Software Cost Estimation with Incomplete Data

Kevin D. Strike

School of Computer Science

McGill University, Montreal

March, 2000

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements of the degree of Master of Science.

0-612-64461-8

Abstract

The construction of software cost estimation models remains an active topic of research. The basic premise of cost modeling is that a historical database of software project cost data can be used to develop a quantitative model to predict the cost of future projects. One of the difficulties faced by workers in this area is that many of these historical databases contain substantial amounts of missing data. Thus far, the common practice has been to ignore observations with missing data. In principle, such a practice can lead to gross biases, and may be detrimental to the accuracy of cost estimation models. In this paper, we describe an extensive simulation where we evaluate different techniques for dealing with missing data in the context of software cost modeling. Three techniques are evaluated: listwise deletion, mean imputation and eight different types of hot-deck imputation. Our results indicate that all the missing data techniques perform well, with small biases and high precision. This suggests that the simplest technique, listwise deletion, is a reasonable choice. However, this will not necessarily provide the best performance. We provide a decision tree to select the best performing missing data techniques depending upon the pattern, mechanism and percentage of missing data.

Résumé

La construction de modèles d'estimation de coût logiciel demeure un sujet de recherche important. Le principe de base de la modélisation de coût est de constituer une base de données sur l'historique des coûts de projets logiciels, afin de développer un modèle quantitatif pour prédire les coûts de projets futurs. Une des difficultés auxquelles on est confronté est que ces bases de données contiennent des quantités importantes de données manquantes. Jusqu'à présent, l'usage a été d'ignorer les observations incomplètes. En principe, de telles pratiques peuvent conduire à d'importantes erreurs, et peuvent nuire à la précision du modèle. Dans ce document, nous présentons une simulation approfondie où nous évaluons différentes techniques pour traiter des données manquantes dans le cadre de la modélisation de coût logiciel. Les trois techniques évaluées sont: l'élimination des enregistrements non complets, l'imputation par la moyenne et huit types différents d'imputation par appariement. Nos résultats montrent que ces techniques sont efficaces, avec peu d'erreurs et une grande précision. La technique la plus simple, la suppression par liste, semble donc un choix raisonnable. Toutefois, elle ne donnera pas nécessairement les meilleures performances. Ainsi, nous fournissons un arbre de décision pour séléctionner la meilleure technique en fonction de la répartition, du mécanisme d'apparition et du pourcentage des données manquantes.

Acknowledgments

I would like to thank my co-supervisor Khaled El Emam for his help in initiating, guiding and editing this work. I would like to thank my co-supervisor Nazim H. Madhavji for his support and valuable suggestions that have encouraged and helped tremendously.

In addition, I wish to thank Risto Nevalainen and Pekka Forselius of Software Technology Transfer Finland (STTF) for defining, collecting and providing me with the data set used in this study, as well as Fabien Deschodt for his help in the French translation of the abstract.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

There exists a vast literature on the construction of software cost estimation models, for example [63][18][1][2][12][25][44][30][51][85][93][83][90]. The basic premise is that one can develop accurate quantitative models that predict development effort[1] using historical project data.[2] The predictors constitute a measure of size, whether measured in terms of LOC or a functional size measure, and a number of *productivity factors* that are collected through a questionnaire, such as questions on required reliability, documentation match to life cycle needs and analyst capability [87].

Knowing the estimated cost of a particular software project early in the development cycle is a valuable asset [63]. Management can use cost estimates to approve or reject a project proposal or to manage the development process more effectively. For example, additional developers may need to be hired for the complete project or for areas that will require a large amount of effort. Furthermore, accurate cost estimates would allow organizations to make more realistic bids on external contracts. Cost estimation models have not been limited to prediction of total project cost. For instance, some recent work constructed a model to estimate the effort to perform a software process assessment [46], and to estimate the effort required to become ISO 9001 certified [72][73], both of which are relevant to contemporary software organizations.

A common practical problem with constructing cost estimation models is that the historical software engineering data sets frequently contain substantial amounts of missing values [15][16][37]. Such missingness would impact the productivity factors in historical data sets most severely since they are the variables collected through a questionnaire.[3] While one should strive to minimize missing values, in practice their

[1] Effort is considered to be the most important ingredient of cost, and is almost always the variable of interest.

[2] Other related models that can be constructed using historical project data include those for size and schedule prediction [92].

[3] Our concern here is not with non-respondents to a questionnaire, but with those who respond to some questions and not others. A great deal of effort should be placed on reducing nonresponse to ensure that resulting conclusions are as accurate as possible. For example, a study by Heberlein and Baumgartner [40] examined the effects of many variables on response rate. It was found that follow up letters and questionnaires judged to be salient to the respondent have the largest effect on the

presence is usually unavoidable. Missing values are not unique to software cost estimation, but is a problem that concerns empirical scientists in other disciplines [50][41][57].

The most common factors that lead to the presence of missing data include individuals not responding to all questions in a questionnaire, either because they run out of time[4], they do not understand the questions or they do not have sufficient knowledge to answer the questions and opt not to respond, [5] or individuals may not wish to divulge certain information that may be harmful or embarrassing to them.[6] Furthermore, the amount of missing values tends to increase as more variables are present in the data set [75]. It is common for cost estimation data sets to have a multitude of productivity factors.

There are many techniques that have been developed to deal with missing data in the construction of quantitative models [58]. The simplest technique is to ignore observations that have missing values (this is called *listwise deletion*). In fact, this is the default approach in most statistical packages [58]. This, however, can result in discarding large proportions of a data set and hence in a loss of information that was costly to collect. For example, Kim and Curry [50] note that with only 2% of the values missing at random in each of 10 variables, one would lose 18.3% of the observations on average using listwise deletion. Furthermore, in a simulation they performed with 5 variables having 10% of their values missing at random, 59% of the observations would be lost with listwise deletion, on average. In addition, listwise deletion may bias correlations downwards. If a predictor variable has many missing high values, for example, then this would restrict its variance, resulting in a deflation

---

response rate. An increase in salience of the study to the respondent, including how much knowledge and interest the respondent has in the subject matter will result in a higher importance placed on the study, resulting in a higher response rate. In addition, increased contact with the respondent, including follow up letters, will further increase perceived importance of the study and also result in higher response rates [76]. Also, the effects of factors such as sponsorship, personalization, length, and monetary rewards on response rates have been examined in the literature. Further details on survey design to maximize the response rate are provided in [28], [27] and [76].

[4] It is generally known that an increase in effort required to complete surveys is likely to result in many respondents complaining and leaving many questions unanswered [24].

[5] It is not uncommon to treat "Don't Know" (DK) responses as missing values when there is no intrinsic interest in the fact that a DK response has been provided [80].

[6] It is common that respondents refuse to provide certain information. For example, in surveys that contain a question about personal income, it is expected that individuals with very high and very low incomes do not offer this information [58][50]. In a cost estimation context, respondents may not wish to express the low level of domain or programming experience of the team leader, especially if the individual is easily identifiable.

of its correlation with a completely observed criterion variable. The same applies if the missing values were on the criterion variable. Measures of central tendency may be biased upwards or downwards depending on where in the distribution the missing data appear. For example, the mean may be biased downward if the data are missing from the upper end of the distribution.

Another set of techniques *impute* the missing values. A common approach is to use mean imputation. This involves filling in the missing values on a variable with the mean of observations that are not missing. However, mean imputation attenuates variance estimates. For example, if there are 30 observations of which 5 have missing values, then we could substitute five means. This would increase the number of observations without affecting the deviations from the overall mean, hence reducing the variance [59][62]. There are alternative forms of imputation that are based on estimates of the missing values using other variables from the subset of the data that have no missing values.

In the context of cost estimation, researchers rarely mention how they dealt with missing values.[7] When they do, their solution tends to be to ignore observations with missing values, i.e. listwise deletion. For example, in the Walston and Felix study [93], different analyses rely on a different number of observations from the historical database, indicating that for some of the variables there were missing values. In one recent cost estimation study of European space and military projects, the authors removed observations that had missing values, resulting in some instances to the loss of approximately 38% of the observations [18]. A comparison study of different cost estimation modeling techniques noted that for approximately 46% of the observations there were missing values [20]. The authors then excluded observations with missing values for the different types of comparisons performed. Another study used a regression model to predict the effort required to perform a software process assessment based on the emerging ISO/IEC 15504 international

[7] This is also a typical problem in other disciplines. For example, in one study [77], articles from the Journal of Applied Psychology and Personnel Psychology were randomly chosen and examined to see what methods were used to handle missing data. It was found that many studies do not state whether or not observations contained missing data. It follows that in these cases the technique for dealing with missing data is not mentioned either. One reason for this could be that journals only accept those studies that are deemed strong and may hesitate to publish studies that report high levels of missing data. Evidence suggested that 1/2 to 2/3 of the studies ignored observations with missing values.

standard [46]. In this study 34% of the total number of observations were excluded from the analysis due to missing values.[8]

To date, there is no evidence that such a simple practice is the best one, or if it is detrimental to the accuracy of cost estimation models. It is plausible that certain types of imputation techniques would save the large proportions of discarded data and result in models with much improved prediction accuracy. It would be of practical utility then to have substantiated guidelines on how to deal with missing values in a manner that would minimize harm to model accuracy. This study takes a step in that direction.

In this paper, we present a detailed simulation study of different techniques for dealing with missing values when building cost estimation models: listwise deletion, (unconditional) mean imputation, and eight different types of hot-deck imputation. We also simulate three different types of missingness mechanisms: (a) missing completely at random, (b) where missingness depends on the size of the project, and (c) where missingness depends on the value of the variable with missing values; two types of missing data patterns: univariate (random) and monotone; and missingness on one productivity factor up to all productivity factors in a model.

Our evaluative criteria focus on the accuracy of prediction, and consist of the common measures: Absolute Relative Error and Pred25 [25]. The summary measures are the bias and variation of the accuracy measures. We focus on ordinary least squares regression as the modeling technique since this is one of the most common modeling techniques used in practice [37], e.g., [70][93][23][22][18][63]. Furthermore, there has been recent evidence that ordinary least squares regression is as good as or better than many competing modeling techniques in terms of prediction accuracy [19][20].

Briefly, our results indicate that all MDTs have a good performance in terms of bias and precision under the different contexts simulated. This suggests that the simplest technique, listwise deletion, is a reasonable choice. However, listwise deletion will

[8] In some other studies missing values were dealt with in a non-traditional manner. For example, a recent cost estimation model construction study replaced "I don't know" and "It does not apply" responses on a productivity factor with the middle point on a

not provide the best performance amongst the different MDTs. We present a decision tree to select the best performing MDT depending upon the pattern, mechanism, and percentage of missing data.

In the following chapter, we present an overview of the missing data problem and the techniques that have been developed for dealing with it. Chapter 3 describes our research method in detail. The results of our simulation are described in Chapter 4 with a discussion of their implications and limitations. We conclude the paper in Chapter 5 with a summary and directions for future research.

5-point scale [80].

Chapter 2: Background

In this chapter, we define some terminology and provide an overview of missing data techniques (MDTs) and their general applicability.

2.1 Terminology

An important distinction when discussing MDTs is the mechanism that caused the data to be missing [58]. Consider a data set with two variables, X_1 and X_2. Let us assume that missing data occurs on variable X_1 only. To make the scenario concrete, let variable X_1 be analyst capability and variable X_2 be project size. If the probability of response to X_1 does not depend on either X_1 or X_2, then it is said that the missing data mechanism is Missing Completely At Random (MCAR). Thus, if the missingness of the analyst capability variable is independent of project size and analyst capability, the mechanism is MCAR. If the probability of response depends on X_2 but not on X_1, then we say that the missing data mechanism is Missing At Random (MAR). This would be exemplified by the situation whereby the missingness on the analyst capability variable is higher for, say, small projects than for large projects. The third mechanism is if the probability of response depends on the value of X_1 itself. This would occur if respondents tend not to answer the question when the analyst capability is, say, low. This is termed non-ignorable response.

In general, the suitability of MDTs will be influenced by the missing data mechanism and the percentage of observations with missing values. We outline some common MDTs below.

2.2 Common Missing Data Techniques (MDTs)

There exist several strategies for dealing with missing data. It is generally accepted that if the data set contains a relatively small amount of missing data and if this data is missing randomly, then all MDTs will be equally suitable [31][53][50][3][74]. It should be noted that caution must be exercised when classifying data sets as having small amounts of missing data. For example, if the small amount of missing data is only found in a few variables and is distributed randomly among all observations, the total percentage of observations containing missing data may be relatively large [50].

The choice of MDT becomes more important as the amount of missing data in the data set increases [74]. Another important factor in choosing a suitable MDT is the mechanism that lead to the missing values, whether MCAR, MAR or non-ignorable [58]. In general, bias will result if the distribution of the missing values is different from that of the observed values [31].

There are two general classes of MDTs that can be applied: deletion methods and imputation methods. These are described below.

2.2.1 Deletion Methods

In summary, deletion methods ignore missing values. These procedures may result in the loss of a significant amount of data, but are widely used because of their simplicity [77][50].

2.2.1.1 Listwise Deletion

Analysis with this method makes use of only those observations that do not contain any missing values. This may result in many observations being deleted but may be desirable as a result of its simplicity [50]. This method is generally accepted when there are small amounts of missing data and when the data is missing randomly. If missingness is not random (e.g. non-ignorable) this method may lead to biases.[58] For example, consider a study in which respondents with low incomes are less likely to respond to a question about personal income. If all the observations with missing values for personal income were discarded, then the conclusions of the analysis would be biased towards individuals with higher incomes. Conversely, if missingness is completely random, excluding observations with missing data would not affect the conclusions of the study.

2.2.1.2 Pairwise Deletion

In an attempt to reduce the considerable loss of information that may result from using listwise deletion, this method considers each variable separately. For each variable, all recorded values in each observation are considered and missing values are ignored. For example, if the objective is to find the mean for the personal income variable, the mean is computed using all recorded incomes. In this case, reported incomes in all observations will be considered, regardless of whether they are missing other variables. This technique will likely result in the sample size changing for each considered variable. Note that pairwise deletion becomes listwise deletion when all the variables are needed for a particular analysis, (e.g. multiple regression). For the same reasons as mentioned for listwise deletion, this method will perform well, without bias, if the data is missing at random [58].

It seems intuitive that since pairwise deletion makes use of all observed data, it should outperform listwise deletion in cases where the missing data is MCAR and correlations are small [58]. This was found to be true in the Kim and Curry study [50]. In contrast, other studies have found that when correlations are large, listwise outperforms pairwise deletion [3]. The disadvantage of pairwise deletion is that it may generate an inconsistent covariance matrix in the case where multiple variables contain missing values. In contrast, listwise deletion will always generate consistent covariance and correlation matrices [50].

In cases where the data set contains large amounts of missing data, or the mechanism leading to the missing values is non-random, Haitovsky proposed that imputation techniques might perform better than deletion techniques [38].

2.2.2 Imputation Methods

The basic idea of imputation methods is to replace missing values with estimates that are obtained based on reported values [81][32]. In cases where much effort has been expended in collecting data, the researcher will likely want to make the best possible use of all available data and prefer not to use a deletion technique [24]. Imputation methods are especially useful in situations where a complete data set is required for the analysis [59]. For example, in the case of multiple regression, all observations must be complete. In these cases, substitution of missing values results in all observations of the data set being used to construct the regression model. It is important to note that no imputation method should *add* information to the data set. In the case of multivariate data, it makes sense that we might be able to obtain information about the missing variable from those observed variables. This forms the basis for imputation methods. The primary reason for using imputation procedures is to reduce the non-response bias that would result if all the observations that have missing values are deleted.

It has been proposed that a good imputation procedure should:

1) use observed values to predict a distribution for the missing values. The method should then impute values based on this distribution [59].
2) use all the observed values for each observation containing missing values [59].
3) use any external constraints about values being imputed. (For example, variables $X_1 + X_2$ must be less than or equal to k.) [59][81]
4) not impute values that have been extrapolated a considerable distance from the observed data. (Except in the case where substantial evidence is available to warrant this.) [59]
5) impute values that do not change the predicted distribution (in 1) [59][81].
6) provide methods that adjust conclusions made from the data set since it has been imputed with missing values [59].

2.2.2.1 Mean Imputation

This method imputes each missing value with the mean of observed values. For example, a missing income could be replaced by the mean of all observed incomes. The advantage of using this method is that it is simple to implement and no observations are excluded, as would be the case with listwise deletion. The disadvantage is that the measured variance for that variable will be underestimated [78][58]. For example, if a question about personal income is less likely to be answered by those with low incomes, then imputing a large amount of incomes equal to the mean income of reported values decreases the variance. Note that mean imputation does not satisfy all of the six points listed above.

2.2.2.2 Hot-Deck Imputation

Hot-deck imputation involves filling in missing data by taking values from other observations in the same data set. The choice of which value to take depends on the observation containing the missing value. The latter property is what distinguishes hot-deck imputation from mean imputation.

In addition to reducing non-response bias and generating a complete data set, hot-deck imputation preserves the distribution of the sample population. Unlike mean imputation, which distorts the distribution by repeating the mean value for all the missing observations, hot-deck imputation attempts to preserve the sample distribution by substituting different observed values for each missing observation.

Hot-deck imputation selects an observation (donor) that best matches the observation containing the missing value (client). The donor then provides the value to be imputed into the client observation. For example, a study may be able to gather a certain variable for all observations, such as geographic location. In this case, a categorical hot-deck is created in which all observations are separated into categories according to one or more classification variables, in this case geographic location. Observations containing missing values are imputed with values obtained from complete observations within each category. In this case, it is assumed that the distribution of the observed values is the same as that of the missing values. This places great importance on the selection of the classification variables and assumes a large correlation between the observed and missing values.

In other cases, there may not be any categorical data and the variables in which to assess 'similarity' may be numerical.

Numerical hot-decks are typically found in data sets that contain multivariate observations with numeric variables. In this case, a donor is selected that is the most similar to the client. Similarity is measured by using a distance-function that calculates the distance between the client and prospective donors. The hot-deck is the set of all complete observations. For each client, a donor (or set of donors) is chosen from the hot-deck that contains the smallest distance to the client. This distance can be based on one or more variables of the observation. The selection of which variables on which to use the distance-function is ideally those variables that are highly correlated to the variable being imputed. In the case where a set of donors has been obtained, the value to impute may be taken from the best donor, random donor, or an average over all donors. The purpose of selecting a set of donors is to reduce the likelihood of an extreme value being imputed one or more times [32][81].

Colledge et al. concluded that hot-deck imputation appears to be a good technique for dealing with missing data, but suggested that further analysis be done before widespread use [24].

2.2.2.3 Cold-Deck Imputation

This method is similar to hot-deck imputation except that the selection of a donor comes from the results of a previous survey [58].

2.2.2.4 Regression Imputation

This is a modeling technique that replaces each missing value with a predicted value based on a regression model. First, a regression model is built using both the complete and incomplete observations. For each incomplete observation, each missing value is replaced by the predicted value found by using the regression model [58].

2.2.2.5 Multiple Imputation Methods

Modeling techniques that impute one value for each missing value underestimate standard error. This is the case because imputing one value assumes no uncertainty. Multiple imputation remedies this situation by imputing more than one value, taken from a predicted distribution of values [61]. The set of values to impute may be taken from the same or different models displaying uncertainty towards the value to impute or the model being used respectively. For each missing value, an imputed value is selected from the set of values to impute, each creating a complete data set. Each data set is analysed individually and final conclusions are obtained by merging those of the individual data sets. This technique introduces variability due to imputation, contrary to the single imputation techniques. The only disadvantage, with respect to single imputation techniques, is that it is more computationally intensive [58]. Consequently, this technique is most useful when the data set contains small amounts of missing data in a small proportion of the observations [79].

2.3 Summary

To our knowledge, there have been no previous studies of MDTs within software engineering. Therefore, it is not possible to determine which MDTs are suitable and under what conditions for software engineering studies.

In our study, we focus on three types of MDTs: listwise deletion, mean imputation, and hot-deck imputation. We chose listwise deletion because it is common practice in software cost estimation studies, and therefore we wished to determine its performance. Furthermore, it has been noted that in general empirical enterprises, listwise deletion and mean imputation are the most popular MDTs [74]. Hot-deck imputation is of interest since it has been adopted in some highly visible surveys, such as the British Census [7][32], the U.S. Bureau of the Census Current Population Survey, the Canadian Census of Construction [32], and the National Medical Care Utilization and Expenditure Survey [57]. Furthermore, some authors contend that the hot-deck is the most common MDT for complex surveys [31].

Chapter 3: Research Method

In this chapter, we outline the objective of the study and describe the data set used. In addition, we discuss the research design and how the empirical study was conducted.

3.1 Objective of Study

The objective of this study is to compare different MDTs for dealing with the problem of missing values in historical data sets when building software cost estimation models. Specifically, listwise deletion, mean imputation and hot-deck imputation were compared. The focus is on simple methods that would allow researchers to easily implement the result. Since for cost estimation models the most important performance measure is their prediction accuracy, we evaluate how this accuracy is affected by using the different missing data techniques. By identifying the most appropriate technique, future researchers would have substantiated guidance as to appropriate ways for dealing with missing values.

3.2 Data Source and Content

The data set used in this study is called the *Experience Database*. The Experience Database began with the co-operation of 16 companies in Finland. Each company must purchase the Experience tool and contribute the annual maintenance fee. This entitles them to the tool that incorporates the database, new versions of associated software and updated data sets. Each company can add their own data to the tool and are encouraged, through incentives, to donate their data to the shared database. For each project donated, the company is given a reduction in the annual maintenance fee. Since all companies collect the data using the same tool and the value of each variable is well defined, integrity and comparability of the data is maintained. In addition, companies that provide data are subsequently contacted in order to verify their submission.

The primary advantage of this data base, with respect to this study, is that it does *not* contain missing values. The fact that this relatively large data set does not contain missing values is due to the careful manner in which the data was collected and extensive follow up. This allows us to simulate various missing data patterns, as will be explained below.

The data set is composed of 206 software projects from 26 different companies. The projects are mainly business applications in banking, wholesale/retail, insurance, public administration and manufacturing sectors. This wide range of projects allows for generalizable conclusions that can be applied to other projects in the business application domain. Six of the companies provided data from more than 10 projects. The system size is measured in unweighted and unadjusted Function Points [2]. For each project, we had the total effort in person-months and values on fifteen productivity factors. The variables considered in our analysis are presented in Table 1. The productivity factors are defined in detail in Appendix B.

Variable	Description	Values / Range / Unit
Effort	Total project effort	Person hours (ph)
FP	System size measured in function points	Unadjusted Unweighted Function Points
PF01-PF15	15 Productivity Factors: Customer Participation, Development Environment Adequacy, Staff Availability, Use of Standards, Use of Methods, Use of Tools, Software Complexity (Logical), Requirements Volatility, Quality Requirements (Software), Efficiency Requirements, Installation Requirements, Staff Analysis Skills, Staff Application Knowledge, Staff Tool Skills, Staff Team Skills	1 – 5 (very small – very large)

Table 1: Variables used in our simulation from the Experience Database.

3.3 Cross Validation

If a cost estimation model is developed using a particular data set and the accuracy of this model is evaluated using the same data set, the accuracy obtained will be optimistic. The calculated error will be low and will not represent the performance of the model on another, separate data set. This study divides the Experience database into two parts. Since 30% of the software projects came from a bank, the database was split into a bank data set (63 projects) and the rest (143 projects). We termed the former the *test* data set, and the latter the *training* data set. The cost estimation model is developed using the 143 projects in the training data set. We then evaluated its prediction performance on the 63 project test data set. This approach is similar to the situation whereby a multi-organizational data set is used to build a model, and then an individual organization applies that model for its own projects (e.g. [73]).

3.4 Regression Model

We used multivariate least squares regression analysis by fitting the data to a specified model that predicts effort [39]. The selected model specification is exponential, because linear models revealed marked heteroscedasticity, violating one assumption for applying regression analysis. [9]

The general form of the regression model is as follows:

$$\log(Effort) = \beta_0 + \beta_1 \log(FP) + \sum_{i>0} \beta_{i+1} \log(T_i) \qquad \textbf{Eqn. 1}$$

where the T_i values are the productivity factors.

It is known that many of the productivity factors are strongly correlated with each other [18][52][86]. Although some cost estimation models have been developed that contain many productivity factors, [93][5][12][64] found that for a given environment, only a small number of significant productivity factors are needed in order to develop an accurate effort estimation model. This conclusion is supported by [52][67] and [8].

Therefore, we first reduced the number of variables down from 15 using only the training data set. Two approaches were followed. A mixed stepwise process was first performed to select variables having a significant influence on effort (alpha=0.05). In

[9] One can make other substantive arguments for selecting this functional form. In software engineering, there has been a debate over whether economies of scale do indeed exist, and if so, what is the appropriate functional form for modeling such economies. The concept of economies of scale states that average productivity increases as the system size increases. This has been attributed, for example, to software development tools whereby the initial tool institutionalization investment may preclude their use on small projects [12]. Furthermore, there may be fixed overhead costs, such as project management, that do not increase directly with system size, hence affording the larger projects economies of scale. On the other hand, it has been noted that some overhead activities, such as documentation, grow at a faster rate than project size [45], contributing to diseconomies of scale. Furthermore, within a single organization, it is plausible that as systems grow larger, then larger teams will be employed. Larger teams introduce inefficiencies due to an increase in communication paths [14], the potential for personality conflicts [12], and more complex system interfaces [25]. A series of studies on the existence of (dis)economies of scales provided inconsistent results [9][10][52]. In another effort to determine whether such (dis)economies of scale exist, Hu [42] compared a simple linear model with a quadratic, log-linear, and translog models, and used objective statistical procedures to make that determination. He also investigated what the appropriate functional form should be. He concluded that the quadratic form is the most appropriate. Subsequently, his study was criticised on methodological grounds [71][18]. Another study that compared functional forms, that addressed some of these shortcomings, concluded that the log-linear form, which we use, is the most plausible one [18]. The general multiplicative form for cost estimation models has also been recommended in [67] since the impact of each of the productivity factors is likely to be proportional to the size of the software.

this case the size measure, FP, was selected due to its strong influence on effort, as well as a subset of the productivity factors. An alternative approach was also investigated, namely using the leaps and bounds algorithm [34]. This is an efficient algorithm for performing regressions for all possible combinations of the 15 productivity factors (size was always included). The model with the largest adjusted R^2 value was selected. Both approaches gave similar results in terms of prediction accuracy on the test data set. We therefore selected the model from the leaps and bounds algorithm since it performs an exhaustive search.

The final model is summarized in Table 2. We refer to this model as the *baseline model* since it has been developed with the complete training data set (i.e., no missing values).

Variable (log)	**Parameter Value**	**Standard Error**	**t value**	**Pr (> I t I)**
Intercept	2.687	0.558	4.813	<0.0001
System size (function points)	0.9104	0.08565	10.629	<0.0001
Customer participation	0.2413	0.19139	1.261	2.095×10^{-1}
Use of tools	-0.3392	0.27905	-1.216	2.263×10^{-1}
Logical complexity of software	0.5274	0.21987	2.399	1.782×10^{-2}
Requirements volatility	0.7223	0.21949	3.291	1.274×10^{-3}
Quality requirements	0.3061	0.25774	1.188	2.370×10^{-1}
Application knowledge of staff	-0.3528	0.14897	-2.368	1.930×10^{-2}

Table 2: Baseline model parameters. The adjusted R^2 is 0.605, and the F test of all parameters equal to zero produced a p value <0.0001.

We used the condition number of Belsley et al. [11] as an indicator of collinearity in this model. It was lower than the threshold of 30, and hence we can be confident that there are no multicollinearity problems in this model.

The baseline model has parameters whose signs are in the expected direction. Perhaps the user participation effect requires further explanation. It would be expected that increased user participation would lead to higher effort since interaction effort with the users increases.

3.5 Scale Type Assumptions

According to some authors, one of the assumptions of the OLS regression model is that all the variables should be measured at least on an interval scale [13]. This assumption is based on the mapping originally developed by Stevens [88] between scale types and "permissible" statistical procedures. In our context, this raises two questions. First, what are the levels of our measurement scales? Second, to what extent can the violation of this assumption have an impact on our results?

Our productivity factors utilized a single item each. In practice, single item measures are treated as if they are interval in many instances. For example, in the construction and empirical evaluation of the User Information Satisfaction instrument, inter-item correlations and principal components analysis are commonly performed [43].

It is also useful to note a study by Spector [84] that indicated that whether scales used have equal or unequal intervals does not actually make a practical difference. In particular, the mean of responses from using scales of the two types do not exhibit significant differences, and that the test-retest reliabilities (i.e., consistency of questionnaire responses when administered twice over a period of time) of both types of scales are both high and very similar. He contends, however, that scales with unequal intervals are more difficult to use, but that respondents conceptually adjust for this.

Given the proscriptive nature of Stevens' mapping, the permissible statistics for scales that do not reach an interval level are distribution-free (or nonparametric) methods (as opposed to parametric methods, of which OLS regression is one) [82]. Such a broad proscription is viewed by Nunnally as being "narrow" and would exclude much useful research [69]. Furthermore, studies that investigated the effect of data transformations on the conclusions drawn from parametric methods (e.g., F ratios and t tests) found little evidence supporting the proscriptive viewpoint [55][54][6]. Suffice it to say that the issue of the validity of the above proscription is, at best, debatable. As noted by many authors, including Stevens [88], the basic point is that of pragmatism: useful research can still be conducted even if, strictly speaking, the proscriptions are violated [13][35][89]. A detailed discussion of this point and the literature that supports our argument is given in [17].

3.6 Evaluative Measures

In order to evaluate the impact of MDTs, we define two different evaluative measures:[10] magnitude of relative error (MRE) and prediction at level l (PRED(l)) [25]. These are calculated from the model developed using the training data set and evaluated on the test data set.

The MRE is defined as:

$$\mathrm{MRE}_i = \frac{|\text{Actual Effort}_i - \text{Predicted Effort}_i|}{\text{Actual Effort}_i} \times 100 \qquad \textbf{Eqn. 2}$$

The MRE value is calculated for each observation i whose effort is predicted. The aggregation of MRE over all predicted observations, N, was achieved by taking the median of the MRE (MdMRE) over N observations.[11] It provides the percentage error in the estimate.

A complementary criterion that is also used is the prediction at level l, $PRED(l) = \frac{k}{N} \times 100$, where k is the number of observations where MRE is less than or equal to l. For our study we set l to 25%. The Pred25 provides the percentage of observations whose effort estimates were within 25% error.

[10] A third potential evaluative measure is concerned with consistency of estimation. This is defined as the correlation between the estimated and the actual effort, and has been used in a number of previous studies [67][2][49]. The logic of using this measure is that the existence of consistency, even if it is consistency in under or overestimation, project managers would be able to easily adjust for that using say a constant multiplier and the estimation model would still be of value. However, it was shown in a recent report [21] that adjusting consistent underestimates with a constant multiplier will increase accuracy but dramatically increase variability, and will reduce the acccuracy of overestimates. In our simulation we, therefore, focus only on accuracy of prediction.

[11] An implicit assumption in using MRE as a measure of predictive accuracy is that the error is proportional to the size of the project. For example, a 15 person-month overestimate for a 15 person-month project is more serious than for a 100 person-month project. On the other hand, Miyazaki et al. have criticised the MRE measure because it penalizes overestimates more than underestimates [66], and propose a new measure to alleviate this. However, Shepperd and Schofield [83] note that the proposed Myiazaki measure of accuracy is effectively two distinct measures that should not be combined. We therefore utilize the commonly used MRE as no alternatives have enjoyed acceptance within the software engineering community.

3.7 Simulation Approach

There are three general approaches that one can use to study the effect of dealing with missing values, two being simulations. The first possibility is to use an actual data set that had missing data, that were subsequently obtained through follow-up activities. This would allow the researcher to compare the performance of models using the data set with missing values after an MDT is applied, and the complete data set after follow-up. This is the approach used in the study by Cox and Folsom [26]. However, it is rare in practice to have such a data set.

The second is a Monte Carlo simulation. Under this approach, one constructs artificial data sets whose variables have known distributions and known inter-correlations. Then one creates missing values in these artificial data sets following various missingness schemes that one wants to study. Subsequently, different MDTs are applied and their performance evaluated in comparison to the known characteristics of the artificial data generation.

An example of such a Monte Carlo simulation is [78]. The analysis considered only cases where observations were partially incomplete, not those in which there was a complete non-response. Techniques considered were listwise deletion, pairwise deletion, mean imputation, regression imputation, and hot-deck imputation. The complete data set used for the analysis was generated from a population correlation matrix. Various statistics were generated based on the complete data set to be compared with those generated after inducing missing values and application of the various MDTs. Missing values were imputed in 10, 20 and 30% of the observations randomly. The effectiveness was based on calculation of root mean squared error and absolute error.

A potential disadvantage of the Monte Carlo approach is that one cannot be certain that the population characteristics that are being simulated are congruent with real data sets. Therefore, one would not know the extent to which the conclusions can be generalized to actual practical situations. This approach can be mitigated by basing the population parameters on values obtained from previous studies, such as in the Monte Carlo simulation performed by Kim and Curry [50].

An alternative approach is to use an actual data set relevant to the problem. This data set would have to be complete (i.e., no missing data). One would then create missing values that follow a known pattern. Subsequently, different MDTs are applied and their performance evaluated in comparison to the results that would be obtained had there been no missing data.

An example of this kind of approach is the study of [53]. This study analysed the performance of five MDTs for dealing with data missing nonrandomly, namely, listwise deletion, pairwise deletion, mean imputation, simple regression imputation, and multiple imputation. Performance of each technique was based on parameter estimates of a two-predictor regression model. Bootstrap samples were taken from actual field data and missing values were assigned to one variable, based on the value of that variable.

The approach we have followed in our study is in the final category. We used the Experience Database as our complete data set. The overall simulation approach is summarized in Figure 1.

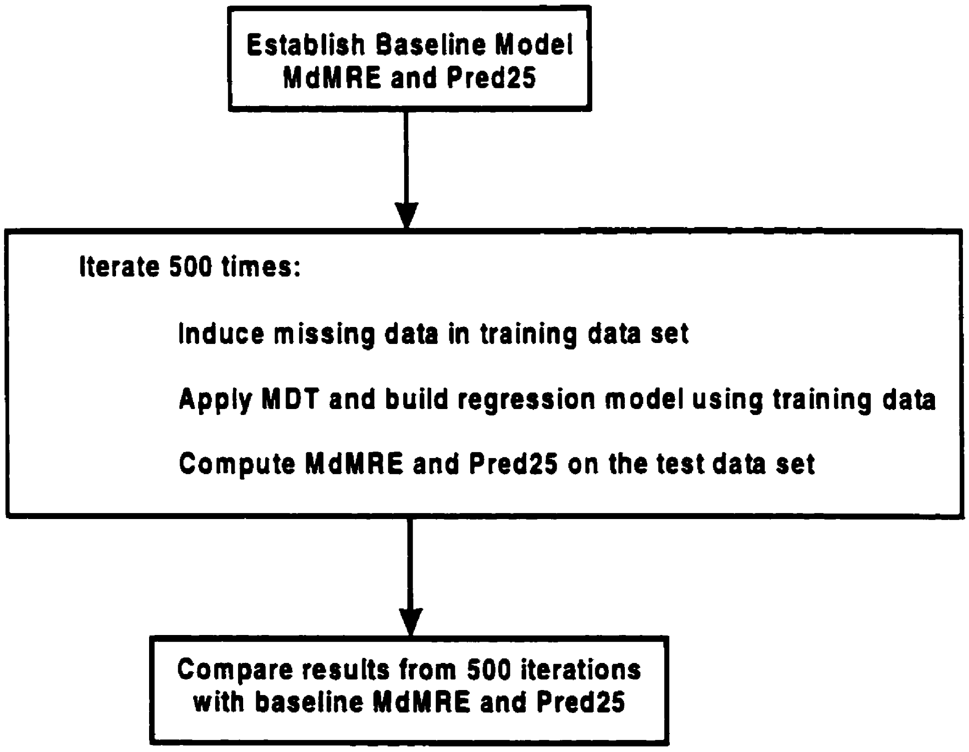
Establish Baseline Model
MdMRE and Pred25
Iterate 500 times:
Induce missing data in training data set
Apply MDT and build regression model using training data
Compute MdMRE and Pred25 on the test data set
Compare results from 500 iterations
with baseline MdMRE and Pred25

Figure 1: Summary of simulation approach.

The baseline model was already described in Section 3.4. Using the baseline model, we predicted the effort on the test data set and found the MdMRE to be 47% and the Pred25 to be 24%. Note that the calculation of these measures is based on the original units, not the log transformed. The exponential function is applied to the predicted effort in order to convert it back to its original unit. These serve as our baseline MdMRE and baseline Pred25 and will be referred to as $MdMRE_{baseline}$ and $Pred25_{baseline}$ respectively.

We describe the remaining steps of our simulation approach below.

3.8 Simulating Missing Data

We consider four parameters in simulating the missing data:

- The percentage of observations with missing data
- The number of variables that have missing data (we only consider missing values on the productivity factors)
- The missing data mechanism
- The pattern of missing data

Five different percentages of missing values were simulated (5,10,15,25,40). It is generally accepted that data sets with more than 40% missing data are not useful for detailed analysis.

Since our final model has six productivity factors, we consider missing values on one up to all six variables.

Three missing data mechanisms were evaluated: missing completely at random (MCAR), missing at random (MAR), and non-ignorable missingness.

The implementation of the MCAR mechanism to impute missing data was straightforward. Missing values were imputed for the relevant variable completely at random.

For MAR, we simulated the situation where missing values depend on the size of the project. The implementation of the MAR mechanism to induce missing data, involves first ordering the observations according to project size. Missing values are induced with biases for both small and large project sizes. Once the data set is ordered, it is split up into quintiles with the first and last quintiles containing 28 observations and the middle three containing 29 observations. Each quintile will receive different percentages of missing values. The total percent of missing values is divided by 10 to get a value for k. The separate quintiles will be induced with 4k, 3k, 2k, k, and 0k missing values. For example, if the total amount of missing data to be created is 10%, then each quintile will be induced with 4, 3, 2, 1, and 0% missing values. It is important to note that within each quintile, missing values are induced randomly.

For non-ignorable missingness, we simulated missing values that depend on the particular variable in question. Implementation was identical to MAR except that the observations were ordered by the variable to be induced with missing data. Quintiles were formed and missing values were created as described for MAR. Missing values are induced with biases for both low and high values for each variable.

Two patterns of missing data were simulated. The first is univariate missing data, whereby the values are missing on each variable separately according to one of the mechanisms described below. Each independent variable was induced with missing values according to the pattern shown in Figure 2 (left panel).

The second is monotone missing data, whereby the variables can be ordered in terms of their extent of missingness. This means that all observations with missing data for X_2 also have missing data for X_1, but the reverse is not always true [61].

The implementation of monotone missing data is illustrated with two variables. It easily generalizes to the case of more than two variables. The monotone pattern of missingness is shown in Figure 2 (right panel). The first variable is induced the same way as for the univariate case. Next, half of the observations that contain missing values at the first variable will have their second variable induced with a missing value. For the MCAR mechanism of missing data, half of the total number of observations containing missing data will have missing data on two variables. For the MAR mechanism, half of the observations *for each quintile* will contain missing data for both variables. It follows that although the total amount of missing values being induced is greater in the monotone case as compared to the univariate case, the total number of observations that contain *at least* one missing value will be the same.

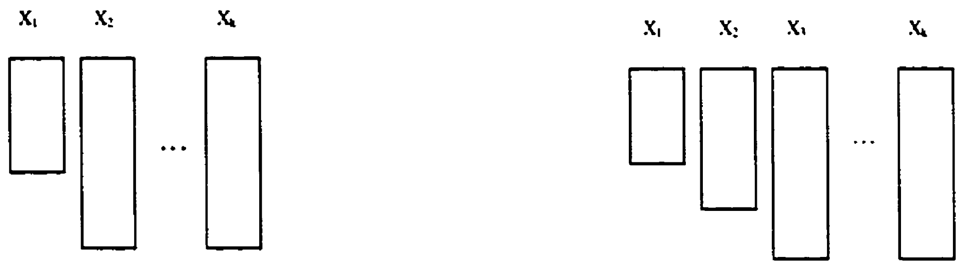

Figure 2: Univariate and monotone patterns of missing data.

In total, then, we defined 29 340 missing data schemes.[12] For each missing data scheme we applied each of the 10 MDTs described below. This gives a total of 293 400 different study points in the simulation. For each study point and MDT combination we performed the simulation 500 times.[13] For each of the 500 iterations, we built a new ordinary least squares regression model. Subsequently, we performed a prediction on the test set and computed the evaluative measures.

3.9 Summary Measures

For each run of the simulation we computed the MdMRE and Pred25. From these numbers we have to produce summary measures. We consider two types of summary measures: *bias* and *precision*. Bias tells us how different the results are from those that would be obtained had there been no missing data (i.e., the baseline). Precision informs us about the dispersion or variability.

[12] At the outset this may seem to be a large number. However, for the monotone patterns having more than one productivity factor with missing values, we must consider all possible permutations of the productivity factors with missing values.

Bias was computed from the 500 simulations as follows:

$$MRE_{StudyPoint} = \underset{1 \le i \le 500}{median} \left| MdMRE_i - MdMRE_{baseline} \right| \qquad \textbf{Eqn. 3}$$

For the MRE, we computed the median absolute difference across all 500 simulations. This provides us a measure of the bias of the MDT compared to the MRE that would be obtained had there been no missing data.

$$Pred25_{StudyPoint} = \underset{1 \le i \le 500}{median} \left| Pred25_i - Pred25_{baseline} \right| \qquad \textbf{Eqn. 4}$$

For the Pred25, we computed the median absolute difference across all 500 simulations. This provides us a measure of bias of the MDT compared to the Pred25 that would be obtained had there been no missing data.

Both of the above measures express bias in terms of the change in percentage. For example, if $MRE_{StudyPoint}$ is 5, that means that overall, using the MDT will be different from the baseline MdMRE by 5 MRE percentage points. Thus, if the baseline MdMRE is 46%, then the overall MDT MdMRE could be 51%. The same applies to Pred25.

Precision was evaluated using the inter-quartile range as follows:

$$\underset{1 \le i \le 500}{IQR} \left(MdMRE_i - MdMRE_{baseline} \right) \qquad \textbf{Eqn. 5}$$

$$\underset{1 \le i \le 500}{IQR} \left(Pred25_i - Pred25_{baseline} \right)$$

We present the precision results in the form of box and whisker plots (for an overview of these types of plots, please see Appendix A).

3.10 MDTs Evaluated

The implementation of the MDTs that we studied is described below.

3.10.1 Listwise Deletion

For listwise deletion, observations containing missing data are ignored. The regression model is built using only observations that contain no missing values.

[13] We randomly selected some of the study points and performed the simulations with 1000 iterations. Our conclusions were not affected, and in fact even the summary values obtained were very similar.

3.10.2 Mean Imputation

After the database has been induced with missing values, each missing value is replaced by the mean calculated for all observed cases for that variable. For the univariate case, the mean is calculated for the variable that contains missing values and this value is imputed for all observations that contain a missing value. For the multivariate case, multiple means are calculated, one for each variable that contains missing values. The same value will be imputed for all observations.

Once all the values have been imputed, the regression model is generated. Unlike listwise deletion, no observations are lost and the regression model is based on the complete data set containing imputed values.

3.10.3 Hot-Deck Imputation

Once the missing observations have been created in the database, each missing value is imputed with a donor value picked from the one of the observations without missing data. We now present some notation to help explain the different types of hot-deck imputation that we implemented.

We divide the data set into those observations with missing values, the missing set, and those observations without missing values, the complete set. Let x_i be the vector of all variables measured on the i^{th} observation in the missing set, and x_{ij} be the value of the j^{th} variable measured on the observation. Further, let c_k be the vector of all variables measured on the k^{th} observation in the complete set, and let c_{kj} be the value of the j^{th} variable measured on that observation.

We constructed a numeric hot-deck function. In setting up a hot-deck function, different hot-deck parameters can potentially have a nonnegligible impact on its performance:

- In calculating a distance between an observation in the missing set and the complete set, variables containing values that are much larger than those in other variables will dominate the distance (e.g., size has a much larger range than the productivity factors). Standardization will prevent those variables from having a larger influence (in effect, treating all variables as being equally important). Which standardization technique should be used ?

- There are multiple possible distance measures that can be employed. Which one should be used?

Below we describe these parameters and the particular values that we evaluated, including justifications for the selections made.

First, we consider three different standardization approaches: z-score, mean absolute, and Z_5, in addition to the case of no standardization.

A simulation study in the area of cluster analysis, found one type of standardization scheme to be superior in recovering the underlying cluster structure under different conditions including error free data, and data with noise and with outliers [65]. However, the parameters of that simulation are not necessarily reflective of software cost data, and therefore this standardization scheme, referred to as Z_5, is evaluated here:

$$Z_5 = \frac{c_{kj} - \min(c_k)}{\max(c_k) - \min(c_k)} \qquad \textbf{Eqn. 6}$$

As will be noted, this has a non-robust denominator in that it will be easily affected by even a single outlier. A more traditional standardization scheme that makes the unit of the variables the sample standard deviation is the z-score:

$$Z - score = \frac{c_{kj} - \frac{1}{n}\sum_k c_{kj}}{\sqrt{\frac{1}{n-1}\sum_k \left(c_{kj} - \frac{1}{n}\sum_k c_{kj}\right)^2}} \qquad \textbf{Eqn. 7}$$

A more robust approach for standardization is to use the *mean absolute deviation* [91] instead of the standard deviation in the denominator. This is robust because the deviations are not squared, therefore atypical points do not exaggerate it [48]. Robustness is desirable because software measurements typically have a few extreme values that may exert strong influence on the analysis results (e.g., see [68]).

$$\text{Mean Absolute} = \frac{c_{kj} - \frac{1}{n}\sum_k c_{kj}}{\frac{1}{n}\sum_k \left| c_{kj} - \frac{1}{n}\sum_k c_{kj} \right|} \qquad \textbf{Eqn. 8}$$

For each observation that contains a missing value, a distance is determined between it and all observations in the complete set. A multitude of different distance functions can reasonably be used in a hot-deck function. We limit ourselves to the common distance functions in other disciplines. We also limit ourselves to the context where the predictor variables are continuous, since we treat them this way in the regression model.

Kaufman and Rousseeaw [48] define the Minkowski distance as follows:

$$d_{ik} = \left(\sum_j \left| c_{kj} - x_{ij} \right|^q \right)^{\frac{1}{q}} \qquad \textbf{Eqn. 9}$$

The most commonly used distance functions for continuous variables are the Euclidean and Manhattan distances.

The Euclidean distance between a component i in the missing data set and a component k in the complete set is given by (i.e., $q = 2$):

$$Euclidean_{ik} = \sqrt{\sum_j \left(c_{kj} - x_{ij} \right)^2} \qquad \textbf{Eqn. 10}$$

The Manhattan distance is defined by (i.e., $q = 1$):

$$Manhattan_{ik} = \sum_j \left| c_{kj} - x_{ij} \right| \qquad \textbf{Eqn. 11}$$

A priori, there is no compelling reason to prefer one distance function over another, and therefore, it is prudent to evaluate them empirically.

Using both functions, a value is determined for the distance between the observation containing a missing value and all observations in the hot-deck. The distance function takes as input the size variable and all productivity factors except the one that is missing. The missing value is imputed with the variable from the observation in the hot-deck that has the smallest distance from the observation containing the missing value.

Once all missing values have been imputed with values obtained from the hot-deck, the data set is complete and the regression model is generated.

In the case of monotone missing data with two variables, the data set is divided into three parts, observations that contain one missing value (M1), observations that contain two missing values (M2), and observations that contain no missing values (complete set). A given percentage of observations will contain missing values on one independent variable, say X_1. In half of these cases the observation will contain a missing value at another independent variable, say X_2. This can be easily generalized to the case with more than two missing variables.

Values are imputed into M1 observations in the same way as for the univariate case. For M2 observations, two values from the same hot-deck observation with the smallest distance to each observation are imputed. It is important to note that the distance function for the M2 observations will be based on one less independent variable than for the M1 observations.

Once all missing values have been imputed with values obtained from the complete set, there are no more missing values, and the regression model is generated.

Once the model has been derived, it is then used to predict the effort for each software cost project in the bank database.

3.11 Summary

In this chapter, we have described the overall simulation approach. In total we had 293 400 study points that were simulated, each with 500 iterations. This includes all techniques for generating missing values combined with the MDTs studied. Such a comprehensive simulation should provide us with a reasonable picture of the strengths and weaknesses of each MDT under different missing data scenarios. Furthermore, the fact that the simulation is based on an actual data set should give us confidence in the applicability of the results within the same application domain.

Chapter 4: Results

We first provide some descriptive statistics for the data set. Then we present the results for each MDT in turn. Due to the large amount of results that can be presented, we are only able to show results that demonstrate the patterns that were observed.

Table 3 presents the abbreviations of each productivity factor that will be used in future figures and tables.

Abbreviation	**Productivity Factor**
CP	Customer Participation
UT	Use of Tools
SC	Software Complexity
RV	Requirements Volatility
QR	Quality Requirements
SK	Staff Application Knowledge

Table 3: Productivity factors with corresponding abbreviations

Evaluative results are given in terms of the $MRE_{StudyPoint}$ and $Pred25_{StudyPoint}$, as described in Eqn. 3 and Eqn. 4, calculated for all 500 iterations. In some instances, we also present box and whisker plots for MdMRE and Pred25 differences from the baseline values to show the extent of variation.

The results are grouped into the four combinations of MCAR and MAR vs. non-ignorable missingness, and univariate vs. monotone patterns. The exception is listwise deletion because there is no difference between univariate and monotone patterns.

4.1 Descriptive Statistics

Table 4 summarizes the descriptive statistics for system size (FP) and project effort (person-hours: ph). The table shows the results for the whole database, the *test* data set, and the *train* data set. Projects in the test data set are from the banking domain and generally have a higher effort than those in the whole database. The breakdown of projects per organization type for the whole data base is 38% banking, 27% insurance, 19% manufacturing, 9% wholesale, and 7% public administration. Table 5 summarizes the descriptive statistics for each of the productivity factors. Figure 3 and Figure 4 illustrate the proportions of projects for different application types and target platforms for the whole database and the test and train data sets. It can be seen that the proportions of application type and target platform are similar for the whole database and the test and train data sets.

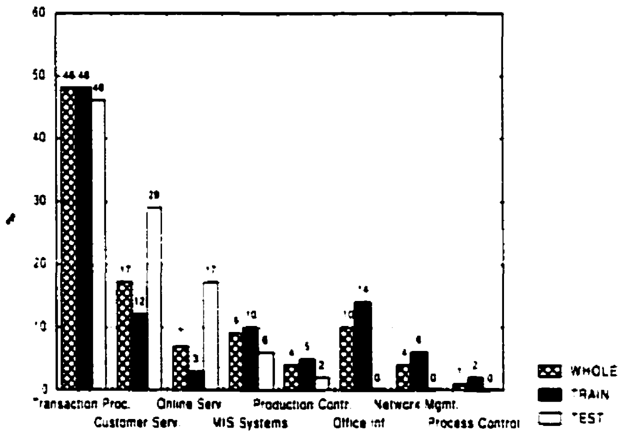

Figure 3: Distribution of projects by application type.

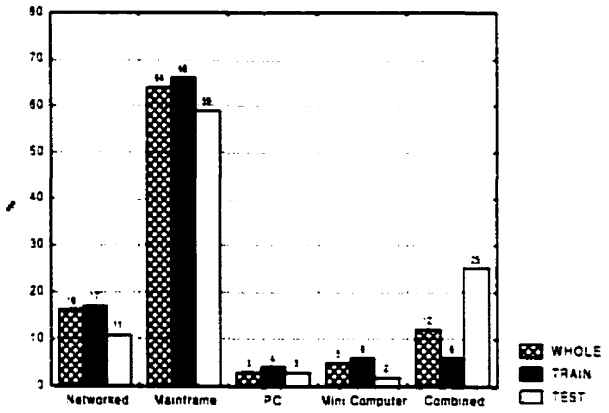

Figure 4: Distribution of projects by target platform.

	Test data set		Train data set		Whole data set	
	Size (FP)	Effort (ph)	Size (FP)	Effort (ph)	Size (FP)	Effort (ph)
min	10	583	10	250	10	250
mean	109.2	8109.5	122.8	5999.6	118.7	6644.9
max	558	63694	487	51100	558	63694
st. dev	111.3	10453.9	97.2	7732.4	101.6	8684.3
obs	63	63	143	143	206	206

Table 4: Descriptive statistics for system size and effort.

	Test data set productivity factors						Train data set productivity factors					
	CP	UT	SC	RV	QR	SK	CP	UT	SC	RV	QR	SK
min	1	1	1	2	2	1	1	1	1	1	1	1
median	3	3	3	4	4	3	3	3	3	3	3	3
mean	3.1	2.9	3.3	3.8	4	3.1	3.2	3.1	3.2	3.1	3.2	3
max	5	4	5	5	5	5	5	5	5	5	5	5
st.dev	1.02	0.7	0.9	0.95	0.75	0.95	0.92	0.68	0.9	0.84	0.8	1.09
obs	63	63	63	63	63	63	143	143	143	143	143	143

Table 5: Descriptive statistics for system productivity factors

4.2 Listwise Deletion

The following section contains the results for the listwise deletion MDT.

4.2.1 MCAR and MAR Mechanisms

Table 6 shows the $MRE_{StudyPoint}$ and $Pred25_{StudyPoint}$ values[14] for listwise deletion under the MCAR and MAR mechanisms. Here, the values are only for the univariate pattern. The results for the monotone pattern are not presented because with listwise deletion they are, by definition, exactly the same as in the univariate case.

[14] For a description of the summary measures, see Section 3.9.

Mech.	Bias	Miss -ing	5%		10%		15%		25%		40%	
			MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25
MCAR	n/a	CP	0.92	1.6	1.1	1.6	1.4	1.6	1.6	1.6	2.3	3.2
		UT	0.83	1.6	1.1	1.6	1.4	1.6	1.8	1.6	2.1	3.2
		SC	0.82	1.6	1.2	1.6	1.4	1.6	1.7	1.6	2.3	1.6
		RV	0.85	1.6	1.2	1.6	1.3	1.6	1.8	1.6	2.3	3.2
		QR	0.88	1.6	1.1	1.6	1.3	1.6	1.8	1.6	2.4	3.2
		SK	0.90	1.6	1.1	1.6	1.2	1.6	1.7	3.2	2.6	3.2
MAR	large	CP	0.81	1.6	1.0	1.6	1.2	1.6	1.6	1.6	1.8	3.2
		UT	0.76	1.6	1.0	1.6	1.2	1.6	1.5	3.2	1.9	3.2
		SC	0.75	1.6	1.1	1.6	1.2	1.6	1.5	1.6	2.0	3.2
		RV	0.75	1.6	1.0	1.6	1.3	1.6	1.6	3.2	2.1	3.2
		QR	0.84	1.6	1.1	1.6	1.2	1.6	1.5	1.6	1.8	3.2
		SK	0.86	1.6	1.0	1.6	1.2	1.6	1.5	3.2	1.9	3.2
	small	CP	0.85	1.6	1.3	1.6	1.4	1.6	1.6	1.6	2.4	1.6
		UT	0.93	1.6	1.2	1.6	1.4	1.6	1.6	1.6	2.5	1.6
		SC	0.92	1.6	1.1	1.6	1.4	1.6	1.7	1.6	2.3	1.6
		RV	0.89	1.6	1.1	1.6	1.2	1.6	1.8	1.6	2.3	1.6
		QR	0.88	1.6	1.1	1.6	1.2	1.6	1.7	1.6	2.3	1.6
		SK	0.94	1.6	1.2	1.6	1.4	1.6	1.6	1.6	2.4	1.6

Table 6: Listwise deletion for univariate missing data on all productivity factors. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

A number of conclusions can be drawn from this table:

- The bias[15] increases as the extent of missing data increases, for both the MCAR and MAR mechanisms.
- For the MAR mechanism, there is no striking difference between biases for large vs. small projects.
- There are no striking differences between the MAR and MCAR mechanisms.
- The Pred25 value peaks at a 3.2% difference for 40% missing data. In our context this would mean that 2 more out of the 63 projects in the test set have an MRE greater than 25% when using listwise deletion.
- MdMRE peaks at a difference of 2.6% for 40% missing data. This means that the estimation accuracy will be different from the case where complete data is collected by an average of only 2.6%.

[15] For a description of this summary measure, see Section 3.9.

- For low percentages of missing data, up to 15%, listwise deletion performs remarkably well, with a negligible bias in its performance. At higher extents of missing data, the bias in terms of MdMRE and Pred25 is still rather small in absolute terms. Since these results hold even for 6 productivity factors with missing values, they indicate that listwise deletion is a reasonable approach to use with MCAR and MAR missing data.

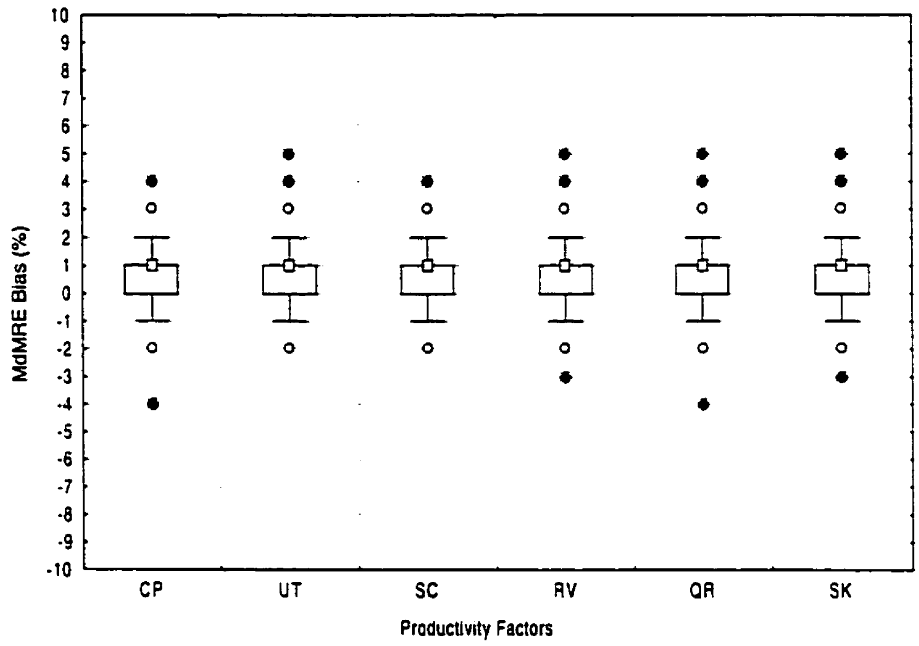

Figure 5: Listwise deletion on each productivity factor with 5% missing data, using MCAR mechanism.

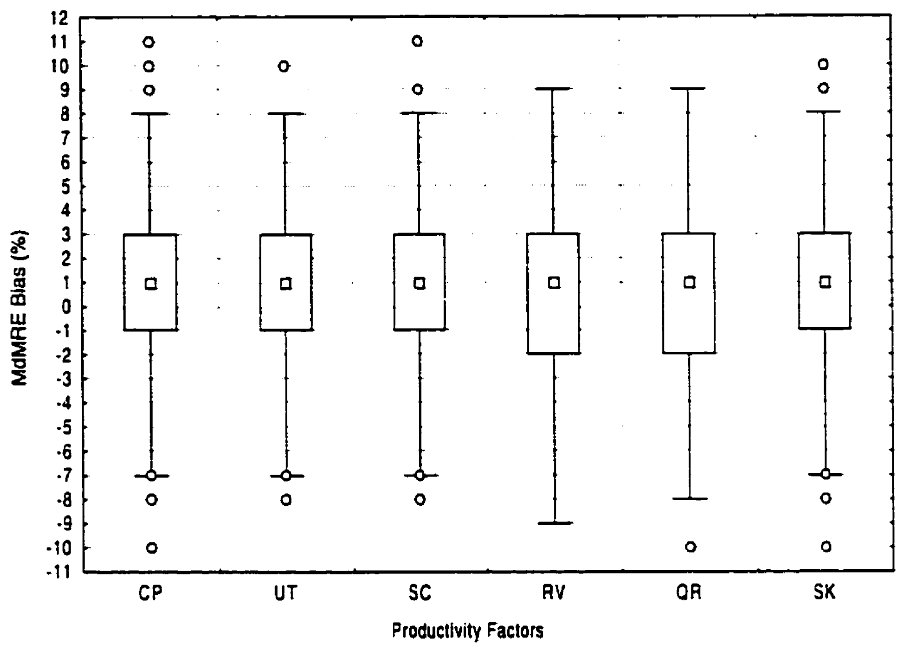

Figure 6: Listwise deletion on each productivity factor with 40% missing data, using MCAR mechanism.

Figure 5 shows box and whisker plots[16] of the MdMRE bias when using listwise deletion for 5% missing values under the MCAR mechanism for all six productivity factors. Contrast this with Figure 6, which shows the MCAR results for 40% missing data. It will be noticed that the variation in MdMRE is larger for 40% missing data. Thus, the variability in the performance of listwise deletion deteriorates as the extent of missing data increases. However, it should be recalled that this deterioration is slight, in that variability increases by about 2-4%. A confirmatory pattern can be seen in Figure 7, where we see that the MdMRE bias from using the listwise deletion MDT for one of the productivity factors increases as the extent of missing data increases. The same pattern was observed for all productivity factors and for MAR.

[16] Box and whisker plots present precision results, as described in Section 3.9.

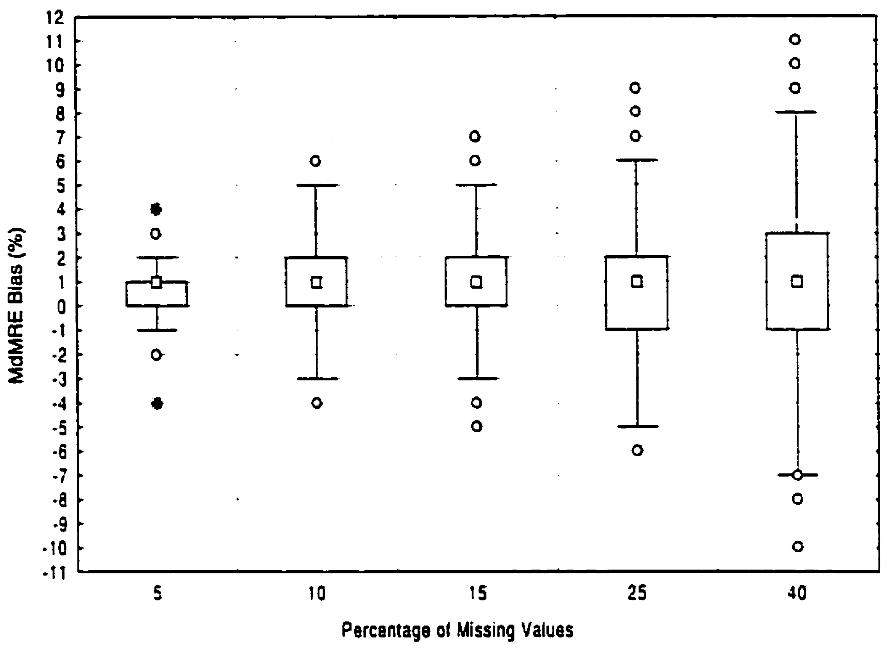

Figure 7: Summary of MdMRE results - listwise deletion for Customer Participation productivity factor under MCAR.

Figure 8 shows a similar increase in variability for the Pred25 measure, indicating that listwise deletion provides less stable results as the extent of missing data reaches 40%. A similar pattern was observed for the other variables.

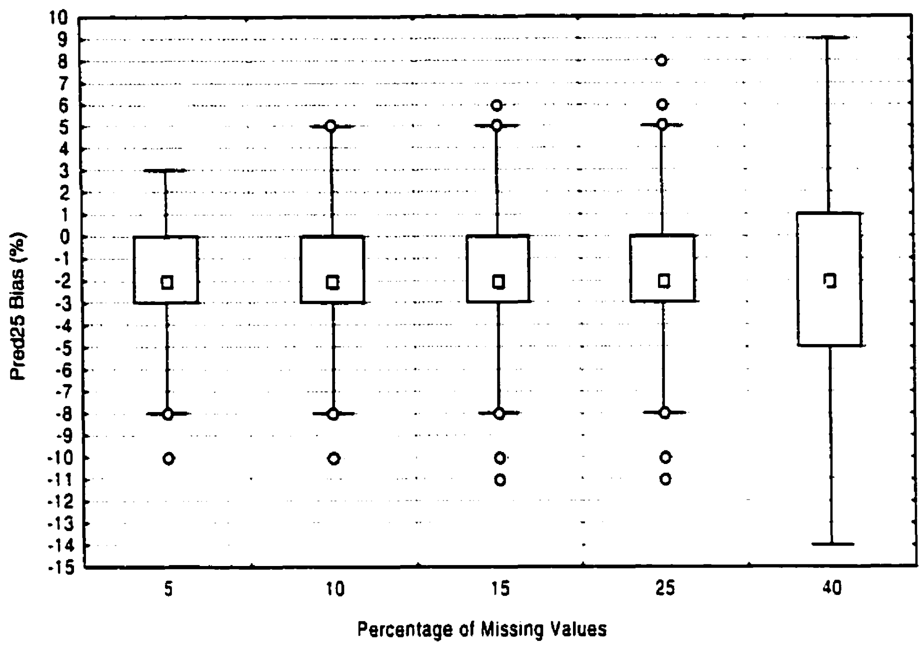

Figure 8: Summary of Pred25 results - listwise deletion for Customer Participation productivity factor under MCAR.

4.2.2 Non-ignorable Missingness Mechanism

		5%		10%		15%		25%		40%	
Missing	**Bias**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
CP	low	0.82	1.6	0.84	1.6	1.1	1.6	1.5	1.6	1.6	3.2
	high	0.77	1.6	1.3	1.6	1.4	3.2	1.7	3.2	2.0	4.8
UT	low	0.86	1.6	1.1	1.6	1.2	1.6	1.5	1.6	1.8	3.2
	high	0.82	1.6	1.0	1.6	1.2	1.6	1.6	1.6	2.5	1.6
SC	low	0.88	1.6	1.2	1.6	1.5	3.2	2.4	3.2	3.5	3.2
	high	0.75	1.6	1.0	1.6	1.2	1.6	1.7	1.6	2.6	1.6
RV	low	0.72	1.6	1.1	1.6	1.5	1.6	2.0	1.6	2.9	1.6
	high	0.72	1.6	1.0	1.6	1.2	3.2	1.6	3.2	2.2	3.2
QR	low	0.86	1.6	1.0	1.6	1.4	1.6	1.5	1.6	2.2	1.6
	high	0.77	1.6	1.1	1.6	1.2	1.6	1.7	3.2	2.5	3.2
SK	low	0.96	1.6	1.2	1.6	1.2	1.6	1.7	1.6	2.2	3.2
	high	0.90	1.6	1.1	1.6	1.3	1.6	1.4	3.2	2.0	3.2

Table 7: Listwise deletion MDT for univariate missing data on all productivity factors using non-ignorable missing data mechanism. It is also indicated whether there is more missingness for low vs. high values.

Table 7 shows the results for non-ignorable missingness. Here we can observe that:

- The performance of listwise deletion deteriorates as the extent of missingness increases.

- There is no systematic pattern of either greater missingness in low vs. high values being better.
- Up to 15% misisng data, listwise deletion still has negligible biases.
- The results, when compared to MCAR and MAR, are not much worse, indicating that the bias performance of listwise deletion remains the same irrespective of the missingness mechanism.

The pattern observed above regarding the variability in our accuracy measures was also observed for the non-ignorable missingness study points.

In general, we can conclude that listwise deletion has a remarkably small bias, even at high percentages of missing data. The disadvantage of listwise is that the variability in the accuracy tends to be large at higher percentages of missing data.

4.3 Mean Imputation

The following section contains the results for the mean imputation MDT.

4.3.1 MCAR and MAR Mechanisms, Univariate Case

Table 8 shows the results for mean imputation for univariate missing data under the MCAR and MAR mechanisms. The MAR results are for higher biases on both small and large projects.

		Miss	5%		10%		15%		25%		40%	
Mech.	Bias	-ing	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25
MCAR	n/a	CP	0.20	0	0.34	0	0.39	0	0.60	1.6	0.77	1.6
		UT	0.16	0	0.28	1.6	0.37	1.6	0.60	1.6	0.82	1.6
		SC	0.29	0	0.41	0	0.53	1.6	0.60	1.6	0.77	1.6
		RV	0.36	1.6	0.64	1.6	0.75	1.6	1.0	3.2	1.5	3.2
		QR	0.53	0	0.76	1.6	0.98	1.6	1.6	1.6	1.8	3.2
		SK	0.36	0	0.67	0	0.84	0	1.0	1.6	1.3	1.6
MAR	large	CP	0.23	0	0.32	0	0.38	0	0.60	1.6	0.75	1.6
		UT	0.11	0	0.25	1.6	0.36	1.6	0.80	1.6	1.4	1.6
		SC	0.23	0	0.40	0	0.62	1.6	0.70	1.6	0.61	3.2
		RV	0.40	0	0.63	1.6	0.75	3.2	1.0	3.2	1.2	4.8
		QR	0.57	0	0.77	0	1.0	1.6	1.5	1.6	2.0	1.6
		SK	0.40	0	0.68	0	0.80	0	1.1	1.6	1.3	1.6
	small	CP	0.20	0	0.32	0	0.41	0	0.60	0	0.66	1.6
		UT	0.16	0	0.34	0	0.43	0	0.50	1.6	0.60	1.6
		SC	0.28	0	0.47	1.6	0.54	1.6	0.70	1.6	1.0	3.2
		RV	0.44	1.6	0.57	1.6	0.80	3.2	1.5	3.2	2.1	4.8
		QR	0.55	0	0.85	1.6	1.0	1.6	1.6	3.2	2.1	4.8
		SK	0.37	0	0.61	0	0.85	0	1.0	1.6	1.2	1.6

Table 8: Mean Imputation for univariate missing data on all productivity factors. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

A number of conclusions can be drawn from this table:

- The bias increases as the extent of missing data increases, for both the MCAR and MAR mechanisms.
- For the MAR mechanism, there is no striking difference between biases for large vs. small projects.
- There are no striking differences between the MAR and MCAR mechanisms.
- The Pred25 value peaks at a 4.8% difference for 40% missing data. In our context this would mean that 3 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation.
- MdMRE peaks at a difference of 2.1% for 40% missing data. This means that the estimation accuracy will be different from the case where complete data is collected by an average of only 2.1%.
- For low percentages of missing data, up to 25%, mean imputation performs remarkably well, with a negligible bias in its performance. At higher extents of

missing data, the bias in terms of MdMRE and Pred25 is still rather small in absolute terms.

Compared to listwise deletion, mean imputation tends to have a slightly smaller bias in terms of MdMRE for MAR and MCAR for univariate missing values. For the Pred25 results, its performance is slightly better up to 25% missing data, after which it is more difficult to see a difference from listwise deletion.

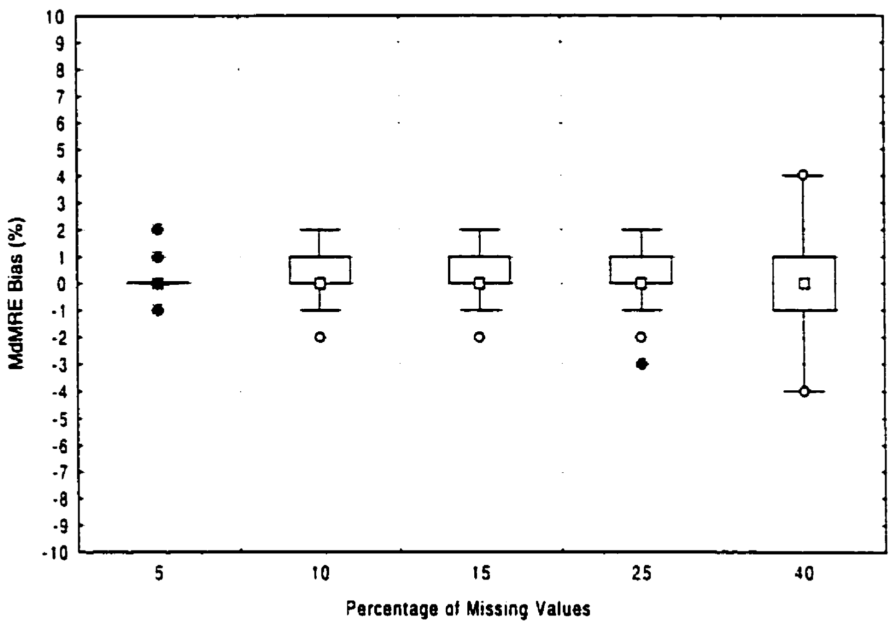

Figure 9: Summary of MdMRE results – mean imputation for Customer Participation productivity factor under MCAR.

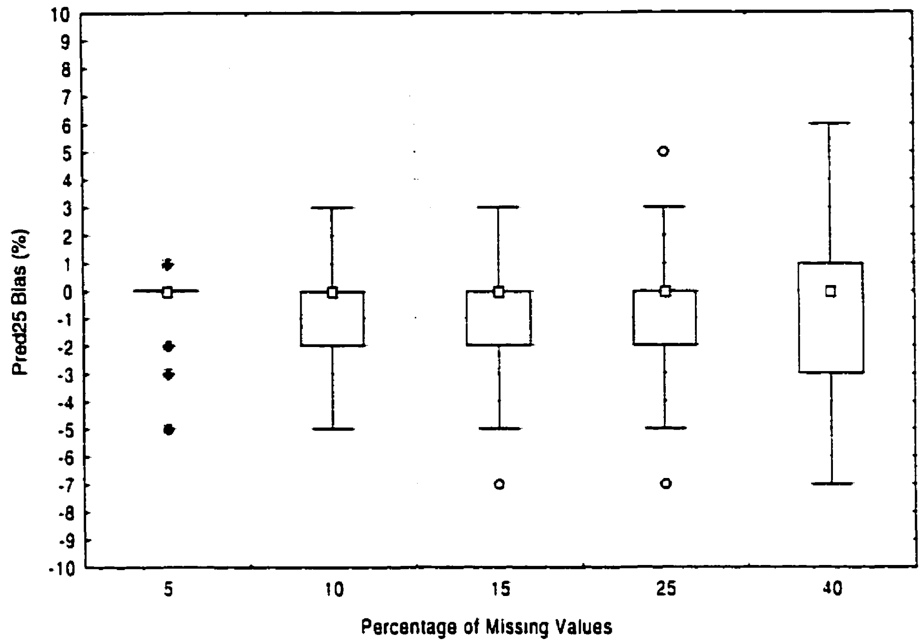

Figure 10: Summary of Pred25 results - mean imputation for Customer Participation productivity factor under MCAR.

Figure 9 and Figure 10 are box and whisker plots for one of the productivity factors. These show the MdMRE and Pred25 results as the missingness percentage increases under MCAR. As can be seen, the variability tends to increase for increased missingness. The same pattern was observed for other variables and under MAR. However, it will be noticed that the extent of variability tends to be smaller than for listwise deletion, indicating more stability in the bias of mean imputation.

In general, we can state that mean imputation under the MCAR and MAR mechanisms with univariate missing data performs rather well. While it does exhibit bias, this is rather small, and tends to be as good as or better than for listwise deletion. Furthermore, the variability of the bias tends to be smaller.

4.3.2 MCAR and MAR Mechanisms, Monotone Pattern

Table 9 shows the results for the MCAR and MAR mechanisms for the monotone pattern of missingness for two variables. The first productivity factor listed in the "Missing" column contains the larger amount of missing data (the conclusions do not change if the order is reversed, and therefore all possible permutations are not presented here).

Mech.	Bias	Miss-ing	5%		10%		15%		25%		40%	
			MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25
MCAR	n/a	CP,UT	0.25	0	0.43	0	0.54	1.6	0.69	1.6	0.96	3.2
		CP,SC	0.32	0	0.45	1.6	0.57	1.6	0.66	1.6	0.83	1.6
		CP,RV	0.37	1.6	0.47	1.6	0.68	1.6	0.81	3.2	0.95	3.2
		CP,QR	0.65	0	0.75	0	0.89	1.6	1.1	1.6	1.4	1.6
		CP,SK	0.35	0	0.49	0	0.50	0	0.63	1.6	0.68	1.6
		UT,SC	0.28	0	0.41	1.6	0.53	1.6	0.73	1.6	1.3	3.2
		UT,RV	0.33	1.6	0.47	1.6	0.54	1.6	0.72	1.6	1.1	3.2
		UT,QR	0.53	0	0.66	1.6	0.73	1.6	1.1	1.6	1.5	1.6
		UT,SK	0.26	0	0.40	0	0.43	1.6	0.56	1.6	0.85	1.6
		SC,RV	0.42	1.6	0.51	1.6	0.68	1.6	0.92	3.2	1.3	3.2
		SC,QR	0.44	0	0.63	1.6	0.73	1.6	1.0	3.2	1.2	3.2
		SC,SK	0.32	0	0.48	1.6	0.61	1.6	0.69	1.6	0.75	3.2
		RV,QR	0.66	1.6	0.87	3.2	1.2	3.2	1.6	4.8	1.9	4.8
		RV,SK	0.52	1.6	0.82	1.6	1.1	1.6	1.6	3.2	2.1	3.2
		QR,SK	0.69	0	0.87	1.6	0.91	1.6	1.3	1.6	1.8	1.6
MAR	large	CP,UT	0.29	0	0.43	0	0.48	1.6	0.92	1.6	1.0	1.6
		CP,SC	0.28	0	0.45	1.6	0.54	1.6	0.66	1.6	0.75	1.6
		CP,RV	0.29	0	0.49	1.6	0.69	1.6	0.75	1.6	0.93	3.2
		CP,QR	0.46	0	0.69	0	0.85	0	1.1	1.6	1.3	1.6
		CP,SK	0.26	0	0.52	0	0.56	0	0.61	1.6	0.64	1.6
		UT,SC	0.23	0	0.39	1.6	0.54	1.6	1.0	3.2	1.5	3.2
		UT,RV	0.24	1.6	0.49	1.6	0.69	3.2	1.0	3.2	1.8	4.8
		UT,QR	0.46	0	0.69	1.6	0.96	1.6	1.4	3.2	1.8	3.2
		UT,SK	0.21	0	0.41	1.6	0.55	1.6	0.93	1.6	1.5	3.2
		SC,RV	0.35	0	0.55	1.6	0.83	1.6	0.84	3.2	1.1	4.8
		SC,QR	0.45	0	0.64	1.6	0.74	1.6	1.1	1.6	1.5	3.2
		SC,SK	0.28	0	0.48	1.6	0.60	1.6	0.66	1.6	0.58	3.2
		RV,QR	0.54	1.6	0.86	1.6	1.1	3.2	1.8	3.2	2.0	4.8
		RV,SK	0.47	1.6	0.79	1.6	1.1	3.2	1.4	3.2	1.3	6.3
		QR,SK	0.56	0	0.77	0	1.1	1.6	1.4	1.6	1.9	1.6
	small	CP,UT	0.25	0	0.39	1.6	0.47	1.6	0.60	1.6	0.85	3.2
		CP,SC	0.32	0	0.42	1.6	0.49	1.6	0.62	1.6	0.82	3.2
		CP,RV	0.37	0	0.57	1.6	0.65	1.6	0.85	3.2	1.1	4.8
		CP,QR	0.55	0	0.75	1.6	0.93	1.6	1.0	1.6	1.3	3.2
		CP,SK	0.30	0	0.48	0	0.53	0	0.63	1.6	0.86	1.6
		UT,SC	0.26	0	0.40	1.6	0.47	1.6	0.63	1.6	1.0	1.6
		UT,RV	0.28	0	0.46	1.6	0.50	1.6	0.58	1.6	0.88	3.2
		UT,QR	0.49	0	0.59	1.6	0.64	1.6	0.95	1.6	1.2	1.6
		UT,SK	0.23	0	0.34	0	0.37	0	0.44	1.6	0.47	1.6
		SC,RV	0.40	1.6	0.52	1.6	0.65	1.6	0.90	1.6	1.5	1.6
		SC,QR	0.44	0	0.68	1.6	0.71	1.6	1.1	3.2	1.7	3.2
		SC,SK	0.34	0	0.48	1.6	0.57	1.6	0.73	1.6	0.94	1.6
		RV,QR	0.63	1.6	0.87	3.2	1.1	3.2	2.4	4.8	3.7	4.8
		RV,SK	0.46	1.6	0.74	1.6	0.88	1.6	1.8	3.2	3.3	3.2
		QR,SK	0.58	0	0.78	1.6	1.1	1.6	1.7	1.6	1.9	3.2

Table 9: Mean Imputation MDT for monotone missing data on combinations of two productivity factors. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

From this table we can observe that:

- The bias increases as the extent of missing data increases, for both the MCAR and MAR mechanisms.
- For the MAR mechanism, there is no striking difference between biases for large vs. small projects.
- There are no striking differences between the MAR and MCAR mechanisms.
- There is no significant difference in performance for different combinations of productivity factors.
- The monotone pattern performs slightly worse than the univariate pattern.
- The Pred25 value peaks at a 6.3% bias for 40% missing data. In our context this would mean that 4 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation.
- MdMRE peaks at a bias of 3.7% for 40% missing data. This means that the estimation accuracy will be different from the case where complete data is collected by an average of only 3.7%.
- For percentages of missing data up to 25%, mean imputation performs remarkably well, with a negligible bias in its performance.

Table 10 shows the results for the case where more than two productivity factors contain 40% missing data under the MCAR and MAR mechanisms. The MAR results are for higher biases on both small and large projects. In agreement with the pattern shown in Figure 2, the first productivity factor listed contains the larger amount of missing data than the second, the second more than the third, and so on. Due to the similarity of the results for the presented permutations, further results with additional permutations are not presented.

Missing	MCAR		MAR (large)		MAR (small)	
	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25
CP,UT,SC	1.1	3.2	1.0	3.2	0.90	3.2
CP,UT,SC,RV	1.0	3.2	0.90	3.2	0.96	3.2
CP,UT,SC,RV,QR	1.0	3.2	1.1	3.2	0.86	3.2
CP,UT,SC,RV,QR,SK	0.93	3.2	1.0	3.2	0.79	3.2
UT,SC,RV	1.3	3.2	1.7	4.8	1.0	1.6
UT,SC,RV,QR	1.4	3.2	1.8	4.8	1.0	1.6
UT,SC,RV,QR,SK	1.3	3.2	1.8	4.8	1.0	1.6
UT,SC,RV,QR,SK,CP	1.4	3.2	1.7	4.8	1.0	1.6
SC,RV,QR	1.4	3.2	1.5	4.8	2.0	1.6
SC,RV,QR,SK	1.5	3.2	1.5	4.8	2.0	2.4
SC,RV,QR,SK,CP	1.5	3.2	1.4	4.8	2.0	1.6
SC,RV,QR,SK,CP,UT	1.6	3.2	1.3	4.8	2.0	1.6
RV,QR,SK	2.1	4.8	1.8	4.8	3.4	4.8
RV,QR,SK,CP	2.2	4.8	1.7	4.8	3.4	4.8
RV,QR,SK,CP,UT	2.1	4.8	1.8	4.8	3.4	4.8
RV,QR,SK,CP,UT,SC	2.1	4.8	1.8	4.8	3.1	4.8
QR,SK,CP	1.8	1.6	1.6	1.6	2.1	3.2
QR,SK,CP,UT	1.8	3.2	1.6	1.6	2.1	3.2
QR,SK,CP,UT,SC	1.6	1.6	1.8	1.6	2.1	3.2
QR,SK,CP,UT,SC,RV	1.8	3.2	2.3	1.6	2.1	3.2
SK,CP,UT	0.90	1.6	0.90	1.6	0.80	1.6
SK,CP,UT,SC	0.90	1.6	0.90	3.2	0.70	1.6
SK,CP,UT,SC,RV	0.90	1.6	1.0	3.2	0.80	1.6
SK,CP,UT,SC,RV,QR	1.1	1.6	0.90	3.2	0.90	1.6

Table 10: Mean Imputation, 40% missing data, monotone pattern, and multiple productivity factors using MCAR and MAR mechanisms. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

Here we can observe that the performance is equivalent to the monotone pattern with two productivity factors.

4.3.3 Non-ignorable Missingness Mechanism, Univariate Case

Table 11 shows the results for the case where the mechanism for missing data is non-ignorable. Cases with biases towards low and high values for each productivity factor are distinguished.

		5%		10%		15%		25%		40%	
Missing	**Bias**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
CP	low	0.33	0	0.57	0	0.88	0	1.1	1.6	1.6	3.2
	high	0.20	0	0.31	0	0.34	0	0.97	1.6	0.71	3.2
UT	low	0.29	0	0.59	0	0.74	1.6	0.89	1.6	1.6	1.6
	high	0.17	0	0.30	1.6	0.37	1.6	0.44	1.6	0.76	1.6
SC	low	0.29	0	0.43	1.6	0.60	1.6	0.73	3.2	1.5	4.8
	high	0.31	0	0.51	1.6	0.75	1.6	0.64	1.6	1.0	1.6
RV	low	0.78	1.6	1.4	3.2	2.2	4.8	3.5	4.8	4.9	1.6
	high	0.32	1.6	0.39	1.6	0.66	1.6	0.88	1.6	1.3	3.2
QR	low	0.71	0	1.1	1.6	1.6	3.2	2.2	4.8	3.2	6.3
	high	0.52	0	0.60	0	0.91	0	1.4	1.6	1.4	1.6
SK	low	0.33	0	0.44	0	0.49	0	0.60	1.6	0.92	1.6
	high	0.49	0	1.0	0	1.5	1.6	2.5	3.2	4.3	6.3

Table 11: Mean imputation MDT for univariate missing data on all productivity factors using non-ignorable missing data mechanism. It is also indicated whether there is more missingness for low vs. high values.

The following observations can be made about the univariate non-ignorable case:

- The bias increases as the extent of missing data increases.
- In general, the non-ignorable case with bias towards high values performs better than MCAR and MAR whereas bias towards low values results in poorer performance.
- In general, having more missing observations with high values performs slightly better than having more missing observations for low values on the variable.
- At 40% missing values, the Pred25 bias peaks at 6.3. In our context this would mean that 4 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation.
- The MdMRE bias peaks at 4.9%, which is considerably larger than any of the values that we have seen so far.
- At and below 15% missing data, both the MdMRE and Pred25 biases tend to be better than for listwise deletion for non-ignorable missingness.

4.3.4 Non-ignorable Missingness Mechanism, Monotone Pattern

Table 12 shows the results for the monotone pattern of missing data and the non-ignorable mechanism. In agreement with the pattern shown in Figure 2, the first productivity factor listed contains the larger amount of missing data than the second. It is important to note that for the monotone pattern of missing data, it is only the first productivity factor listed in the "Missing" column that contains missing values according to the non-ignorable mechanism.

		5%		10%		15%		25%		40%	
Bias	**Missing**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
high	CP,UT	0.24	0	0.39	1.6	0.50	1.6	1.3	3.2	1.9	3.2
	CP,SC	0.32	0	0.36	1.6	0.55	1.6	0.88	1.6	0.97	3.2
	CP,RV	0.27	0	0.42	1.6	0.50	1.6	0.71	3.2	1.1	4.8
	CP,QR	0.50	0	0.68	0	0.74	1.6	0.90	1.6	0.78	3.2
	CP,SK	0.22	0	0.35	0	0.36	1.6	0.62	1.6	0.79	3.2
	UT,SC	0.24	0	0.35	1.6	0.41	1.6	0.49	1.6	1.3	1.6
	UT,RV	0.28	1.6	0.39	1.6	0.49	1.6	0.66	1.6	1.2	1.6
	UT,QR	0.49	0	0.62	1.6	0.83	1.6	1.3	1.6	1.4	1.6
	UT,SK	0.28	0	0.39	1.6	0.52	1.6	0.71	1.6	0.77	1.6
	SC,RV	0.38	1.6	0.58	1.6	0.76	1.6	0.90	1.6	1.8	1.6
	SC,QR	0.36	0	0.59	1.6	0.72	1.6	1.3	1.6	1.7	1.6
	SC,SK	0.32	0	0.47	1.6	0.58	1.6	0.71	1.6	1.0	1.6
	RV,QR	0.45	1.6	0.57	1.6	0.72	1.6	1.2	1.6	1.8	1.6
	RV,SK	0.39	0	0.57	1.6	0.72	1.6	0.90	3.2	1.3	3.2
	QR,SK	0.58	0	0.78	0	0.95	0	1.2	1.6	1.8	1.6
low	CP,UT	0.35	0	0.59	0	0.77	1.6	1.1	1.6	1.3	4.8
	CP,SC	0.39	0	0.66	1.6	0.73	1.6	1.1	1.6	2.2	4.8
	CP,RV	0.47	1.6	0.89	1.6	0.94	1.6	1.3	3.2	1.4	4.8
	CP,QR	0.58	0	0.92	1.6	0.93	1.6	1.2	1.6	1.6	3.2
	CP,SK	0.40	0	0.69	0	0.82	0	0.99	1.6	1.2	3.2
	UT,SC	0.33	0	0.51	1.6	0.72	1.6	1.1	3.2	1.1	1.6
	UT,RV	0.33	1.6	0.55	1.6	0.64	3.2	1.0	3.2	1.7	3.2
	UT,QR	0.44	0	0.58	1.6	0.69	1.6	0.94	1.6	1.3	1.6
	UT,SK	0.27	0	0.45	1.6	0.58	1.6	0.66	1.6	0.96	1.6
	SC,RV	0.40	1.6	0.70	3.2	0.93	3.2	1.6	4.8	3.5	3.2
	SC,QR	0.53	0	1.0	3.2	1.2	4.8	2.0	4.8	3.6	3.2
	SC,SK	0.35	0	0.65	1.6	0.76	1.6	1.1	3.2	1.9	4.8
	RV,QR	0.97	3.2	2.1	4.8	2.8	4.8	4.7	4.8	6.2	3.2
	RV,SK	0.81	1.6	1.6	3.2	2.4	4.8	4.0	4.8	6.0	1.6
	QR,SK	0.66	0	0.85	1.6	1.4	1.6	1.9	3.2	3.6	4.8

Table 12: Mean Imputation MDT for monotone missing data on combinations of two productivity factors using non-ignorable missing data mechanism. It is also indicated whether there is more missingness for low vs. high values.

The following observations can be made about the monotone non-ignorable case:

- The bias increases as the extent of missing data increases.
- The performance is similar to that of MCAR and MAR when more high values of the productivity factor contain missing values, but slightly worse for low values.
- In general, bias towards high values results in slightly better performance than low values on the variable.
- The performance is slightly worse than the univariate case.
- There is no marked difference in performance for different combinations of productivity factors.
- At 40% missing values, the Pred25 bias peaks at 4.8. In our context this would mean that 3 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation.
- The MdMRE bias peaks at 6.2%. This value is relatively large compared to the results seen so far.

Next, we present the results where more than two productivity factors contain missing data under the non-ignorable mechanism. Table 13 shows the results for the case where more than two productivity factors contain 40% missing data. The conclusions do not change for additional permutations, and therefore they are not presented here.

	bias=low		bias=high	
Missing	MdMRE	Pred25	MdMRE	Pred25
CP,UT,SC	1.4	4.8	1.5	3.2
UT,SC,RV	1.3	1.6	1.4	1.6
SC,RV,QR	4.1	3.2	2.2	1.6
RV,QR,SK	6.5	3.2	1.6	3.2
QR,SK,CP	3.4	4.8	1.8	1.6
SK,CP,UT	1.2	1.6	3.5	4.8
CP,UT,SC,RV	1.5	4.8	1.7	3.2
UT,SC,RV,QR	1.3	1.6	1.4	3.2
SC,RV,QR,SK	4.3	3.2	2.3	1.6
RV,QR,SK,CP	6.3	3.2	1.8	1.6
QR,SK,CP,UT	3.1	4.8	1.9	1.6
SK,CP,UT,SC	1.1	1.6	3.3	4.8
CP,UT,SC,RV,QR	1.6	4.8	1.6	3.2
UT,SC,RV,QR,SK	1.3	1.6	1.3	1.6
SC,RV,QR,SK,CP	4.1	3.2	2.2	1.6
RV,QR,SK,CP,UT	6.3	3.2	1.9	3.2
QR,SK,CP,UT,SC	3.1	4.8	1.6	1.6
SK,CP,UT,SC,RV	1.1	1.6	3.0	4.8
CP,UT,SC,RV,QR,SK	1.7	4.8	1.7	3.2
UT,SC,RV,QR,SK,CP	1.3	1.6	1.4	3.2
SC,RV,QR,SK,CP,UT	4.2	3.2	2.3	1.6
RV,QR,SK,CP,UT,SC	6.3	3.2	1.8	3.2
QR,SK,CP,UT,SC,RV	3.1	4.8	2.1	1.6
SK,CP,UT,SC,RV,QR	1.1	1.6	3.3	4.8

Table 13: Mean Imputation MDT for 40% monotone missing data on multiple productivity factors using non-ignorable missing data mechanism. It is also indicated whether there is more missingness for low vs. high values.

It can be seen that the results are similar to those in Table 12, where two productivity factors contained missing values. Specifically, in cases where the first productivity factor is the same the results are equivalent. Therefore, adding productivity factors according to the monotone pattern does not affect the results.

4.4 Hot-Deck

The following section contains the results for the hot-deck imputation MDTs.

4.4.1 MCAR and MAR Mechanisms, Univariate Case

The MCAR and MAR missing data mechanisms are presented, with bias towards both large and small projects (MAR only). The Euclidean and Manhattan distance functions are labelled "E" and "M" respectively. Table 14 summarizes the results for the univariate missing data pattern on the Customer Participation productivity factor.

No Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	n/a	M	0.26	0	0.38	0	0.45	1.6	0.60	1.6	0.80	1.6
		E	0.27	0	0.43	0	0.48	1.6	0.76	1.6	0.96	1.6
MAR	large	M	0.22	0	0.27	0	0.36	0	0.65	1.6	0.82	1.6
		E	0.21	0	0.43	0	0.58	0	0.73	1.6	1.0	1.6
	small	M	0.25	0	0.47	1.6	0.58	1.6	0.75	1.6	1.1	1.6
		E	0.26	0	0.45	1.6	0.54	1.6	0.82	1.6	1.3	1.6

Z – score Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	n/a	M	0.25	0	0.43	0	0.44	1.6	0.63	1.6	0.85	1.6
		E	0.30	0	0.46	0	0.48	1.6	0.73	1.6	1.0	1.6
MAR	large	M	0.24	0	0.40	0	0.44	0	0.55	1.6	0.79	1.6
		E	0.30	0	0.43	0	0.52	0	0.71	1.6	0.74	1.6
	small	M	0.23	0	0.39	1.6	0.48	1.6	0.67	1.6	0.80	1.6
		E	0.32	0	0.49	1.6	0.55	1.6	0.72	1.6	1.0	1.6

Mean Absolute Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	n/a	M	0.28	0	0.46	0	0.49	0	0.64	1.6	0.70	1.6
		E	0.29	0	0.52	0	0.56	0	0.71	1.6	0.90	1.6
MAR	large	M	0.28	0	0.39	0	0.46	0	0.70	1.6	0.80	1.6
		E	0.26	0	0.49	0	0.50	0	0.73	1.6	0.86	1.6
	small	M	0.31	0	0.48	1.6	0.46	1.6	0.74	1.6	0.76	1.6
		E	0.29	0	0.44	1.6	0.58	1.6	0.71	1.6	0.81	1.6

z5 Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	n/a	M	0.28	0	0.38	0	0.49	1.6	0.60	1.6	0.76	1.6
		E	0.25	0	0.39	1.6	0.51	1.6	0.56	1.6	0.85	1.6
MAR	large	M	0.23	0	0.29	0	0.43	0	0.58	1.6	0.79	1.6
		E	0.21	0	0.31	0	0.40	0	0.60	1.6	0.80	1.6
	small	M	0.31	0	0.36	1.6	0.56	1.6	0.69	1.6	0.89	1.6
		E	0.30	0	0.43	1.6	0.54	1.6	0.69	1.6	0.86	1.6

Table 14: All Hot-Deck MDTs for univariate missing data on Customer Participation productivity factor. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

The following conclusions can be drawn from this table:

- The bias increases as the extent of missing data increases, for both the MCAR and MAR mechanisms.
- For the MAR mechanism, there is no striking difference between biases for large vs. small projects.
- There are no striking differences between the MAR and MCAR mechanisms.
- There are no significant differences between any of the standardization techniques or the two distance measures for both the MCAR and MAR mechanisms.
- The Pred25 value peaks at a 1.6% bias for 40% missing data. In our context this would mean that 1 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation. Although the difference is relatively small, this is smaller than that of both the listwise deletion and mean imputation MDTs.
- MdMRE peaks at a bias of 1.3% for 40% missing data. This means that the estimation accuracy will be different from the case where complete data is collected by an average of only 1.3%. Again, although the difference is small, it is better than listwise deletion and mean imputation.
- For all percentages of missing data, hot-deck imputation performs remarkably well, with a negligible bias in its performance. This holds regardless of the standardization technique or distance measure used.

Given that the different hot-deck implementations have similar results, we focus below on the one with z-score standardization and Euclidean distance.

			5%		40%	
Factor	**Mech.**	**Bias**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
CP	MCAR	n/a	0.30	0	1.0	1.6
	MAR	large	0.30	0	0.74	1.6
		small	0.32	0	1.0	1.6
UT	MCAR	n/a	0.25	0	0.85	1.6
	MAR	large	0.21	1.6	1.7	1.6
		small	0.25	0	0.52	1.6
SC	MCAR	n/a	0.32	0	1.2	3.2
	MAR	large	0.25	0	0.88	3.2
		small	0.39	0	1.4	3.2
RV	MCAR	n/a	0.46	1.6	2.4	1.6
	MAR	large	0.40	1.6	2.1	3.2
		small	0.55	1.6	2.8	1.6
QR	MCAR	n/a	0.62	0	2.6	4.8
	MAR	large	0.60	0	2.5	4.8
		small	0.67	0	2.9	4.8
SK	MCAR	n/a	0.37	0	0.93	1.6
	MAR	large	0.51	0	0.85	1.6
		small	0.35	0	1.1	1.6

Table 15: Traditional[17] Hot-Deck Imputation for univariate missing data on each productivity factor. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

Table 15 summarizes the results for traditional hot-deck imputation for each productivity factor at both extremes of percentage of data missing. It can be seen that there are no striking differences between the different productivity factors nor each of the mechanisms.

[17] The hot-deck method using z-score standardization and Euclidean distance function.

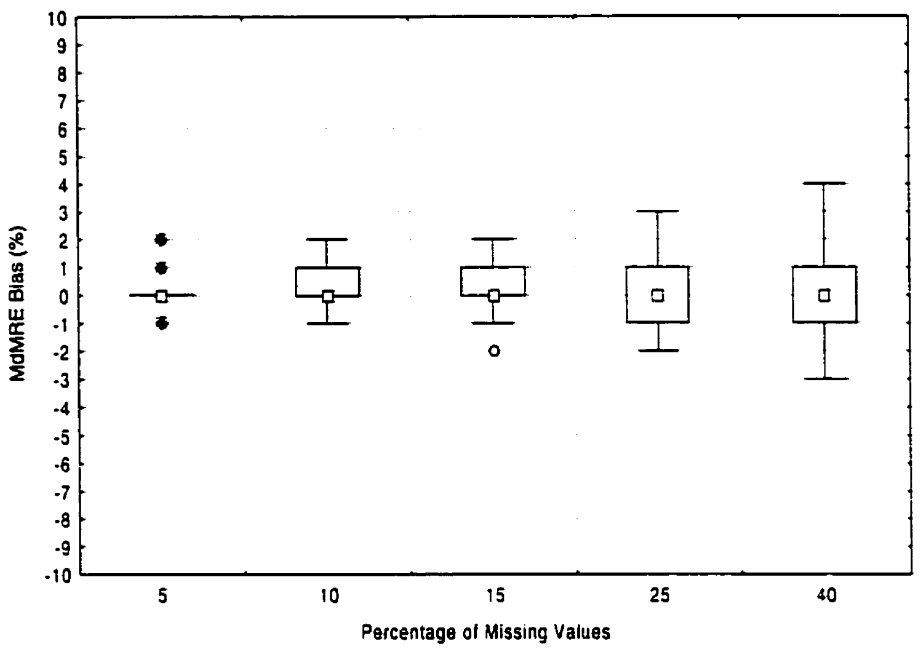

Figure 11: Summary of MdMRE results - traditional[18] Hot-Deck Imputation for Customer Participation productivity factor under MCAR.

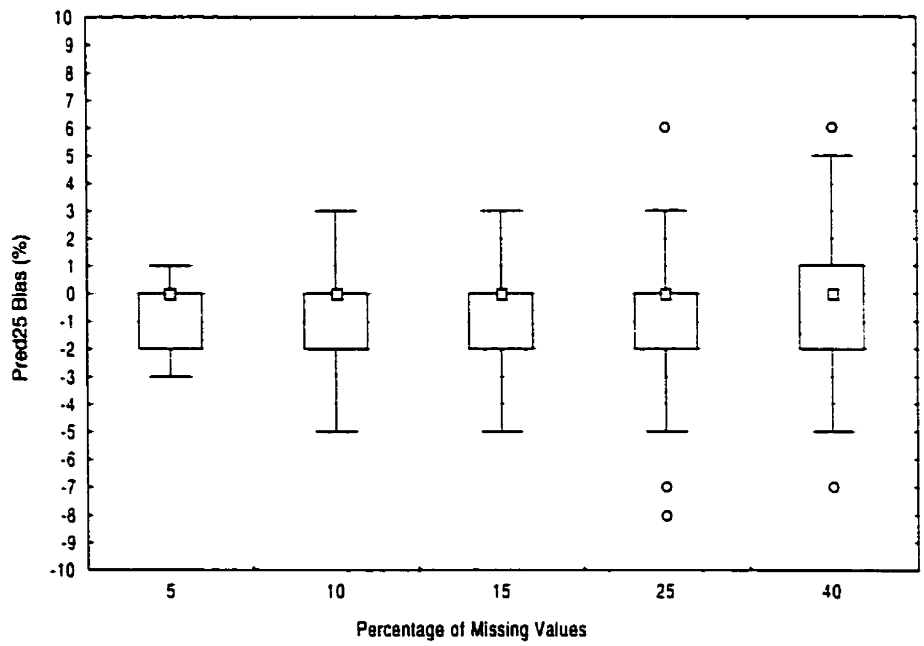

Figure 12: Summary of Pred25 results – traditional[18] Hot-Deck Imputation for Customer Participation productivity factor under MCAR.

[18] The hot-deck method using z-score standardization and Euclidean distance function.

The traditional[18] hot-deck imputation results for imputation using the MCAR mechanism on one productivity factor, Customer Participation, are summarized in Figure 11 and Figure 12. It can be seen that the MdMRE and Pred25 for all percentages of missing values are equal to the MRE and Pred25 of the complete data set analysis. The variance increases slightly for 40% missing data, but is not as pronounced as that for listwise deletion and mean imputation.

4.4.2 MCAR and MAR Mechanisms, Monotone Pattern

Table 16 summarizes the results for the monotone missing data pattern on The Customer Participation and Use of Tools productivity factors (the conclusions do not change for other permutations of productivity factors, and are therefore not presented here).

No Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	n/a	M	0.31	0	0.47	1.6	0.60	1.6	0.74	1.6	1.0	1.6
		E	0.30	0	0.48	1.6	0.58	1.6	0.73	1.6	1.0	1.6
MAR	large	M	0.29	0	0.52	1.6	0.63	1.6	0.93	1.6	1.1	3.2
		E	0.30	0	0.44	0	0.71	1.6	0.82	1.6	1.1	1.6
	small	M	0.29	1.6	0.47	1.6	0.55	1.6	0.76	1.6	0.97	1.6
		E	0.30	1.6	0.46	1.6	0.55	1.6	0.82	1.6	1.0	3.2

Z – score Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	n/a	M	0.35	1.6	0.50	1.6	0.61	1.6	0.79	1.6	0.96	1.6
		E	0.36	0	0.58	0	0.61	1.6	0.74	1.6	1.0	1.6
MAR	large	M	0.33	0	0.48	1.6	0.67	1.6	0.83	1.6	1.0	1.6
		E	0.31	0	0.57	1.6	0.73	1.6	0.86	1.6	1.1	1.6
	small	M	0.33	1.6	0.46	1.6	0.61	1.6	0.69	1.6	1.0	1.6
		E	0.35	0	0.44	1.6	0.60	1.6	0.80	1.6	0.97	1.6

Mean Absolute Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	n/a	M	0.31	0	0.49	1.6	0.67	1.6	0.81	1.6	1.1	1.6
		E	0.37	0	0.52	1.6	0.63	1.6	0.83	1.6	1.0	1.6
MAR	large	M	0.36	0	0.57	1.6	0.67	1.6	0.94	1.6	0.99	1.6
		E	0.35	0	0.59	0	0.69	1.6	0.91	1.6	0.98	1.6
	small	M	0.35	0	0.52	1.6	0.64	1.6	0.78	1.6	1.00	1.6
		E	0.32	0	0.52	1.6	0.57	1.6	0.73	1.6	0.94	1.6

z5 Standardization

			5%		10%		15%		25%		40%	
Mech.	**Bias**	**Dist**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**	**MdMRE**	**Pred25**
MCAR	none	M	0.34	0	0.51	1.6	0.60	1.6	0.77	1.6	0.99	1.6
		E	0.32	0	0.44	1.6	0.57	1.6	0.72	1.6	1.0	1.6
MAR	large	M	0.30	0	0.48	1.6	0.62	1.6	0.84	1.6	1.1	1.6
		E	0.27	0	0.45	1.6	0.60	1.6	0.83	1.6	1.0	1.6
	small	M	0.36	0	0.50	1.6	0.65	1.6	0.71	1.6	0.88	1.6
		E	0.34	0	0.46	1.6	0.59	1.6	0.68	1.6	0.98	1.6

Table 16: Hot-Deck MDTs for monotone missing data on Customer Participation and Use of Tools productivity factors. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

From this table we can observe that:

- Bias tends to increase as the extent of missing data increases, for both the MCAR and MAR mechanisms.
- For the MAR mechanism, there is no striking difference between biases for large vs. small projects.

- There are no striking differences between the MAR and MCAR mechanisms.
- There are no significant differences between any of the standardization techniques or the two distance measures for both the MCAR and MAR mechanisms.
- The Pred25 value peaks at a 3.2% difference for 40% missing data. In our context this would mean that 2 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation. Although the difference is small, this value is smaller than that for mean imputation.
- MdMRE peaks at a difference of 1.1% for 40% missing data. This means that the estimation accuracy will be different from the case where complete data is collected by an average of only 1.1%. Again, this is slightly better than mean imputation.
- For all percentages of missing data, hot-deck imputation performs remarkably well, with a negligible bias in its performance. This holds regardless of the standardization technique or distance measure used.
- Hot-deck imputation technique on monotone missing data performs slightly worse than for the univariate case.

Given that the different hot-deck implementations have similar results, we focus below on the one with z-score standardization and Euclidean distance.

Table 17 summarizes the results for the case where more than two productivity factors contain low and high extremes of missing data under the MCAR and MAR mechanisms for the traditional hot-deck imputation. The MAR results are for higher biases on both small and large projects.

			5%		40%	
Factor	Mech.	Bias	MdMRE	Pred25	MdMRE	Pred25
CP,UT	MCAR	n/a	0.36	0	1.0	1.6
	MAR	large	0.31	0	1.1	1.6
		small	0.35	0	0.97	1.6
UT,SC	MCAR	n/a	0.33	0	1.1	1.6
	MAR	large	0.25	1.6	1.7	3.2
		small	0.32	0	0.81	1.6
SC,RV	MCAR	n/a	0.50	1.6	1.5	1.6
	MAR	large	0.37	1.6	1.2	3.2
		small	0.50	1.6	1.7	1.6
RV,QR	MCAR	n/a	0.77	1.6	3.4	3.2
	MAR	large	0.66	1.6	2.5	3.2
		small	0.88	1.6	3.7	1.6
QR,SK	MCAR	n/a	0.77	0	2.4	3.2
	MAR	large	0.74	0	2.4	3.2
		small	0.63	0	2.0	3.2
SK,CP	MCAR	n/a	0.53	0	0.97	1.6
	MAR	large	0.64	0	0.93	1.6
		small	0.44	0	0.99	1.6

Table 17: Traditional[19] Hot-Deck Imputation for monotone missing data on combinations of productivity factors. For the MAR case, it is indicated whether there is more missingness for large vs. small projects.

In general, the results are similar to those presented in Table 16, where the Customer Participation and Use of Tools productivity factors contained missing data. It can be seen that there is no significant difference in performance for all combinations of two productivity factors.

4.4.3 Non-ignorable Missingness Mechanisms, Univariate Case

Table 18 shows the results for the case where the mechanism for missing data is non-ignorable. Cases where there is non-ignorable missing data with biases towards low and high values for each productivity factor are distinguished. The traditional hot-deck method, using z-score standardization and Euclidean distance measure, is presented. The results for the other standardization techniques and Manhattan distance function are similar and are thus not presented here.

[19] The hot-deck method using z-score standardization and Euclidean distance function.

Bias	CP		UT		SC		RV		QR		SK	
	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25
Low	1.5	3.2	0.86	1.6	1.8	3.2	4.8	1.6	3.4	6.3	0.86	1.6
High	1.1	1.6	2.0	1.6	1.3	1.6	1.6	1.6	1.4	1.6	3.4	4.8

Table 18: Traditional[20] Hot-Deck Imputation MDT for 40% univariate missing data on each productivity factor using non-ignorable missing data mechanism. It is also indicated whether there is more missingness for low vs. high values.

The following observations can be made about the univariate non-ignorable case:

- The performance of hot-deck for the non-ignorable case is slightly worse than for MCAR and MAR.
- At 40% missing values, the Pred25 bias peaks at 6.3. In our context this would mean that 4 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation.
- The MdMRE bias peaks at 4.8%.
- At 40% missing data, both the MdMRE and Pred25 biases are slightly better overall compared with mean imputation with non-ignorable missingness.

4.4.4 Non-ignorable Missingness Mechanism, Monotone Pattern

Table 19 shows the results for 40% missing data with monotone pattern and the non-ignorable mechanism. Cases where there is non-ignorable missing data with biases towards low and high values for each productivity factor are distinguished. In agreement with the pattern shown in Figure 2, the first productivity factor listed contains the larger amount of missing data than the second. It is important to note that for the monotone pattern of missing data, it is only the first productivity factor listed in the "Missing" column that contains missing values according to the non-ignorable mechanism. The traditional hot-deck method, using z-score standardization and Euclidean distance measure, is presented as the results for the other standardization techniques and Manhattan distance function are similar and are thus not presented here.

[20] The hot-deck method using z-score standardization and Euclidean distance function.

Bias	CP,UT		UT,SC		SC,RV		RV,QR		QR,SK		SK,CP	
	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25	MdMRE	Pred25
low	1.3	3.2	0.99	1.6	3.9	3.2	7.0	3.2	2.7	3.2	0.99	1.6
high	1.2	1.6	1.4	1.6	1.2	1.6	1.7	1.6	1.5	1.6	2.8	3.2

Table 19: Traditional[21] Hot-Deck Imputation MDT for 40% monotone missing data on combinations of productivity factors using non-ignorable missing data mechanism. It is also indicated whether there is more missingness for low vs. high values.

The following observations can be made about the monotone non-ignorable case:

- The performance is slightly worse than that of MCAR and MAR.
- The results are similar or slightly worse overall compared to the univariate pattern for the non-ignorable case (Table 18).
- The Pred25 bias for this high level of missing data peaks at 3.2. In our context this would mean that 2 more out of the 63 projects in the test set have an MRE greater than 25% when using mean imputation.
- The MdMRE bias peaks at 7.0%.

The hot-deck results presented indicate that there is no marked difference in the performance of different types of hot-deck. Furthermore, while the MdMRE and Pred25 results are slightly better than listwise deletion and mean imputation, but the improvement is so slight that they would easily be offset by the complications of setting up a hot-deck.

4.5 Concordance Between Train and Test Set

While constructing cost estimation models, it is a common assumption that the population represented by the training data set would be the same as the population represented by the test data set, and similar to any new projects for which cost predictions are made. If this is not the case, then this has implications on the performance of MDTs.

Note from Table 5 that the Requirements Volatility (RV) variable distribution has a different central tendency (mean or median) for the train and test data sets. Specifically, the test data set tends to have projects with a larger Requirements

[21] The hot-deck method using z-score standardization and Euclidean distance function.

Volatility than the training data set. When there are non-MCAR missing values on RV such that there are more values missing on smaller values of RV, we witness a deterioration of the imputation MDTs, for univariate and monotone patterns. We explain this below.

RV is correlated with the size of the project, in that projects with little effort are likely to have a low Requirements Volatility, and also are likely to be smaller. Therefore, the pattern of missingness for non-ignorable missingness with more missing values for low RV will tend to be similar to MAR when smaller projects are more likely to have missing values.

Also, recall that the accuracy measures we use tend to penalize smaller projects more. For instance, a 10 person-month overestimate for a 10 person-month project would have a much worse MRE than for a 100 person-month project.

When there are more missing values for small RV projects, values imputed by mean imputation will by definition be larger than the actual RV values. Therefore, the regression model will be predicting worse for small projects in the test data set, and because of the behavior of MRE, this will result in a small prediction accuracy.

Hot-deck imputation works best when the hot-deck covariates are strongly correlated with the variable with missing values. In our case, the covariates that we used were not strongly correlated with RV. Therefore, the hot-deck tended to impute values that were larger than the actual RV values, resulting in behavior similar to mean imputation.

Because of the above, listwise deletion performed better than the imputation techniques when low values of RV had many missing values. In fact, this kind of outcome would be expected whenever the training data set has projects that have, on average, smaller values on a particular variable and missingness occurs more frequently on smaller values on that variable.

4.6 Summary

Below we provide an overall summary of our findings:

- In general, all MDTs tend to perform well in absolute terms, with their bias being consistently of a low percentage. This has some implications on previous software engineering research. Given that the most common practice thus far has been to use listwise deletion, our results indicate that the detrimental consequences of this would be rather minor. Listwise deletion has the appeal of being a simple approach. However, researchers could do better, as we discuss below.
- For all mechanisms and patterns, we found that the performance of the MDTs (in terms of bias and precision) deteriorates as the percentage of missing data increases.
- The precision of listwise deletion tends to be worse than the other MDTs, especially as the percentage of missing data increases (Figure 13 exemplifies this trend).
- We did not find marked differences in the performance between MCAR and MAR mechanisms. The only exception is for monotone patterns with a large percentage of missing data, where the imputation techniques tend to perform slightly worse under MAR in terms of Pred25. This is evident in the summary plot in Figure 13.
- For hot-deck imputation, the differences amongst the various types of hot-deck parameters were not marked. Therefore, we suggest a traditional hot-deck is appropriate. A traditional hot-deck uses Euclidean distance and z-score standardization. Euclidean distance is the most common distance measure, and thus its appeal. Standardization is a reasonable thing to do given that otherwise size would systematically dominate any distance.
- In general, the MDTs tend to perform slightly worse with monotone patterns of missingness compared with the univariate pattern.[22]

[22] Except for listwise deletion, where the results are exactly the same as for the univariate missingness case.

- For monotone patterns of missingness, increasing the number of productivity factors with missing values to more than two does not have a marked impact on the performance of MDTs.[23]
- The MDTs tend to perform slightly worse under non-ignorable missingness compared with MCAR and MAR.
- Any hot-deck imputation technique will work well in absolute terms, with accuracy differences from a complete data set rarely, if ever, going above 3%. A summary of the performance of the three MDTs is presented in Figure 13. Mean imputation performs slightly worse than hot-deck, but the degradation is minor. Listwise deletion still performs very well, but its performance is a slight degradation over mean imputation and hot-deck. The above conclusions hold irrespective of the missingness mechanism (MCAR or MAR), missingness pattern (univariate or monotone on two variables), extent of observations with missing values (up to 40%), and productivity factors under consideration.

[23] Except for listwise deletion, where the results are exactly the same as for the univariate missingness case.

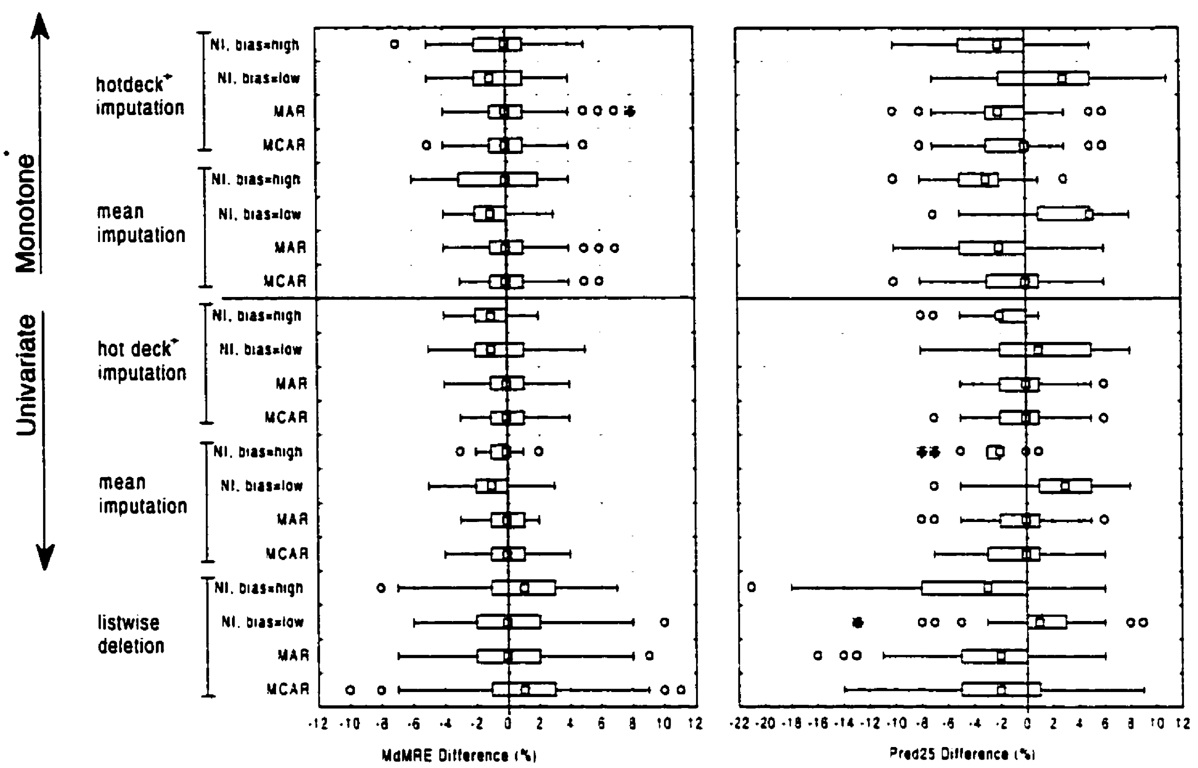

Figure 13: 40% Missing Data on Customer Participation productivity factor using MCAR, MAR and Non-Ignorable(NI). *Use of Tools productivity factor also contains missing data. *The hot-deck method using z-score standardization and Euclidean distance function.

4.7 Discussion

The best way to handle missing data is to maximize response in the original sample. In some cases, it may be desirable to resample if large amounts of missing data are in the original sample. But in the most likely scenario, in which there is missing data, we can make some practical recommendations. Furthermore, if ensuring complete data sets would be too costly, then our recommendations would result in models that would be almost as accurate had complete data sets been attained.

We found that in general, all MDTs perform rather well in terms of bias and precision. The differences between the MDTs under the various simulated conditions are small. This implies that common practices thus far of using listwise deletion would not have penalized the accuracy of cost estimation models too greatly.

One potential reason for this finding is that the impact of the productivity factors is rather small compared with the effect of size. This is similar to the conclusion drawn in a previous study, where Matson et al. showed that there was only a small improvement when productivity factors were added to the baseline model relating effort to project size [63]. If this is the general case, then ignoring or imputing the missing values on the productivity factors would have little influence on the accuracy of the prediction models.

However, one would in principle prefer to use an MDT with the smallest bias and greatest precision. Therefore, it is up to the analyst to decide whether the added complexity of other MDTs, such as hot-deck, is worth the improvement in bias and precision.

Amongst the MDTs examined, we found that imputation techniques perform better than listwise deletion.[24] To obtain the best performance from cost estimation models, it would therefore be prudent to apply the appropriate imputation technique. Our recommendations are summarized in the decision tree of Figure 14. The tree has a decision point at the non-terminal nodes, and the edges indicate the value. For example, starting from the root, if one believes that the missingness mechanism is MCAR or MAR, then take the edge on the left, otherwise take the edge on the right. The terminal nodes indicate the recommended MDT.

[24] The exception is when there is a discrepancy between the distribution of variables between the training data set and the test data set, or future projects, whereby the training data set tends to have smaller values, as described in Section 4.5.

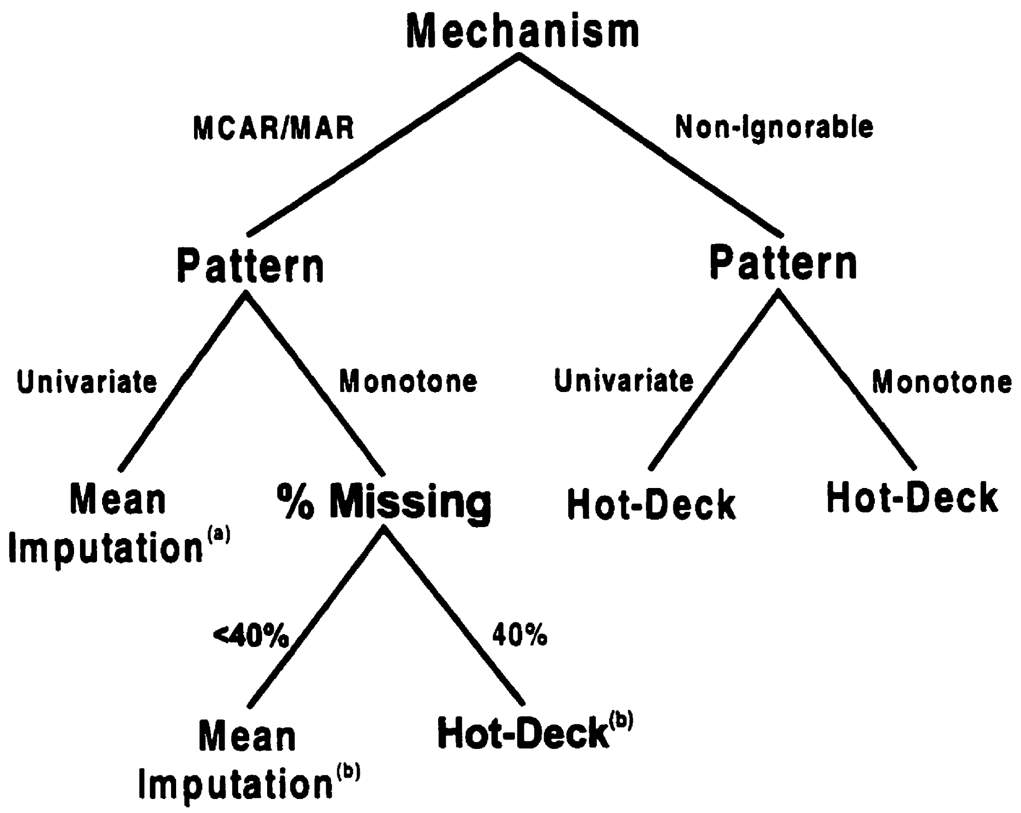

Figure 14: Recommendations for selecting MDTs. (a) This is generally the case, irrespective of the percentage of missing data. (b) It should be noted that here the difference between the choice of mean imputation and hot-deck is slight.

In order to apply the above decision tree, one has to determine whether the missing data mechanism is MCAR, MAR, or non-ignorable. It is possible to test whether data is missing completely at random [60]. To test for MAR, one could construct a logistic regression model with the dependent variable being the missing status and the covariates being those fully observed. If any of the covariates is found to be associated with the missingness indicator, then one could decide that the missingness mechanism is MAR. The remaining parameters necessary to use the decision tree (i.e., pattern and percentage of missing data) are easily determined from a data set.

4.8 Comparison to Previous Work

The literature suggests that under MCAR and a low percentage of missing data, listwise deletion tends to perform well. For instance, Gilley and Leone [36] state "If the item nonresponse is nonsystematic and represents a small percentage of a reasonably large sample, then excluding the nonrespondents from the sample would have a negligible effect on any statistical results", and Frane [33] notes "If the number of subjects with missing data is small, if data are missing at random, and if interest lies in statements regarding a population rather than individuals, the elimination of subjects with missing data is likely to lead to a satisfactory analysis." Furthermore, one Monte Carlo simulation provides corroborative results in that at low percentages of missing values (10% or less), they found that listwise deletion does not give markedly distorted estimates of regression coefficients and R^2 when data is missing at random [74].

Another study [53] analysed the performance of five MDTs for dealing with data missing nonrandomly, namely listwise deletion, pairwise deletion, mean imputation, simple regression imputation, and multiple imputation. Performance of each technique was based on parameter estimates of a two predictor regression model. The results showed that in general the three imputation techniques examined, mean imputation, simple regression imputation and multiple imputation, did not perform well with nonrandomly missing data. In contrast, the listwise and pairwise deletion techniques performed well when the level of missing data was less than 30%. This is consistent with our results, in that listwise deletion produces reasonable results under a variety of different missingness schemes and relatively high percentages of missing data.

For low percentages of missing values under MCAR, Roth [77], based on a literature review, recommended using hot-deck imputation. A simulation study by Lee and Chiu [56] led to the conclusion that listwise deletion is a preferred MDT to mean imputation when computing the polychoric correlation. We are, however, assuming a regression model.

Roth and Switzer [78] performed a Monte Carlo simulation comparing different MDTs. Techniques considered were listwise deletion, pairwise deletion, mean imputation, regression imputation, and hot-deck imputation under an MCAR mechanism. The results showed that pairwise deletion had the least amount of dispersion and average error around true scores for bivariate correlations. In the case of multiple regression, the performance of pairwise deletion was similar to that of listwise deletion. Furthermore, the authors recommended against the use of mean imputation. For regression parameters, the authors noted that there are slight differences between the various MDTs studied. This is consistent with our conclusions, even for MAR and non-ignorable missingness mechanisms. In our study, however, we did find that mean imputation performed better than listwise deletion for the MCAR setting.

Another simulation found that for data missing at random, mean imputation tended to perform slightly better than listwise deletion [74]. This is consistent with our results. For high percentages of missing values under MCAR and MAR, Roth recommended hot-deck imputation [77]. This is the same conclusion that we drew based on our simulations for the software cost estimation problem. Similarly, he recommends hot-deck for low percentages of data that is not missing at random. At high percentages of data that are not missing at random, Roth suggested maximum likelihood estimation [77]. However, we did not evaluate this as part of our study. We did find that hot-deck imputation worked best under these conditions for the software cost estimation problem.

One study by Kaiser found that the performance of hot-deck methods decreases with an increase in the proportion of records containing missing values, the increase in the number of missing values in each record, or the combination of both [47]. This is somewhat consistent with our results, except that we did not find that increases in the number of variables that have missing values beyond two had a substantial impact on the accuracy of hot-deck imputation. Further evaluations of the hot-deck procedure were performed [31][4][26][29].

Cox and Folsom [26] performed a simulation using an actual data set, where they were evaluating estimates of univariate means and proportions under different MDTs. They found that for discrete questionnaire items, hot-deck tended to reduce bias compared to listwise deletion, with better performance for items that had the greatest missingness. However, they also noted that the variance of the estimates was inflated by hot-deck. This was also noted by Ernst [29]. For continuous items, they concluded that the performance of hot-deck was inferior: it did not reduce the bias in the estimates and when it did, the variance was large. This is consistent with our results, in that the productivity factors can be considered as discrete variables by their definition, and we found that hot-deck tended to outperform listwise deletion. Another simulation was performed by Ford [31] where he compared six different MDTs, including four variants of hot-deck imputation. His criteria were estimates of the mean and its variance. Ford found that there was no difference in the performance amongst the different MDTs, although they were better than listwise deletion. This finding is also consistent with our simulation results. Both of the above studies, however, focus on parameters that are different from those that we evaluated (i.e., the accuracy of predictions from a regression model).

Perhaps the most important message from the above review is that it is critical to evaluate quantitative techniques using software engineering data sets. There is a mixed correspondence between our results and those obtained from previous studies. This is not surprising since the previous work used different evaluation criteria, and the distributions of the variables they simulated were different. Furthermore, techniques such as hot-deck, which tend to work best when there are strong correlations between the covariates and the variable with missing values, will perform differently depending on the correlation structure amongst the variables. This correlation structure may not be transportable from other disciplines.

4.9 Limitations

In this paper, we reported on a simulation study to evaluate some MDTs for dealing with missing data during the construction of software cost estimation models. While we have attempted to design the simulation to be as comprehensive as possible, covering many plausible different missingness scenarios, we cannot claim that this study is the last word on missingness for cost estimation modeling. Below we identify the limitations of the study, which in turn suggest avenues for further research. However, it should be noted that most of these limitations are a consequence of the scope that we have defined for our study. We already simulated more than 290 000 study points, which is quite extensive. Further simulations can be performed to address situations that we have not considered.

Our simulation study was performed using one data set in the business application domain. Although the data set was large, giving us confidence in the conclusions we draw, other simulations would have to be performed on different data sets to confirm our findings. We have attempted to be precise in describing the details of our simulation as an aid for future replications.

We only simulated missingness on the productivity factors. Our rationale has been that if data on the size variable and the effort variable are missing, when building a cost estimation model, then one has little to go on. This may be a reflection of a serious data collection problem. Whereas missingness on questionnaire responses, while not desirable, is more likely to happen.

Our evaluation criteria concerned only prediction accuracy. We did not consider the parameter estimates in the regression models or the effect of MDTs on statistical tests of significance. The objective of our study was to focus on the utility of such models for making effort predictions for new projects. It is plausible that a simulation that used other evaluative criteria may come up with different recommendations.

The simulations we performed selected six productivity factors out of an initial fifteen. We did not utilize the dropped nine productivity factors during the imputation procedures we used. Since the excluded factors are likely strongly correlated with some of the included factors (and this is one reason for dropping them in our analysis due to the increased risk of collinearity in the regression model otherwise), they may have served as useful covariates in a hot-deck and have resulted in even better performance for this imputation MDT. However, in practice, researchers would be more likely to exclude variables with many missing values from their analysis, and therefore excluded variables would not necessarily serve as good covariates.

We only considered up to 40% of the observations having missing values, which encompasses most practical situations. It is possible that in some studies there would be more than 40% missing values. However, it is contended that this would be an indicator of a severe data collection problem. For instance, Raymond and Roberts [74] state "With 40 percent of the data missing, one would have to question seriously the appropriateness of conducting any analysis." Furthermore, the wisdom of doing any analysis with 30-40% missing data has been questioned by Roth [77] and Ford [31].

When one is constructing a cost estimation model, it is not known a priori whether the available data set is concordant with future projects. Therefore, when the choice of MDT depends on this knowledge, as alluded to in Section 4.5, it is difficult to choose the optimal MDT. We therefore based our recommendations on the assumption that there is such a concordance. This is the same assumption invoked in all previous cost estimation studies that utilize historical data.

Finally, our simulation focused on ordinary least squares regression as the modeling technique. This is the most popular technique for building cost estimation models, and has been found in recent studies to perform at least as good as alternatives. It is plausible that other modeling techniques would require different MDTs.

Chapter 5: Conclusions

The objective of this paper was to perform a comprehensive simulation to evaluate techniques for dealing with missing data in cost estimation models. We simulated a total of 293 400 study points, varying: the number of variables with missing data (from 1 to six), the percentage of missing data (from 5% to 40%), the missing data mechanism (MCAR, MAR, and non-ignorable missingness), the missing data pattern (univariate and monotone), and comparing 10 different techniques for dealing with missing data. The performance of this study was greatly facilitated by the existence of a large and complete software project data set. This is not very common, as many cost data sets do have missing values.

Our results provide practical and substantiated guidelines for researchers and practitioners constructing cost estimation models when their data sets have missing values. Although we show that deleting observations with missing values has a small penalty, better performance would be obtained from applying imputation techniques.

We encourage the replication of this simulation study on alternative data sets to confirm, or otherwise, our conclusions. If indeed further replications confirm our findings, then this has important practical significance to those building cost estimation models. Furthermore, future work ought to examine alternative imputation techniques, as these may provide even better performance than the ones we studied.

Appendix A: Box and Whisker Plots

In this paper, box and whisker plots are used quite frequently. This appendix is intended to explain how to interpret such a diagram.

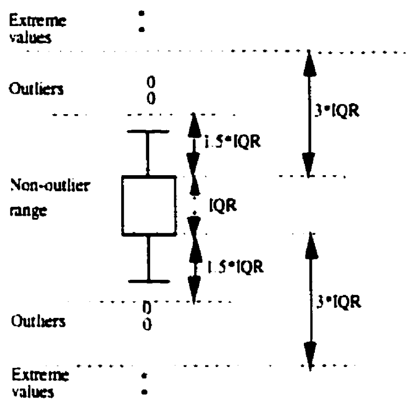

Figure 15: Description of a box and whisker plot.

Box and whisker plots are used to show the variation in a particular variable. Figure 15 shows how such a plot is constructed. The box represents the inter-quartile range (IQR). The IQR bounds the 25th and 75th percentiles. The 25th percentile is the value of the variable where 25% or less of the observations have equal or smaller values. The same is true for the 75th percentile. The whiskers are the largest values within 1.5 times the size of the box. This value of 1.5 is conventional. Outliers are within 1.5 times the size of the box beyond the whiskers, and extremes are beyond the outliers. Finally, usually there is a dot in the box. This dot denotes the median, or the 50th percentile.

The box and whisker plot provides a versatile way for visualizing the obtained values on a variable.

Appendix B: Definitions of the Productivity Factors

This appendix provides a short description of the productivity factors in the LATURI data set, as well as the question from the questionnaire used to collect data pertaining to that factor. Each question contains the five possible answers, based on the 1 – 5 numerical scale. In addition, each productivity factor is separated into one of four groups: project, process, product, and people factors.

B.1 Project factors

The following list contains those productivity factors that assess elements of the software project.

B.1.1 Customer Participation

How actively the customer (user) participates in the development

1 – Very Small: The customer does not have time to participate in the project definition or in project development.

2 – Small: Customer participation is passive. The customer has approved a small amount (less than 30%) of software function.

3 – Nominal: The customer participates in the project at satisfactory level. The customer has approved approximately half of the functions (30-70%).

4 – Much: The customer actively participates in the project. The customer has defined and approved most of the functions (over 70%), including all of the most important functions.

5 – Very Much: The customer participates very actively. Consequently, most of the functions will be slightly volatile.

B.1.2 Development Environment Adequacy

The performance level of tool and equipment resources during the project

1 – Very Small: Development facilities continually fall short of expectations. Constructing test environments requires special arrangements.

2 – Small: There are shared equipment/machine resources. Delays exist in some stages (e.g. compiling and testing).

3 – Nominal: There is enough equipment and tool resources during development work. All members have their own workstation.

4 – Much: There is enough equipment and tool resources to handle capacity peaks (e.g. efficiency, storage, response time).

5 – Very Much: There exist dedicated development environments specifically for this project.

B.1.3 Staff Availability

The availability of software personnel during the project

1 – Very Small: Availability of the key software personnel to perform most tasks is small. Key personnel are involved in many customer and maintenance responsibilities simultaneously.

2 – Small: The team members are involved in other simultaneous projects in addition to maintenance responsibilities. This project is low priority.

3 – Nominal: The key members of this project are involved in one other project at most. Responsibilities of this other project can effect their availability for this project.

4 – Much: Members are involved in this project close to full time. Some key staff may have some availability problems.

5 – Very Much: Qualified software personnel are available when needed, and can fully participate in this project. Personnel are ready for short bursts of high overload during the project.

B.2 Process factors

The following list contains those productivity factors that assess elements of the software process.

B.2.1 Use of Tools

Use and quality of tools available for the project

1 – Very Small: Minimal amount of tools are available (e.g. editors, compilers, simple debugging tools).

2 – Small: Basic tools are available (e.g. interpreters, editors, compilers, debuggers, databases, libraries).

3 – Nominal: There exists a development environment, data base management system and support for most phases.

4 – Much: Modern tools are available (e.g. CASE, project planning, application generators). There are standardized interfaces between phases and/or tools.

5 – Very Much: Integrated CASE environment that covers the whole life cycle. All tools can support each other.

B.2.2 Requirements Volatility

Stability of customer (user) requirements

1 – Very Small: Requirements are continuously changing. More than 30% of the functions are new or modified versions of the original requirements.

2 – Small: Some essential changes are made that impact total architecture. Project must return to previous phases and modify previous results. 15-30% of the functions are new or modified.

3 – Nominal: Changes to specifications occur, but they are managed and their impact is minor (less than 15% of the functions are new or modified).

4 – Much: Some changes to specifications, some new or adapted functions, some minor changes in data contents.

5 – Very Much: No new features added during the project.

B.2.3 Use of Standards

The quality of the existing standards and procedures applied on this project.

1 – Very Small: Standards and basic practices are developed during the project.

2 – Small: Standards are partially known. Additional procedures must be developed for some tasks.

3 – Nominal: Standards are well known. General standards are used that have been applied in the past. Some tailoring is needed for most major tasks.

4 – Much: Detailed standards are used that have been applied in the same environment in the past.

5 – Very Much: Stable and detailed standards are used that are familiar to the team.

B.2.4 Use of Methods

The use and quality of methods to be applied during the project

1 – Very Small: The project does not use any modern software engineering methods (e.g. mostly meetings, individual use, trials, etc.).

2 – Small: The use of methods is minimal. Traditional concepts are used (e.g. structured analysis and design, top-down design, etc.)

3 – Nominal: Well known methods are used (e.g. structured analysis and design, conceptual analysis, entity relationship modeling, etc.)

4 – Much: Detailed methods are integrated that cover most activities. Support exists for most methods and all members of the project use methods.

5 – Very Much: Methods are used that cover the whole lifecycle and are tailored to satisfy specific project needs. Methods are supported for each individual project.

B.3 Product factors

The following list contains those productivity factors that assess elements of the software product.

B.3.1 Software Complexity (Logical)

Computing, I/O-needs, algorithmic features and user interface requirements

1 – Very Much: Require functionally and technically difficult solutions. User interface is very demanding. Distributed databases (many database systems).

2 – Much: Require processing that is more demanding than normal. Database is large and demanding.

3 – Nominal: Are functionally typical. Normal standard database.

4 – Small: Are functionally clear. Database solution is clear.

5 – Very Small: Are routine. No need for user interfaces. Simple database.

B.3.2 Quality Requirements (Software)

The quality goals that product must satisfy

1 – Very Much: There exist quantified quality requirements. 100% satisfaction of the technical and functional goals must be satisfied, minimizing the amount of the maintenance work.

2 – Much: There exist formal reviews and inspections between all phases. Attention devoted towards documentation, usability and maintenance.

3 – Nominal: There exists proper documentation and critical features. Design and implementation is tested, modules/job flows tested, as well as walkthroughs. The maintenance work is planned.

4 – Small: Basic requirements are satisfied (documentation, implementation testing, system testing, module testing). No statistical control or reviews.

5 – Very Small: No explicit or measurable quality requirements. "Quick and dirty" is allowed.

B.3.3 Efficiency Requirements

The efficiency goals for the software

1 – Very Much: Efficiency of software is essential. There exist strict efficiency goals that require continuous attention and specific skills.

2 – Much: Specific quantified goals are in effect. Response time, transaction processing and turnaround time requirements can be reached by advanced design and implementation techniques.

3 – Nominal: The capacity level of the software is stable and predictable. The response time, transaction load and turnaround times are typical.

4 – Small: The efficiency goals and requirements are easy to attain.

5 – Very Small: No efficiency requirements are present that need attention and planning.

B.3.4 Installation Requirements

The training needs for users and number of different platforms

1 – Very Much: Software developed for a thousand or more users. The expected lifetime is long. There are several user organizations and several different platforms.

2 – Much: Large amount of training is needed for several organizations. Extra software needed for conversions, possible parallel runs. Several platforms.

3 – Nominal: Typical amount of training. Number of users is approximately 10 to 50.

4 – Small: Some training needed. Approximately 10 users.

5 – Very Small: No training needs. Only a few users.

B.4 People factors

The following list contains those productivity factors that assess elements of the people involved in the software project.

B.4.1 Staff Application Knowledge

Knowledge of project team (supplier and the customer) of the application domain

1 – Very Small: The application experience of the team is less than six months.

2 – Small: The application experience is small. Some members of the project staff have application experience. The average experience is 6 to 12 months.

3 – Nominal: The team has good experience in the application domain. The average experience is 1 to 3 years.

4 – Much: The team has good experience in the application domain. In addition, business dynamics is known. The average experience is 3 to 6 years.

5 – Very Much: The application area is well known to the team and the business as a whole. The average experience is more than six years.

B.4.2 Staff Analysis Skills

The analysis skills of the project staff at the project outset

1 – Very Small: No experience in requirements analysis or from similar projects.

2 – Small: Approximately one third of the staff has experience on analysis and design activities in similar projects.

3 – Nominal: 30 to 70% of the project staff has experience in analysis work. The project has also one very experienced member.

4 – Much: Most of the project staff has experience in specifications and analysis. The project manager is a professional in analysis work.

5 – Very Much: Project staff consists of first-class professionals with a strong vision and experience in requirements analysis.

B.4.3 Staff Tool Skills

The experience of the project team (supplier and customer) with development and documentation tools at the project outset

1 – Very Small: The team has no experience with the necessary tools. The average experience is less than 6 months.

2 – Small: The experience with tools is below average. Some members have experience with some of the tools. The average experience is 6 to 12 months.

3 – Nominal: The experience with tools is good for approximately half the team. Development and documentation tools are well known. The average experience is 1 to 3 years.

4 – Much: The tools needed for the project are well known to most team members. Some members can give support for their use. The average experience is 3 to 6 years.

5 – Very Much: All tools are well known to all team members. Support is available for specific needs of the project. The average experience is more than 6 years.

B.4.4 Staff Team Skills

The ability of the project team to work effectively and according to best project practices

1 – Very Small: Team is scattered. There are minimal project and management skills.

2 – Small: Only some team members have previous experience with similar projects. Team is not united as a group.

3 – Nominal: Most team members have experience with similar projects. The commitment towards project goals is good. No motivation to practice true team spirit.

4 – Much: The project group is very active and knows how to exploit the team concept effectively.

5 – Very Much: Team anticipates possible problems very well. Team can solve, in an effective way, most personal and team conflicts. There exists superior team spirit.

Appendix C: Source Code

This appendix contains the S-plus source code for the developed functions used in the simulation.

C.1 Univariate Missing Data Functions

The following section contains the source code for functions used in the simulation of univariate missing data.

C.1.1 generate( )

This is the highest level function, called from the command line, when evaluating the performance of MDTs on univariate missing data. There is a maximum of nine parameters that must be passed to this function, depending on which MDT is being evaluated. For listwise deletion and mean imputation, only the first six parameters must be passed: "random.pattern", "bias", "percent", "attribute", "technique", and "iterations". The "stand", "k" and "minkowski" parameters are only necessary when evaluating one of the hot-deck imputation MDTs. The following table describes each parameter.

Parameter	Description
random.pattern	Mechanism of missing data ("MCAR" or "MAR")
bias	Indicates bias towards "large" or "small" projects (MAR only)
percent	Percentage of missing data (e.g. 15)
attribute	Attribute in data set to be imputed with missing values (e.g. "T01")
technique	MDT to be evaluated ("CC" for listwise deletion, "MeanMethod" for mean imputation and "HotDeck" for hot-deck imputation)
iterations	Number of iterations to be performed in the simulation (e.g. 500)

stand (hot-deck only)	Standardization technique, if any, to be applied ("none" for no standardization, "z5" for Z_5, "trad" for z-score, and "meanabsolute" for mean absolute)
k (hot-deck only)	Number of nearest neighbours (e.g. 1)
minkowski (hot-deck only)	Distance function (1 for Manhattan, 2 for Euclidean)

For example, to evaluate listwise deletion for 15 percent missing data, using the MAR mechanism of missing data with bias towards small projects, on the "T01" attribute, over 500 iterations, one would type "generate("MAR","small",15,"T01","CC",500)" at the command line. This function calls three other functions listed in this appendix: hot.deck( ), naVector( ) and MARVector( ), which are discussed below.

This function returns the two summary measures, $MRE_{StudyPoint}$ and $Pred25_{StudyPoint}$ as described in Eqn. 3 and Eqn. 4.

The following is the source code:

```
generate <-
function(random.pattern,bias,percent,attribute,technique,iterations,stand="none",k=1,minkowski=2) {

#set up vector with desired percentage of NAs (MCAR mechanism)
#vector contains 143 elements (number of observations in the data set)
#NAs represent missing values
navector <- naVector(percent)

#set up vectors, one for each quintile, with different percentages of NAs (MAR mechanism)
#NAs represent missing values
#first and last quintiles contain 28 elements and the middle three contain 29 elements
#total number of elements in all vectors is 143 (number of observations in the data set)
G1Vector <- MARVector(percent,4)
G2Vector <- c(MARVector(percent,3),0)
G3Vector <- c(MARVector(percent,2),0)
G4Vector <- c(MARVector(percent,1),0)
G5Vector <- rep(0,28)

#get regression model and evaluative measures for train data set (no missing values)
#LATURIRestLog is the log-transformed train data set
#LATURIBank is the test data set
#LATURIBankLog is the log-transformed test data set
#WORKSUP represents effort, UUFP represents size, and T01 – T13 are productivity factors
#generate model using ordinary least squares regression
f.clean <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, LATURIRestLog)
#using model, predict effort in the test data set
```

```
worksup.predict.clean <- exp(predict(f.clean,LATURIBankLog))
#calculate magnitude relative error (MRE)
MRE.clean <- abs((worksup.predict.clean -
LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
#aggregate set of MRE values
MedianMRE.clean <- median(MRE.clean)
#calculate prediction at level 25 (Pred25)
Pred25.clean <- length(MRE.clean[MRE.clean <= 0.25]) / 63

#main body of simulation, repeat simulation of imputing missing data and using MDT "iterations"
#number of times
results.matrix <- sapply(seq(1:iterations),

function(x,navector,random.pattern,bias,attribute,technique,iterations,stand,k,minkowski,G1Vector,G2
Vector,G3Vector,G4Vector,G5Vector,f.clean,worksup.predict.clean){

  #impute missing values into train data set according to desired missing value mechanism
  switch(random.pattern,
   MCAR =
     {
     #make copy of train data set
     newLATURIRest <- LATURIRestLog
     #set desired attribute of data set with appropriate percentage of missing values
     newLATURIRest[,attribute] <- newLATURIRest[,attribute] + sample(navector)
     },
   MAR =
     {
     #make copy of train data set
     #if larger projects are to contain higher percentages of missing data
     if (bias == "large")
     #order data set accordingly
     newLATURIRest <- LATURIRestLog[rev(order(LATURIRestLog[,"UUFP"])),]
     else
     #smaller projects are to contain higher percentages of missing data.
     #order the data set accordingly
     newLATURIRest <- LATURIRestLog[order(LATURIRestLog[,"UUFP"]),]

     #set desired attribute of data set with appropriate distribution of NA
     newLATURIRest[,attribute] <- newLATURIRest[,attribute] +

       c(sample(G1Vector),sample(G2Vector),sample(G3Vector),sample(G4Vector),G5Vector)
     }
  )#switch

  #apply desired MDT
  switch(technique,
   #listwise deletion
   CC =
     {
     #generate model
     f <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, newLATURIRest)
     #predict effort of test data set based on model generated
     worksup.predict <- exp(predict(f,LATURIBankLog))

     #find magnitude relative error
     MRE <- abs((worksup.predict - LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
```

```
      #aggregate set of MREs
      MedianMRE <- median(MRE)
      #calculate Pred25
      Pred25 <- length(MRE[MRE <= 0.25]) / 63

      #return results
      return(c(MedianMRE,Pred25))
      },
    #mean imputation
    MeanMethod =
      {
      #find mean for attribute containing missing values
      MeanForAttribute <- mean(newLATURIRest[,attribute],na.rm=TRUE)
      #impute mean of attribute for each missing value
      newLATURIRest[,attribute] <- sapply(newLATURIRest[,attribute],
                           function(x,MeanForAttribute){if(is.na(x)) return(MeanForAttribute) else
return(x)
                          },MeanForAttribute)
      #generate model
      f <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, newLATURIRest)
      #predict effort of test data set based on model generated
      worksup.predict <- exp(predict(f,LATURIBankLog))

      #find magnitude relative error
      MRE <- abs((worksup.predict - LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
      #aggregate MREs
      MedianMRE <- median(MRE)
      #calculate Pred25
      Pred25 <- length(MRE[MRE <= 0.25]) / 63

      #return results
      return(c(MedianMRE,Pred25))
      },
    #hot-deck imputation
    HotDeck =
      {
      #identify observations in data set containing a missing value
      ccVector <- sapply(newLATURIRest[,attribute],function(x){if(is.na(x)) return(T) else return(F)})

      #separate complete observations from those that contain a missing value
      complete.cases <- newLATURIRest[!ccVector,]
      missing.cases <- newLATURIRest[ccVector,]

      #get list of attributes of data set that do not contain any missing values
      all.attribute.names <- names(newLATURIRest)
      complete.case.attributes <- all.attribute.names[all.attribute.names != attribute]
      #remove attribute containing missing values from "missing.cases"
      missing.cases <- missing.cases[complete.case.attributes]

      # perform standardization, if any
      if (stand != "none")
      {
        #get number of attributes in data set
        length.names.dataset <- length(names(newLATURIRest))
        #compute the central tendency value, depending on standardization method to be applied
```

```
  Central.complete <- switch(stand,
                z5=sapply(seq(length.names.dataset), function(x,d)
                        {min(d[,x])},newLATURIRest),
                traditional=sapply(seq(length.names.dataset), function(x,d)
                        {mean(d[,x])},newLATURIRest),
                meanabsolute=sapply(seq(length.names.dataset), function(x,d)
                        {mean(d[,x])},newLATURIRest))
    #compute the dispersion value, depending on standardization method to be applied
    Dispersion.complete <- switch(stand,
                z5= sapply(seq(length.names.dataset), function(x,d)
                        {diff(range(d[,x]))},newLATURIRest),
                traditional= sapply(seq(length.names.dataset), function(x,d)
                        {sqrt(var(d[,x],na.method="include"))},newLATURIRest),
                meanabsolute= sapply(seq(length.names.dataset), function(x,d)
                        {mean(abs(d[,x]- mean(d[,x]))) },newLATURIRest))
    #scale the data set
    newLATURIRest <- scale(newLATURIRest, Central.complete, Dispersion.complete)
   }
   #prepare data to be passed to hot.deck( )
   #observations that form the hot-deck
   hotdeck.obs <- newLATURIRest[!ccVector,]
   #observations that contain a missing value
   missing.obs <- newLATURIRest[ccVector,]

   #call to hot.deck( ),  function returns vector of values to impute.
   impute.values <- hot.deck(hotdeck.obs[,complete.case.attributes],
                missing.obs[,complete.case.attributes],complete.cases[,attribute],k,minkowski)

   #convert vector of values to impute to a data frame
   impute.values <- data.frame(impute.values)
   #add the name of the attribute
   names(impute.values) <- attribute
   #add the column with the imputed values to the observations with a missing value
   missing.cases <- cbind.data.frame(missing.cases, impute.values)

   #combine the complete observations with the "repaired" missing value ones
   newLATURIRest <- rbind.data.frame(complete.cases,missing.cases)

   #create model, compute evaluative measures and return results
   f <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, newLATURIRest)
   worksup.predict <- exp(predict(f,LATURIBankLog))
   MRE <- abs((worksup.predict - LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
   MedianMRE <- median(MRE)
   Pred25 <- length(MRE[MRE <= 0.25]) / 63
   return(c(MedianMRE,Pred25))
   }
  )#switch
 }#function
,navector,random.pattern,bias,attribute,technique,iterations,stand,k,minkowski,G1Vector,G2Vector,G
3Vector,G4Vector,G5Vector,f.clean,worksup.predict.clean)

#aggregate results over the number of iterations
#return the median of the absolute difference between calculated and actual(complete data set)
MdMRE <- median(abs(results.matrix[1,] - MedianMRE.clean))
MdPred25 <- median(abs(results.matrix[2,] - Pred25.clean))
```

```
#return summary of results
ResultSummary <- c(MdMRE,MdPred25)
return(ResultSummary)
}
```

C.1.2 hot.deck( )

This function is called by generate( ) when applying the hot-deck MDT. In total, five parameters are passed to this function. The following table describes each parameter.

Parameter	**Description**
complete.cases	The subset of observations that have no missing values
missing.cases	The subset of observations with missing values
complete.mv	The vector for the variable in the data set that contains missing values
k	The number of nearest neighbours
minkowski	The value to differentiate between Euclidean and Manhattan distances, 2 or 1 respectively

Note that the missing value column is separate from the rest of the columns. This means that if "T01" is the missing value column, then "complete.cases" and "missing.cases" do not have this column. That "T01" column for the complete data set is in "complete.mv".

This function returns a vector of values to impute, corresponding to the observations in missing.cases.

The following is the source code:

```
hot.deck <-
function(complete.cases,missing.cases,complete.mv,k,minkowski)
 {
 #get number of observations in the hot-deck
 complete.Nobs <- length(complete.cases[,1])
 #get number of observations that contain a missing value
 missing.Nobs <- length(missing.cases[,1])

 #get number of attributes in "complete.cases" and "missing.cases"
 length.names.cc <- length(names(complete.cases))
length.names.miss <- length(names(missing.cases))
#check that both "complete.cases" and "missing.cases" data sets have the same
#number of attributes
if (length.names.cc != length.names.miss)
  stop("Complete and missing set have different no of variables")

# weights are set to 1 by default
weights <- rep(1,7)

#convert to a matrix, this reduces computation time
missing.cases <- as.matrix(missing.cases)
complete.cases <- as.matrix(complete.cases)

#for each observation with a missing value
estimate <- sapply(seq(1:missing.Nobs),
          function(i,complete.Nobs,missing.cases,complete.cases,complete.mv,weights,k,minkowski)
           {
            #for each observation in the hot-deck
            #calculate the distance to the missing value observation
           distances <- sapply(seq(1:complete.Nobs),
                     function(j,i,missing.cases,complete.cases,weights,minkowski)
                      {
                       #calculate distance (Euclidean or Manhattan) for each individual attribute
                      dvec <- abs(missing.cases[i,]-complete.cases[j,])^minkowski
                      #aggregate over all attributes
                      d <- (sum(weights*dvec))^(1/minkowski)
                      #return distance from the current missing value observation to
                      #the current hot-deck observation
                      return (d)
                      }, i, missing.cases,complete.cases,weights,minkowski)

           #sort the possible values to impute based on the calculated distances
           orderedObsNo <- complete.mv[order(distances)]
           #select the number of nearest neighbours to be considered
           hot.deck.subset <-  orderedObsNo[1:k]

           #here a decision is made based on the value of k (number of nearest neighbours)
           #If k is 1, return the "hot.deck.subset" as the value to impute.
           #If k > 1, then must select one out of "hot.deck.subset" at random.
           if(k==1) return(hot.deck.subset)
           else
             {
             #create list of possible indices to choose from
             possible.indices <- seq(k)
             #find random index and return value in hot.deck.subset for this index
             return(hot.deck.subset[sample(possible.indices)[1]])
             }
           },
           complete.Nobs,missing.cases,complete.cases,complete.mv,weights,k,minkowski)

 # return the values to impute
 return(estimate)
}
```

C.2 Monotone Missing Data Functions

The following section contains the source code for functions used in the simulation of monotone missing data.

C.2.1 generate.multi( )

This is the highest level function, called from the command line, when evaluating the performance of MDTs on monotone missing data. There is a maximum of ten parameters that must be passed to this function, depending on which MDT is being evaluated. For listwise deletion and mean imputation, only the first seven must be passed: "random.pattern", "bias", "percent", "attribute1", "attribute2", "technique", and "iterations". The "stand", "k" and "minkowski" parameters are only necessary when evaluating one of the hot-deck MDTs. The parameters only differ from generate( ) in the addition of "attribute2". This is the second attribute that will be imputed with missing values. The parameter "attribute1" represents the attribute that will be imputed with more missing values than "attribute2", according to the pattern shown in Figure 2.

For example, to evaluate listwise deletion for 15 percent missing data, using the MAR mechanism of missing data with bias towards small projects, on the "T01" and "T02" attributes, and over 500 iterations, one would type "generate.multi("MAR","small",15,"T01","T02","CC",500)" at the command line. This function calls three other functions listed in this appendix: hot.deck.multi( ), naVector( ) and MARVector( ), which are discussed below.

This function returns the two summary measures, $MRE_{StudyPoint}$ and $Pred25_{StudyPoint}$ as described in Eqn. 3 and Eqn. 4.

The following is the source code:

```
generate.multi <-
function(random.pattern,bias,percent,attribute1,attribute2,technique,iterations,stand="none",k=1,mink
owski=2) {

#set vectors with desired percentage of NAs (MCAR mechanism)
#both vectors contain 143 elements (number of observations in the data set)
#the second vector contains half the amount of NAs than the first
#NAs represent missing values
navector1 <- naVector(percent)
```

```
navector2 <- naVector(percent / 2)

#set up vectors, one for each quintile, with different percentages of NAs (MAR mechanism) for
#each attribute ("attribute1" and "attribute2")
#final character in vector name("1" or "2") denotes vector for "attribute1" and "attribute2", respectively
#the vector for "attribute2" contains half the amount of NAs than that for "attribute1"
#first and last quintiles contain 28 elements and the middle three contain 29 elements
#total number of elements in all vectors for each attribute is 143 (number of observations in
#the data set)
G1Vector1 <- MARVector(percent,4)
G1Vector2 <- MARVector(percent / 2,4)
G2Vector1 <- c(MARVector(percent,3),0)
G2Vector2 <- c(MARVector(percent / 2,3),0)
G3Vector1 <- c(MARVector(percent,2),0)
G3Vector2 <- c(MARVector(percent / 2,2),0)
G4Vector1 <- c(MARVector(percent,1),0)
G4Vector2 <- c(MARVector(percent / 2,1),0)
G5Vector1 <- rep(0,28)
G5Vector2 <- rep(0,28)
#assemble two vectors with NAs for each attribute that will be imputed with  missing values.
MARVector1 <- c(G1Vector1,G2Vector1,G3Vector1,G4Vector1,G5Vector1)
MARVector2 <- c(G1Vector2,G2Vector2,G3Vector2,G4Vector2,G5Vector2)

#get regression model and evaluate measures for train data set (no missing values)
#LATURIRestLog is log-transformed train data set
#LATURIBank is the test data set
#LATURIBankLog is the log-transformed test data set
#WORKSUP represents effort, UUFP represents size, and T01 – T13 are productivity factors
#generate model using ordinary least squares regression
f.clean <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, LATURIRestLog)
#using model, predict effort in the test data set
worksup.predict.clean <- exp(predict(f.clean,LATURIBankLog))
#calculate magnitude relative error (MRE)
MRE.clean <- abs((worksup.predict.clean -
LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
#aggregate set of MRE values
MedianMRE.clean <- median(MRE.clean)
#calculate prediction at level 25 (Pred25)
Pred25.clean <- length(MRE.clean[MRE.clean <= 0.25]) / 63

#main body of simulation, repeat simulation of imputing missing data and using MDT "iterations"
#number of times
results.matrix <- sapply(seq(1:iterations),
  function(x,navector1,navector2,random.pattern,bias,attribute1,attribute2,technique,iterations,stand,
          k,minkowski,MARVector1,MARVector2,f.clean,worksup.predict.clean){

    #impute missing values into train data set according to desired missing value mechanism
    switch(random.pattern,
     MCAR =
        {
        #make copy of train data set
        newLATURIRest <- LATURIRestLog
        #create a random sample of the data set
        newLATURIRest <- newLATURIRest[sample(seq(143)),]
```

```
      #set desired attributes of data set with appropriate percentage of missing values
      newLATURIRest[,attribute1] <- newLATURIRest[,attribute1] + navector1
      newLATURIRest[,attribute2] <- newLATURIRest[,attribute2] + navector2
      },
    MAR =
      {
      #make copy of train data set
     if (bias == "large")
      #if larger projects are to contain higher percentages of missing data
      #order data set accordingly
      newLATURIRest <- LATURIRestLog[rev(order(LATURIRestLog[,"UUFP"])),]
      else
      #smaller projects are to contain higher percentages of missing data
      # order the data set accordingly
      newLATURIRest <- LATURIRestLog[order(LATURIRestLog[,"UUFP"]),]
      #create a random sample of the data set. The observations in the train data set are arranged
      #at random within each quintile.
      #generate a random sequence of observation numbers
      sequence_Obs <- c(sample(seq(1,28,1)),
                     sample(seq(29,57,1)),
                     sample(seq(58,86,1)),
                     sample(seq(87,115,1)),
                     sample(seq(116,143,1)))
      #order the data set according to sequence_Obs
      newLATURIRest <- newLATURIRest[sequence_Obs,]
      #set desired attributes of train data set with appropriate number of missing values
      newLATURIRest[,attribute1] <- newLATURIRest[,attribute1] + MARVector1
      newLATURIRest[,attribute2] <- newLATURIRest[,attribute2] + MARVector2
      }
   )#switch

   #apply desired MDT
   switch(technique,
    #listwise deletion
    CC =
     {
     #generate model
     f <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, newLATURIRest)
     #predict effort of test data set based on model created
     worksup.predict <- exp(predict(f,LATURIBankLog))

     #find magnitude relative error
     MRE <- abs((worksup.predict - LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
     #aggregate set of MREs
     MedianMRE <- median(MRE)
     #calculate Pred25
     Pred25 <- length(MRE[MRE <= 0.25]) / 63

     #return results
     return(c(MedianMRE,Pred25))
     },
    #mean imputation
    MeanMethod =
     {
     #get the mean for "attribute1"
     MeanForAttribute1 <- mean(newLATURIRest[,attribute1],na.rm=TRUE)
```

```
  #get the mean for "attribute2"
  MeanForAttribute2 <- mean(newLATURIRest[,attribute2],na.rm=TRUE)

  #replace each missing value of attribute with the mean for that attribute
  newLATURIRest[,attribute1] <- sapply(newLATURIRest[,attribute1],
                     function(x,MeanForAttribute1){if(is.na(x)) return(MeanForAttribute1)
                                                       else return(x)
                    },MeanForAttribute1)
  newLATURIRest[,attribute2] <- sapply(newLATURIRest[,attribute2],
                     function(x,MeanForAttribute2){if(is.na(x)) return(MeanForAttribute2)
                                                       else return(x)
                    },MeanForAttribute2)

  #generate model
  f <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, newLATURIRest)
  #predict effort in test data set based on model generated
  worksup.predict <- exp(predict(f,LATURIBankLog))

  #find magnitude relative error
  MRE <- abs((worksup.predict - LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
  #aggregate MREs
  MedianMRE <- median(MRE)
  #calculate Pred25
  Pred25 <- length(MRE[MRE <= 0.25]) / 63

  #return results
  return(c(MedianMRE,Pred25))
 },
#hot-deck imputation
HotDeck =
  {
  #identify observations in data set containing one or more missing values
  ccVector <- sapply(newLATURIRest[,attribute1],function(x){if(is.na(x)) return(T) else return(F)})
  #generate the set of observations that do not contain any missing values
  complete.cases <- newLATURIRest[!ccVector,]
   #generate the set of observations that contain one or two missing values
  missing.cases <- newLATURIRest[ccVector,]

  #generate vectors containing possible values to impute for "attribute1" and "attribute2" of
  #observations with missing values
  complete.mv1 <- complete.cases[,attribute1]
  complete.mv2 <- complete.cases[,attribute2]

  #get list of all attribute names
  all.attribute.names <- names(newLATURIRest)
  #get list off all attribute names excluding "attribute1"
  complete.case.attributes <- all.attribute.names[all.attribute.names!=attribute1]
  #get list of all attribute names excluding "attribute1" and "attribute2"
  #(this is the list of attributes that do not contain any missing values)
  complete.case.attributes.two <- all.attribute.names[all.attribute.names!=attribute1 &
                                                      all.attribute.names!=attribute2]

  #identify observations in data set that contain two missing values
  miss2Vector <- sapply(newLATURIRest[,attribute2],function(x){if(is.na(x)) return(T)
                                                      else return(F)})
  #get the set of observations that contain two missing values
```

```
missing.cases.two <- newLATURIRest[miss2Vector,]
#remove attributes that contain missing values
missing.cases.two <- missing.cases.two[complete.case.attributes.two]
#get the set of observations that contain only one missing value
missing.cases.one <- newLATURIRest[xor(miss2Vector,ccVector),]
#remove the attribute that contains missing values
missing.cases.one <- missing.cases.one[complete.case.attributes]

# perform standardization, if any
if (stand != "none")
{
  #The complete train data set is separated into three separate parts(A, B and C)
  #part A is the train data set excluding attribute1 and attribute2
  #part B is "attribute2" of the train data set
  #part C is "attribute1" of the train data set
  #This is done because standardization cannot be performed with NAs in the data set.

  #standardization of part A
  #get train data set excluding "attribute1" and "attribute2"
  newLATURIRest.xAtt2 <- newLATURIRest[,complete.case.attributes.two]
  #get number of attributes in "newLATURIRest.xAtt2"
  length.names.dataset <- length(names(newLATURIRest.xAtt2))
  #compute the central tendency value, depending on standardization method to be applied
  Central.complete <- switch(stand,
            z5=sapply(seq(length.names.dataset), function(x,d)
                    {min(d[,x])},newLATURIRest.xAtt2),
            traditional=sapply(seq(length.names.dataset), function(x,d)
                    {mean(d[,x])},newLATURIRest.xAtt2),
            meanabsolute=sapply(seq(length.names.dataset), function(x,d)
                    {mean(d[,x])},newLATURIRest.xAtt2))
  # compute the dispersion value, depending on the standardization method to be applied
  Dispersion.complete <- switch(stand,
            z5= sapply(seq(length.names.dataset), function(x,d)
                    {diff(range(d[,x]))},newLATURIRest.xAtt2),
            traditional= sapply(seq(length.names.dataset), function(x,d)
                    {sqrt(var(d[,x]))},newLATURIRest.xAtt2),
            meanabsolute= sapply(seq(length.names.dataset), function(x,d)
                    {mean(abs(d[,x]- mean(d[,x]))) },newLATURIRest.xAtt2))

  # scale the data set
  newLATURIRest.xAtt2 <- scale(newLATURIRest.xAtt2, Central.complete, Dispersion.complete)

  #standardization of part B
  #create a data frame of "attribute2" from train data set
  newLATURIRest.Att2 <- data.frame(newLATURIRest[,attribute2])
  #remove NAs from "newLATURIRest.Att2"
  newLATURIRest.Att2 <- newLATURIRest.Att2[!miss2Vector,]
  #add correct name of attribute
  newLATURIRest.Att2 <- data.frame(newLATURIRest.Att2)
  names(newLATURIRest.Att2) <- attribute2
  #set length of list of attribute names for "newLATURIRest.Att2"
  length.names.dataset <- length(names(newLATURIRest.Att2))
  # compute the central tendency value, depending on standardization method to be applied
Central.complete <- switch(stand,
          z5=sapply(seq(length.names.dataset), function(x,d)
                {min(d[,x])},newLATURIRest.Att2),
          traditional=sapply(seq(length.names.dataset), function(x,d)
                {mean(d[,x])},newLATURIRest.Att2),
          meanabsolute=sapply(seq(length.names.dataset), function(x,d)
                {mean(d[,x])},newLATURIRest.Att2))
# compute the dispersion value, depending on standardization method to be applied
Dispersion.complete <- switch(stand,
          z5= sapply(seq(length.names.dataset), function(x,d)
                {diff(range(d[,x]))},newLATURIRest.Att2),
          traditional= sapply(seq(length.names.dataset), function(x,d)
                {sqrt(var(d[,x],na.method="include"))},newLATURIRest.Att2),
          meanabsolute= sapply(seq(length.names.dataset), function(x,d)
                {mean(abs(d[,x]- mean(d[,x]))) },newLATURIRest.Att2))

# scale the data set
newLATURIRest.Att2 <- scale(newLATURIRest.Att2, Central.complete, Dispersion.complete)

#separate the set of standardized observations that contain two missing values from the rest
missing.obs2 <- newLATURIRest.xAtt2[miss2Vector,]
newLATURIRest.xAtt2 <- newLATURIRest.xAtt2[!miss2Vector,]

#get part C   (Note: part C does not need to be standardized)
#get "attribute1" column from "newLATURIRest"
att1.col <- data.frame(newLATURIRest[,attribute1])
#shorten "att1.col" by removing NAs that are from observations that also have a
#NA for "attribute2"
#this is done so that it will be the same length as parts A and B
att1.col <- att1.col[!miss2Vector,]
#convert to data frame and add appropriate name to column
att1.col <- data.frame(att1.col)
names(att1.col) <- attribute1

#combine parts A, B and C
#combine "newLATURIRest.xAtt2" (standardized), "newLATURIRest.Att2" (standardized) and
#"att1.col" (not standardized)
#this data set contains all observations except for those with two missing values
newLATURIRest <- cbind.data.frame(newLATURIRest.xAtt2,newLATURIRest.Att2,att1.col)
#identify observations in data set containing one missing value
ccVector <- sapply(newLATURIRest[,attribute1],function(x){if(is.na(x)) return(T) else return(F)})

#observations that form the hot-deck
hotdeck.obs1 <- newLATURIRest[!ccVector,]
#observations that contain only one missing value
missing.obs1 <- newLATURIRest[ccVector,]

#call to hot.deck.multi( ), function returns values to impute
impute.values <- hot.deck.multi(hotdeck.obs1[,complete.case.attributes],
                  missing.obs1[,complete.case.attributes],
                  missing.obs2[,complete.case.attributes.two],
                  complete.mv1,complete.mv2,attribute2,
                  k,minkowski)
}
```

```
else #no standardization
{
#prepare data to be passed to hot.deck.multi( )
#observations that form the hot-deck
hotdeck.obs1 <- newLATURIRest[!ccVector,]
#observations that contain only one missing value
missing.obs1 <- newLATURIRest[xor(ccVector,miss2Vector),]
#observations that contain two missing values
missing.obs2 <- newLATURIRest[miss2Vector,]

#call to hot.deck.multi( ), function returns values to impute
impute.values <- hot.deck.multi(hotdeck.obs1[,complete.case.attributes],
                     missing.obs1[,complete.case.attributes],
                     missing.obs2[,complete.case.attributes.two],
                     complete.mv1,complete.mv2,attribute2,
                     k,minkowski)
}#if - else

#add values to impute for observations with only one missing value
#get the values in "impute.values" that correspond to the values to impute
impute.values1 <- data.frame(impute.values[[1]])
#add the name of "attribute1"
names(impute.values1) <- attribute1
#add the column with the values to impute
missing.cases.one <- cbind.data.frame(missing.cases.one, impute.values1)

#add values to impute for observations with two missing values
#get the values in "impute.values" that correspond to the values to impute
impute.values2 <- impute.values[[2]]
attribute1.column <- impute.values2[1,]
attribute2.column <- impute.values2[2,]
#convert to a data frame
attribute1.column <- data.frame(attribute1.column)
attribute2.column <- data.frame(attribute2.column)
#add the names of the attributes
names(attribute1.column) <- attribute1
names(attribute2.column) <- attribute2
#add both columns of values to impute
missing.cases.two <- cbind.data.frame(missing.cases.two, attribute1.column, attribute2.column)

#combine the complete cases with the 'repaired' missing value ones
newLATURIRest <- rbind.data.frame(complete.cases,missing.cases.one,missing.cases.two)

#create model, compute evaluative measures and return results
f <- ols(WORKSUP ~ UUFP + T01 + T06 + T07 + T08 + T09 + T13, newLATURIRest)
worksup.predict <- exp(predict(f,LATURIBankLog))
MRE <- abs((worksup.predict - LATURIBank[,"WORKSUP"])/LATURIBank[,"WORKSUP"])
MedianMRE <- median(MRE)
Pred25 <- length(MRE[MRE <= 0.25]) / 63
return(c(MedianMRE,Pred25))
}
)#switch
}#function
,navector1,navector2,random.pattern,bias,attribute1,attribute2,technique,iterations,stand,k,minkowski,
MARVector1,MARVector2,f.clean,worksup.predict.clean)
```

```
#aggregate results over the number of iterations
#return the median of the absolute difference between calculated and actual(complete data set)
MdMRE <- median(abs(results.matrix[1,] - MedianMRE.clean))
MdPred25 <- median(abs(results.matrix[2,] - Pred25.clean))

#return summary of results
ResultSummary <- c(MdMRE,MdPred25)
return(ResultSummary)
}
```

C.2.2 hot.deck.multi( )

This function is called by generate.multi( ) when applying the hot-deck MDT. In total, eight parameters are passed to this function. The parameters are the same as those for hot.deck( ) except for the following:

Parameter	Description
missing.cases.one	The subset of observations with one missing value
missing.cases.two	The subset of observations with two missing values
complete.mv	The vector for the variable "attribute1" that contains missing values in the data set
complete.mv2	The vector for the variable "attribute2" that contains missing values in the data set

Note that "missing.cases.one", "missing.cases.two" and "complete.cases" do not have the column represented in "complete.mv". In addition, "missing.cases.two" does not have the column represented in "complete.mv2".

This function returns the values to impute, corresponding to the observations in "missing.cases.one" and "missing.cases.two".

The following is the source code:

```
hot.deck.multi <-
function(complete.cases,missing.cases.one,missing.cases.two,complete.mv,complete.mv2,attribute2,
k=1,minkowski=2)
 {
 #get list of attributes in "complete.cases"(the hot-deck)
 all.attribute.names <- names(complete.cases)
 #get list of all attributes excluding "attribute2"
 complete.case.attributes.two <- all.attribute.names[all.attribute.names!=attribute2]
 #get number of observations in hot-deck
 complete.Nobs <- length(complete.cases[,1])

 #get number of attributes in "complete.cases", "missing.cases.one" and "missing.cases.two"
 length.names.cc <- length(names(complete.cases))
 length.names.miss1 <- length(names(missing.cases.one))
 length.names.miss2 <- length(names(missing.cases.two))
 #check that "complete.cases" and "missing.cases.one" data sets have the same
 #number of attributes
 if (length.names.cc != length.names.miss1)
   stop("Complete and missing set have different no of variables")

 # weights are set to 1 by default
 weights <- rep(1,length.names.cc)
 weights2 <- rep(1,length.names.cc - 1)

 #create hot-deck for observations that contain two missing values
 #This hot-deck is the same as the hot-deck for observations containing one missing value except
 #the second missing attribute, "attribute2", is removed. (Observations with two missing values
 #do not have this attribute and so it is not needed in the hot-deck)
 complete.cases.two <- complete.cases[,complete.case.attributes.two]

#hot-deck MDT is performed in two parts (A and B).
#Part A applies the hot-deck MDT to observations that contain one missing value and
#Part B applies the hot-deck MDT to observations that contain two missing values.

#Part A
#get number of observations that contain one missing value
missing.Nobs1 <- length(missing.cases.one[,1])
#convert to matrix, this reduces computation time
complete.cases <- as.matrix(complete.cases)
missing.cases.one <- as.matrix(missing.cases.one)

#for each observation containing one missing value
estimate1 <- sapply(seq(1:missing.Nobs1),

function(i,complete.Nobs,missing.cases.one,complete.cases,complete.mv,weights,k,minkowski)
          {
          #for each observation in hot-deck
          #calculate the distance to the missing value observation
          distances <- sapply(seq(1:complete.Nobs),
                    function(j,i,missing.cases.one,complete.cases,weights,minkowski)
                     {
                     #calculate distance (Euclidean or Manhattan) for each individual attribute
                     dvec <- abs(missing.cases.one[i,]-complete.cases[j,])^minkowski
                     #aggregate over all attributes
                     d <- (sum(weights*dvec))^(1/minkowski)
                     #return distance from the current missing value observation to
                     #the current observation in the hot-deck
                     return (d)
                     }, i, missing.cases.one,complete.cases,weights,minkowski)
          #sort the possible values to impute based on the calculated distances
          orderedObsNo <- complete.mv[order(distances)]
          #select the number of nearest neighbours to be considered
          hot.deck.subset <-  orderedObsNo[1:k]
          #here a decision is made based on the value of k (number of nearest neighbours.
          #If k is 1, return the "hot.deck.subset" as the value to impute.
          #If k > 1, then must select one out of "hot.deck.subset" at random.
          if(k==1) return(hot.deck.subset)
          else
            {
            #create list of possible indices to choose from
            possible.indices <- seq(k)
            #find random index and return value in "hot.deck.subset" for this index
            index.to.pick <- sample(possible.indices)[1]
            #return value to impute
            return(hot.deck.subset[index.to.pick])
            }
          },
          complete.Nobs,missing.cases.one,complete.cases,complete.mv,weights,k,minkowski)

 #Part B
 #get number of observations that contain two missing values
 missing.Nobs2 <- length(missing.cases.two[,1])
 #convert to matrix, this reduces computation time
 complete.cases.two <- as.matrix(complete.cases.two)
 missing.cases.two <- as.matrix(missing.cases.two)

#for each observation containing two missing values
estimate2 <- sapply(seq(1:missing.Nobs2),
          function(i,complete.Nobs,missing.cases.two,complete.cases.two,
                 complete.mv,complete.mv2,weights2,k,minkowski)
          {
          #for each observation in hot-deck
          #calculate the distance to the missing value observation
          distances <- sapply(seq(1:complete.Nobs),
                  function(j,i,missing.cases.two,complete.cases.two,weights2,minkowski)
                   {
                   #calculate distance (Euclidean or Manhattan) for each individual attribute
                   dvec <- abs(missing.cases.two[i,]-complete.cases.two[j,])^minkowski
                   #aggregate over all attributes
                   d <- (sum(weights2*dvec))^(1/minkowski)
                   #return distance from the current missing value observation to
                   #the current observation in the hot-deck
                   return (d)
                   }, i, missing.cases.two,complete.cases.two,weights2,minkowski)
          #sort the possible values to impute, for both attributes, based on the calculated distances
          orderedObsNo <- complete.mv[order(distances)]
          orderedObsNo2 <- complete.mv2[order(distances)]
          #select the number of nearest neighbours to be considered
          hot.deck.subset <-  orderedObsNo[1:k]
          hot.deck.subset2 <-  orderedObsNo2[1:k]

          #here a decision is made based on the value of k (number of nearest neighbours)
          #If k is 1, return "hot.deck.subset" and "hot.deck.subset2" as the values to impute.
          #If k > 1, then must select one out of "hot.deck.subset" and "hot.deck.subset2" at random.
          if(k==1) return(c(hot.deck.subset,hot.deck.subset2))
          else
            {
            #create list of possible indices to choose from
```

```
                possible.indices <- seq(k)
                #find random index
                index.to.pick <- sample(possible.indices)[1]
                #return both values to impute for this index
                return(c(hot.deck.subset[index.to.pick],
                      hot.deck.subset2[index.to.pick]))
                }
              },
              complete.Nobs,missing.cases.two,complete.cases.two,complete.mv,
              complete.mv2,weights2,k,minkowski)
      #return the values to impute
      return(list(estimate1,estimate2))
}
```

C.3 Commonly Used Functions

The following section contains the source code for functions used in the simulation of both univariate and monotone missing data.

C.3.1 naVector( )

This function is called by generate( ) and generate.multi( ). The purpose of this function is to create a vector of NAs, representing missing values, to be imputed in one attribute in the data set when applying the MCAR mechanism. The function assumes that there are 143 observations in the data set and so if x = 10, then this function would produce a vector of 143 elements, 14 of which are NAs and the rest are zeros.

The following is the source code:

```
naVector <-
function(x)
{
  #number of NAs for vector
  numTrue <- round(143 * x * 0.01, digits=0)
  #number of zeros for vector
  numFalse <- 143 - numTrue
  #concatenate
  c(rep(NA,numTrue),rep(0,numFalse))
 }
```

C.3.2 MARVector( )

This function is called by generate( ) and generate.multi( ). The purpose of this function is to create a vector with the appropriate percentage of NAs, representing missing values, to be imputed in one quintile of the data set when applying the MAR mechanism. This corresponds to the quintiles of the 143 observation data set and discussed in Section 3.8. The function contains two parameters, "x", representing the total percentage of missing values to be imputed for the attribute, and "k", the number identifying the quintile.

The following is the source code:

```
MARVector <- function(x,k)
 {
  #number of NAs for vector
  numTrue <- round(143 * x / 10 * k * 0.01, digits=0)
  #number of zeros for vector
  numFalse <- 28 – numTrue
  #concatenate
  c(rep(NA,numTrue),rep(0,numFalse))
 }
```

References

[1] A. Albrecht: "Measuring Application Development Productivity". In *SHARE/GUIDE: Proceedings of the IBM Applications Development Symposium*, 83-92, 1979.

[2] A. Albrecht and J. Gaffney: "Software Function, Source Lines of Code, and Development Effort Prediction". In *IEEE Transactions on Software Engineering*, 9(6):639-648, 1983.

[3] S. Azen and M. Van Guilder: "Conclusions Regarding Algorithms for Handling Incomplete Data". In *Proceedings of the Statistical Computing Section*, American Statistical Association, 53-56, 1981.

[4] J. Bailar III, B. Bailar: "Comparison of Two Procedures For Imputing Missing Survey Values". In *Proceedings of the Section on Survey Research Methods*, American Statistical Association, 462-467, 1978.

[5] J. Bailey and V. Basili: "A Meta-Model For Software Development Resource Expenditures". In *Proceedings 5th International Conference on Software Engineering*, Lund, Sweden, July 1983, 107-116, 1981.

[6] B. Baker, C. Hardyck, and L. Petrinovich: "Weak Measurements vs. Strong Statistics: An Empirical Critique of S. S. Stevens' Proscriptions on Statistics". In *Educational and Psychological Measurement*, 26:291-309, 1966.

[7] K. Baker, P. Harris, and J. O'Brien: "Data Fusion: An Appraisal and Experimental Evaluation". In *Journal of the Market Research Society*, 31:153-212, 1989.

[8] R. Banker, S. Datar, and C. Kemerer: "A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects". In *Management Science*, 37(1):1-18, 1991.

[9] R. Banker and C. Kemerer: "Scale Economies in New Software Development". In *IEEE Transactions on Software Engineering*, 15(10):1199-1205, 1989.

[10] R. Banker and C. Kemerer: "The Evidence on Economies of Scale in Software Development". In *Information and Software Technology*, 36(5):275-282, 1994.

[11] D. Belsley, E. Kuh, and R. Welsch: *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. John Wiley and Sons, 1980.

[12] B. Boehm: *Software Engineering Economics*. Prentice-Hall, 1981.

[13] G. Bohrnstedt and T. Carter: "Robustness in Regression Analysis". In: H. Costner (ed.): Chapter 5, *Sociological Methodology*. Jossey-Bass, 1971.

[14] F. Brooks: *The Mythical Man Month*. Addison-Wesley, 1975.

[15] L. Briand, V. Basili, and W. Thomas: "A Pattern Recognition Approach for Software Engineering Data Analysis". In *IEEE Transactions on Software Engineering*, 18(11):931-942, 1992.

[16] L. Briand: "Quantitative Empirical Modeling for Managing Software Development: Constraints, Needs and Solutions". In: H.D. Rombach, V. Basili, and R. Selby (eds.): *Experimental Software Engineering Issues: Critical Assessment and Future Directions*. Springer-Verlag, 1993.

[17] L. Briand, K. El Emam, and S. Morasca: "On the Application of Measurement Theory in Software Engineering". In *Empirical Software Engineering: An International Journal*, 1(1):61-88, 1996.

[18] L. Briand, K. El Emam, and I. Wieczorek: "Explaining the Cost of European Space and Military Projects". In *Proceedings of the International Conference on Software Engineering*, 303-312, 1999.

[19] L. Briand, K. El Emam, D. Surmann, I. Wieczorek, and K. Maxwell: "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques". In *Proceedings of the International Conference on Software Engineering*, 313-322, 1999.

[20] L. Briand, T. Langley, and I. Wieczorek: "A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques". Technical Report, International Software Engineering Research Network, ISERN-99-15, 1999.

[21] L. Briand, K. El Emam, B. Freimut, and O. Laitenberger: "A Comprehensive Evaluation of Capture-Recapture Models for Estimating Software Defect Content". To appear in *IEEE Transactions on Software Engineering*, 2000.

[22] B. Clark: *The Effects of Software Process Maturity on Software Development Effort.* PhD Thesis, University of Southern California, 1997.

[23] B. Clark, S. Devnani-Chulani, and B. Boehm: "Calibrating the COCOMO II Post-Architecture Model". In *Proceedings of the 20th International Conference on Software Engineering*, 477-480, 1998.

[24] M. Colledge, J. Johnson, R. Pare, I. Sande: "Large Scale Imputation of Survey Data". In *Proceedings of the Section on Survey Research Methods*, American Statistical Association, 431-436, 1978.

[25] S. Conte, H. Dunsmore, and V. Shen: *Software Engineering Metrics and Models.* Benjamin/Cummings Publishing Company, 1986.

[26] B. Cox and R. Folsom: "An Empirical Investigation of Alternate Item Nonresponse Adjustments". In *Proceedings of the Section on Survey Research Methods*, American Statistical Association, 219-223, 1978.

[27] D. Dillman: *Mail and Telephone Surveys: The Total Design Method.* New York: Wiley-Interscience, 1978.

[28] D. Dillman: "Mail and Other Self-Administered Questionnaires". In: P.H. Rossi, J.D. Wright and A.B. Anderson (eds.): *Handbook of Survey Research.* New York: Academic Press, 359-378, 1983.

[29] L. Ernst: "Weighting to Adjust For Partial Nonresponse". In *Proceedings of the Section on Survey Research Methods*, American Statistical Association, 468-472, 1978.

[30] G. Finnie and G. Wittig: "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models". In *Journal of Systems and Software*, 39:281-289, 1997.

[31] B. Ford: "Missing Data Procedures: A Comparative Study". In *Proceedings of the Social Statistics Section*, American Statistical Association, 324-329, 1976.

[32] B. Ford: "An Overview of Hot-Deck Procedures". In W. Madow, I. Olkin, and D. Rubin (eds.): *Incomplete Data in Sample Surveys, Volume 2: Theory and Bibliographies.* Academic Press, 1983.

[33] J. Frane: "Some Simple Procedures for Handling Missing Data in Multivariate Analysis". In *Psychometrika*, 41(3):409-415, 1976.

[34] G. Furnival and R. Wilson: "Regressions by Leaps and Bounds". In *Technometrics*, 16(4):499-511, 1974.

[35] P. Gardner: "Scales and Statistics". In *Review of Educational Research*, 45(1): 43-57, 1975.

[36] O. Gilley and R. Leone: "A Two-Stage Imputation Procedure for Item Nonresponse in Surveys". In *Journal of Business Research*, 22:281-291, 1991.

[37] A. Gray and D. MacDonnell: "A Comparison of Techniques for Developing Predictive Models of Software Metrics". In *Information and Software Technology*, 39:425-437, 1997.

[38] Y. Haitovsky: "Missing data in regression analysis". In *Journal of the Royal Statistical Society*, B30:67-81, 1968.

[39] W. Hayes: *Statistics.* Fifth Edition, Hartcourt Brace College Publishers, 1994.

[40] T. Heberlein and R. Baumgartner: "Factors Affecting Response Rates to Mailed Questionnaires: A Quantitative Analysis of the Published Literature". In *American Sociological Review*, 43:447-462, 1978.

[41] D. Heitjan: "Annotation: What Can Be Done about Missing Data? Approaches to Imputation". In *American Journal of Public Health*, American Public Health Association, 548-550, 1997.

[42] Q. Hu: "Evaluating Alternative Software Production Functions". In *IEEE Transactions on Software Engineering*, 23(6):379-387, 1997.

[43] B. Ives, M. Olson, and J. Baroudi: "The Measurement of User Information Satisfaction". In *Communications of the ACM*, 26(10):785-793, 1983.

[44] R. Jensen: "A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Models". In *Proceedings of the International Society of Parametric Analysis*, 96-106, 1984.

[45] C. Jones: *Programming Productivity*. McGraw-Hill, 1986.

[46] H. Jung and R. Hunter: "Modeling the Assessor Effort in Software Process Assessment". *Submitted for publication, 1999.*

[47] J. Kaiser: "The Effectiveness of Hot-Deck Procedures in Small Samples". *Paper presented at the Annual Meeting of the American Statistical Association*, 1983.

[48] L. Kaufman and P. Rousseeuw: *Finding Groups in Data.* John Wiley & Sons, 1990.

[49] C. Kemerer: "An Empirical Validation of Software Cost Estimation Models". In *Communications of the ACM*, 30:416-429, 1987.

[50] J. Kim and J. Curry: "The Treatment of Missing Data in Multivariate Analysis". In *Social Methods & Research*, 6:215-240, 1977.

[51] B. Kitchenham and N. Taylor: "Software Project Development Cost Estimation". In *Journal of Systems and Software*, 5:267-278, 1985.

[52] B. Kitchenham: "Empirical Studies of Assumptions that Underlie Software Cost-Estimation Models". In *Information and Software Technology*, 34(4):211-218, 1992.

[53] J. Kromrey and C. Hines: "Nonrandomly Missing Data In Multiple Regression: An Empirical Comparison Of Common Missing-Data Treatments". In *Educational and Psychological Measurement*, 54(3):573-593, 1994.

[54] S. Labovitz: "Some Observations on Measurement and Statistics". In *Social Forces*, 46(2):151-160, 1967.

[55] S. Labovitz: "The Assignment of Numbers to Rank Order Categories". In *American Sociological Review*, 35:515-524, 1970.

[56] S-Y Lee and Y-M Chiu: "Analysis of Multivariate Polychoric Correlation Models with Incomplete Data". In *British Journal of Mathematical and Statistical Psychology*, 43:145-154, 1990.

[57] J. Lepowski, J. Landis, and S. Stehouwer: "Strategies for the Analysis of Imputed Data From A Sample Survey: The National Medical Care Utilization and Expenditure Survey". In *Medical Care*, 25:705-716, 1987.

[58] R. Little and D. Rubin: *Statistical Analysis with Missing Data.* John Wiley & Sons, 1987.

[59] R. Little: "Missing-Data Adjustments in Large Surveys". In *Journal of Business & Economic Statistics*, 6(3):287-296, 1988.

[60] R. Little: "A Test of Missing Completely at Random for Multivariate Data with Missing Values". In *Journal of the American Statistical Association*, 83(404):1198-1202, 1988.

[61] R. Little: "Regression With Missing X's: A Review". In *Journal of the American Statistical Association*, 87(420):1227-1237, 1992.

[62] N. Malhorta: "Analyzing Marketing Research Data with Incomplete Information on the Dependent Variable". In *Journal of Marketing Research*, 24:74-84, 1987.

[63] J. Matson, B. Barrett, and J. Mellichamp: "Software Development Cost Estimation Using Function Points". In *IEEE Transactions on Software Engineering*, 20(4):275-287, 1994.

[64] K. Maxwell, L. Van Wassenhove, and S. Dutta: "Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation". In *Management Science*, 45(6), 1999.

[65] G. Milligan and M. Cooper: "A Study of Standardization of Variables in Cluster Analysis". In *Journal of Classification*, 5:181-204, 1988.

[66] Y. Miyazaki, A. Takanou, H. Nozaki, N. Nakagawa, and K. Okada: "Method to Estimate Parameter Values in Software Prediction Models". In *Information and Software Technology*, 33:239-243, 1991.

[67] T. Mukhopadhyay and S. Kekre: "Software Effort Models For Early Estimation of Process Control Applications". In *IEEE Transactions on Software Engineering*, 18(10):915-924, 1992.

[68] A. Myrvold: "Data Analysis for Software Metrics". In *Journal of Systems and Software*, 12:271-275, 1990.

[69] J. Nunnally and I. Bernstein: *Psychometric Theory*. McGraw Hill, 1994.

[70] S. Oligny, P. Bouque, and A. Abran: "An Empirical Assessment of Project Duration Models in Software Engineering". In *Proceedings of the Eighth European Software Control and Metrics Conference*, 1997.

[71] L. Pickard, B. Kitchenham, and P. Jones: "Comments on: Evaluating Alternative Software Production Functions". In *IEEE Transactions on Software Engineering*, 25(2):282-283, 1999.

[72] S. Rahhal: *An Effort Estimation Model for Implementing ISO 9001 in Software Organizations.* Master's Thesis, School of Computer Science, McGill University, October 1995.

[73] S. Rahhal and N. Madhavji: "An Effort Estimation Model for Implementing ISO 9001". In *Proceedings of the Second IEEE International Software Engineering Standards Symposium*, 278-286, 1995.

[74] M. Raymond and D. Roberts: "A Comparison of Methods for Treating Incomplete Data in Selection Research". In *Education and Psychological Measurement*, 47:13-26, 1987.

[75] M. Raymond: "Missing Data in Evaluation Research". In *Evaluation & the Health Profession*, 9(4):395-420, 1986.

[76] L. Rea and R. Parker: *Designing and Conducting Survey Research: A Comprehensive Guide.* Jossey-Bass, 1992.

[77] P. Roth: "Missing Data: A Conceptual Review for Applied Psychologists". In *Personnel Psychology*, 47:537-560, 1994.

[78] P. Roth and F. Switzer: "A Monte Carlo Analysis of Missing Data Techniques in a HRM Setting". In *Journal of Management*, 21(5):1003-1023, 1995.

[79] D. Rubin: *Multiple Imputation for Nonresponse in Surveys.* John Wiley & Sons, 1987.

[80] D. Rubin, H. Stem, and V. Vehovar: "Handling 'Don't Know' Survey Responses: The Case of the Slovenian Plebiscite". In *Journal of the American Statistical Association*, 90(431):822-828, 1995.

[81] I. Sande: "Hot-Deck Imputation Procedures". In: W. Madow and I. Olkin (eds.): *Incomplete Data in Sample Surveys, Volume 3: Proceedings of the Symposium.* Academic Press, 1983.

[82] S. Siegel and J. Castellan: *Nonparametric Statistics for the Behavioral Sciences.* McGraw Hill, 1988.

[83] M. Shepperd and C. Schofield: "Estimating Software Project Effort Using Analogies". In *IEEE Transactions on Software Engineering*, 23(12):736-743, 1997.

[84] P. Spector: "Ratings of Equal and Unequal Response Choice Intervals". In *The Journal of Social Psychology*, 112:115-119, 1980.

[85] K. Srinivasan and D. Fisher: "Machine Learning Approaches to Estimating Software Development Effort". In *IEEE Transactions on Software Engineering*, 21(2):126-137, 1995.

[86] G. Subramanian and S. Breslawski: "Dimensionality Reduction in Software Development Effort Estimation". In *Journal of Systems and Software*, 21:187-196, 1993.

[87] University of Southern California: *COCOMO 2.0 Model User's Manual.* Version 1.1, 1994.

[88] S. Stevens: "Mathematics, Measurement, and Psychophysics". In: S. Stevens (ed.): *Handbook of Experimental Psychology.* John Wiley, 1951.

[89] P. Velleman and L. Wilkinson: "Nominal, Ordinal, Interval, and Ratio Typologies Are Misleading". In *The American Statistician*, 47(1):65-72, 1993.

[90] F. Walkerden and R. Jeffery: "Software Cost Estimation: A Review of Models, Process, and Practice". In *Advances in Computers*, 44:59-125, 1997.

[91] H. Weisberg: *Central Tendency and Variability.* Sage Publications, 1992.

[92] C. Wrigley and A. Dexter: "Software Development Estimation Models: A Review and Critique". In *ASAC 1987 Conference*, University of Toronto, 1987.

[93] C. Walston and C. Felix: "A Method of Programming Measurement and Estimation". In *IBM Systems Journal*, 1:54-73, 1977.