or numerical methods), or any other required numerical or visual technique could be used measure and compare their attributes.

- The proposed method has been used to estimate the accuracy of the imputed values generated by the EM and NN imputation techniques in this thesis. However, the method could be used to estimate the accuracy of the imputed values generated by any other imputation technique, so that the effectiveness of these techniques could be compared. In particular, the method could be adapted to estimate the accuracy of the imputed values generated by multiple imputation techniques, which are generally recognised as being the most effective imputation methods.

- The imputation evaluation statistics produced by the proposed method could be compared with the statistics produced by other imputation evaluation methods. This would be best achieved by implementing several evaluation methods alongside one another in a single integrated software application. In particular, the results produced by the proposed approach could be compared with the results produced by uncertainty estimation methods, which have some similarity with the proposed approach.

REFERENCES

Aguirre, F. and Sun, T., (2003), *SSC Case Study 2002 - Handling Missing Data in the 1994 National Population Health Survey,* SSC Annual Meeting, May 2002, Proceedings of the Survey Methods Section.

Aho, A.V., Hopcroft, J.E. and Ullman, J.D., (1983), *Data Structures and Algorithms*, Addison Wesley, MA.

Allison, P. D., (2001), *Missing Data (Quantitative Applications in the Social Sciences, series no. 07-136),* Sage Publications, California.

Allison, P.D., (2000), *Multiple Imputation for Missing Data: A Cautionary Tale*, Sociological Methods and Research, 28(3), pp. 301-309.

Barnard, J., and Meng, X. L. (1999), *Applications of Multiple Imputation in Medical Studies: From AIDS to NHANES*, Statistical Methods in Medical Research, 8, pp. 17-36.

Beale, E. M. L. and Little, R.J.A., (1975), *Missing Values in Multivariate Analysis*, Journal of the Royal Statistical Society. Series B (Methodological), 37 (1), pp. 129-145.

Beaton, A. E. (1964), *The Use of Special Matrix Operations in Statistical Calculus*, Educational Testing Service Research Bulletin, RB-64-51.

Bello, A.L., (1995), *Imputation Techniques in Regression Analysis: Looking Closely at Their Implementation*, Computational Statistics & Data Analysis, 20, pp. 45-57.

Bernier, J., Haziza, D., Nobrega, K. and Whitridge, P., (2002), *Statistical Society of Canada: Handling Missing Data Case Study*, Available at; http://www.ssc.ca/documents/case_studies/2002/missing_e.html Accessed 17th July 2007.

Berson, A. and Smith, S. J., (1997), *Data Warehousing, Data Mining, and OLAP*, McGraw-Hill, New York, USA.

Binder, D.A., (1996), *On Variance Estimation with Imputed Survey Data: Comment*, Journal of the American Statistical Association, Vol. 91 (434), pp. 510-512.

Binder, D. A. and Sun, W., (1996), *Frequency Valid Multiple Imputation for Surveys with a Complex Design,* Proceedings of the section on survey research methods, American Statistical Association, pp. 281-286.

Box, G.E.P. and Cox, D.R., (1964), *An Analysis of Transformations,* Journal of the Royal Statistical Society, Series B, 26, pp 211-243.

Buck, S. F., (1960), *A Method of Estimation of Missing Values in Multivariate Data Suitable For Use With an Electronic Computer*. Journal of the Royal Statistical Society, Series B (Methodological), 22, (2), pp. 302-306.

Chambers, R., (2001), *Evaluation Criteria for Statistical Editing and Imputation,* National Statistics Methodological Series No. 28, Office for National Statistics, Also available from http://www.cs.york.ac.uk/euredit/ as *EUREDIT Project deliverable D3.3,* Accessed 2nd June 2007.

Chen, J. and Shao, J., (2000), *Nearest neighbour Imputation for Survey Data,* Journal of Official Statistics, 16 (2), pp. 113-131.

Chen, J. and Shao, J., (2001), *Jackknife Variance Estimation for Nearest-Neighbour Imputation,* Journal of the American Statistical Association, Vol. 96, (453), pp. 260-269.

Coleman, C., (2004), *A Fast, High-Precision Implementation of the Univariate One-Parameter Box-Cox Transformation Using the Golden Section Search in SAS/IML®,* NESUG 2004, Baltimore, USA, Proceedings of the Statistics and Pharmacokinetics Section.

Dear, R.E., (1959), *A Principal-Component Missing Data Method for Multiple Regression Models*, Report SP-86, System Development corporation, Santa Monica, CA.

Dempster, A. P., (1969), *Elements of Continuous Multivariate Analysis,* Addison Wesley, MA.

Dempster, A.P., Laird, N.M. and Rubin, D.B., (1977), *Maximum Likelihood from Incomplete Data Via the EM Algorithm,* Journal of the Royal Statistical Society, Series B, 39, pp 1-38.

Dixon, J. K., (1979), *Pattern Recognition With Partly Missing Data,* IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-9 (10), pp.617-621.

Dunham, M. H., (2003), *Data Mining Introductory and Advanced Topics,* Prentice-Hall, New Jersey.

Durrant, G. B., (2005), *Imputation Methods for Handling Item-Nonresponse in the Social Sciences: A Methodological Review*, ESRC National Centre for Research Methods: Methods Review Working Paper. Also available at; http://www.ncrm.ac.uk/publications/index.php Accessed 4th March 2007.

Efron, B., (1994), *Missing Data, Imputation, and the Bootstrap*, Journal of the American Statistical Association, Vol. 89 (426), pp. 463-47.

Eltinge, J.L., (1996), *On Variance Estimation with Imputed Survey Data: Comment*, Journal of the American Statistical Association, Vol. 91 (434), pp. 513-515.

Ester, M., Kriegel, H., Sander, J. and Xu, X., (1996), *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,* In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, USA, pp. 226-231.

Ezzati-Rice, T.M., Johnson, W., Khare, M., Little, R.J.A., Rubin, D.B. and Schafer, J.L., (1995), *A Simulation Study to Evaluate the Performance of Multiple Imputations in NCHS Health Examination Survey,* Proceedings of the Bureau of the Census, Eleventh Annual Research Conference, pp. 257-266.

Fay, R. E., (1991), *A Design Based Perspective on Missing Data Variance,* Proceedings of the 1991 Annual Research Conference, Washington DC, U.S. Bureau of the Census, pp. 429-440.

Fay, R. E., (1992), *When are Inferences from Multiple Imputation Valid?*, Proceedings of the Survey Research Methods Section, American Statistical Association, pp. 227-232.

Fay, R. E., (1996a), *Alternative Paradigms for the Analysis of Imputed Survey Data*, Journal of the American Statistical Association, Vol. 91 (434), pp. 490-498.

Fay, R. E., (1996b), *On Variance Estimation with Imputed Survey Data: Rejoinder*, Journal of the American Statistical Association, Vol. 91 (434), pp. 517-519.

Fay, R. E., (1999), *Theory and Applications of Nearest Neighbor Imputation in Census 2000*, Proceedings of the section on survey research methods, American Statistical Association 1999, pp. 112-121.

Fayyad, U. M., Piatetsky-Shapiro, G. and Smyth, P., (1996), *From Data Mining to Knowledge Discovery: An Overview*, Advances in Knowledge Discovery and Data Mining (Eds., Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. and Uthursamy, R.), pp. 1–30, AAAI Press, Menlo Park, CA.

Feynman, R. P., (2001), *The Pleasure of Finding Things Out: The Best Short Works of Richard P. Feynman*, (Editor; Robbins, J, Foreword; Dyson, F.), Penguin Books (Science), London.

Gilks, W.R., Richardson, S. and Spiegelhalter, D.J., (Eds.) (1995), *Markov Chain Monte Carlo in Practice*, Chapman and Hall, London.

Goodnight, J. H., (1979), *A Tutorial on the SWEEP Operator*, American Statistician 33(3), pp 149-158. Also available from http://support.sas.com/documentation/onlinedoc/trindex.html (accessed 1st June 2007) as SAS Technical Report R-106, *The Sweep Operator: Its Importance in Statistical Computing*

Graham, J. W., Hofer, S. M. and Piccinin, A. M., (1994), *Analysis With Missing Data in Drug Prevention Research*, Advances in Data Analysis for Prevention Intervention Research, NIDA Research Monograph 142 (Eds. Collins, L.M., Seitz, L. A.), pp. 13-63, National Institute on Drug Abuse, Washington, DC.

Hartley, H.O., (1958), *Maximum Likelihood Estimation from Incomplete Data*, Biometrics, 14, pp. 174-194.

Hejlsberg, A., Wiltamuth, S. and Golde, P., (2004), *The C# Programming Language*, Addison-Wesley, Boston.

Horton N. J. and Lipsitz, S. R., (2001), *Multiple Imputation in Practice: Comparison of Software Packages for Regression Models With Missing Variables*, American Statistician 55(3), pp. 244-254.

Huang, X. and Zhu, Q., (2002), *A Pseudo-nearest-neighbor Approach for Missing Data Recovery on Gaussian Random Data Sets*, Pattern Recognition Letters, 23 (13), pp. 1613-1622.

Iglewicz, B. and Hoaglin, D., C., (1993), *How to Detect and Handle Outliers (ASQC Basic References in Quality Control, Vol. 16),* American Society for Quality Control (ASQC/Quality Press), Milwaukee, WI.

Judkins, D.R., (1996), *On Variance Estimation with Imputed Survey Data: Comment*, Journal of the American Statistical Association, Vol. 91 (434), pp. 507-510.

Junninen, H., Niska, H., Tuppurainen, K., Ruuskanen, J., Kolehmainen, M., (2004), *Methods for Imputation of Missing Values in Air Quality Data Sets*, Atmospheric Environment, Vol. 38, pp. 2895-2907.

Kalton, G., (1982), *Compensating for Missing Survey Data.* Ann Arbor, MI: Survey Research Center, University of Michigan.

Kalton, G., (1983), *Introduction to Survey Sampling (Quantitative Applications in the Social Sciences, series no. 07-35)*, Sage Publications, California.

Knuth, D. E., (1997), *The Art of Computer Programming: Vol. 1 Fundamental Algorithms, Third Revised Ed. Edition,* Addison Wesley, MA.

Krzanowski, W.J., (1988). *Missing Value Imputation in Multivariate Data Using the Singular Value Decomposition of a Matrix,* Biometrical Letters 25(1,2), pp. 31-39

Lavori, P., Dawson, R. and Shera, D., (1995), *A Multiple Imputation Strategy for Clinical Trials with Truncation of Patient Data*, Statistics in Medicine (14), pp. 1913-1925.

Lazzeroni, L.C., Schenker, N. and Taylor, J.M.G., (1990), *Robustness of Multiple-imputation Techniques to Model Misspecification,* Proceedings of the Survey Research Methods Section, American Statistical Association 1990, pp. 260-265.

Lee, H., Rancourt, E. and Sarndal, C.E, (2002), *Variance Estimation from Survey Data under Single Imputation*, Chapter 21, in Survey Nonresponse (Eds., Groves, R.M., Dillman, D.A., Eltinge, J.L. and Little, R.J.A), Wiley, New York.

Lee, C.H. and Shin, D.G., (1999), *Using Hellinger Distance in a Nearest neighbour Classifier for Relational Databases*, Knowledge-Based Systems, 12 (7), pp. 363-370.

Little, R.J.A. and Rubin, D.B., (2002), *Statistical Analysis with Missing Data – Second Edition,* Wiley, New York.

Mahalanobis, P.C., (1936), *On the Generalised Distance in Statistics,* Proceedings of the National Institute of Science of India, pp. 49-55.

Manly, B.F.J., (1986), *Multivariate Statistical Methods: A Primer,* Chapman and Hall, New York.

McKendrick, A. G., (1926), *Applications of Mathematics to Medical Problems*, Proceedings of the Edinburgh Mathematical Society, Vol. 44, pp. 98-130.

McLachlan, G. J. and Krishnan, T., (1996), *The EM Algorithm and Extensions,* Wiley Series in Probability and Statistics: Applied Section, Wiley, New York.

Meng X. L., and Pellow S, (1992), *EM: A Bibliographic Review with Missing Articles*, Statistical Computing Section, Proceedings of the American Statistical Assoc., pp. 244-247.

Meng, X. L., (1990), *Towards Complete Results for Some Incomplete-data Problems*, PhD dissertation, Department of Statistics, Harvard University, Ann Arbor, MI.

Meng, X. L., (1994), *On the Rate of Convergence of the ECM Algorithm*. The Annals of Statistics, Vol. 22, (1), pp. 326-339.

Nie, N.H., Hull, C.H., Jenkins, J.G., Steinbrenner K. and Bent, D.H, (1975), *SPSS. Statistical Package for the Social Sciences (Second Edition)*, McGraw-Hill, New York.

Ogwang, T. and Rao, U.L.G., (1997), *A Simple Algorithm for Estimating Box-Cox Models*, The Statistician, 46, pp 399-409.

Orchard, T. and Woodbury, M.A., (1972), *A Missing Information Principle: Theory and Applications*, Proceedings of the Sixth Berkeley Symposium on Mathematics, Statistics, and Probability, Vol. 1, pp. 697-715.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P., (1992), *Numerical Recipes in C: The Art of Scientific Computing, Second Edition,* Cambridge University Press, New York.

Pyle, D., (1999), *Data Preparation for Data Mining, (The Morgan Kaufmann Series in Data Management Systems)*, Morgan Kaufmann, San Francisco.

Rancourt, E., Sarndal, C.E. and Lee H., (1994), *Estimation of the Variance in the Presence of Nearest Neighbor Imputation*, Proceedings of the section on survey research methods, American Statistical Association 1994, pp. 883-893.

Rao, J.N.K., (1996a), *On Variance Estimation with Imputed Survey Data*, Journal of the American Statistical Association, Vol. 91 (434), pp. 499-506.

Rao, J.N.K., (1996b), *On Variance Estimation with Imputed Survey Data: Rejoinder*, Journal of the American Statistical Association, Vol. 91 (434), pp. 519-520.

Rousseeuw, P.J. and Leroy, A.M., (1987), *Robust Regression and Outlier Detection*, John Wiley & Sons, New York.

Rubin, D.B., (1976), *Inference and Missing Data,* Biometrika, 63, pp. 581-592.

Rubin, D.B., (1978), *Multiple Imputations in Sample Surveys - A Phenomenological Bayesian Approach to Nonresponse*, Proceedings of the Survey Research Methods Section, American Statistical Association 1978, pp. 20-34.

Rubin, D.B., (1987), *Multiple Imputation for Nonresponse in Surveys*, Wiley, New York.

Rubin, D.B., (1996a), *Multiple Imputation After 18+ year*s, Journal of the American Statistical Association, Vol. 91 (434), pp. 473-489.

Rubin, D.B., (1996b), *On Variance Estimation with Imputed Survey Data: Rejoinder*, Journal of the American Statistical Association, Vol. 91 (434), pp. 515-517.

Rubin, D.B. and Schenker, N., (1986), *Multiple Imputation for Interval Estimation From Simple Random Samples With Ignorable Nonresponse*, Journal of the American Statistical Association, 81, pp. 366-374.

RSC Analytical Methods Committee, (2001), *Robust Statistics: A Method of Coping with Outliers,* Royal Society of Chemistry (RSC, London, UK), AMC Technical Brief No. 6., Apr 2001, Available at; http://www.rsc.org/images/brief6_tcm18-25948.pdf Accessed 14th August 2007.

Ryan, B.F., and Joiner, B.L., (1994), *Minitab Handbook - Third Edition,* Wadsworth, Belmont, CA.

Schafer, J.L., (1997), *Analysis of Incomplete Multivariate Data,* Chapman and Hall, London.

Shao, J., (2002), *Replication Methods for Variance Estimation in Complex Surveys with Imputed Data*, Chapter 20, in Survey Nonresponse (Eds., Groves, R.M., Dillman, D.A., Eltinge, J.L. and Little, R.J.A), Wiley, New York.

Shao, J. and Sitter, R.R., (1996), *Bootstrap for Imputed Survey Data*, Journal of the American Statistical Association, Vol. 91 (435), pp. 1278-1288.

Simonoff, J.S., (1988), *Regression Diagnostics to Detect Nonrandom Missingness in Linear Regression,* Technometrics, Vol. 30 (2), pp 205-214.

Solomon, N., Oatley, G. and McGarry, K, (2007a), *A Dynamic Method for the Evaluation and Comparison of Imputation Techniques*, In: Proceedings of the World Congress on Engineering 2007 (International Conference of Computational Statistics and Data Engineering), ISBN: 978-988-98671-2-6, pp. 974-982, Newswood Limited, International Association of Engineers, Hong Kong.

Solomon, N., Oatley, G. and McGarry, K, (2007b), *A Fast Multivariate Nearest Neighbour Imputation Algorithm,* In: Proceedings of the World Congress on Engineering 2007 (International Conference of Computational Statistics and Data Engineering), ISBN: 978-988-98671-2-6, pp. 940-947, Newswood Limited, International Association of Engineers, Hong Kong.

Stage, A.R. and Crookston, N.L., (2002), *Measuring Similarity in Nearest-neighbor Imputation: Some New Alternatives*, In: Proceedings of the Symposium on Statistics and Information Technology in Forestry, Blacksburg, Virginia, pp. 91-96

Starick, R. and Watson, N., (2006), *Evaluation of Alternative Income Imputation Methods: the HILDA Experience*, Methodology of Longitudinal Surveys International Conference, University of Essex, Colchester, UK.

Tabachnick, B.G, and Fidell, L.S., (2000), *Using Multivariate Statistics,* Allyn & Bacon, MA, USA.

Tanner, M. A. and Wong, W. H., (1987), *The Calculation of Posterior Distributions by Data Augmentation*, Journal of the American Statistical Association, Vol. 82 (398), pp. 528-540.

Tanner, M. A., (2005), *Tools for Statistical Inference: Methods for the Exploration of Posterior Distributions and Likelihood Functions Third Edition (Springer Series in Statistics)*, Springer-Verlag, New York.

Toutenburg, H. and Fieger, A., (2000), *Using Diagnostic Measures to Detect Non-MCAR Processes in Linear Regression Models With Missing Covariates*, Collaborative Research Center 386, Discussion Paper 204, Also available from Electronic Publications of the University of Munich at; http://epub.ub.uni-muenchen.de/archive/00001594/ Accessed 16th June 2007.

Tseng, S., Wang, K. and Lee, C., (2003), *A Pre-processing Method to Deal With Missing Values by Integrating Clustering and Regression Techniques*, Applied Artificial Intelligence, 17 (5/6), pp. 535-544.

Vieira, R., (2003), *Professional SQL Server 2000 Programming (Programmer to Programmer Series)*, Wiley, Indianapolis.

Wang, X., and Hamilton, H.J., (2003), *DBRS: A Density-Based Spatial Clustering Method with Random Sampling*, Tech. Report CS-2003-13, University of Regina, Canada, ISBN 0-7731-0465-8.e

Wasito, I. and Mirkin, B., (2005), *Nearest Neighbour Approach in the Least Squares Data Imputation Algorithms*, Information Sciences, 169 (1), pp. 1-25.

Wasito, I. and Mirkin, B., (2006), *Nearest Neighbours in Least-Squares Data Imputation Algorithms With Different Missing Patterns*, Computational Statistics & Data Analysis, 50 (4), pp. 926-949.

APPENDICES

A TCD Imputation Experimental Results

Tables of experimental results for the case study given in chapter six.

B Complete EM Algorithm Pseudo-code

Gives a complete pseudo-code version of the EM imputation algorithm.

C Software and Hardware Platform Used

Describes the computer platform used to do all of the work described in this thesis.

D Notation and Terminology Used in This Thesis

Describes the mathematical notation and the associated nomenclature used in this thesis.

E Thesis Publications

Details of the two published papers resulting from the work described in this thesis.

A TCD Imputation Experimental Results

This appendix presents the imputation accuracy statistics that were generated during the experimental imputation of the financial variables in the collaborating company's database. The contents are referenced extensively from within chapter six. The sub-sections on the following pages contain two sets of associated tables and histograms, as follows;

SME Firm financial variable imputation experiments

A.1 - Table showing imputation accuracy statistics for all 24 imputation experiments.

A.2 - Histograms showing accuracy variance for the 6 most accurate imputation methods.

A.3 - Tables showing segmented accuracy for the 6 most accurate imputation methods.

LARGE Firm financial variable imputation experiments

A.4 - Table showing imputation accuracy statistics for all 24 imputation experiments.

A.5 - Histograms showing accuracy variance for the six most accurate imputation methods.

A.6 - Tables showing segmented accuracy for the six most accurate imputation methods.

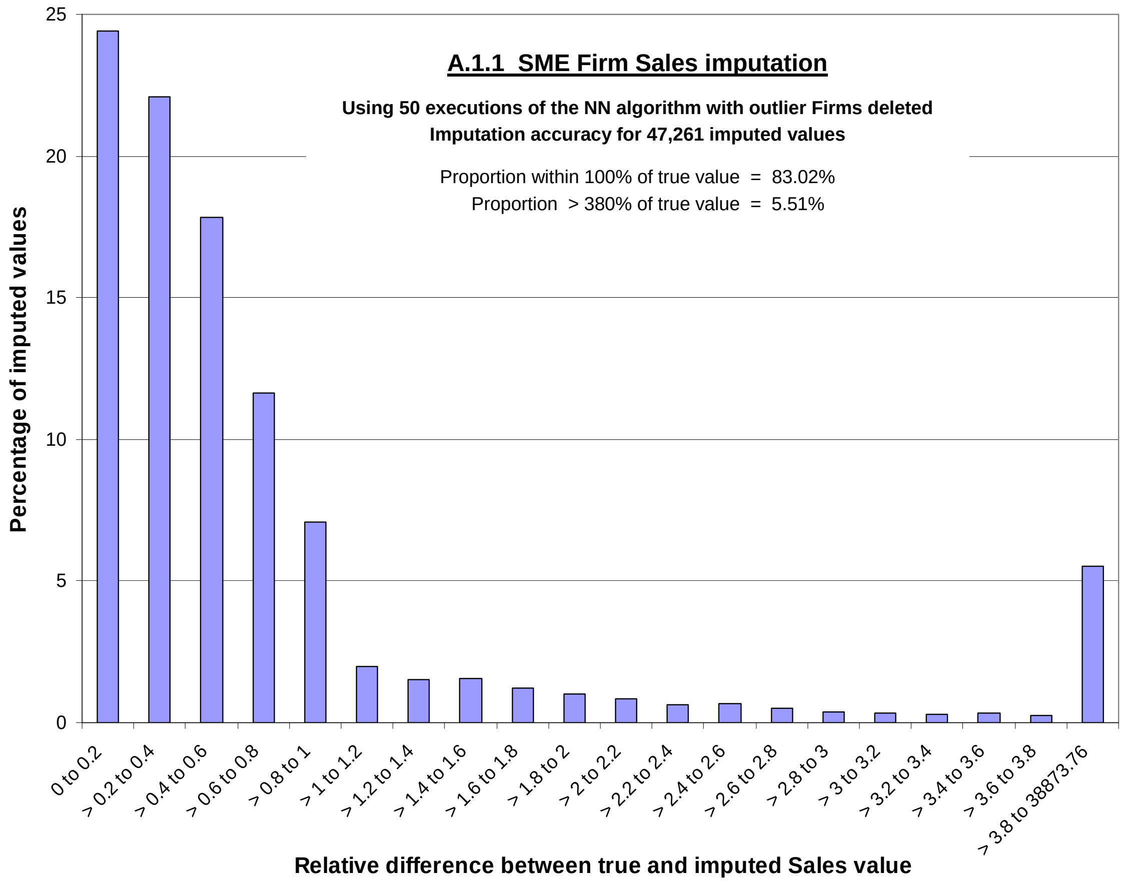
Table A.1

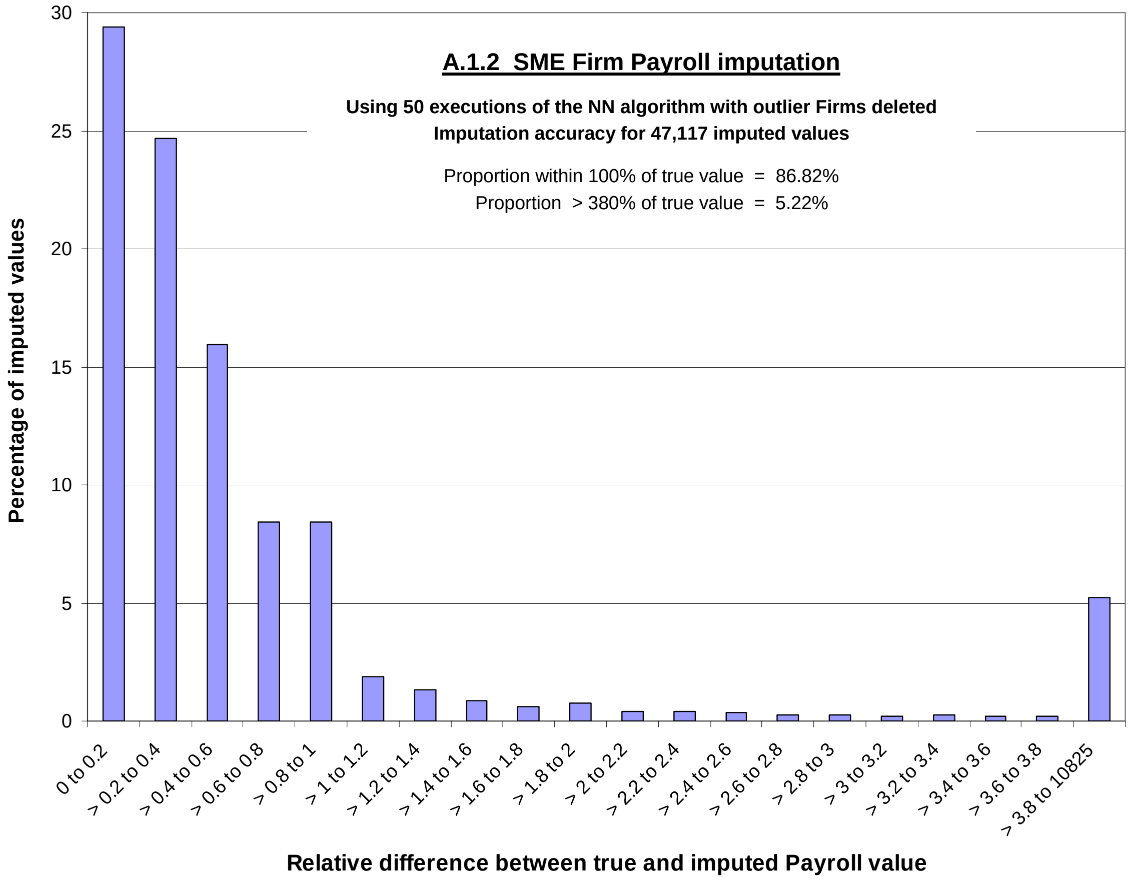
Imputation accuracy statistics for SME Firm financial variables - Using 50 executions of the EM and NN algorithms with and without outlier Firms

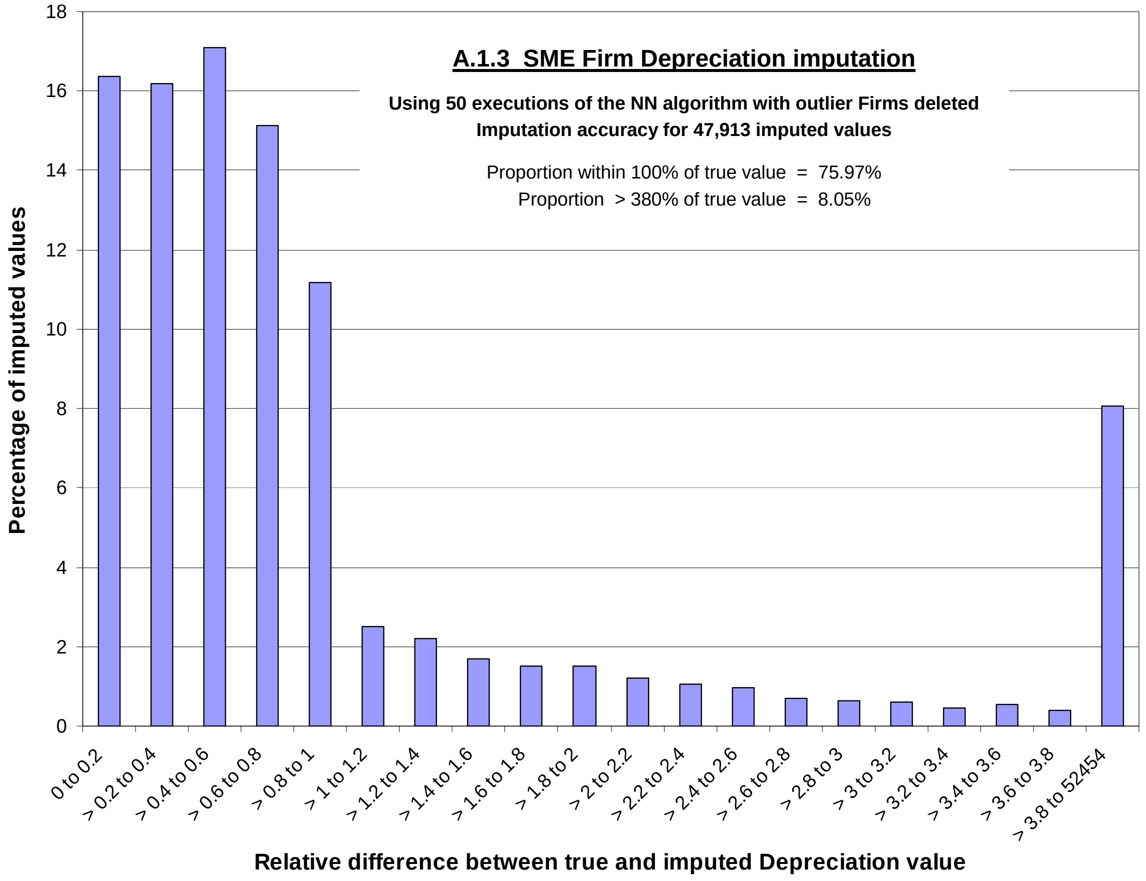
The highlighted rows show the most accurate method for each variable - Variable reference numbers (e.g. A.1.1 Sales) refer to the charts and tables which follow

Firm variable imputed and evaluated	Imputation method used for the experiment (50 executions per row)	MRD		SRD		MRZ			% CHANGE in Mean		% CHANGE in STD	
		$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	% Outliers	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A.1.1 Sales	EM with outlier Firms	54.58	66.54	1,298.61	1,912.29	18.58	8.34	0.32	-42.68	0.19	-38.14	0.93
	EM deleting outlier Firms	48.03	55.40	1,092.29	1,583.56	16.68	8.25	0.41	-27.43	0.11	-46.15	0.04
	NN with outlier Firms	30.92	27.16	833.36	954.06	20.39	9.85	0.26	-29.85	0.89	-28.38	6.08
	NN deleting outlier Firms	**15.38**	**15.97**	**358.84**	**450.20**	**17.10**	**8.72**	**0.45**	**-22.32**	**0.27**	**-11.51**	**0.34**
A.1.2 Payroll	EM with outlier Firms	5.51	3.71	76.96	96.97	12.75	7.01	0.66	-30.42	0.30	-40.22	0.04
	EM deleting outlier Firms	6.38	3.45	83.16	93.29	11.97	5.78	0.65	-20.15	0.08	-35.13	0.05
	NN with outlier Firms	4.66	4.62	85.13	147.84	13.44	6.81	0.52	-16.98	0.55	-8.31	3.96
	NN deleting outlier Firms	**3.52**	**2.68**	**54.31**	**73.02**	**14.88**	**8.08**	**0.51**	**-14.95**	**0.24**	**-1.65**	**0.33**
A.1.3 Depreciation	EM with outlier Firms	33.78	57.39	900.30	1,622.68	17.27	9.86	0.52	-45.97	0.23	-38.55	0.83
	EM deleting outlier Firms	28.23	44.16	718.86	1,229.59	16.19	9.57	0.54	-29.92	0.09	-41.40	0.03
	NN with outlier Firms	37.65	127.41	1,222.04	4,653.49	18.01	10.14	0.38	-23.94	0.77	-17.09	2.28
	NN deleting outlier Firms	**16.24**	**25.31**	**381.64**	**655.77**	**15.20**	**8.76**	**0.53**	**-18.58**	**0.43**	**-7.74**	**0.68**
A.1.4 DirectorPay	EM with outlier Firms	16,211.46	7,032.13	58,958.34	208,060.42	5.80	6.04	0.13	-46.81	0.12	-37.15	0.23
	EM deleting outlier Firms	**11,574.16**	**779.77**	**21,909.73**	**1,196.48**	**4.50**	**0.37**	**1.14**	**-50.62**	**0.13**	**-35.67**	**0.02**
	NN with outlier Firms	44,708.42	5,306.07	198,334.78	68,299.78	6.29	1.35	1.54	-22.38	0.56	-31.36	6.13
	NN deleting outlier Firms	25,852.51	2,863.12	87,034.84	7,965.85	4.80	0.21	2.84	-22.96	0.32	-7.54	0.28
A.1.5 NetWorth	EM with outlier Firms	148,532.24	27,323.87	753,671.86	89,631.70	5.44	0.33	2.71	-15.60	0.25	-23.65	0.00
	EM deleting outlier Firms	**33,729.05**	**5,671.18**	**140,904.07**	**13,559.47**	**4.94**	**0.30**	**3.10**	**-4.70**	**0.03**	**-27.60**	**0.01**
	NN with outlier Firms	235,595.16	213,992.19	3,892,032.86	4,940,644.72	12.98	6.33	0.60	-21.81	1.05	-22.35	1.96
	NN deleting outlier Firms	39,572.53	9,986.93	256,427.87	52,331.23	6.67	0.73	1.79	-1.40	0.33	-0.86	0.19
A.1.6 PBT	EM with outlier Firms	5,392.76	8,595.93	93,017.92	242,404.33	12.38	4.09	0.67	9.87	4.02	-36.44	0.02
	EM deleting outlier Firms	**1,208.20**	**419.79**	**12,039.58**	**2,788.43**	**9.93**	**2.17**	**0.98**	**-21.63**	**0.19**	**-42.29**	**0.02**
	NN with outlier Firms	6,077.73	6,430.00	125,461.32	184,234.01	15.22	5.66	0.39	-0.68	5.39	-34.51	1.43
	NN deleting outlier Firms	2,221.97	1,168.25	35,719.64	17,026.37	13.50	3.71	0.45	-17.64	0.78	-9.12	0.33

A.2 Most Accurate Imputation Methods for SME Firms



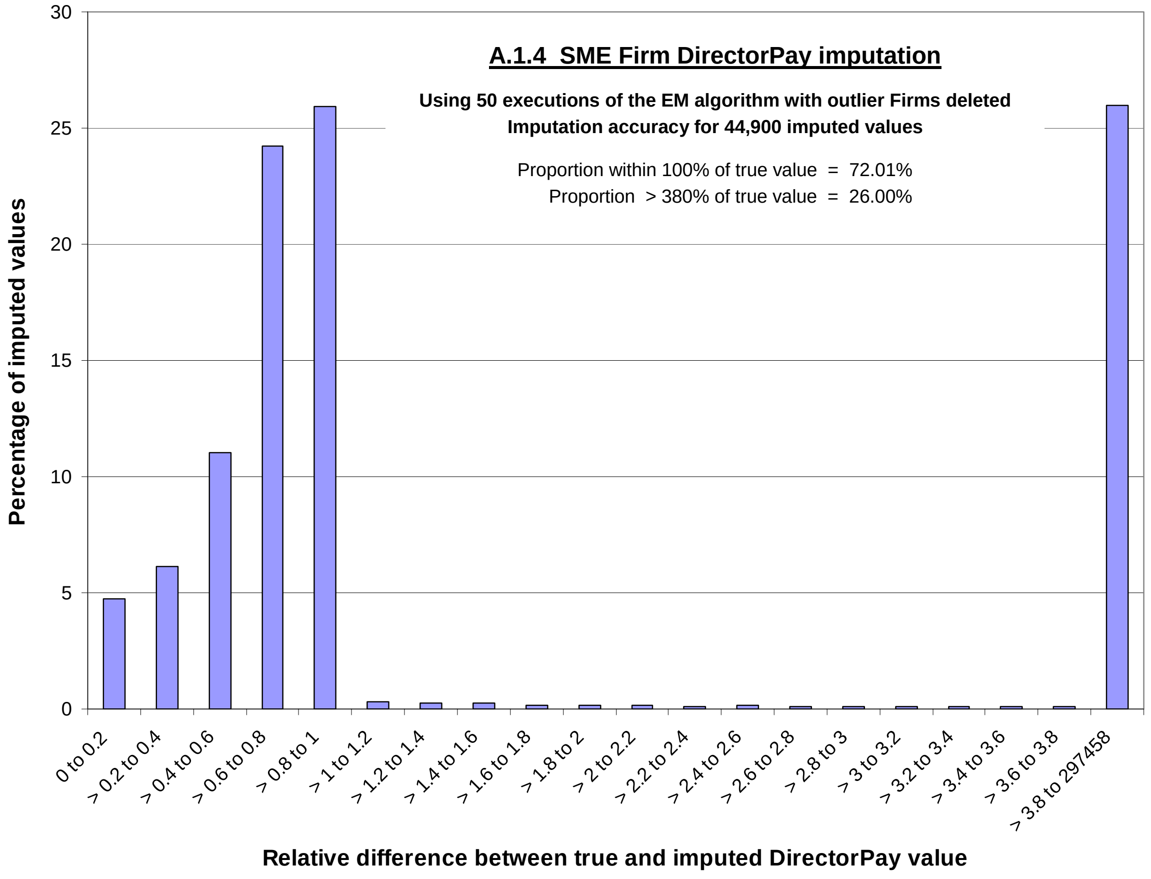




A.1.3 SME Firm Depreciation imputation

Using 50 executions of the NN algorithm with outlier Firms deleted
Imputation accuracy for 47,913 imputed values

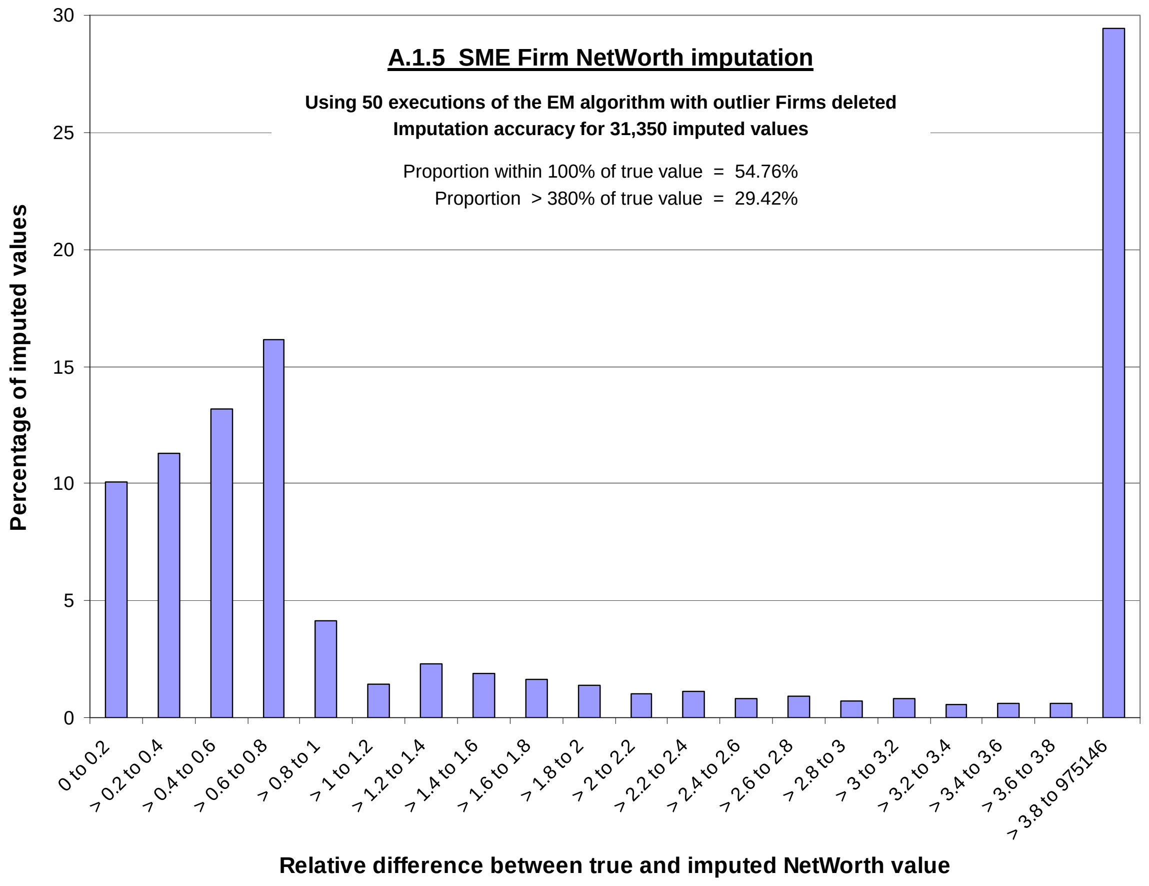
Proportion within 100% of true value = 75.97%
Proportion > 380% of true value = 8.05%

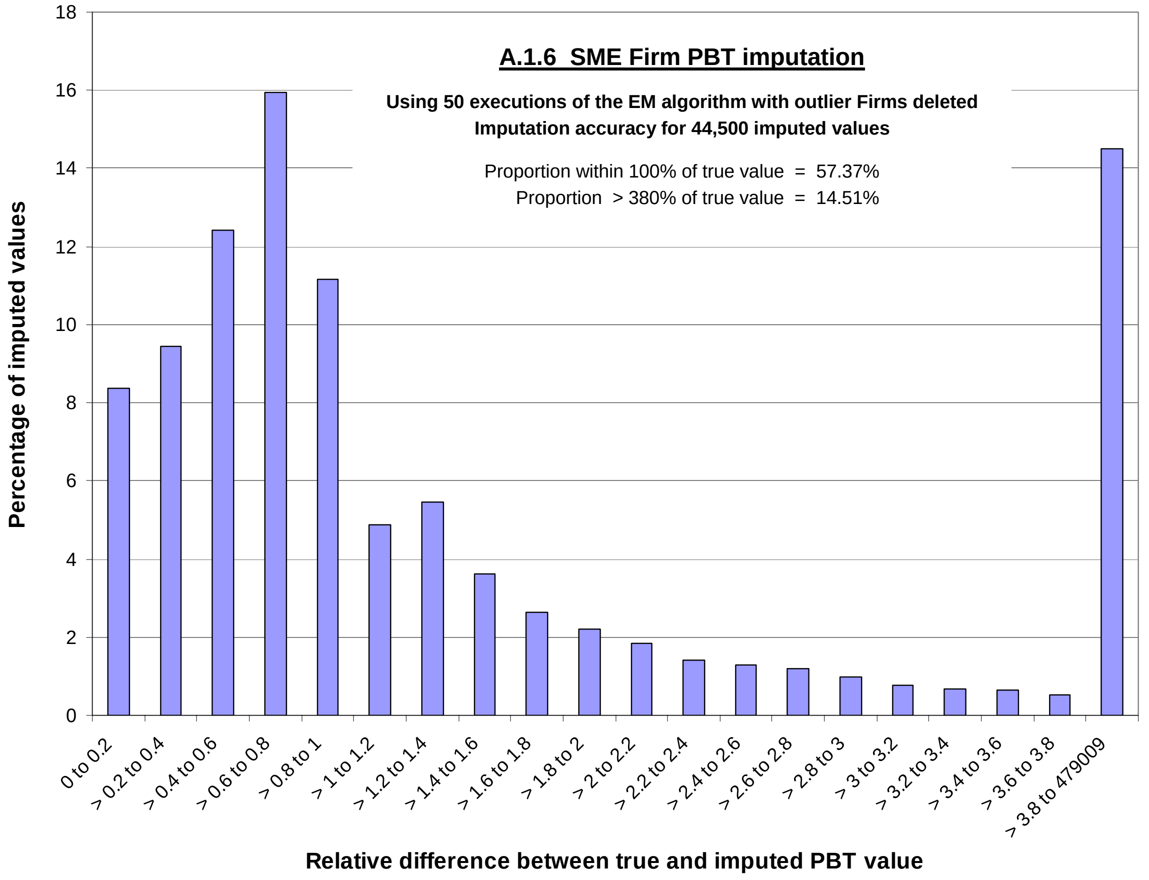


A.1.4 SME Firm DirectorPay imputation

Using 50 executions of the EM algorithm with outlier Firms deleted
Imputation accuracy for 44,900 imputed values

Proportion within 100% of true value = 72.01%
Proportion > 380% of true value = 26.00%


A.1.5 SME Firm NetWorth imputation
Using 50 executions of the EM algorithm with outlier Firms deleted
Imputation accuracy for 31,350 imputed values
Proportion within 100% of true value = 54.76%
Proportion > 380% of true value = 29.42%
Percentage of imputed values
30
25
20
15
10
5
0
0 to 0.2
> 0.2 to 0.4
> 0.4 to 0.6
> 0.6 to 0.8
> 0.8 to 1
> 1 to 1.2
> 1.2 to 1.4
> 1.4 to 1.6
> 1.6 to 1.8
> 1.8 to 2
> 2 to 2.2
> 2.2 to 2.4
> 2.4 to 2.6
> 2.6 to 2.8
> 2.8 to 3
> 3 to 3.2
> 3.2 to 3.4
> 3.4 to 3.6
> 3.6 to 3.8
> 3.8 to 975146
Relative difference between true and imputed NetWorth value


A.1.6 SME Firm PBT imputation
Using 50 executions of the EM algorithm with outlier Firms deleted
Imputation accuracy for 44,500 imputed values
Proportion within 100% of true value = 57.37%
Proportion > 380% of true value = 14.51%
Percentage of imputed values
18
16
14
12
10
8
6
4
2
0
0 to 0.2
> 0.2 to 0.4
> 0.4 to 0.6
> 0.6 to 0.8
> 0.8 to 1
> 1 to 1.2
> 1.2 to 1.4
> 1.4 to 1.6
> 1.6 to 1.8
> 1.8 to 2
> 2 to 2.2
> 2.2 to 2.4
> 2.4 to 2.6
> 2.6 to 2.8
> 2.8 to 3
> 3 to 3.2
> 3.2 to 3.4
> 3.4 to 3.6
> 3.6 to 3.8
> 3.8 to 479009
Relative difference between true and imputed PBT value

A.3 Most Accurate Imputation Method Segments for SME Firms

Table A.1.1 - SME Sales imputation - 50 executions of NN with outlier Firms deleted
Accuracy of imputed values in UKSIC categories - With Firms segmented at UKSIC level 1

UKSIC Category Level 1	% missing Sales values	MRD	SRD	MRZ	% of RD outlier rows
4	5.26	89.91	702.02	7.51	1.74
6	8.65	27.04	232.82	7.18	1.93
7	21.03	19.83	231.60	10.33	1.06
9	5.16	6.40	32.36	5.66	2.74
1	3.40	6.13	31.55	5.67	3.04
5	22.79	5.15	53.46	10.14	1.12
8	21.18	4.36	22.47	6.52	2.27
0	0.76	3.71	9.37	1.80	4.66
2	9.48	2.28	15.99	9.19	1.19
3	3.23	1.59	8.37	6.73	1.92

Table A.1.2 - SME Payroll imputation - 50 executions of NN with outlier Firms deleted
Accuracy of imputed values in UKSIC categories - With Firms segmented at UKSIC level 1

UKSIC Category Level 1	% missing Payroll values	MRD	SRD	MRZ	% of RD outlier rows
9	5.63	13.27	66.64	4.88	3.49
6	9.04	8.36	60.76	7.41	1.84
7	22.63	6.21	56.53	8.24	1.49
4	4.49	3.81	22.29	6.81	2.22
8	21.66	3.09	12.59	5.42	2.78
0	0.70	1.90	3.54	1.91	4.46
2	7.77	1.50	10.81	9.32	1.27
5	22.39	1.26	7.09	8.35	1.56
1	3.01	1.26	4.66	5.92	2.36
3	2.62	0.95	3.70	6.22	2.44

Table A.1.3 - SME Depreciation imputation - 50 executions of NN with outlier Firms deleted
Accuracy of imputed values in UKSIC categories - With Firms segmented at UKSIC level 1

UKSIC Category Level 1	% missing Depreciation values	MRD	SRD	MRZ	% of RD outlier rows
4	4.53	126.37	1042.74	6.97	2.04
2	7.87	15.29	194.02	8.25	1.41
6	9.19	5.53	30.33	6.29	2.35
1	3.00	4.93	26.56	6.54	2.26
7	22.57	4.66	25.92	8.69	1.41
9	5.59	4.45	16.32	4.83	3.20
8	21.60	3.65	16.38	5.42	2.86
5	22.82	2.66	17.42	9.98	1.10
3	2.64	2.50	11.76	7.22	1.86
0	0.69	1.65	3.17	1.85	4.31

Table A.1.4 - SME DirectorPay imputation - 50 executions of EM with outlier Firms deleted
Accuracy of imputed values in missingness patterns - With DirectorPay in pattern position 4

DirectorPay Missingness Pattern	% missing DirectorPay values	MRD	SRD	MRZ	% of RD outlier rows
0000101	28.32	12,662.65	26,139.14	4.13	2.55
0010111	0.01	11,543.89	0.00	0.00	0.00
0000111	0.34	11,337.58	14,771.77	0.00	0.00
0000001	69.93	11,176.79	19,491.34	0.00	0.00
1000111	1.36	10,433.19	25,239.07	1.95	4.46
1010111	0.02	6,593.23	0.00	0.00	0.00
1000101	0.01	5,328.30	0.00	0.00	0.00

Table A.1.5 - SME NetWorth imputation - 50 executions of EM with outlier Firms deleted
Accuracy of imputed values in missingness patterns - With NetWorth in pattern position 5

NetWorth Missingness Pattern	% missing NetWorth values	MRD	SRD	MRZ	% of RD outlier rows
0000001	99.98	33,836.98	141,116.31	4.93	3.10
0011001	0.01	1.47	0.00	0.00	0.00
0111001	0.01	1.46	0.00	0.00	0.00

Table A.1.6 - SME PBT imputation - 50 executions of EM with outlier Firms deleted
Accuracy of imputed values in missingness patterns - With PBT in pattern position 6

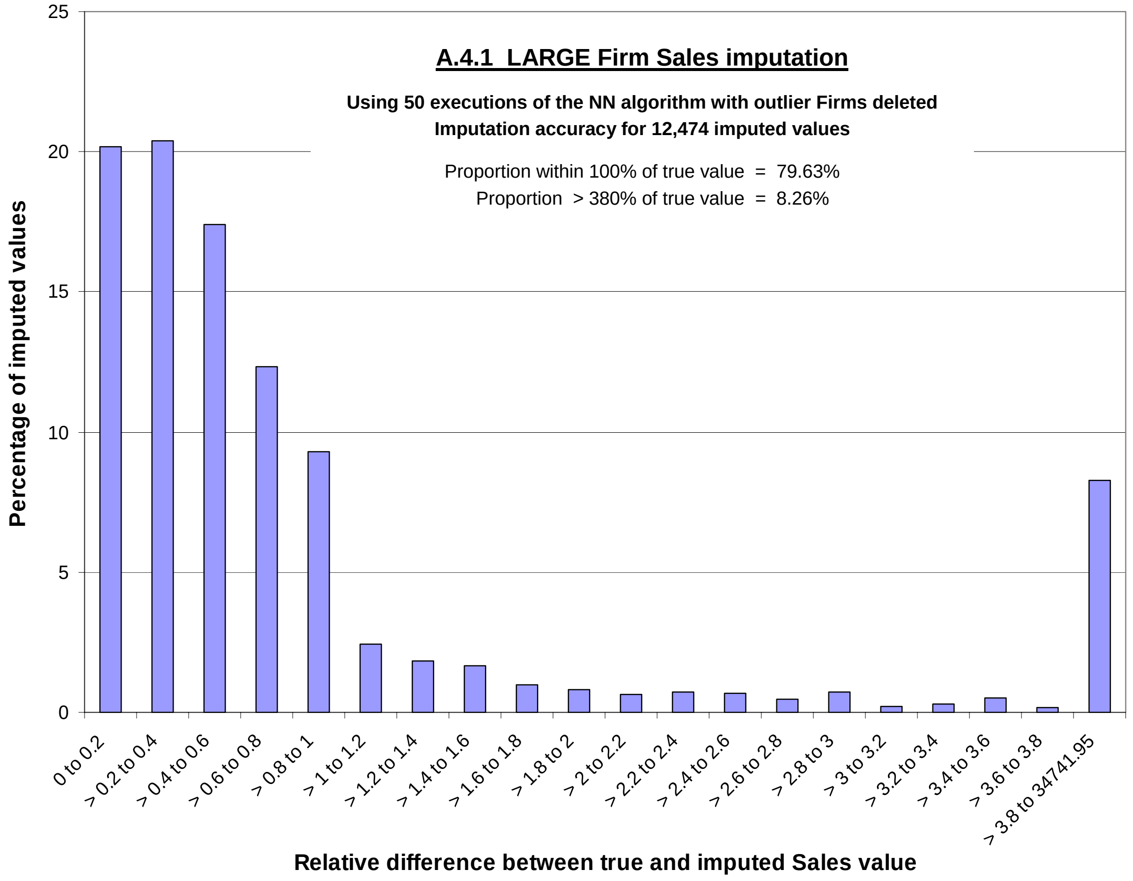
PBT Missingness Pattern	% missing PBT values	MRD	SRD	MRZ	% of RD outlier rows
0000001	70.78	1,377.02	12,660.87	10.23	1.08
0000101	28.66	840.25	8,202.69	11.00	0.96
0001101	0.30	661.97	935.06	0.00	0.00
0101101	0.01	53.58	0.00	0.00	0.00
1101101	0.04	22.10	0.00	0.00	0.00
1111101	0.02	6.20	0.00	0.00	0.00
1000101	0.01	4.07	0.00	0.00	0.00
0111001	0.01	2.27	0.00	0.00	0.00
0011001	0.01	2.15	0.00	0.00	0.00
1001101	0.01	2.01	0.00	0.00	0.00
0111101	0.01	1.96	0.00	0.00	0.00
1011101	0.01	1.67	0.00	0.00	0.00

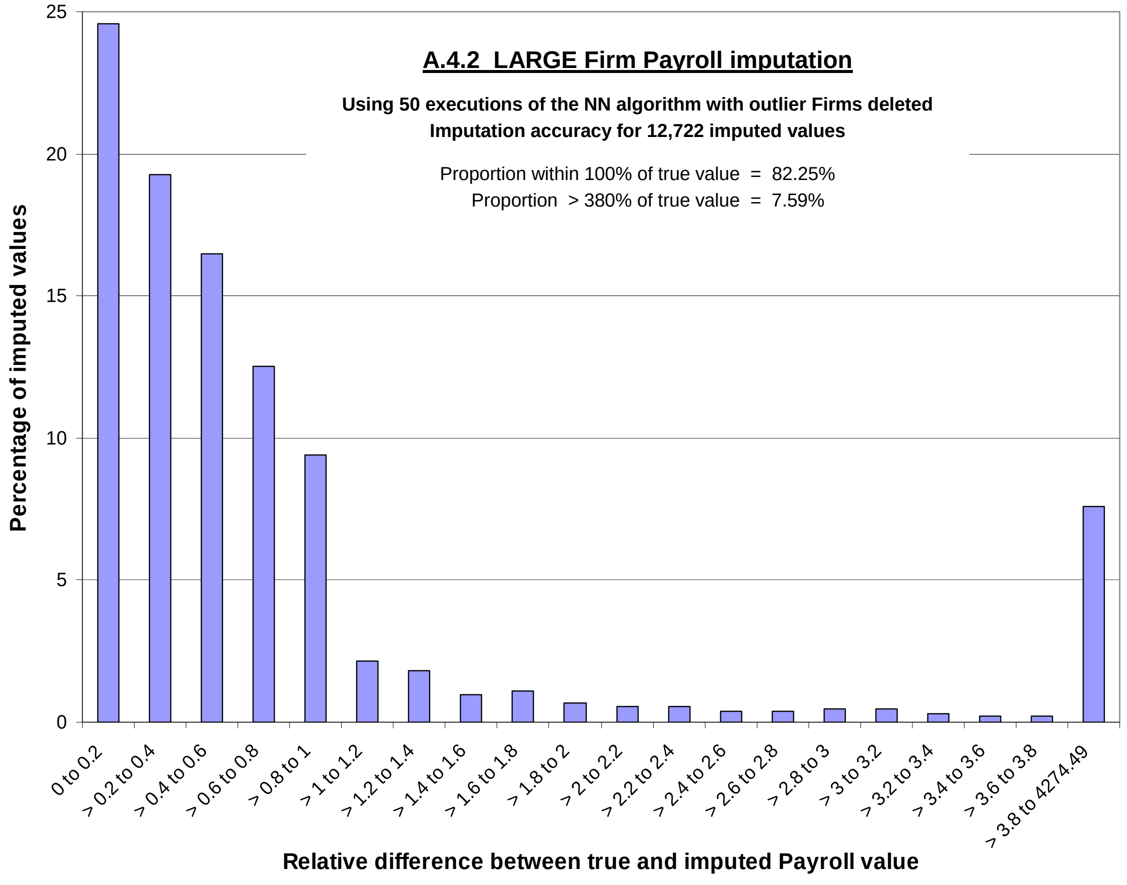
Table A.4

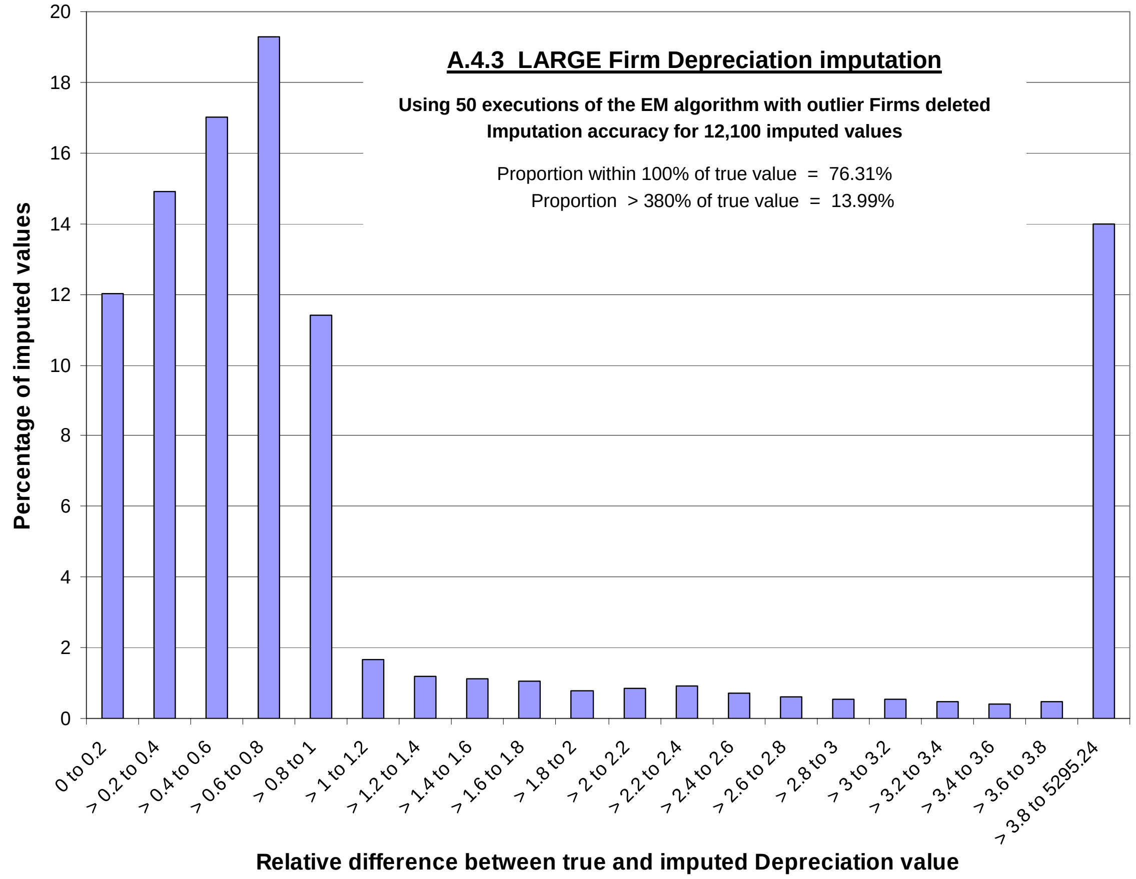
Imputation accuracy statistics for LARGE Firm financial variables - Using 50 executions of the EM and NN algorithms with and without outlier Firms
The highlighted rows show the most accurate method for each variable - Variable reference numbers (e.g. A.4.1 Sales) refer to the charts and tables which follow

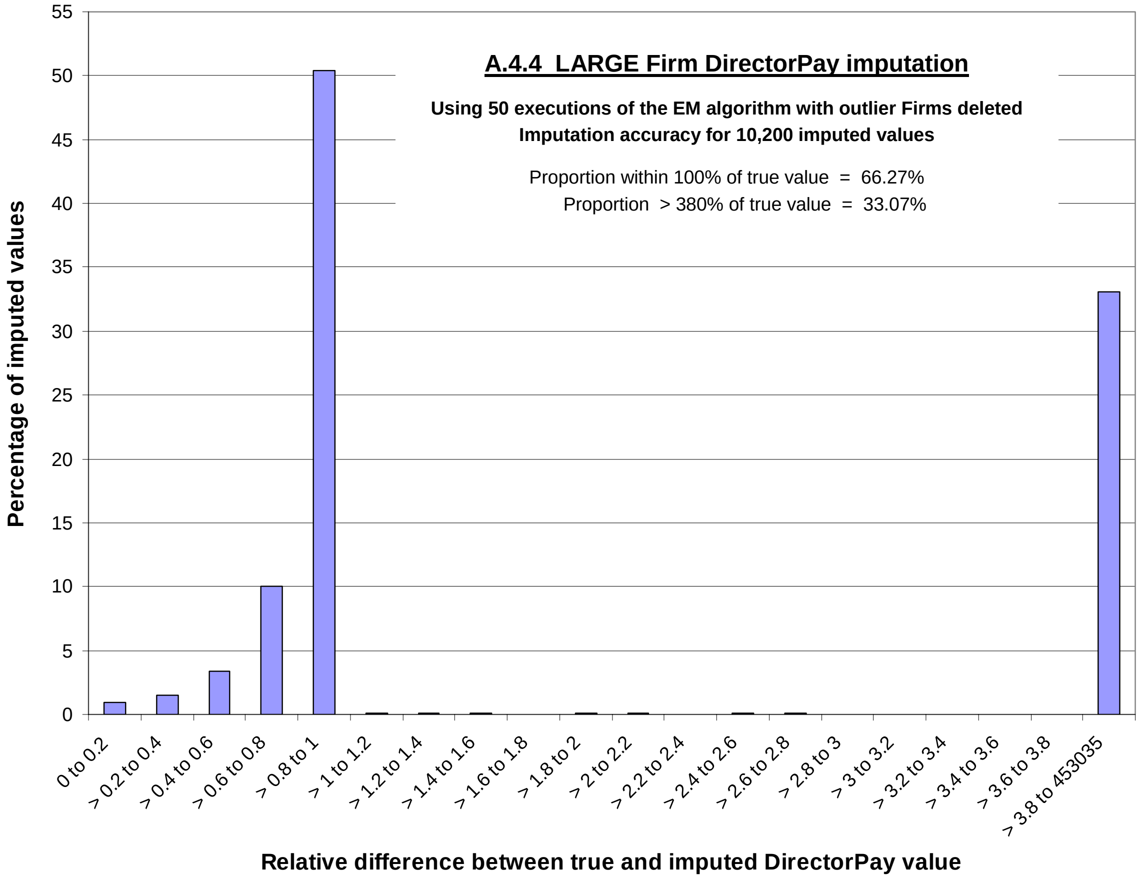
Firm variable imputed and evaluated	Imputation method used for the experiment (50 executions per row)	MRD		SRD		MRZ			% CHANGE in Mean		% CHANGE in STD	
		$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$	% Outliers	$\hat{\mu}$	$\hat{\sigma}$	$\hat{\mu}$	$\hat{\sigma}$
A.4.1 Sales	EM with outlier Firms	698.32	1,999.09	10,391.28	30,862.94	11.64	3.43	0.81	-30.33	2.21	35.37	25.20
	EM deleting outlier Firms	246.96	541.35	3,277.08	7,629.30	11.40	3.34	0.86	-29.16	0.11	-32.70	0.04
	NN with outlier Firms	424.82	1,527.01	7,343.15	27,368.50	13.21	4.32	0.64	-25.42	0.76	-18.03	0.72
	NN deleting outlier Firms	**69.18**	**69.89**	**747.65**	**736.47**	**11.08**	**3.44**	**0.90**	**-21.36**	**0.76**	**-10.48**	**0.54**
A.4.2 Payroll	EM with outlier Firms	18.67	36.17	228.85	558.89	10.61	3.70	0.98	-43.78	0.33	-32.54	0.14
	EM deleting outlier Firms	12.75	22.90	138.81	332.99	9.46	3.59	1.22	-24.38	0.12	-32.53	0.05
	NN with outlier Firms	7.93	7.08	79.67	102.00	10.63	4.07	0.94	-19.83	1.27	-4.09	2.61
	NN deleting outlier Firms	**5.14**	**4.90**	**48.23**	**72.07**	**9.49**	**3.31**	**1.12**	**-16.20**	**0.62**	**-2.84**	**0.55**
A.4.3 Depreciation	EM with outlier Firms	11.16	7.78	93.11	109.20	9.37	3.15	1.15	-50.48	0.29	-33.74	0.12
	EM deleting outlier Firms	**9.23**	**5.06**	**67.48**	**60.92**	**9.44**	**3.50**	**1.21**	**-36.65**	**0.12**	**-34.89**	**0.03**
	NN with outlier Firms	14.98	10.62	167.25	166.00	11.69	3.72	0.76	-34.34	1.60	-29.17	2.68
	NN deleting outlier Firms	20.45	21.10	246.21	293.60	11.22	3.57	0.83	-19.79	1.54	-7.23	0.46
A.4.4 DirectorPay	EM with outlier Firms	18,311.84	3,158.18	38,139.02	27,944.88	4.49	4.55	0.36	-41.12	0.06	-25.62	0.24
	EM deleting outlier Firms	**9,415.28**	**1,552.82**	**19,302.37**	**4,440.20**	**5.47**	**2.21**	**1.50**	**-50.56**	**0.09**	**-23.64**	**0.03**
	NN with outlier Firms	117,380.58	21,445.38	419,769.60	110,591.52	6.26	2.08	1.94	-15.39	1.15	-12.33	5.31
	NN deleting outlier Firms	67,941.77	11,573.54	189,884.37	24,061.41	4.55	0.57	2.86	-21.82	0.58	-8.41	0.31
A.4.5 NetWorth	EM with outlier Firms	2,690,235.09	1,490,351.67	13,362,848.25	5,740,207.98	6.26	1.54	2.39	-11.41	0.10	-14.71	0.00
	EM deleting outlier Firms	**217,467.93**	**58,376.49**	**777,225.19**	**134,158.24**	**4.41**	**0.47**	**3.94**	**-3.37**	**0.03**	**-19.41**	**0.00**
	NN with outlier Firms	1,918,119.56	5,161,620.72	23,868,083.51	76,975,512.57	9.78	2.97	1.08	-5.84	1.31	7.71	2.97
	NN deleting outlier Firms	276,864.10	111,033.05	1,568,598.11	529,779.65	6.56	1.27	2.03	2.06	3.45	0.78	1.63
A.4.6 PBT	EM with outlier Firms	167,159.25	112,553.11	1,514,183.24	834,385.60	10.10	2.94	1.08	-6.81	2.07	-24.88	0.04
	EM deleting outlier Firms	**12,136.88**	**7,499.75**	**95,026.91**	**53,176.02**	**8.44**	**2.53**	**1.51**	**-15.08**	**0.21**	**-32.13**	**0.02**
	NN with outlier Firms	88,376.40	98,270.00	1,135,090.68	1,170,841.87	13.32	3.53	0.60	-25.15	5.47	-11.70	1.09
	NN deleting outlier Firms	21,820.40	19,409.05	242,274.70	202,342.06	11.50	2.84	0.76	-11.67	2.39	-8.10	0.81

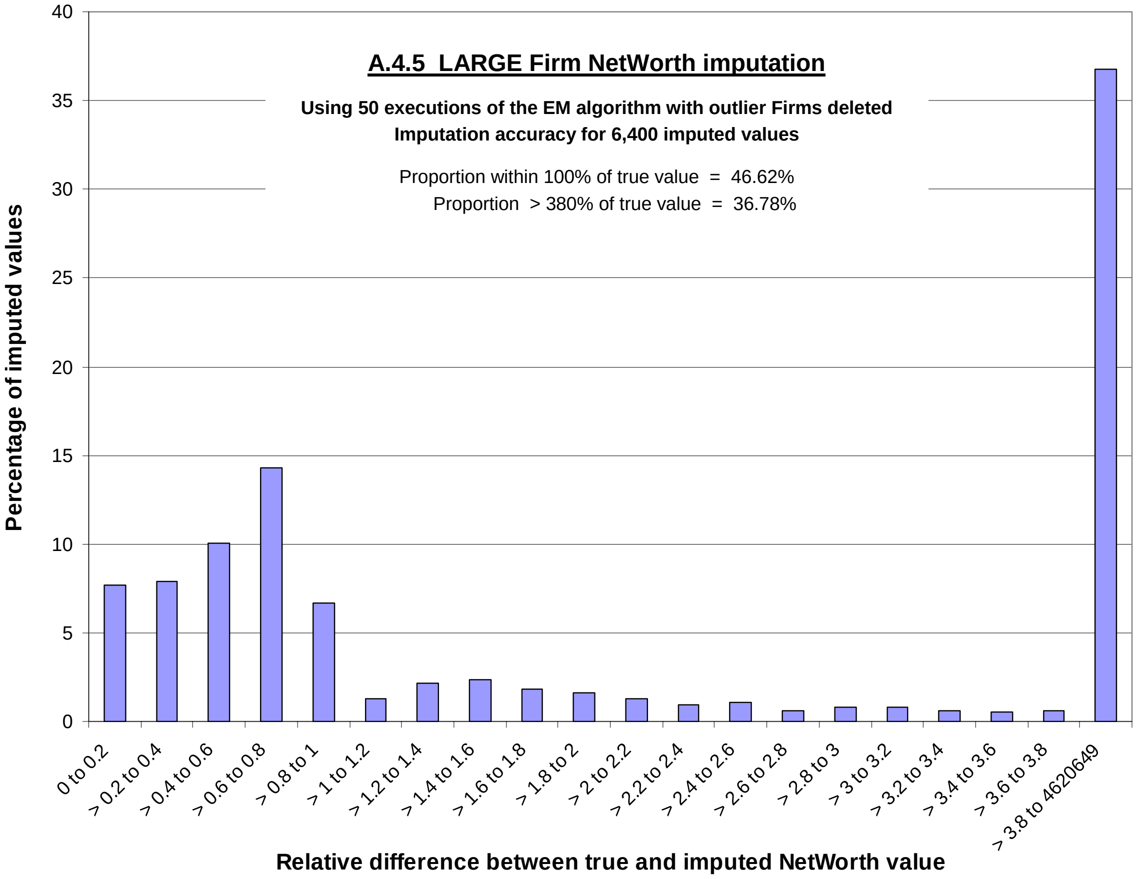
A.5 Most Accurate Imputation Methods for LARGE Firms

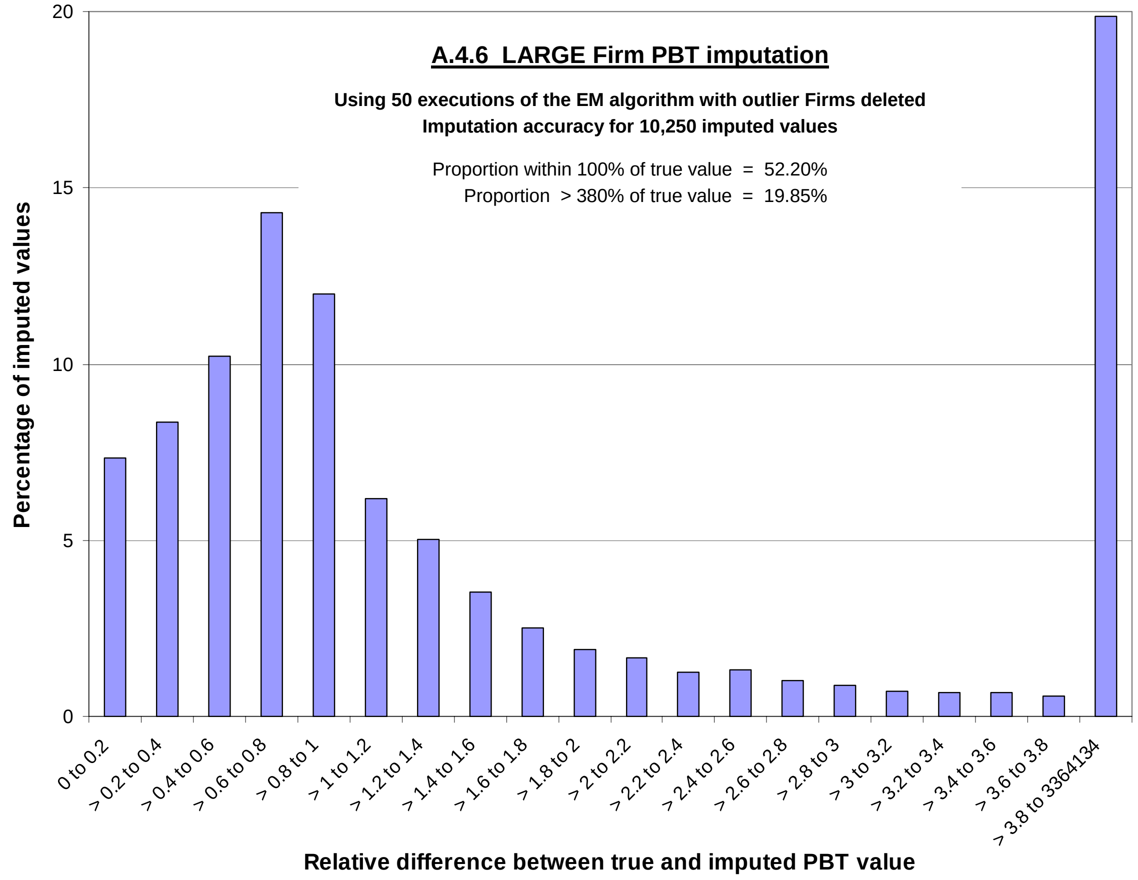





A.4.3 LARGE Firm Depreciation imputation
Using 50 executions of the EM algorithm with outlier Firms deleted
Imputation accuracy for 12,100 imputed values
Proportion within 100% of true value = 76.31%
Proportion > 380% of true value = 13.99%
Percentage of imputed values
0
2
4
6
8
10
12
14
16
18
20
0 to 0.2
> 0.2 to 0.4
> 0.4 to 0.6
> 0.6 to 0.8
> 0.8 to 1
> 1 to 1.2
> 1.2 to 1.4
> 1.4 to 1.6
> 1.6 to 1.8
> 1.8 to 2
> 2 to 2.2
> 2.2 to 2.4
> 2.4 to 2.6
> 2.6 to 2.8
> 2.8 to 3
> 3 to 3.2
> 3.2 to 3.4
> 3.4 to 3.6
> 3.6 to 3.8
> 3.8 to 5295.24
Relative difference between true and imputed Depreciation value


A.4.4 LARGE Firm DirectorPay imputation
Using 50 executions of the EM algorithm with outlier Firms deleted
Imputation accuracy for 10,200 imputed values
Proportion within 100% of true value = 66.27%
Proportion > 380% of true value = 33.07%
Percentage of imputed values
0
5
10
15
20
25
30
35
40
45
50
55
0 to 0.2
> 0.2 to 0.4
> 0.4 to 0.6
> 0.6 to 0.8
> 0.8 to 1
> 1 to 1.2
> 1.2 to 1.4
> 1.4 to 1.6
> 1.6 to 1.8
> 1.8 to 2
> 2 to 2.2
> 2.2 to 2.4
> 2.4 to 2.6
> 2.6 to 2.8
> 2.8 to 3
> 3 to 3.2
> 3.2 to 3.4
> 3.4 to 3.6
> 3.6 to 3.8
> 3.8 to 453035
Relative difference between true and imputed DirectorPay value


A.4.5 LARGE Firm NetWorth imputation
Using 50 executions of the EM algorithm with outlier Firms deleted
Imputation accuracy for 6,400 imputed values
Proportion within 100% of true value = 46.62%
Proportion > 380% of true value = 36.78%
Percentage of imputed values
0
5
10
15
20
25
30
35
40
0 to 0.2
> 0.2 to 0.4
> 0.4 to 0.6
> 0.6 to 0.8
> 0.8 to 1
> 1 to 1.2
> 1.2 to 1.4
> 1.4 to 1.6
> 1.6 to 1.8
> 1.8 to 2
> 2 to 2.2
> 2.2 to 2.4
> 2.4 to 2.6
> 2.6 to 2.8
> 2.8 to 3
> 3 to 3.2
> 3.2 to 3.4
> 3.4 to 3.6
> 3.6 to 3.8
> 3.8 to 4620649
Relative difference between true and imputed NetWorth value


A.4.6 LARGE Firm PBT imputation
Using 50 executions of the EM algorithm with outlier Firms deleted
Imputation accuracy for 10,250 imputed values
Proportion within 100% of true value = 52.20%
Proportion > 380% of true value = 19.85%
Percentage of imputed values
0
5
10
15
20
0 to 0.2
> 0.2 to 0.4
> 0.4 to 0.6
> 0.6 to 0.8
> 0.8 to 1
> 1 to 1.2
> 1.2 to 1.4
> 1.4 to 1.6
> 1.6 to 1.8
> 1.8 to 2
> 2 to 2.2
> 2.2 to 2.4
> 2.4 to 2.6
> 2.6 to 2.8
> 2.8 to 3
> 3 to 3.2
> 3.2 to 3.4
> 3.4 to 3.6
> 3.6 to 3.8
> 3.8 to 3364134
Relative difference between true and imputed PBT value

A.6 Most Accurate Imputation Method Segments for LARGE Firms

Table A.4.1 - LARGE Firm Sales imputation - 50 executions of NN with outlier Firms deleted
Accuracy of imputed values in UKSIC categories - With Firms segmented at UKSIC level 1

UKSIC Category Level 1	% missing Sales values	MRD	SRD	MRZ	% of RD outlier rows
9	4.08	452.57	1,434.67	2.49	6.33
8	13.82	272.73	720.57	0.00	0.00
6	10.58	85.17	405.84	4.30	3.81
1	4.25	84.02	388.09	4.57	3.62
2	8.51	56.38	358.04	5.64	2.45
5	18.23	40.38	228.35	5.47	3.06
7	33.49	18.52	119.76	6.48	2.23
0	0.51	6.68	10.56	0.00	0.00
4	3.18	3.97	9.09	1.32	3.82
3	3.32	1.69	4.99	3.67	3.67

Table A.4.2 - LARGE Firm Payroll imputation - 50 executions of NN with outlier Firms deleted
Accuracy of imputed values in UKSIC categories - With Firms segmented at UKSIC level 1

UKSIC Category Level 1	% missing Payroll values	MRD	SRD	MRZ	% of RD outlier rows
9	4.40	32.86	103.30	2.36	6.16
0	0.54	12.54	21.13	0.00	0.00
4	3.29	8.12	21.25	0.00	0.00
6	11.30	7.53	22.26	4.13	4.56
7	33.44	6.52	34.86	6.28	2.41
1	4.11	2.48	8.96	4.20	3.55
3	3.23	1.76	5.03	4.10	3.76
8	13.68	1.32	2.31	0.00	0.00
5	17.74	1.25	3.82	4.75	2.95
2	8.26	0.91	2.09	5.11	3.11

Table A.4.3 - LARGE Firm Depreciation imputation - 50 executions EM with outlier Firms deleted Accuracy of imputed values in missingness patterns - With Depreciation in pattern position 3

Depreciation Missingness Pattern	% missing Depreciation values	MRD	SRD	MRZ	% of RD outlier rows
0000111	0.52	18.06	16.65	0.00	0.00
0000001	53.80	11.60	75.44	8.45	1.47
0001001	0.02	10.49	0.00	0.00	0.00
0000101	29.36	8.89	44.61	6.46	2.35
0001101	0.61	8.29	7.61	0.00	0.00
0001111	5.53	3.67	8.01	1.87	4.31
1000111	2.51	3.03	4.52	0.00	0.00
0101111	0.60	2.35	1.70	0.00	0.00
0101001	0.02	2.22	0.00	0.00	0.00
1001111	3.58	1.99	3.52	0.00	0.00
0101101	0.01	1.80	0.00	0.00	0.00
1001101	0.12	1.52	0.00	0.00	0.00
1000101	0.02	1.46	0.00	0.00	0.00
1101101	0.09	1.44	0.00	0.00	0.00
1101111	3.17	1.29	1.78	0.00	0.00

Table A.4.4 - LARGE Firm DirectorPay imputation - 50 executions of EM with outlier Firms deleted Accuracy of imputed values in missingness patterns - With DirectorPay in pattern position 4

DirectorPay Missingness Pattern	% missing DirectorPay values	MRD	SRD	MRZ	% of RD outlier rows
1010111	0.09	13,980.01	0.00	0.00	0.00
1000101	0.02	13,005.70	0.00	0.00	0.00
0000111	0.61	11,135.54	9,421.53	0.00	0.00
1000111	2.90	11,085.65	20,296.14	0.00	0.00
0000101	34.02	10,142.37	20,777.49	4.47	2.52
0000001	62.35	8,840.99	15,339.61	2.24	1.52

Table A.4.5 - LARGE Firm NetWorth imputation - 50 executions of EM with outlier Firms deleted Accuracy of imputed values in missingness patterns - With NetWorth in pattern position 5

NetWorth Missingness Pattern	% missing NetWorth values	MRD	SRD	MRZ	% of RD outlier rows
0000001	99.86	222,440.15	785,472.93	4.39	3.98
0001001	0.04	30,868.23	0.00	0.00	0.00
0111001	0.06	5.35	0.00	0.00	0.00
0101001	0.04	2.31	0.00	0.00	0.00

Table A.4.6 - LARGE Firm PBT imputation - 50 executions of EM with outlier Firms deleted

Accuracy of imputed values in missingness patterns - With PBT in pattern position 6

PBT Missingness Pattern	% missing PBT values	MRD	SRD	MRZ	% of RD outlier rows
0000001	63.92	14,295.15	95,696.56	7.70	1.98
0000101	34.88	10,135.42	67,981.28	7.10	2.06
0001101	0.73	856.74	855.99	0.00	0.00
1000101	0.02	40.86	0.00	0.00	0.00
1101101	0.11	29.86	0.00	0.00	0.00
0001001	0.02	16.42	0.00	0.00	0.00
1001101	0.14	5.39	0.00	0.00	0.00
0101001	0.02	4.99	0.00	0.00	0.00
0111101	0.05	3.35	0.00	0.00	0.00
0101101	0.01	2.93	0.00	0.00	0.00
1111101	0.04	2.78	0.00	0.00	0.00
0111001	0.04	2.16	0.00	0.00	0.00

B Complete EM Algorithm Pseudo-code

The following set of pseudo-code listings and explanations give a complete algorithmic description of the EM algorithm for numeric multivariate imputation. The pseudo-code should allow programmers to implement the algorithm as a software application using any programming language. Particular attention is paid to the use of implementation techniques which decrease algorithm execution time, as explained in chapter 2.

Complete pseudo-code listings for all functions called from within the main algorithm follow, including detailed descriptions of the purpose, creation and processing of every data structure needed to support the implementation. Note that the pseudo-code uses operators such as += and ++ etc. These operators are commonly used in object oriented programming languages such as Java and C#. See the standard texts on these languages for more details.

The following notations are used within the algorithm

$M(r)$	Refers to a particular row r in the matrix M.
M_{ij}	Refers to the element at row i and column j in matrix M.
V_j	Refers to element j in vector V, where V_j can be a number or a matrix.
matrix $M = 0$	Removes all rows from matrix M.
vector $V = 0$	Removes all elements from vector V.
for int $j \in V$	Where j is an integer loop variable which takes every value in the vector V.

The following simple functions are called from within the algorithm.

The pseudo-code for these functions is not given here, since the first three should be included as part of the built in functionality in most modern programming languages, and the final two can be easily written by any experienced programmer. See the following sections for full and detailed descriptions of the more complex functions called.

num_rows_in (X)	Returns the number of rows in matrix or vector X.
num_columns_in (X)	Returns the number of columns in matrix or vector X.
remove_matrix_row ($X(r)$)	Removes row r from matrix X. All rows below then move up so that row $r + 1$ becomes row r etc. and the matrix becomes compacted.
missingness_pattern_in ($X(r)$)	Returns a binary vector V representing row r in matrix X where, $V_j = 1$ if X_{rj} is present or $V_j = 0$ if X_{rj} is missing - e.g. $V = \{ 1, 0, 1 \}$ indicates that the values in elements X_{r1} and X_{r3} are present in row r of X, and that the value in element X_{r2} is missing.
num_missing_values_in ($X(r)$)	Returns the number of missing values found in row r of matrix X

The following parameters are passed to the top level EM algorithm, as given below

1. Y is a data matrix with one or more missing values in one or more of its columns. The estimates for the missing values in Y are imputed just before the algorithm terminates.

2. m is the maximum number of iterations that can be performed before the convergence loop terminates. This parameter should be passed as a suitably large number (such as 500 or above).

3. e is the $\theta_{new} - \theta_{old}$ difference used to terminate the algorithm's convergence loop. This parameter should be passed as a suitably small value (such as 0.0001) to ensure close convergence.

```
function matrix EM_algorithm_for_missing_data_imputation ( matrix Y, int m, double e )
    Y = pattern_ordered_matrix (Y )
    int n = 0, int s = 0, int f = 0, matrix R = 0, vector Z = 0
    while ( num_rows_in (Y ) > 0 )
        s ++
        R (s) = missingness_pattern_in (Y (1))
        int i = 0, matrix X = 0
        while ( R (s) = = missingness_pattern_in (Y (1)) )
            i ++
            X (i) = Y (1)
            remove_matrix_row ( Y (1) )
        end while
        if ( num_missing_values_in (R (s)) = = num_columns_in (R) )
            remove_matrix_row (R (s))
            s - -
        else
            if ( num_missing_values_in (R (s)) = = 0 )
                f = s
            end if
            $Z_s$ = X
            n += i
        end if
    end while
    matrix $\theta_{new}$ = initial_parameter_estimate (Z, f )
    matrix $T_{obs}$ = observed_data_sufficient_statistics (Z, R)
    int t = 0
    repeat
        matrix T = $T_{obs}$
        matrix $\theta$ = $\theta_{new}$
        for int s = 1 to num_rows_in (R)
            int a = 0, int b = 0, vector o = 0, vector m = 0
            for int j = 1 to num_columns_in (R)
                if ( $R_{sj}$ = = 1 )
```

```
                    $a{+}{+} :\ o_a = j$
                else
                    $b{+}{+} :\ m_b = j$
                end if
                if ( $R_{sj}$ == 1 && $\theta_{jj} > 0$ )
                    $\theta$ = sweep_matrix_on ( $j, \theta$ )
                end if
                if ( $R_{sj}$ == 0 && $\theta_{jj} < 0$ )
                    $\theta$ = reverse_sweep_matrix_on ( $j, \theta$ )
                end if
            next j
            matrix $X = Z_s$
            for int $i = 1$ to num_rows_in ( $X$ )
                vector $c = 0$
                for int $j \in m$
                    $c_j = \theta_{0j}$
                    for int $k \in o$
                        $c_j \mathrel{+}= \theta_{kj} X_{ik}$
                    next k
                next j
                for int $j \in m$
                    $T_{0j} + = c_j$
                    $T_{j0} = T_{0j}$
                    for int $k \in o$
                        $T_{kj} \mathrel{+}= c_j X_{ik}$
                        $T_{jk} = T_{kj}$
                    next k
                    for int $k \in m$ && $k \geq j$
                        $T_{kj} \mathrel{+}= \theta_{kj} + c_k c_j$
                        $T_{jk} = T_{kj}$
                    next k
                next j
            next i
        next s
        $\theta_{old} = \theta_{new}$
        $\theta_{new}$ = sweep_matrix_on ( $0, n^{-1}T$ )
        $t{+}{+}$
    until (em_has_converged( $\theta_{new}$ , $\theta_{old}$ , e) || $t = m$ )
    Z = impute_missing_values ( $\theta_{new}$ , R , Z )
    return Z
end function
```

B.1 Missingness Patterns and Associated Data Structures

The most important data structure processed within the main EM pseudo-code algorithm given above is the Y data matrix, which has one or more missing values in one or more of its columns. This matrix is passed as a parameter to the algorithm, which estimates the missing values, then returns a completed version of Y, with the missing values "filled in". For example, the Y matrix shown below has $n = 6$ rows and $p = 5$ columns. The missing values are represented by ? symbols, and the present values are represented by $-$ symbols.

$$Y = \begin{bmatrix} - & ? & - & ? & - \\ - & ? & ? & - & - \\ ? & ? & - & - & ? \\ - & ? & - & ? & - \\ - & - & - & - & - \\ - & ? & ? & - & - \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad Z = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

Rows 1 and 4 of Y both have the same missingness pattern, where missing values occur in columns Y_2 and Y_4, but this pattern does not occur in any other rows of Y. Notice that rows 2 and 6 have the same missingness pattern, and that rows 3 and 5 both have unique patterns. These four distinct patterns are stored in the R matrix, where 0 represents a missing value and 1 represents a present value. The corresponding elements in the Z vector each contain a matrix, where each such matrix $X(s)$ contains copies of all the rows in Y that have the pattern in $R(s)$. For example, we can see that row one of R contains the missingness pattern found in rows 1 and 4 of Y, and the X_1 matrix will contain copies of these two Y rows.

A new pair of temporary workspace matrices o and m are also generated, then discarded, for each iteration of the algorithm's ***repeat…..until*** loop. These matrices are used to facilitate processing by storing the column numbers corresponding to the observed and missing elements of each missingness pattern in R. For example, row 1 of R has observed values in columns 1, 3 and 5, and has missing values in columns 2 and 4. The corresponding o and m matrices store these values, as shown below.

$$R = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \begin{array}{ll} o = [1, \; 3, \; 5] & m = [2, \; 4] \\ o = [1, \; 4, \; 5] & m = [2, \; 3] \\ o = [3, \; 4] & m = [1, \; 2, \; 5] \\ o = [1, \; 2, \; 3, \; 4, \; 5] & m = [\;] \end{array}$$

To facilitate the creation of the R and Z matrices, the rows in Y must first be sorted into missingness pattern blocks, where each such block contains all of the rows in Y that have the same missingness pattern. It is absolutely crucial for fast algorithm performance that this sorting of the rows in Y is achieved using an algorithm which requires no more than (*n log n*) row comparisons, where n is the number of rows in Y.

B.2 Overview of Pseudo-code Functionality

The R matrix and the set of X matrices contained in the Z vector are created at the start of the algorithm using the two nested ***while……end while*** loops shown at the top of the pseudo-code. When this process is complete each X matrix in the Z vector will contain a different set of rows - where each such set corresponds to a unique missingness pattern. This process improves algorithm performance by removing the need to repeatedly search Y for the rows corresponding to each missingness pattern. After each Y row has been copied into an X matrix it is immediately deleted from Y, using the function call *remove_matrix_row* (*Y*(1)). This is essential, since keeping two copies of each Y row would be very wasteful of RAM, particularly in cases where the number of rows in Y was large. Note that the data for the set of Y rows which have missing values in *all elements* are excluded from both the R matrix and the Z vector, since these rows contribute nothing to the EM process, and they can cause significant deterioration in algorithm performance.

Each missingness pattern in the R matrix is processed separately within the algorithm's ***repeat……until*** loop, using the nested *for* loops contained within the *for int s = 1 to num_rows_in* (*R*) loop. The processing of each missingness pattern - which is stored in a *separate* X matrix - is outlined below.

1. The coefficients of the regression equation used to estimate the missing values in each row of the X matrix are created and stored in the θ matrix. This is achieved by repeatedly "sweeping" θ using the pattern specified in $R(s)$. Note that the same regression equation is used to estimate the missing values in every row of X.

2. The missing values in each row of X are estimated by summing the products of the observed values in each row and the regression equation coefficients contained within the swept version of θ. Note that each row in X is processed separately within the loop *for int i = 1 to num_rows_in* (*X*) and that the estimated values for each X row are stored in the c vector, which is a temporary workspace.

3. The estimated values in the c vector are added to the appropriate elements within the T matrix using a set of nested *for* loops. This completes the processing for the current missingness pattern. The process now repeats from step 1, and continues until every missingness in R has been processed.

When the above three step iteration has completed the estimates for every missing value in *Y* will be fully accumulated into *T*. The completed *T* matrix is then used to recalculate θ_{new} ready for the start of the next iteration, using the pseudo-code function call θ_{new} = *sweep_matrix_on* $(0, n^{-1}T)$.

The ***repeat…...until*** loop stop condition (*em_has_converged*(θ_{new} , θ_{old} , *e*) || $t = m$) compares each corresponding pair of elements in θ_{new} and θ_{old} using the function below. The *e* parameter should be passed to the EM algorithm as a small value (such as 0.0001) to ensure close convergence. The $t = m$ test is included in the stop condition to ensure that the algorithm always terminates after a maximum of m iterations (where m is passed to the EM algorithm as a parameter). This will be important when θ convergence is slow.

```
function boolean  em_has_converged($\theta_{new}$ , $\theta_{old}$ , $e$)
    int  p  =  num_columns_in  ($\theta_{new}$ )
    for  int  i  =  0  to  p
        for  int  j  =  i  to  p
            if  ($|\theta_{new\ (i,j)} - \theta_{old\ (i,j)}| > e|\theta_{new\ (i,j)}|$)
                return  false
        next  j
    next  i
    return  true
end function
```

B.3 Complex Functions Called From Within the Algorithm

At the start of the EM pseudo-code function the *Y* matrix is partitioned into a related set of *X* matrices, where each *X* matrix represents a particular missingness pattern in *Y*, as described in the previous section. However, for conceptual simplicity the sections that follow will refer to the *Y* matrix itself, rather than to the equivalent set of *X* matrices in *Z*.

***impute_missing_values* (*matrix* θ , *matrix R*, *vector Z*)**

This function is called at the end of the main algorithm's ***repeat…...until*** loop, just after EM convergence is achieved. The function imputes the missing values in the set of *X* matrices contained within the passed *Z* vector using; (1) The passed θ parameter, which contains the final value of the augmented covariance matrix, as generated within the algorithms convergence loop, and (2) The passed *R* parameter, which contains a list of each unique missingness pattern found in the *Y* matrix.

This function can be called from any procedure that generates a maximum likelihood estimate for the covariance matrix θ , since creation of the other parameters passed to the function is simple. Notice in particular the *sweep_matrix_on* (j,θ) and *reverse_sweep_matrix_on* (j,θ) sub-function calls, which transform the θ matrix into the state required to impute the missing values for the current missingness pattern, which is in turn retrieved from the R matrix for the current iteration of the outer loop *for int s = 1 to num_rows_in* (*R*) Finally, the missing values in every row of the X matrix (which is retrieved from the Z vector for the current missingness pattern) are imputed using the command $X_{ij} = c$. Where c is computed by summing the products of the observed values in the current row of X and the regression equation coefficients contained within the swept version of θ .

```
function matrix  impute_missing_values ( matrix $\theta$ , matrix R,  vector Z )
    for  int  s = 1  to  num_rows_in  (R)
        int  a = 0,   int  b = 0,   vector  o = 0,   vector  m = 0
        for  int  j = 1  to  num_columns_in  (R)
            if  ( $R_{sj}$ = = 1 )
                a ++ :   $o_a = j$
            else
                b ++ :   $m_b = j$
            end  if
            if  ( $R_{sj}$ = = 1 && $\theta_{jj} > 0$ )
                $\theta$ = sweep_matrix_on ( $j,\theta$ )
            end  if
            if  ( $R_{sj}$ = = 0 && $\theta_{jj} < 0$ )
                $\theta$ = reverse_sweep_matrix_on ( $j,\theta$ )
            end  if
        next  j
        matrix  $X = Z_s$
        for  int  i = 1  to  num_rows_in  (X )
            double  c = 0
            for  int  $j \in m$
                $c = \theta_{0j}$
                for  int  $k \in o$
                    $c \mathrel{+}= \theta_{kj} X_{ik}$
                next  k
                $X_{ij} = c$
            next  j
        next  i
    next  s
    return  Z
end function
```

initial_parameter_estimate (vector Z, int f)

The EM process requires that the initial value of the $\theta = (\mu, \Sigma)$ parameter, which describes the distribution of the *complete* dataset in the Y matrix, must be estimated before the algorithm's convergence loop is started. The pseudo-code given below returns θ as a (p + 1) x (p + 1) augmented covariance matrix, where p is the number of columns in *Y*. For implementation simplicity, the rows and columns of the calculated matrix T are indexed from 0 to p, rather than from 1 to (p + 1).

```
function matrix  initial_parameter_estimate ( vector Z,  int f )
    matrix X = $Z_f$
    int n = num_rows_in (X)
    int p = num_columns_in (X)
    if ($n \le p$)
        $\theta$ = alternative_parameter_estimate (Z)
        return $\theta$
    end if
    matrix T = new matrix (p + 1, p + 1)
    $T_{00} = n$
    for int i = 1 to n
        for int c = 1 to p
            $T_{0c}$ += $X_{ic}$
            $T_{c0} = T_{0c}$
            for int j = c to p
                $T_{cj}$ += $X_{ic} X_{ij}$
                $T_{jc} = T_{cj}$
            next j
        next c
    next i
    $T = T/n$
    $\theta$ = sweep_matrix_on (0, T)
    return $\theta$
end function
```

It is important to note that the function above estimates the starting value of $\theta = (\mu, \Sigma)$ by performing calculations based *only* on the complete data rows in the Y matrix (i.e. those rows with no missing values), and this method can be used in most cases. However, notice that the function given above exits (near the top of the pseudo-code) when the number of complete rows is less than or equal to the number of columns in *Y*, which should rarely occur in practice. When this does occur an alternative method of estimating the initial value of θ should be used, depending on the nature of the distribution in *Y*, as described by Schafer

(1997). For example, the means and covariances could be calculated using only the observed data values. Little and Rubin (2002) explain that caution should be exercised in cases where difficulty arises in estimating the starting value of θ. In these cases the Y dataset may not be amenable to the EM process, and it could be sensible to simply terminate the algorithm by displaying a warning message to the user, explaining that the Y dataset should be examined further before proceeding with the imputation process.

observed_data_sufficient_statistics (vector Z, matrix R)

This function returns the T_{obs} matrix, which is created only once (at the start of algorithm) then stored and used repeatedly. The structure of the $T(s)$ and $T_{obs}(s)$ matrices, and the methods for calculating their elements are given below.

$$T(s) = \begin{bmatrix} n_s & \Sigma y_{i1} & \Sigma y_{i2} & \cdots & \Sigma y_{ip} \\ \Sigma y_{i1} & \Sigma y_{i1}^2 & \Sigma y_{i1} y_{i2} & \cdots & \Sigma y_{i1} y_{ip} \\ \Sigma y_{i2} & \Sigma y_{i1} y_{i2} & \Sigma y_{i2}^2 & \cdots & \Sigma y_{i2} y_{ip} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma y_{ip} & \Sigma y_{i1} y_{ip} & \Sigma y_{i2} y_{ip} & \cdots & \Sigma y_{ip}^2 \end{bmatrix} \quad \text{and} \quad T = \sum_{s=1}^{S} T(s)$$

Where the sums in each individual $T(s)$ element (such as Σy_{i1} etc.) are taken over the subset of rows in Y that have the same missingness pattern (as stored in the corresponding $R(s)$ row) and where n_s gives the total number of rows in Y that have that particular missingness pattern. The matrix T contains the sum of all such $T(s)$ matrices, where each element stores the calculated sums for *both* the observed and the missing elements of the rows in Y. However, the T_{obs} matrix we require should contain *only* the calculated sums for the observed elements in Y. Despite this difference, the required T_{obs} matrix can be calculated in a very similar way to T, the only difference being that the elements in the rows and columns that correspond to missing values in Y are set equal to zero. For example, consider a Y matrix that has four columns labelled Y_1 to Y_4, where the data for a particular missingness pattern s is present for columns Y_1, Y_3 and Y_4, but is missing for column Y_2. In this case T_{obs} can be calculated as shown below (notice that the elements in $T_{obs}(s)$ are the same as those in $T(s)$, except for the row and column corresponding to Y_2).

$$T_{obs}(s) = \begin{bmatrix} n_s & \Sigma y_{i1} & 0 & \Sigma y_{i3} & \Sigma y_{i4} \\ \Sigma y_{i1} & \Sigma y_{i1}^2 & 0 & \Sigma y_{i1} y_{i3} & \Sigma y_{i1} y_{i4} \\ 0 & 0 & 0 & 0 & 0 \\ \Sigma y_{i3} & \Sigma y_{i1} y_{i3} & 0 & \Sigma y_{i3}^2 & \Sigma y_{i3} y_{i4} \\ \Sigma y_{i4} & \Sigma y_{i1} y_{i4} & 0 & \Sigma y_{i3} y_{i4} & \Sigma y_{i4}^2 \end{bmatrix} \quad \text{and} \quad T_{obs} = \sum_{s=1}^{S} T_{obs}(s)$$

Where the elements in each $T_{obs}(s)$ contain the calculated sums for a particular missingness pattern s. However, we require the matrix T_{obs} which contains the sum of all such matrices, where s = 1 to S and S gives the total number of unique missingness patterns present in Y. The pseudo-code function below calculates and returns T_{obs} using the method shown above. For implementation simplicity, the rows and columns in the passed parameters Z and R, and in the matrix X, are indexed starting at 1, whereas the rows and columns in the augmented matrices T_{obs} and T are indexed from 0 to p.

```
function matrix  observed_data_sufficient_statistics  ( vector Z,  matrix R )
    int  n  =  num_rows_in  (R)
    int  p  =  num_columns_in  (R)
    matrix  $T_{obs}$  =  new  matrix  (p + 1,  p + 1)
    for  int  s  = 1  to  n
        matrix  $T_{obs}(s)$  =  0
        matrix  X  =  $Z_s$
        int  r  =  num_rows_in  (X)
        $T_{obs}(s)_{00}$  =  r
        for  int  i  = 1  to  r
            for  int  c = 1  to  p
                $T_{obs}(s)_{0c}$ +=  $X_{ic} R_{sc}$
                $T_{obs}(s)_{c0}$  =  $T_{obs}(s)_{0c}$
                for  int  j  =  c  to  p
                    $T_{obs}(s)_{cj}$ +=  $X_{ic} R_{sc} X_{ij} R_{sj}$
                    $T_{obs}(s)_{jc}$  =  $T_{obs}(s)_{cj}$
                next  j
            next  c
        next  i
        $T_{obs}$  +=  $T_{obs}(s)$
    next  s
    return  $T_{obs}$
end function
```

The use of the T_{obs} matrix within the EM algorithm, and its relationship to the T and θ_{new} matrices, is outlined below (refer to the main EM pseudo-code listing for further clarification).

1. The T matrix is set equal to T_{obs} at the start of the algorithm's ***repeat…..until*** loop. At this stage of the processing T will contain *only* the data for the observed values in *Y*.
2. The missing values in *Y* are estimated using the current value of the θ parameter, then accumulated into T using the nested *for…. next* loops contained within the overall ***repeat…..until*** code structure.
3. Just before the current iteration of the ***repeat…..until*** loop ends the T matrix is used to recalculate θ_{new} using the function call $\theta_{new} = \mathit{sweep\_matrix\_on}\ (0, n^{-1}T)$
4. The process begins again at step 1. Note that the T_{obs} matrix does not need to be recalculated at this stage, since it has already been stored in RAM by the above function (as previously explained).

This processing method is very performance efficient because it allows θ_{new} to be recalculated at the end of each EM iteration using a single call of the sweep function. The alternative method of recalculating θ_{new} would require accessing and processing every row in the *Y* matrix, which would take much longer, particularly if the number of rows in *Y* was large. This efficient procedure is perhaps the most procedurally elegant aspect of the EM algorithm. It also explains why the estimated values are accumulated into the *T* matrix, rather than being imputed directly into *Y*, during each iteration of the algorithm. Another important performance benefit of the above process is that the missing values in the elements of the *Y* matrix need only be imputed once – just before the algorithm terminates.

C Software and Hardware Platform Used

All of the software development and all of the experimentation described in this thesis were carried out in the period between October 2004 and July 2007. The following computer platform was used to develop the software, and to perform the experiments.

- All software was developed using the Microsoft C# programming language (Hejlsberg et al, 2004) within the Microsoft .NET 2003 software development environment (Version 7.1.3088 utilising the .NET Framework, Version 1.1.4322 SP1).

- The Microsoft Windows XP Professional® operating system Version 2002 was used. No other applications were running while any of the experiments were being performed.

- A Dell Dimension® 8400 desktop personal computer (PC) with an Intel Pentium® 4 CPU running at 3GHz was used. This computer had 3 gigabytes of RAM.

D Notation and Terminology Used in This Thesis

This appendix describes the mathematical notation and the associated nomenclature used in this thesis. ***In a sense, this section defines the central problem addressed within the thesis,*** in that the proposed imputation evaluation method has been devised to assess the feasibility of imputing missing values in numeric data matrices such as the one shown below.

$$\begin{array}{r|ccccc|}
i = 1 & 1 & 0 & 1 & 0 & 1 \\
2 & 1 & 0 & 0 & 1 & 1 \\
3 & 0 & 0 & 1 & 1 & 0 \\
4 & 1 & 0 & 1 & 0 & 1 \\
5 & 1 & 1 & 1 & 1 & 1 \\
6 & 1 & 0 & 0 & 1 & 1 \\
\multicolumn{1}{r}{} & \multicolumn{1}{c}{j = 1} & \multicolumn{1}{c}{2} & \multicolumn{1}{c}{3} & \multicolumn{1}{c}{4} & \multicolumn{1}{c}{5}
\end{array} = Y$$

The matrix rows are indexed as $\boldsymbol{i = 1 \text{ to } n}$

The matrix columns are indexed as $\boldsymbol{j = 1 \text{ to } p}$

Matrix elements are referenced using Y_{ij}

Known values are represented by a value of 1

Missing values are represented by a value of 0

e.g. rows 1 and 4 have “missingness pattern” 10101

Fig D.1 – Matrix notation used to represent missing value datasets in this thesis

The matrix notation shown in Fig. D.1 is used across the entire thesis. Each column in the matrix stores the values taken by a particular numeric variable. And each row in the matrix stores the values of a set of related variables - such as a statistical observation or a set of values describing the attributes of a particular object. It is important to note that the actual values stored in the matrix elements *are not shown* in the matrix above. Instead, the state of “missingness” for each value is shown - in such a way that the “missingness pattern” for each row in the matrix can be seen. These missingness patterns are referred to throughout the thesis using the notation given in Fig. D.1. Exceptions to this notation are clearly stated where they are used.

However, the nomenclature used to describe rectangular datasets with missing values differs depending on the discipline that refers to them, and the problem of missing data is common to many of these disciplines. For example, in statistics the dataset with missing values is generally referred to as a sample, the rows in the dataset are referred to as observations and the columns as variables. In survey sampling, the dataset is also referred to as a sample, but the rows are called respondents and the columns are known as responses. To further confuse matters, disciplines such as the study of relational databases and data mining theory use their own distinctive terminology to describe rectangular datasets.

To avoid confusion this thesis *generally* refers to the dataset as a “matrix” which contains “rows” and “columns”. Although matrix rows are occasionally be referred to as “observations” and matrix columns are sometimes be referred to as “variables”, where this is required to clarify the meaning of the explanations given (depending on context).

E Thesis Publications

This appendix contains copies of the two published papers that were written by the author as the work described in this thesis progressed. The first paper discusses the ideas that led to the development of the imputation evaluation method described in chapter four. The second paper discusses the work described in chapter three. Full publication details are given below.

Solomon, N., Oatley, G. and McGarry, K, (2007a), *A Dynamic Method for the Evaluation and Comparison of Imputation Techniques*, In: Proceedings of the World Congress on Engineering 2007 (International Conference of Computational Statistics and Data Engineering), ISBN: 978-988-98671-2-6, pp. 974-982, Newswood Limited, International Association of Engineers, Hong Kong.

Solomon, N., Oatley, G. and McGarry, K, (2007b), *A Fast Multivariate Nearest Neighbour Imputation Algorithm*, In: Proceedings of the World Congress on Engineering 2007 (International Conference of Computational Statistics and Data Engineering), ISBN: 978-988-98671-2-6, pp. 940-947, Newswood Limited, International Association of Engineers, Hong Kong.