Universidad del País Vasco
Euskal Herriko Unibertsitatea

Unibertsitate Masterra
Konputazio **I**ngeniaritza eta **S**istema **A**dimentsuak

Konputazio Zientziak eta Adimen Artifiziala Saila –
Departamento de Ciencias de la Computación e Inteligencia Artificial

K I S A I C S I

Master Thesis

An investigation of imputation methods for discrete databases and multi-variate time series

Unai Garciarena Hualde

Advisor:

Roberto Santana Hermida

Department of Computer Science and Artificial Intelligence
Faculty of Informatics

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

KZAA /CCIA

September, 2016

Master eta Doktorego Eskola
Escuela de Máster y Doctorado
Master and Doctoral School

Master Degree Thesis:

An investigation of imputation methods for discrete databases and multi-variate time series

Unai Garciarena Hualde

Advisor: Roberto Santana
Intelligent Systems Group,
Department of Computer Science and Artificial Intelligence,
University of the Basque Country UPV/EHU,
Paseo Manuel de Lardizabal, 1
Donostia, 20018 Gipuzkoa, Spain
ugarciarena001@ikasle.ehu.es

Abstract. When it comes to information sets in real life, often pieces of the whole set may not be available. This problem can find its origin in various reasons, describing therefore different patterns. In the literature, this problem is known as Missing Data. This issue can be *fixed* in various ways, from not taking into consideration incomplete observations, to guessing what those values originally were, or just ignoring the fact that some values are missing. The methods used to estimate missing data are called Imputation Methods.

The work presented in this thesis has two main goals.
The first one is to determine whether any kind of interactions exists between Missing Data, Imputation Methods and Supervised Classification algorithms, when they are applied together. For this first problem we consider a scenario in which the databases used are discrete, understanding discrete as that it is assumed that there is no relation between observations. These datasets underwent processes involving different combinations of the three components mentioned. The outcome showed that the missing data pattern strongly influences the outcome produced by a classifier. Also, in some of the cases, the complex imputation techniques investigated in the thesis were able to obtain better results than simple ones.

The second goal of this work is to propose a new imputation strategy, but this time we constrain the specifications of the previous problem to a special kind of datasets, the multivariate Time Series. We designed new imputation techniques for this particular domain, and combined them with some of the contrasted strategies tested in the previous chapter of this thesis. The time series also were subjected to processes involving missing data and imputation to finally propose an overall *better* imputation method.
In the final chapter of this work, a real-world example is presented, describing a water quality prediction problem. The databases that characterized this problem had their own original latent values, which provides a real-world benchmark to test the algorithms developed in this thesis.

Glossary. Missing Data (MD), Imputation Method (IM), Missing Data Type (MDT), Time Series (TS), Supervised Classification (SC), Missing Completely At Random (MCAR), Missing At Random (MAR), Missingness that depends on unObserved Values (MUOV), Missingness that depends on Its own Value (MIV), Expectation-Maximization (EM), Multiple Imputation by Chained Equations (MICE), Hot Deck (HD), Database (DB), Last Value Carried Forward (LVCF),

Keywords: Missing Data, Imputation, Supervised Classification, Time Series

Acknowledgments

These paragraphs do not pretend to be a standard acknowledgments section. There has been a massive amount of people who has made an indispensable contribution towards me throughout my life, to get me who and where I am right now. For this reason, i cannot start mentioning names, first because it would cost me hours remembering each and every person, and second because I would forget to mention some names for sure, and that simply would not be fair.

For this reason, I have decided that these acknowledgments will be a self-service-like acknowledgments. However, I feel the need to thank some people particularly.

First of all I would like to thank my advisor in this thesis, Roberto in representation of all the people who has believed in me. Particularly, Roberto has helped me to take this thesis to an end, something that I could not have been able to do on my own, or without the help of such a great director. Also, he has made me much more confident on my capabilities, more than I had ever been.

On second place, but not least important, I would like to thank my mom, Irene, in representation of all the people who has supported me. That person who never, ever lets me down, and has done everything on her power to help me.

These groups are not exclusive, as there are lots of people that belong in both of them.

As I said, anyone who believes that they have a place in any of the groups may have my gratitude.

I would like also to thank the SIRENE® project, developed by the Rivages Pro Tech company, for providing a real-world example to enrich this work.

Table of Contents

CHAPTER 3
New Imputation Methods for Time Series Based on Regression and Temporality

CHAPTER 4
Advanced Imputation Methods for a real-world time series prediction problem

CHAPTER 5
Conclusions

List of Figures

List of Tables

Chapter 1

Introduction

Missing Data (MD), one of the most relevant problems in data quality nowadays, is a term used to refer to those attributes in an observation for which we have no value recorded. This could happen for various reasons, such as; a sensor not working properly, a worker not typing results in a program, or simply a person's refusal to answer a question. This issue finds a trivial solution in case deletion, which simply omits observations with missing values. However, when data is meant to be used in a machine learning task, this loss of information may not be acceptable.

This scenario raises a situation in which information wastage is not an option and more advanced strategies are required. One of the most common solutions for this problem are Imputation Methods(IM) [44]. These algorithms attempt to compute an accurate estimation of the unrecorded values, using the rest of the information in the database (DB).

As it will be discussed, an enormous variety of IMs have been developed. Roughly, IMs can be classified according to their computational complexity, since they may simply derive the value from a missing record using statistics from known values (e.g. the mean of a feature), or follow more complex strategies, i.e. Multiple Imputation [59].

A particular instance of the imputation problem needs to be addressed when the DB possesses certain characteristics, i.e. Time Series (TS) DBs. The main characteristic of these information sets is that the relation between observations differ depending on the distance separating them. This singularity invites IMs to contemplate the information available in near time-stamps in higher consideration rather than the one found in far ones, in temporal terms.

This thesis first investigates the effect of IMs over machine learning tasks, such as supervised classification over discrete DBs (databases in which it is assumed that there is no relation between observations), depending on the distribution of the missing data. Then, the domain of TS DBs is addressed, benefiting from the experience obtained in the discrete DB part. Finally, a single IM is proposed and used in a real world problem.

1.1 Objectives

This chapter will cover the task of placing the reader in a situation from which the rest of the thesis is understandable. First, the basic concepts are introduced. Then, machine learning concepts are illustrated. Finally, we can find the MD problem characterization.

1.2 Basic Concepts

1.2.1 DataBase

A DB is a collection of related data, considering data as known facts that can be recorded and have implicit meaning [14]. These recordings should attend to the following two restrictions:

1. A DB should be related to a certain field and all the data it contains should provide us with information (only) about it.
2. A DB must be structured and have a coherent order, which will help us gather its information.

A DB can be any size and as complex as imaginable. Also, it can be structured in many different ways.

We generated a fictional medical DB example in order to ease the understanding of this basic concept. For this instance, we will use a table-like DB (Table 1.1). This fictional dataset has been created simulating a telephonic poll, in which the participants were asked about the results they obtained from their last blood test.

Table 1.1: Table-structured Medical DB.

Age (years)	Weight (kg)	Sex (M, F)	Hemoglobin (g/dL)	Leukocytes (u/mcL)	Platelets (u/mcL)	Glucose (mg/dL)	Cholesterol T. (mg/dL)	Calcium (mg/dL)	Sodium (mEq/L)
23	74	M	15.3	8.700	136.000	80	174	9.9	145
25	92	F	13.4	11.300	146.000	113	247	8.3	150
49	81	F	16.5	6.600	147.000	92	145	8.7	136
46	104	M	13.1	12.100	148.000	116	264	8.1	143
15	60	F	14.4	4.500	145.000	80	185	9	143
53	83	M	14.2	7.300	145.000	93	194	9.8	147
27	94	F	13.1	5.400	85.000	123	238	8.7	151
52	100	M	17.9	11.900	144.000	126	8.7	8.3	154

The rows of Table 1.1 represent the observations of the structured DB since each one represents an observation of a different individual. Columns will be mentioned as features of those observations, which will be divided into two classes, namely, variables and attributes. We will use variables when we focus on features containing numeric values, commonly real ones. And attributes in case a column encloses discrete values.

Looking for correspondence between this example and the rules we mentioned before, the particular field would be blood test results, it is structured as a table, and it respects the same feature order for all the different observations.

1.2.2 Time Series

TS are successions of observed values for a certain variable referred to different moments [15]. The main difference between a TS and a *traditional*, discrete DB is that in a TS, observations follow a chronological order, which strengthens the relationship among close observations, unlike discrete ones, in which observation closeness mean nothing.

A TS can be defined as a sequence of variable values, ordered by the time they were measured:

$$TS = (t_i, x_i), i = 1, 2, ..., N$$

where we assume that each measurement (x_i) corresponds to a *timestamp* (t_i), which may take positive and ascending real values [50]. Even if time is a continuous variable, while working with this problem, it will be discretized, and referred as timestamp.

The TS analysis problem can be approached in two different ways:

1. Univariate approach: This uses a feature's historic values in order to make a model that describes its past behaviour. It may be used to make predictions, using its projection.
2. Multivariate approach: In this case, relations between two or more features will be exploited. With more than a single piece of information for each timestamp, regression models can be calculated, which will presumably give a more accurate prediction.

As it will be explained later in this work (Chapters 3 and 4), this work will treat DBs with multiple variables, which leads us to the possibility of approaching the problem from a multivariate approximation.

1.2.3 Missing Data

Missing Data (MD) can be caused by many reasons, and may follow different patterns. Recognizing these distribution types will probably be very important

for a later treatment. Here, we present the different types of MD, according to previous related work [5] [22] [24] [8] [44]. After its formal introduction, examples will be presented using the same synthetic medical DB described in Table 1.1.

- Missingness Completely at Random (MCAR): When a DB's lost values follow no pattern, in other words, measurements fail randomly. This type of MD presumably affects all the features equally, which uniformly distributes the information loss between them. If the amount of MD does not reach high percentages, case deletion could be a reasonable option to clear a DB from missing values. However, following this strategy may not be a ideal when machine learning techniques are set to be applied. IM could have a determinant effect in this task.

- Missingness At Random (MAR): MD is cataloged as MAR when a pattern can be identified, i.e., we can find a common factor in all the observations with missing values. For example, when a certain variable (with no MD) takes extreme values for an observation, two other variables tend to be missing for that same observation. A visual example can be found in Table 1.3 where we can come up with an *explanatory rule*, as all the cases with missing values are from males under 25 years old. Since all the missing records are concentrated in certain features, this Missing Data Type (MDT) may harm our data depending on the amount of information they hold.

- Missingness not at Random (MNAR): A dataset named to have MNAR values contains some lost observations that follow a pattern, but it results to be unidentifiable. This last kind of MD can be divided into two types:

  - Missingness that depends on unobserved variables (MUOV): It is very similar to MAR, but in this case the explanatory attributes mentioned above have not been collected. For example, in Table 1.4 we could have asked some extra questions, such as "Are you concerned about your health?" or "When was the last time you suffered a disease that required a blood test?". Possibly the most common reasons for which a person would get a test of this kind. Regarding classification and IMs, similar results to MAR are expected in both areas.

  - Missingness that depends on its value itself (MIV): This type of MD refers to the case where the loss of an attribute observation depends on its own value. MIV could have many causes, from a machine being unable to represent a value to a worker not knowing how to introduce it. A simple example can be found in Table 1.5, in which we can deduce that the missing values will probably be high in all cases. High values in Cholesterol, Glucose and Sodium are common in people with high weight/age. These individuals will possibly not be at ease with themselves, and just refuse to give that information away. MIV may be the

most harmful MDT, since basically all the information provided by a variable could be lost. IMs will probably offer poor results in case of a high percentage of MD, since missing values are usually focused in a small amount of features. However, if the variable does not contain important information about the class in a Supervised Classification (SC) problem, it probably won't harm its results.

Table 1.2: DB containing Missingness Completely At Random.

Age (years)	Weight (kg)	Sex (M, F)	Hemoglobin (g/dL)	Leukocytes (u/mcL)	Platelets (u/mcL)	Glucose (mg/dL)	Cholesterol T. (mg/dL)	Calcium (mg/dL)	Sodium (mEq/L)
23	74	M	15.3	8.700	136.000	?	174	9.9	145
25	?	?	13.4	11.300	?	113	247	?	150
?	81	F	16.5	6.600	147.000	92	145	8.7	136
46	?	M	?	12.100	148.000	?	264	8.1	?
15	60	F	14.4	4.500	145.000	80	?	9	143
53	83	M	14.2	7.300	?	93	194	9.8	147
27	94	?	13.1	5.400	85.000	123	?	8.7	151
?	100	M	17.9	?	144.000	126	8.7	8.3	154

Table 1.3: DB containing Missing At Random MDT. In this case, the causative variable would be Sex, and the key value, M.

Age (years)	Weight (kg)	Sex (M, F)	Hemoglobin (g/dL)	Leukocytes (u/mcL)	Platelets (u/mcL)	Glucose (mg/dL)	Cholesterol T. (mg/dL)	Calcium (mg/dL)	Sodium (mEq/L)
23	?	M	?	?	?	?	?	?	?
25	92	F	13.1	11.300	85.000	113	247	8.7	150
35	81	F	16.5	6.600	147.000	92	145	8.7	136
17	57	M	?	?	?	?	?	?	?
46	100	M	17.9	12.100	148.000	126	264	8.1	154
15	60	F	14.4	4.500	145.000	80	120	9	143
53	83	M	14.2	7.300	133.000	93	194	9.8	147
24	79	M	?	?	?	?	?	?	?

Table 1.4: DB containing Missingness depending on unobserved Variables.

Age (years)	Weight (kg)	Sex (M, F)	Hemoglobin (g/dL)	Leukocytes (u/mcL)	Platelets (u/mcL)	Glucose (mg/dL)	Cholesterol T. (mg/dL)	Calcium (mg/dL)	Sodium (mEq/L)
23	74	M	15.3	8.700	136.000	83	174	9.9	145
25	92	F	13.1	11.300	85.000	113	247	8.7	150
35	81	F	16.5	6.600	147.000	92	145	8.7	136
46	?	M	?	?	?	?	?	?	?
15	?	F	?	?	?	?	?	?	?
53	83	M	14.2	7.300	133.000	93	194	9.8	147

Table 1.5: DB containing Missingness depending on Its Value itself.

Age (years)	Weight (kg)	Sex (M, F)	Hemoglobin (g/dL)	Leucocites (u/mcL)	Platelets (u/mcL)	Glucose (mg/dL)	Cholestherol T. (mg/dL)	Calcium (mg/dL)	Sodium (mEq/L)
23	74	M	15.3	8.700	136.000	83	174	9.9	145
25	?	F	13.1	11.300	85.000	113	247	8.7	150
35	81	F	16.5	6.600	147.000	92	145	8.7	136
?	?	M	17.9	12.100	148.000	?	?	8.1	154
15	60	F	14.4	4.500	145.000	80	120	9	143
?	83	M	14.2	7.300	133.000	93	194	9.8	147

1.2.4 Mutual Information

The amount of MI between two random variables (X and Y) measures the uncertainty reduction in X when Y's value is known:

$$I(X,Y) = H(X) - H(X|Y)$$

Being H the Shannon's entropy of a variable:

$$H(X) = E(I(X)) = -\sum_{i=1}^{n} p(x_i) log_2 p(x_i)$$

This measure is used later, in Chapter 3, when it enables us to reduce the amount of information used by a proposed IM, thus, improving its performance in time consumption matter.

1.3 Classification

The world we currently inhabit generates and consumes an enormous amount of information. This volume can be measured in various zettabytes [25] (10^{21} bytes, while a gigabyte represents 10^9 and a terabyte 10^{12}, to put it into perspective).

Data Mining consists of extracting knowledge from all this information. Several procedures have been created with that target in mind. *Machine Learning (ML)* [6] holds some of them, processes that generalize patterns from a big volume of given observations. These operations are able to both predict and describe data. ML can be divided into three major branches, from which two are Supervised and Unsupervised Classification [60].

Both SC and UC attempt to categorize observations with an important difference, in SC these categories or *classes* are known beforehand, while in UC, they are not. SC algorithms need tagged observations (named training set), which means that we need to know what class each observation belongs to. This way, each class will have its own patterns, and we will be able to categorize new, untagged observations. Also, when the prediction aims for a continuous value rather than a discrete class, the process is not named SC, but Regression. UC simply groups measurements by the resemblance they have with each other. Since the main motivation of this work is to tag untagged observations, extracting information from tagged ones, we will use SC techniques [25, 64].

Within SC techniques three different groups can be distinguished; [44]

1. Rule Induction Learning: Algorithms that infer rules from the training set. These algorithms seek for regularities shared by observations belonging to the same class, and infer rules. Data with missing values is susceptible of being used by these classifiers, since it can be interpreted as a characteristic itself. Decision trees would be classified here.
2. Approximate Models: Algorithms that use the training set to build models. Since these algorithms build *functions* that (usually) depend on all features, missing values are not an option for these algorithms. Neural Networks, Regression and Support Vector Machines would be included here.
3. Lazy Learning: Algorithms that use the training set directly for classification purposes. This process implies the presence of measures of similarity of some kind, and uses them to classify. K nearest neighbors (K-NN) would be a perfect example of this type of methods.

1.4 Supervised Classification in Time Series

SCTS is a specific kind of SC applied to TS data. This problem consists on *extracting* information from a raw dataset. Raw datasets are formed by long regular time intervals that probably will contain some sub-intervals that break

monotony. The primary task is to recognize these *anomalies*, so that we can difference the time sequences when *nothing* is happening, from the time sequences in which *something* happened.

Imagine that we have sensors in different parts of a truck engine. These sensors control various measurements, i.e. the amount of fuel introduced in the cylinders, the temperature variance or smoke volume they produce when it is burnt, the revolutions per minute of the pistons, etc. We know what the range of these parameters should be when the entire engine works properly, but detecting anomalies and interpreting what they mean (for example, knowing what part is going to break down) could be performed by SCTS.

For our SCTS task, several classification techniques can be found, but they may be divided into three categories [69]:

FBC Feature based classification: These methods consist on *vectorizing* the TS, and applying a conventional classification method, as in a common SC problem.

DBC Distance based classification: In this case, functions that measure the distance between two series are used.

MBC Model based classification: These methods consist on building statistical models, such as Hidden Markov Model. These methods assume that the sequences in a class are generated by an underlying model.

1.5 Imputation Methods

Once the MDT has been identified, a reasonable solution is needed. A solution will or will not be suitable depending on the MD problem type. The most straightforward solution is to simply ignore cases with missing values and replace them by other complete observations, or omit unstable attributes (features with many missing entries). Those methods that discard incomplete observations are not very efficient when there is a high probability of having lost values, since we would discard big amounts of relevant information. Another possible approach is trying to predict the lost values. This method is called imputation. Several strategies have been proposed in the literature. In the following, we discuss some of the most commonly used: [22]:

I1. *Mean-Mode*, *Median* and *Most Frequent Value* imputation. These methods are self-explained by their names, as they calculate the statistics and simply ascribe them to gaps.

These methods can be very harmful to data, since they have a high probability of changing the distribution of the variables.

I2. Simple Random Imputation: This method takes a random value from the attribute containing an unrecorded value and duplicates it. This method excels in computational cost, but will likely bias the data distribution.

I3. *Last Value Carried Forward* (LVCF): This method consists of taking the last recorded value for that attribute and simply copying it.

This method could bias the inference process. It is mainly applied to TSs, therefore its application to other type of DB will possibly produce wrong values. Also, it is not valid for TSs in some cases. For example, when a value should experiment a considerably high increase or decrease between two timestamps, like in the probability of a newborn suffering sudden death as it grows, or when adjacent observations are unrelated.

I4. *Interpolation*: The working system of this method can also be easily deduced from its name. The interpolation method computes a function that fits both the previous and later values of a missing value stretch and fills it using the function.

As I3, this method can be very effective when observations are measurements of the same attribute on different timestamps, but it will provide unstable values when not.

I5. *Hot Deck(HD)*: When this method is applied over a missing value, a known value for the same feature in a different observation is assigned. This particular data entry is the nearest neighbor, although other similar approaches (such as clustering imputation) can be classified under this tag as well.

This IM may have a high computational cost, since it depends on the number of observations and variables the DB manages

I6. Imputation based on logical rules: Sometimes we can make use of our common sense to impute values when the MD volume is reasonably reduced and some expert knowledge may infer values *manually*.

For example, if a woman is polled and refuses to give information about her salary, we could infer it from other answers, like what her job is, and how many hours she spends at her workplace. Even if this method is not very accurate, it may work in some cases.

This imputation approach requires individual treatment for each DB, along with expert knowledge.

I7. SC algorithms as IM: This strategy uses a SC method to compute missing values. First, it sets a discrete feature containing missing values as the target class. Then, the observations with recorded values for that feature are selected as training set, and the classifier is learned. Finally, observations with unrecorded values are processed by the classification algorithm, and results are assigned to the missing spots.

Analogously, when the feature is continuous, regression may be applied. Also, UC could be exploited in a similar manner, lightly linking this method to I5.

I8 *Iterative Imputation*: Basically these methods impute the same missing values multiple times for a number of iterations. The common operation of these methods is to make initial guesses for missing values (e.g. mean imputation) and *reimputes* the same missing values using other (more complex, usually) methods. Once all variables have been *reimputed*, the cycle is repeated. An iteration limit or a tolerance value could be set as halting condition. These are its phases:

1 Initial values are imputed with a starting method, I1 for example.

2 A model is created.

3 Original missing values are recalculated by new values estimated from the model.

4 New values are imputed in the dataset.

Steps 2-4 are repeated [42].

I9 *Multiple Imputation*: These IMs also belong to the spectrum of computationally complex methods. This strategy replicates the DB m times and imputes them separately. *Submethods* used to perform imputation on these replicas may or not be different strategies (in case they are not, the submethod should be stochastic). Then one of two strategies can be followed;

The first one applies the analysis algorithm (Supervised Classification in our case) and then combine their results. The second strategy performs first the combination step, forming one single DB again and then the analysis is applied. The recommended m value used for these procedures has increased along with computation capacity, being the classical advice $3 \leq m \leq 5$ and having reached $20 \leq m \leq 100$ by some authors [67].

In Figure 1.1 an example diagram of these methods can be found. In this example we are searching for parameters α, β, γ and ω. With this objective, the dataset is cloned four times, being each copy processed with different imputation methods. As a final step, parameters are calculated, and combined.

There are many techniques for improving the performance of an imputation method, such as creating an auxiliary binary variable that takes '0' value when an attribute has its value missing, and '1' when it is known, or weighting correlated variables in the model, which requires little effort, and could be very useful in some cases. Also, some classification algorithms have their own internal imputation methods. For example, Autoclass [37] [38], calculates the probability of an instance belonging to each possible class, and infers missing data using those values.

The first 3 imputation methods (I1-I3) are quite simple and have no significant requirements in either implementation or execution. However, these methods will not calculate new values standing out of the data distribution, which will bias the variance of the imputed results towards 0. Since the amount of information provided by a variable is usually proportional to its variance, this is not a very good sign.

However, the last 6 methods, (I4-I9), are more complex in terms of implementation and computational time. While the previous methods (I1-I3) focus on choosing one value from the range of known values in a simple approach, methods I4 to I9 exploit all data available to compute those guesses. They tend to obtain better results than simple imputation, and may be required in some scenarios, but the computational cost must be considered when these strategies are chosen.

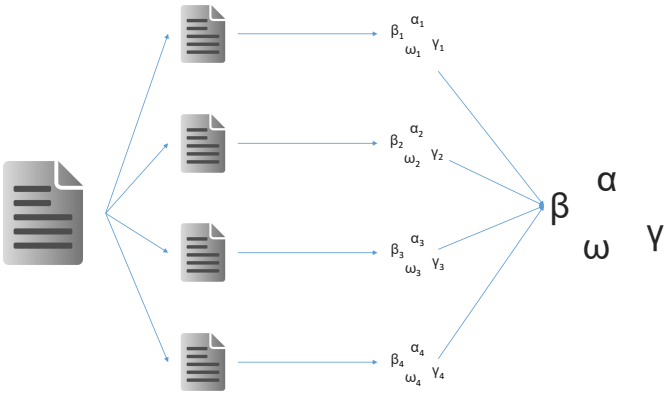


Fig. 1.1: Multiple imputation diagram.

1.6 Research motivation

The research presented in this thesis is focused on two related problems. First (Chapter 2), analyzes the relationship between the three principal components mentioned in this chapter: IMs, SC algorithms, and MDTs.

This is a problem that can be found in the real world with assiduity. Any aspect of work or life of society members can be recorded and potentially monetized, and when the amount of information is large enough that it becomes impossible to be processed by a human being, SC techniques are applied. Also, no machine is perfect, and can miss capturing, recording or sending data, which produces

MD. The IMs can help to automatically improve the quality of this data, for which they should be in a high consideration.

In this particular approach, we consider four different MDTs, while the rest of the literature fails to detect more than three patterns. Also, three types of IMs are considered. Simple, *bad* and *good* methods. The simple ones impute just performing a single computing operation. The *bad* methods are designed for a concrete type of DBs, the TSs. For this reason, the results they would produce will probably harm the data. Finally, the *good* ones will potentially produce good results. This way, we can use the inadequate methods to put into perspective the simple and the elaborated ones, to make a valid final conclusion.

For the second stage, we add some restrictions to the problem being treated, as it only contemplates TSs instead of discrete DBs (Chapter 3). This type of DB is also very common, as in some cases we need to monitor the evolution of an agent over the time. Analogously to the previous problem specification, this problem can also suffer from the MD issue, and IMs are again a good strategy to improve the data quality.

The experimental part considered the two most common types of MD in TSs and presented various simple IMs. Also, some combinations of these methods were developed in order to improve the performance offered by the simple ones, regarding both result accuracy and time consumption. The outcome of this problem is a IM ranking for TSs.

Finally (Chapter 4), the results obtained in the previous part are applied in a real world problem, with original MD on it. This problem describes the water quality regarding various aspects. The IM that got the best performance in terms of result accuracy was applied to the data, among other simple IM, to be able to put the results into perspective.

Chapter 2

An investigation of the relationship between Missing Data Types, Imputation Methods and Supervised Classification in Discrete Data

2.1 Objectives

The main goal of this chapter is to determine whether interactions between MDTs, IMs and SC algorithms exist or not. To fulfill this goal, some intermediate objectives are set. First, an in-depth analysis of the existing work is produced, to find out to what extent the hypothesis that supports the existence of the mentioned interactions can be valid. This analysis will enable us to start working on the border of the knowledge available in this research field. This exploration of related work will also provide ideas about the methodology that the subsequent experiments should be based on.

The next objective is to design experiments according to the knowledge obtained in the previous study of the literature. These experiments need to be proposed considering all the plausible variations in the three components being investigated in this chapter (MDT, IM and SC). Finally, the experiments will be executed, and the results that the experimentation provides will be analyzed to reach a final conclusion that validates or refuses the relation existence hypothesis proposed at the beginning.

2.2 Related work

Not many recent studies have researched the interaction between missing data configurations, MD handling methods, and supervised classification algorithms in such depth as ours, i.e., considering a large amount of MD, IM and SC types. In the following paragraphs, a number papers that have investigated this question are reviewed.

The most exhaustive investigation on the joint behavior of IMs and classification algorithms is the paper by Luengo, García, and Herrera [44]. They performed an analysis with 23 classification algorithms (grouped in three classes) combined

with 14 ways of dealing with MD. They selected 21 DBs, all of them containing natural missing values, ranging from 0.06 to 21.82 percent from the total values in the tables. Since they had no knowledge about the MD distribution, they assumed it to follow a MAR pattern. However, the experiments conducted by Luengo, García, and Herrera proved the positive effect of imputing data, and found evidence of the relation between the classifier type and the IM. The same authors developed the work presented in [43], which focuses on the Radial Basis Function Network (RBFN) classifier, pointing out the improvement of the classifier results when combined with a specific IM, an Event-Covering approach.

Batista and Monard [5] studied the effects of imputation over the C4.5 and CN2 classifiers. They chose four DBs from which three had no MD, and inserted varying percentages of missing values (10, 20, 30, 40, 50 and 60%) and then proceeded to classify them. The DB containing natural MD had its observations with MD removed, to have complete control over the missing values. Once the artificial MD was introduced, the DBs were put through a classification process with no imputation (note that C4.5 and CN2 have their own way of working with MD), mean/mode imputation, and K-nearest neighbors imputation (KNNI) (with $k = 10$). 10-NNI showed better performance than the other simpler treatments, but the experiments found limitations in the way MD was inserted, since only some of all the attributes chosen were affected. Also, only two basic IMs were considered.

Acuña and Rodríguez [1] evaluated the effect of three methods for dealing with MD on the misclassification error rate. They used the mean, median and KNNI IMs, and also considered case deletion. Then they evaluated the results these algorithms produced with two classifiers, Linear Discriminant Analysis (LDA) and K-nearest neighbors (KNN). This work adopted 12 datasets, 4 of which contained natural MD. These four DBs were pretreated in order to make their starting points similar by applying the IM. The results achieved by Acuña and Rodríguez showed that the IM effect on accuracy has a higher dependence on the problem rather than on the classification algorithm.

Farhangfar, Kurgan and Dy [16] examined the impact of performing MD imputation as a preprocessing step for posterior classification. They combined different versions of four imputation methods (HD, Naïve Bayes Polynomial multiple Regression and Mean) with six classifiers (Ripper 2, C4.5, RBFN, Support Vector Machine (SVM), KNN and naïve Bayes). They selected 15 DBs and considered 5, 10, 20, 30, 40 and 50% of MCAR-type MD in their experiments. They concluded that the improvement in the accuracies obtained from the dataset with MD and the imputed one had no relation with the percentage of MD. Also, differences in the improvement were shown for different IM-classification algorithm combinations, which led them to conclude that no universally best IM exists. The paper finally states that imputation is beneficial for machine learning tasks, overall.

Hruschka et al. [29] introduced two IMs based on Bayesian Networks and contrasted them with four other classic methods, namely, Expectation-Maximization (EM), Data Augmentation, Decision Trees, and Mean/Mode. Four datasets were used as a benchmark, which combined natural and artificial missing values introduced in the experiments. Four classifiers were considered in the experiments. This work concluded that IM obtaining closer values to original data (comparing imputations in artificially generated MD to values originally in those cells) did not necessarily produce better results when classifying.

Song et al. [63] performed an analysis similar to the one presented in [5], in which the effect of KNN-imputation over C4.5 classification was investigated. However, in their paper they considered three MDTs (MCAR, MAR and NMAR). Their conclusions regarding IM-C4.5 coincide with those found by Batista and Monard [5], and also state that the MD mechanism influences the classification task. Nevertheless, an in-depth investigation of this question was not addressed in [63].

Twala et al [66] also studied the three MDTs, along with 21 DBs, 7 IMs (containing both types, single and multiple IMs). However, their work focused on decision tree learners. The paper concluded that Multiple Imputation offers better results when the MD proportion is high, and also mentioned the importance of addressing MD patterns at the time of using the IM.

García-Laencina et al. [21] evaluated four different IMs (KNN, Self-Organizing Map, Multi-Layer Perceptron and EM) on three datasets (with varying amounts of artificially introduced MD) for posterior classification via Artificial Neural Network with six hidden neurons. They conclude their work stating that the data imputation paradigm generally required detailed study, probably due to the problem dependency factor mentioned in other works in the same area.

Ding and Simonoff [13] studied three types of MD (MCAR, MAR and NMAR), and introduced them into a set of 36 DBs considering eight patterns. These patterns vary in whether or not the missing value is related to other missing values, observed predictors, and the class variable. They used the combination of six IMs and two decision tree classifiers concluding that they are not highly affected by MD.

Gheyas and Smith [23] developed two new IMs: Generalized regression neural network Ensemble for Multiple Imputation (GEMI) and its single imputation version (GESI). They compared these two methods with other 25 IMs in 98 real-world datasets. They concluded that even though GEMI's computational cost is relatively high, both GEMI and GESI outperformed other multiple and single IMs, respectively.

Nogueira et al. [52] selected a real imbalanced DB with a high percentage of MD (over 23% and incomplete observations). Their DB preprocessing step consisted on removing attributes and observations with several missing observations and then imputing the gaps left. They concluded that Artificial Neural Network are

the right choice to solve this problem, but an expert understanding of the DB and the problem is set as necessary.

Saar-Tsechansky and Provost [61] compared three different methods to predict missing values (predictive value imputation, the distribution based imputation used by C4.5, and using reduced models) and followed the same approach of Luengo et al. [44] when it comes to imputing once the test and training set are separated. Their results showed that the reduced models strategy outperforms the other two classic IMs.

Matsubara et al. [45] present Corai, a semi supervised learning algorithm based IM, and compare it to KNNI and Mode imputation over three datasets with artificial MD introduced in varying percentages. They conclude that Corai's performance over other IMs improves as the MD percentage rises.

2.3 Experimental setup

The presented question that investigates the possible existence of interactions between MDT, IM and SC is explored by designing procedures that generate DBs with different types of MD, and using them as a benchmark, evaluate the effect of the MDT and the IMs on the accuracy of the classifier. Complete real-world DBs have been chosen to fulfill this experiment, introducing artificial MD in them. Following this strategy, the characteristics of the MD can be controlled. Then, the incomplete DBs are set to undergo an imputation process, prior to being tested via SC algorithms. The dependencies of the involved factors are determined once the whole procedure is measured when accuracies are computed for each combination.

2.3.1 Description of the databases investigated

A set of 10 datasets from the UCI Machine Learning Repository [41] have been chosen, all of them built in completely different contexts in order to achieve results as generic as possible, avoiding biasing the IMs and classifiers with a specific behavior from the data. DBs that describe diverse natural aspects (Forest, Biodeg, Climate, Leaf), containing measurements taken from a medical domain (Diabetic, BUPA, Thoracic), credit denial/approval (German), vehicle dimension analysis (Vehicle) and image pixel interpretation (Segmentation) have been used.

Table 2.1 contains a structural description of the DBs. The first three columns (Categorical, Integer and Real) provide information about the type of the elements contained in the DB. The following two columns, (N. Variables, N. Cases) refer to the number of measurements for each observation (including the class tag) and the amount of observations that the DB contains.

DB	Categorical	Integer	Real	N. Variables	N. Cases	DB Index
Leaf	✗	✗	✓	16	340	1
Vehicle	✗	✓	✗	18	946	2
German	✓	✓	✗	20	1000	3
BUPA	✓	✓	✓	7	345	4
Biodeg	✗	✓	✓	41	1055	5
Forest	✓	✓	✓	27	326	6
Diabetic	✗	✓	✓	20	1151	7
Climate	✗	✗	✓	18	540	8
Segmentation	✗	✗	✓	19	2310	9
Thoracic	✓	✓	✗	17	470	10

Table 2.1: Description of the original datasets used to generate the benchmark.

2.3.2 Strategies for generating Missing Data

As previously mentioned, MD can find its cause in different origins, which may result in various MDTs. The DBs used in this work do not contain MD, so that the patterns the missing values follow could be totally under control. In this section the strategies proposed to infuse MD into the DBs and the rationale behind each of these strategies are introduced.

One method has been implemented for each one of the four different MDTs described in this work. Note that all four algorithms introduce an amount of missing values equal to the 7% of all the values stored in the DB.

In order to produce results as close as possible to reality, these introduced methods generate MD in a stochastic way. The positions of the DB that are going to be modified are randomly selected according to the criterion defined for each type of MD. This way, it is possible to produce different results (instances of the DB with MD) each time the algorithm is executed, making our inference much richer.

Before presenting the strategies, a number of functions that serve as building blocks of the algorithms are presented:

- *random(I,nsamples)*: Takes one or two input parameters, depending on how many random numbers will be generated. I is the interval from which the random number will be produced. *nsamples* is the number of samples (numbers) that will be returned. For example, *random([0,10])* will generate a single arbitrary integer between 0 and 10, and *random([1,20], 3)* will create three distinct integer numbers between 1 and 20.
- *minIdex(A)*: Returns the index of the minimum value in the array A. It is used to obtain extreme values.
- *numObservations*: Represents the amount of observations in a DB (commonly corresponding to the number of rows in the DB).
- *numVariables*: Represents the number of variables in a DB, (analogously, usually corresponds to the number of columns in the DB).

– *length(A)*: Returns the length of A.

Variable *nV*, used in MAR (Section 2.3.2.2) and MuOV (Section 2.3.2.4), represents the amount of variables susceptible of losing their values. As explained previously about these MD patterns (Sections 2.3.2.2 and 2.3.2.4), the fact that their missing values are found in the same variables is their main characteristic. Therefore, when *random([0,y], nV)* is called, it is used to determine which variables will have their values go missing.

2.3.2.1 Missing completely At Random For this instance a simple algorithm is followed, in which two random numbers are generated and used as indexes in the DB. The element they point at has its value changed to *"NaN"*. This process is repeated until the predefined percentage is reached. The pseudocode is shown in Algorithm 2.1.

Algorithm 2.1: MCAR generating algorithm.

```
Input: DB
mdp: MD percentage
Output: DB with mdp% generated MD
begin
    x = numObservations(data)
    y = numVariables(data)
    for i ∈ [0, x · y · mdp//100] do
        data[random([0,x]), random([0,y])] = "NaN"
    return (data)
```

This algorithm simply generates two coordinates of the data matrix and sets the corresponding entry in the DB to "NaN".

2.3.2.2 Missing At Random This method is less straightforward than the previous one, since first it is needed to determine which variable is the *causative* of the MD. This implementation assumes a single variable as causative but *causative* variables of MAR could be multiple. The next step is to choose the variables that will have some of their values as missing, named *dependent* features. Since MAR causative variables tend to *cause* MD when they take extreme values, the observations with minimum values for those variables were chosen to get "NaN"s introduced in their dependent variables. Here, *maxInt* means the maximum interpretable integer for a programming language, and it is used in order to make that index ineligible. The pseudocode describing this method is shown in Algorithm 2.2.

This algorithm first chooses the causative variable and copies its values to another vector (*aux*), so that the original data is not modified apart from the

Algorithm 2.2: MAR generating algorithm.

```
1  Input:data: DB
2  mdp: MD percentage
3  nV: number of variables losing their values (3 in this paper)
4  Output: DB with mdp% generated MD
5  begin
6      x = numObservations(data)
7      y = numVariables(data)
8      causative = random([0,y])
9      aux = data[:,causative]
10     MDVariables = random([0,y]-{causative}, nV)
       /* Find observations with minimum value in causative                */
11     for i ∈ [0, x · y · mdp/100//nV] do
12         observations[i] = minIndex(aux)
13         aux[observation[i]]=maxInt
14     for i ∈ [0,length(observations)] do
15         for j ∈ [0, length(MDVariables)] do
16             data[observations[i], MDVariables[j]] = "NaN"
17     return (data)
```

introduction of the "NaN" values. Then the variables that will lose their values (*MDVariables*) are chosen. The first loop determines which observations will be the ones losing values for the *MDVariables*. It selects the minimum values for the causative variable (*aux*), and then makes them ineligible by assigning them a huge number. Finally, the two nested loops use all the information to introduce the MD.

2.3.2.3 Missingness that depends on its own Value

This algorithm is very similar to the one used to create the MAR pattern of MD, but instead of generating variables that will lose their values depending on the values of a causative variable, the missing values will be introduced in the causative itself. The pseudocode is shown in Algorithm 2.3

2.3.2.4 Missingness that depends on Unobserved Values

This algorithm is also quite similar to the one proposed for generating MAR MDT, but in this case the causative variable is unobserved. Therefore, the observations that will have missing values will be chosen randomly. The pseudocode is shown in Algorithm 2.4.

Algorithm 2.3: MIV generating algorithm.

```
1  Input:data: DB to MD be introduced
2  mdp: MD percentage
3  nV: number of variables losing their values (4 for our instance)
4  Output:DB with mdp% generated MD
5  begin
6      x = numObservations(data)
7      y = numVariables(data)
8      causatives = random([0,y], nV)
9      for i ∈ [0,length(causatives)] do
10         aux = data[:,causatives[i]]
11         for j ∈ [0, x · y · mdp/100//nV] do
12             observations[j] = minIndex(aux)
13             aux[observation[j]]=maxInt
14         for i ∈ [0,length(observations)] do
15             data[observations[i], causative] = "NaN"
16         return (data)
```

2.3.3 Imputation Methods

IMs differ in the class of DBs they can be applied to, their computational complexity, and the sophistication of the methods used to replace the MD. There are straightforward techniques to replace missing observations [5, 22, 38], that simply copy values from other observations (using similarity as a criterion, for example). Other more elaborated imputation algorithms use Parameter Estimation [27, 71] to replace the missing values from the available data. The first class of methods require less computational time and their results could be sufficiently good for some DBs. When these simpler strategies do not produce results that fit the expectations, parameter estimation can presumably provide more accurate results.

For this research, 8 IMs have been selected, including some of the most frequently used and more sophisticated approaches:

The initial three methods considered are *Mean-Mode*, *Median* and *Most Frequent Value* imputation. These methods have self-explanatory names, as they calculate the statistics and simply ascribe them to gaps.

Last Value Carried Forward imputation is also self-explanatory. This function searches the variable for the last available value, and uses it as its guess for the missing value.

Interpolation's working system can also be easily deduced from its name. The interpolation method computes a function that fits both the previous and later values of a missing value stretch and fills it using the function.

Algorithm 2.4: MuOV generating algorithm.

```
1  Input:data: DB
2  mdp: MD percentage
3  mde: Natural MD percentage in the DB
4  nV: number of variables losing their values (3 in this paper) Output:DB with mdp%
     generated MD begin
5      x = numObservations(data)
6      y = numVariables(data)
7      MDVariables = random([0,y], nV)
8      for i ∈ [0, x · y · mdp/100)//nV] do
9          observations[i] = random([0,x])
10     for i ∈ [0,length(observations)] do
11         for j ∈ [0, length(MDVariables)] do
12             data[observations[i], MDVariables[j] = "NaN"
13     return (data)
```

HD imputation can also have a high computational cost, since it depends on the number of observations and variables the DB has. This method computes the distance (e.g., euclidean distance) between an observation with one (or more) missing value(s) and replicates the value for that certain variable in the nearest observation.

Iterative Imputation basically imputes the same missing values multiple times. The common operation of these methods is to make initial simple guesses for missing values (e.g. mean imputation) and *reimputates* the same missing values using other (more complex, usually) methods. Once all the variables have been *reimputed*, the cycle is repeated. An iteration limit or a tolerance threshold could be set as a halting condition.

Multiple Imputation also belongs to the computationally complex method spectrum. This strategy duplicates the DB m times and imputes them separately. *Submethods* used to perform imputation on these replicas may or may not be different strategies. Next, one of two strategies can be followed. The first one applies the analysis algorithm (Supervised Classification in our case) and then combines their results. The second option first performs the combination step, forming one single DB again, being able this way to know the variance of the imputed values. Then the analysis is applied. The recommended m value used for this procedure has (and still is) increased as computation capacity has risen, being the classical advice $3 \leq m \leq 5$, having reached to $20 \leq m \leq 100$ by some authors [67].

All the IMs used in this work have been applied using third party implementations, as follows:

– For Mean, Mode and Most Frequent IMs, *Sci-Kit Learn*'s implementation was used [36].

– For Interpolation and LVCF, were applied using *Pandas* Python library's implementations [46].

– HD implementation in this work is the one implemented in the **R** package *HotDeckImputation* [3].

– For EM IM, the implementation in **R** package *Amelia II* was used [28].

– For Multiple Imputation using Chained Equations(MICE) as a *submethod*, this work used a Bootstrap approach algorithm implemented in the **R** package *mice* [10].

Note that some of the IMs considered here are more appropriate for a particular type of data (e.g. the interpolation method is conceived for data with real domain representation). If a given IM is not appropriate for one DB, this fact will probably be translated in the accuracy of the classifier for that particular DB.

2.3.4 Supervised Classification methods

In this section the classification methods evaluated in this work are presented. All the classifiers were implemented using the scikit-learn software [58] programmed in Python language. Also, when no details about the parameters used by the classifiers are provided it is assumed that they were applied with their defaults parameters in scikit-learn[1].

1. Regularized logistic regression with norm l1 [70]
2. Regularized logistic regression with norm l2 [70]
3. Linear discriminant analysis (LDA) [17]
4. Quadratic discriminant analysis (QDA) [18]
5. Support Vector Machine [30]
6. Radial Basis Function [56]
7. Gaussian Naïve Bayes classifier (GNB) [72]
8. Gradient boosting (GB) [19]
9. Random forests (RF) [9]
10. Decision tree (DT) [54]
11. k-nearest neighbor classifier (1NN) algorithm [2]
12. k-nearest neighbor classifier (3NN) algorithm [2]

[1] See `http://scikit-learn.org/stable/index.html` for more details on the code.

The following Tables 2.2 to 2.5 and paragraphs describe the main parameters chosen for each classifier:

Parameters/Classifier	Logistic Regression L1	Logistic Regression L2
Sklearn Function	Logistic Regression	Logistic Regression
Penalty	L1	L2
Dual Formulation	-	False
C (Regularization strength inverse)	1.0	1.0
Class Weights	Default (1 each)	Default (1 each)
Iteration Limit	100	100
Solver	Liblinear	Liblinear
Tolerance	0.0001	0.0001
Multiclass	One Versus Rest	One Versus Rest

Table 2.2: Logistic Regression classifiers with norms L1 and L2.

For the Linear Discriminant Analysis classification, a Least Squares approach was considered. Whereas for Quadratic Discriminant Analysis standard parameters were used.

Parameters/Classifier	SVM	RBF
Sklearn Function	SVC	SVC
Kernel	Linear	RBF
C	1.0	1.0
Gamma (Kernel Coefficient)	-	0.1 + 10e-17
Tolerance	0.001	0.001
Iteration Limit	5.000.000	5.000.000

Table 2.3: Parameters for the SVM based classifiers.

Parameters/Classifier	Gradient Boosting
SKlearn Function	GradientBoostingClassifier
Optimized Function	Deviance
Learning Rate	0.1
Amount of Estimators	100
Max. Depth of the Estimators	11

Table 2.4: Parameters for the Gradient Boosting Strategy classifier.

The last two algorithms are euclidean based K-NN classifiers. The first one considers one single neighbor, while the second uses three of them.

Classifier	RF	DT
SKlearn Function	RandomForestClassifier	DecissionTreeClassifier
Amount of Estimators	10	-
Criterion to measure quality of split	Gini Impurity	Gini Impurity
Splitting strategy	-	Best split
Max. Depth	Unlimited	Unlimited
Amount of features considered in splits	sqrt(n_features)	n_features

Table 2.5: Parameters for the Random Forest and Decision Tree classifier.

Some of these classifiers consider interactions between the features, some others incorporate regularization techniques, or take into account similarity metrics between the data. According to [44], the KNN classifiers would be lazy classifiers, while 9 and 10 would be tagged in the decision tree class. The rest of the classifiers, 1-8, are model constructing classifiers.

2.4 Experiments

2.4.1 Introduction

This section contains the description of the experiments carried out, i.e., which their goal was, the hypothesis they attempt to prove, its practical design and its execution. Finally, the outcome produced is evaluated, with the purpose of deducing whether the hypothesis is valid or not, and in case it is, to what extent.

2.4.2 Goal

The goal of this experimental setup is to prove right or wrong the hypothesis this work is based on; the one that suggests that the three different components (MDT, IM and SC) involved in the problem are interrelated and will offer better or worse results when combined with different partners. More specifically, the following questions are investigated:

- Which is the overall behavior of the imputation methods when all datasets, MDTs, and classification methods are considered?
- Which is the overall behavior of the classification algorithms when all datasets, MDTs, and imputation methods are considered?
- Have the different types of MD a different effect in global classification accuracies achieved by the classifiers?
- Is it possible to identify any type the relationship between MDTs, IM and classifiers?

In order to set a base knowledge that allows valid conclusion extraction, the three components are evaluated separately. This will provide a *standard* behavior for different configurations of each component that will be used as an indicative of a possible increase or decrease of accuracy of a determined factor when used along others.

2.4.3 Design

First, ten non-temporal DBs from the UCI Machine Learning Repository [41] were chosen, aiming for the ones containing no natural missing values. Then, these DBs were treated to have MD artificially inserted. Next step consisted in imputing/estimating the missing values using eight different IMs. Finally, the data was used in a SC process, and validated using 5-fold cross validation.

This paragraph explains the process used to study the MD-IM-SC relation, roughly. First of all, all 4 types of MD were introduced into each and every one of the 10 selected datasets, generating 4 different DBs with the same origin, but different missing values patterns. This process was executed 30 times, to obtain a sample of 30 different DBs with the same combination. This part of the process produces 10 DBs $\times$ 4 MDtypes $\times$ 30 instances = 1,200 DBs with missing values. All DBs will be different, even if their origin is the same data, the inserted missing values will make them unique. Then, eight different imputation methods (Section 2.3.3) were applied to each of the generated DBs, to obtain a total 1,200 DB $\times$ 8 IM = 9,600 different, complete DBs. Finally, all these DBs were subjected to 12 distinct SC algorithms (Section 2.3.4), obtaining a final output of 9,600 $\times$ 12 = 115,200 distinct *accuracies*. Figure 2.1 captures the complete process for one single DB.

2.4.4 Analysis

Once the experiments were finished, the next task is to analyze the data produced. As an initial step of the analysis, the independent effect of two of the three factors considered in this work are investigated, focusing in the accuracies obtained by the classifiers for each of the MDTs.

2.4.4.1 Overall Missing Data Types behavior Table 2.6 shows the mean accuracy obtained by all classifiers for each MDT. To compute the accuracy shown in each row of Table 2.6, the mean is computed using 10 DB $\times$ 30 Inst. $\times$ 8IM $\times$ 12 SC = 2880 executions of the classifiers, using in every case those DBs in the benchmark with same MDT.

As it can be observed in Table 2.6, the mean accuracies of the classifiers for MAR, MIV and MuOV are very similar, showing an insignificant gap of 0.0016

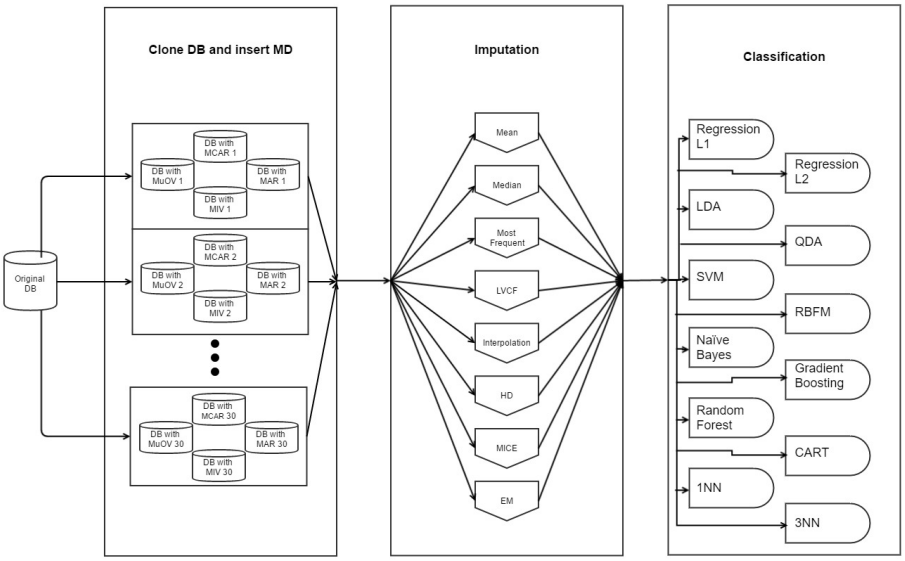


Fig. 2.1: The process DBs undergo.

MD Type	Mean Accuracy
MAR	0.5627
$MCAR$	0.5514
$MuOV$	0.5611
MIV	0.5624

Table 2.6: Average accuracies for each MDT.

among them. However, the mean accuracy generated for MCAR shows an average difference with respect to the other three MDTs of around 0.01.

For further understanding of the effect produced by the MD pattern over the accuracy of the SCs, statistical differences were searched for in the accuracies generated by the classifiers. First, all accuracies were grouped by the original DB they were obtained from, and then separated into 4 subgroups considering the MDT as a criterion. Next, the Kruskal-Wallis H-test [2] was applied to determine whether the 4 groups originated from the same distribution. If the null hypothesis was rejected, ($p - value < 0.05$), a post-hoc test was applied to all pairs of MDTs looking for differences between them. The Dunn's test[3] with *Bonferroni* correction was the one chosen for pairwise comparisons. Table 2.7 shows the results of the tests for the 10 original DBs introduced in Table 2.1.

	DB 1	DB 2	DB 3	DB 4	DB 5	DB 6	DB 7	DB 8	DB 9	DB 10
MAR vs MCAR	✓	–	✓	✓	✓	✓	✓	✓	✗	✓
MAR vs MuOV	✗	–	✗	✗	✗	✗	✓	✓	✗	✗
MAR vs MIV	✗	–	✓	✗	✗	✗	✓	✓	✓	✓
MCAR vs MuOV	✓	–	✗	✓	✓	✓	✓	✓	✗	✓
MCAR vs MIV	✓	–	✓	✓	✓	✓	✗	✓	✓	✗
MuOV vs MIV	✗	–	✓	✓	✗	✓	✗	✓	✓	✓

Table 2.7: Statistical differences between MDTs for each DB. "✓" represent that significant differences were found, while "✗" represents the opposite.

The analysis of Table 2.7 reveals that only one DB, DB2, offered no initial difference between the four MDTs when Kruskal-Wallis was applied. For all other DBs the Dunn's test showed statistical evidence of differences between at least three pairs of MDTs. From the 35 cases where statistical differences were found in pair-wise comparison, MCAR was involved in 22, which makes around 63% of the total, while it appeared in only half of the comparisons. Together with an average classifier accuracy lower than the other three MDTs (as shown in Table 2.6), this fact indicates that MCAR has a different effect in terms of the challenge it imposes to the classification task.

2.4.4.2 Overall Imputation Methods behavior To search for statistical differences between the IMs, another similar independent experiment was designed. In this case, the problem to be solved was determining the existence of differences for a particular combination of the 10 original DBs, the 4 MDTs and 12 classifiers. For each of these possible combinations, 30 instances and the correspondent accuracy results for the 8 IMs are available. This means that IMs

[2] http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.mstats.kruskalwallis.html
[3] https://cran.r-project.org/web/packages/dunn.test/dunn.test.pdf

for $10 \times 4 \times 12 = 480$ different combinations will be compared. Note that in each combination all the accuracies would have been calculated from the same original DB, having the same MDT inserted, and having been applied the same classification algorithm. For each of the 480 combinations, the accuracy results of the IMs can be sorted, in order to be compared in a table of 30 rows (one for each unique DB instance) and 8 columns (one for each IM).

In each of these 480 tables the Kruskal-Wallis H-test was applied to determine whether significant differences exist between the IMs. If such differences exist, a Dunn test is used as a post-hoc analysis to determine which pair(s) of IMs had significant differences between them, resulting a similar procedure as the one used before.

In Table 2.8 the results of the comparison between each pair of IMs for all MDTs are summarized. Cell (r, c) of Table 2.8 indicates the number of times IM c was significantly better than the other imputation method r in all possible combinations of DBs, MDT, and classifiers. The last row shows the number of times each algorithm was outperformed by the others (o^-). The last column shows the number of times each algorithm outperforms the rest (o^+).

IMs	Mean	Median	Most Frequent	LVCF	Interpolation	HD	MICE	EM	Total
Mean	0	7	27	59	92	22	18	20	245
Median	3	0	19	55	95	18	15	22	227
Most Frequent	8	7	0	51	89	10	15	31	211
LVCF	27	29	41	0	11	14	18	20	160
Interpolation	24	29	35	7	0	18	17	17	147
HD	39	33	52	65	83	0	3	54	329
MICE	41	34	49	65	74	15	0	47	325
EM	19	22	37	35	58	15	9	0	195
Total	161	161	260	337	502	112	95	211	1,839

Table 2.8: Results of the statistical tests on the difference between the performance of the IMs.

The number of pair-wise comparisons for each group was $\frac{8\times(8+1)}{2} = 36$. As it can be observed from Table 2.8, the number of significant differences was 1839 out of the 36 comp. $\times$ 12 class. $\times$ 10 MD $\times$ 4 MDT = 17,208 total number of pair-wise comparisons done. Therefore, 10.64% of the comparisons were cataloged as significantly different.

The analysis of Table 2.8 reveals a number of facts about the behavior of the IMs:

- HD and MICE are the best IMs outperforming the other IMs more frequently.
- LVCF and Interpolation exhibit the poorest performance being outperformed more often by the other IMs.
- Despite their simplicity, the Mean, Median, and Most Frequent IMs show an acceptable performance.

2.4.4.3 Imputation Method behavior for each Missing Data Type

Since significant differences were found among MDT and IM separately, this section analyzes potential relations between the two components.

In Tables 2.9, 2.10, 2.11 and 2.12 the results shown in Table 2.8 are classified by MDT. From these tables, the influence of MDT on the IM can be analyzed.

IMs	Mean	Median	Most Frequent	LVCF	Interpolation	HD	MICE	EM	Total
Mean	0	0	0	5	20	2	3	6	36
Median	0	0	0	5	18	2	3	6	34
Most Frequent	0	0	0	7	18	1	2	7	35
LVCF	2	3	2	0	1	3	2	1	14
Interpolation	5	6	5	1	0	4	2	1	24
HD	1	2	3	8	13	0	0	6	33
MICE	4	4	4	9	11	1	0	3	36
EM	3	3	3	2	8	2	0	0	21
Total	15	18	17	37	89	15	12	30	233

Table 2.9: Table 2.8 results filtered by MAR MDT.

IMs	Mean	Median	Most Frequent	LVCF	Interpolation	HD	MICE	EM	Total
Mean	0	3	11	20	30	8	7	10	89
Median	1	0	9	18	36	6	8	12	90
Most Frequent	5	6	0	25	38	5	10	19	108
LVCF	4	6	15	0	8	3	4	3	43
Interpolation	4	7	12	5	0	6	6	3	43
HD	1	1	9	18	26	0	0	9	64
MICE	1	0	7	19	24	0	0	11	62
EM	2	6	12	7	19	5	2	0	53
Total	18	29	75	112	181	33	37	67	552

Table 2.10: Table 2.8 results filtered by MIV MDT.

IMs	Mean	Median	Most Frequent	LVCF	Interpolation	HD	MICE	EM	Total
Mean	0	0	0	3	8	2	1	1	15
Median	0	0	0	5	7	2	1	1	16
Most Frequent	0	0	0	3	9	1	0	1	14
LVCF	5	4	5	0	0	3	4	3	24
Interpolation	3	3	3	0	0	4	3	2	18
HD	3	0	3	8	10	0	1	9	34
MICE	1	1	1	5	6	1	0	1	16
EM	0	1	2	0	3	2	1	0	9
Total	12	9	14	24	43	15	11	18	146

Table 2.11: Table 2.8 results filtered by MuOV MDT.

IMs	Mean	Median	Most Frequent	LVCF	Interpolation	HD	MICE	EM	Total
Mean	0	4	16	31	34	10	7	3	105
Median	2	0	10	27	34	8	3	3	87
Most Frequent	3	1	0	16	24	3	3	4	54
LVCF	16	16	19	0	2	5	8	13	79
Interpolation	12	13	15	1	0	4	6	11	62
HD	34	30	37	31	34	0	2	30	198
MICE	35	29	37	32	33	13	0	32	211
EM	14	12	20	26	28	6	6	0	112
Total	116	105	154	164	189	49	35	96	908

Table 2.12: Table 2.8 results filtered by MCAR MDT.

Analysis of Tables 2.9 - 2.12 shows how the performance of the IMs is in relation to the MDTs.

- Influence differences have been detected depending on the MDT applied to the DB. This is clearly illustrated by comparing the case of the MuOV and MCAR MDTs. In the first case there are relatively few statistical differences between the IMs (146), while in the second case these differences are much higher (908).
- The highest scores performed by MICE and HD were achieved in MCAR suggesting that simple IMs can offer good enough results in MIV, MAR and MuOV, but overall, and specifically in MCAR, computationally complex methods are the most reliable ones.

The pair-wise comparison between IMs reveals only a partial picture of the overall behavior of the IMs. To further understanding of the relation between IM and MDT, another experiment was designed to evaluate the rank of the IMs. First, all the accuracy results were divided in 40 groups. Each group contains all classification results for the combinations of the 10 original DBs and the 4 MDTs. Each group comprises 8IM $\times$ 12SC $\times$ 30*Inst.* = 2880 classification accuracies. In each group, the accuracies were sorted from the highest to the lowest, and split into 3 equally sized groups: *High*: the experiments with the highest accuracy; *Low*: experiments with the lowest accuracy; and *Medium*: the rest of the experiments. By inspecting each group it can be determined, for each original DB and MD method, which combination of IMs and classifiers were the most frequent in each of the groups, allowing us to detect high performing IMs, SCs, and combinations of the two.

Then, all groups representing the same level (High, Medium or Low) from all DBs, MDTs, IMs and SC algorithms were merged, resulting three supergroups that divided the overall performance of all combinations in three. Figure 2.2 shows the polar charts describing the frequency of the IMs in the supergroups *High* and *Low* according to the MDT. Considering this, an IM being present with larger frequency in its high section rather than in the corresponding low section suggests that for a concrete MDT the IM had a relatively good performance. However, a section representing *low* surpassing *high*, denotes a bad performance

by the IM, which potentially means that the imputation hurt the data. Finally, both sections being even shall be prove of an imputation procedure having little effect on posterior classification.

Similar conclusions to ones extracted from the Tables 2.8-2.12 are deductible from Figure 2.2. First, a regularity may be found in all four figures in the Interpolation and in LVCF (to a lesser extent) sections. This uniformity consists on their low section *defeating* their high equivalent, which means that the IM possibly hurt the data. Also, another, more subtle, consistency can be noticed between the three simple IMs (Mean, Most Frequent and Median). Their appearance frequency in high sections is greater (or similar) than in their corresponding lows in MAR, MuOV and MIV (Figures 2.2b-2.2d). Furthermore, these three charts have a quite regular shape in the complex IMs (HD, MICE and EM).

Finally, Figure 2.2a seems to have little in common with its peers. However, it does sustain the inefficacy of LVCF and Interpolation. Nevertheless, in regard to the simple IMs, this MDT was the only one not letting them have a good performance, as their presence frequency was higher in the low section (in varying degrees). On top of this, this MDT finally concedes an overwhelming victory to the complex IMs, as HD and MICE high representation clearly outscores their low correspondents. Also EM high section slightly beats its low representation.

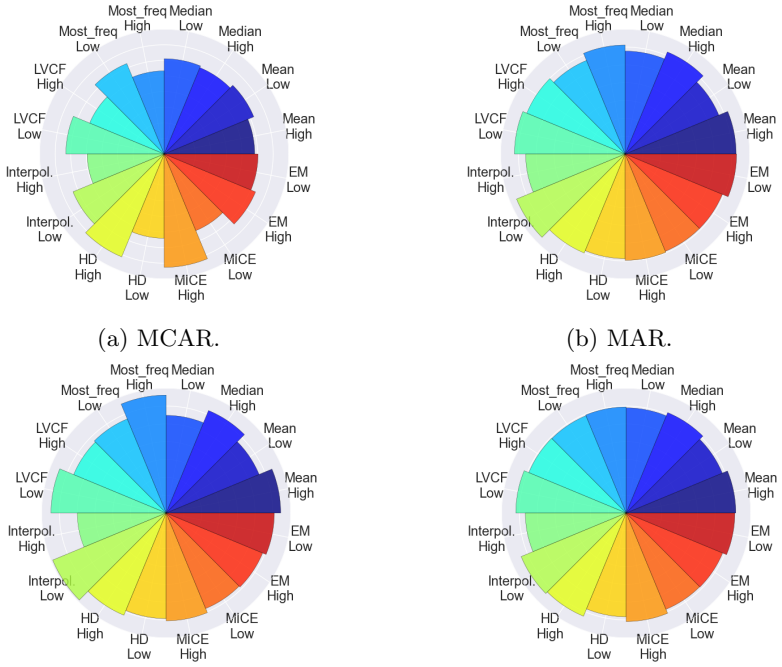
2.4.5 Interactions between Missing Data Types, Imputation Methods, and Classifiers

To investigate this question the same supergroups of *High* and *Low* configurations were used, but in this case the frequencies of all $8 \times 12 = 96$ pairs of IMs and SCs were computed. This information is shown in Figure 2.3, in which values are ranked by classification performance (x axis).

From Figure 2.3 it seems reasonable to assert that a significant interaction between IM and SC does not exist when it comes to high performance. The top three classifiers (Two Regressions and one SVM) remain consistent in any combination with IM, i.e., the best result obtained by any IM-classifier cooperation from the low section of classifiers does not match the worst result from the top three.

However, in the lower section, LDA shows some good partnerships with MICE and HotDeck, precisely the two IMs that generated the best results. Also, the largest difference obtained by this pair of methods, was shown in MCAR, the MDT that seemed to be the one offering the most significant contrasts.

Overall, it may be concluded that all three factors interact in some aspects. In terms of classifier-IM performance similar results are obtained for MAR, MIV and MuOV. Nevertheless, both the IM and the classifier achieve different results for MCAR, as the classifier rank changes (x-axis in figure 2.3). DBs with MCAR MDT seem to be a particularly good domain of application for MICE and HD



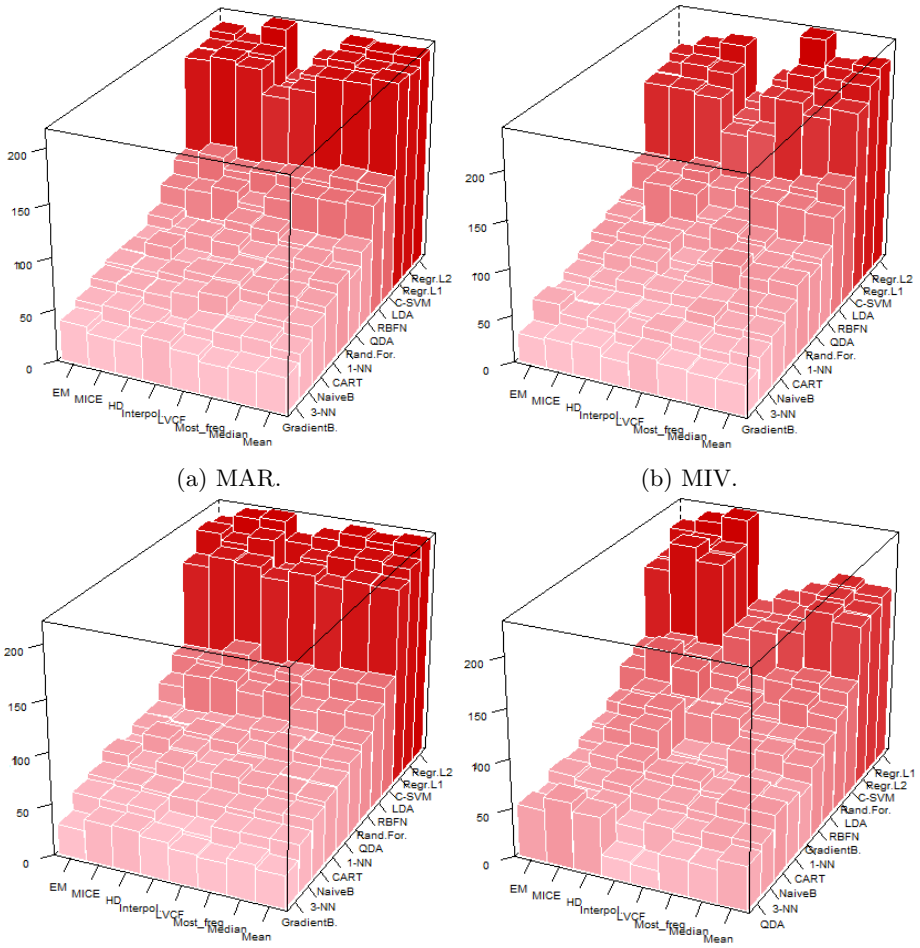
(a) MCAR.

(b) MAR.

(c) MIV.

(d) MuOV.

Fig. 2.2: Frequency of the IMs in the configuration with highest (*High*) and lowest (*Low*) classification accuracy.



(a) MAR.

(b) MIV.

(c) MuOV.

(d) MCAR.

Fig. 2.3: Amount of IM-Classifier pairs present in the High accuracy section.

(y-axis in figure 2.3), especially in the top performances, but noticeable in almost all of them.

2.5 Conclusions

The main objective of this chapter is to determine whether relations between MDT, IM and SC exist or not.

Considering the results discriminated by the MDT, a significative difference has been found, as MCAR obtained different results compared to the other MDTs. The classification accuracies were generally worse, and on top of that a statistical test reaffirmed this fact. Randomness affects all the variables equally, and this could make that the IM could gain importance, thus making the difference. The rest of MDTs affect only some concrete variables, and the SC algorithm could learn to ignore these variables, making the IM algorithm useless.

Regarding the IMs, overall the more complex methods obtained lightly better results, specially HD and MICE. However, the simple methods obtained good accuracies after applying the classifiers, Median imputation could be an example of this.

With respect to the pairwise combinations, MuOV and MAR showed no performance improvement in combination with any IM in particular. However MIV had a relatively good relation with simple IMs. Finally, MCAR, the MDT that gave the most importance to the IMs, have a clear positive interaction with the complex IMs.

Finally, some light interactions between the three components have also been found. Random Forest Classifier performed considerably better for MCAR (and thus, with complex IMs) than the other three MDTs, and LDA showed good performance when combined with complex IMs (also, particularly in MCAR).

The fact that the MDT is the most influential component determines that imputation procedures performance is strongly problem dependent. So, as an overall conclusion, an in-depth problem analysis is suggested before imputing data. This study should enclose expert knowledge on why records go missing. Depending on this first question, the MD pattern should be identified, and finally, an imputation procedure should be applied.

Chapter 3

New Imputation Methods for Time Series Based on Regression and Temporality

3.1 Objectives

The previous chapter has treated the MD problem in its full generic version in terms of DB types. This is, considering no restrictions over the databases used. In this chapter, we will go from the broadness to the specificity, as only multivariate TSs will be contemplated where discrete DBs (DBs with uncertified relations among observations) were allowed in the problem addressed in the previous chapter.

Since this is a concrete instance of the MD problem, all the information obtained in the previous experimentation part (Section 2.4) is applicable to this specification, and therefore will be exploited in this chapter.

The major focus of this chapter is the investigation of IMs specifically conceived to treat multivariate TSs. In Figure 3.1 an example of a TS with MD can be found. These charts show three univariate TSs which known segments are represented by a blue line, and the unknown ones are filled with red blocks. In the example shown in Figure 3.1, two types of MD can be identified, the MDT that defines the thin blocks, and the type producing large blocks. These MD classes are characterized by the length of the missing segments, which can be short or long. It is noteworthy that the distance between observations represents the time elapse between them.

Note that the imputation problem addressed in this chapter is slightly different, as now it is assumed that the relation among observations that belong to the same TS is inversely proportional to the time distance existing between them in the TS. These relations provide us with more information to impute missing values, as each piece of data is related to its previous and posterior values. Also, as in any other type of DB, we may find relations between the different TSs (e.g. correlations between TSs).

With the target of developing a new IM specific for TSs in mind, this chapter first explores a considerable amount of the work done in the field of MD analysis (specifically problems that address MD in TSs), using a similar work flow that

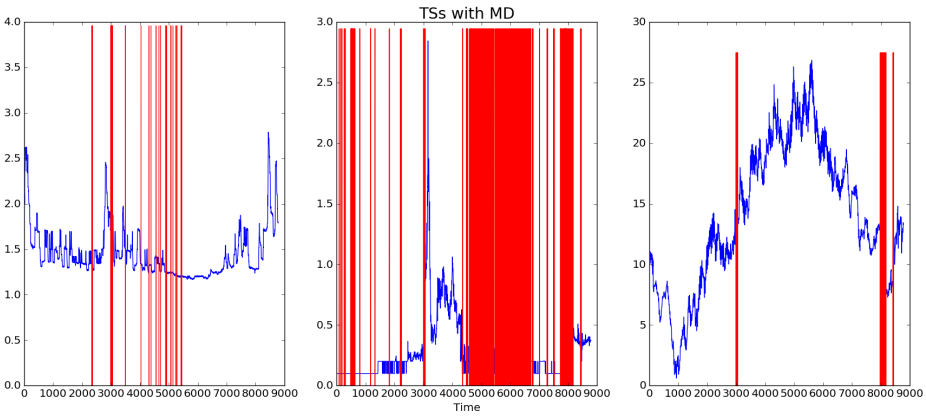


Fig. 3.1: Three TSs with MD.

the one described in the previous chapter. Next, we introduce a number of algorithms for treating MD in TSs. These algorithms have been conceived to address the lacks detected on the existing literature examination and on the findings described in the previous chapter. The suitability of the introduced algorithms for TSs is also considered, i.e. whether they can incorporate and exploit temporal information. Then, we empirically investigate the behavior of the IMs using TSs with different patterns of MD, to measure the performance of different IMs over TSs with MD.

In order to make a detailed analysis of the IMs, the chapter also addresses the question of how to introduce MD in the TSs, since different algorithms operate differently in distinct scenarios. Finally, a variety of distances between TSs are selected so that the performance of the imputation can be measured. Using more than one dissimilarity metric allows to observe the result from various perspectives, eliminating this way a possible bias of conclusions depending on the metric used.

3.2 Related Work

Three major types of approaches can be considered when facing a MD problem in a multivariate TS, when the main goal is SC. The option of deleting observations is still available, but the information loss produced as a result is disadvantageous. Some other preprocessing methods are modified to allow MD in the inputs. For instance, by applying a machine learning technique on a reformulation of the problem [62].

Authors	Reference	Domain	DBs	MD
Schoellhamer	[62]	Geophysics TS	1 Synth. Univariate	convinient values
Kim et al.	[33]	Microarray	Two multivariate	Random 5-25%
Honaker and King	[27]	Social Science	6 multivariate	Original
King et al.	[35]	Social Science	2 multivariate	original
Honaker, king et al.	[26, 28]	-	-	-
Troyanskaya et al.	[65]	Microarray	3 (2 multivariate)	random
Kim et al.	[34]	Microarray	5 Multivariate	1,5,10,15,20%
Oba et al.	[53]	Microarray	2, multivariate	1,2,5,10,20%
Gan et al.	[20]	Microarray	7	1,5,10 and 15%
Ouyang et al.	[55]	Microarray	2, multivariate	0.3, 0.5, 0.7, 0.9%
Jörnsten et al	[31]	Microarray	2, multivariate	1,4,7 %
Chiu et al.	[12]	Microarray	10 MVTS + 3 DB	1,5,10,15,20%
Bar Joseph et al.	[4]	Microarray	1, mvts	natural
Li and Parker	[40]	Sensor	1 MVTS	10-20–80-90%
Li and Parker	[39]	Sensor	1 multivariate	40%

Table 3.1: Table sumarizing related work on different methods of dealing with MD in TSs (majorly IMs). The table represents the authors, references where the work was published, application domain, characteristics of the TSs included in the DBs, and the characteristics of the MD.

Reference	IMs	Findings
[62]	Unimputed treatment	Unimputed treatment
[33]	-	Unimputed Treatment
[27]	Multiple and EM	EM>Multiple
[35]	Multiple Imputation	Implemented software
[26, 28]	Multiple Imputation	Amelia Software
[65]	SVD, KNN, mean	KNN>SVD>Mean
[34]	Local Least Squares, KNN	LLS>KNN
[53]	BPCA, KNN, SVD	BPCA>KNN & BPCA>SVD
[20]	POCS, Mean, KNN LS	POCS>>KNN, Mean & POCS>LS
[55]	Gaussian Mixture Cluster EM, KNN, SVD	GMEM>> KNN, SVD
[31]	Mean, KNN, SVD, BPCA, GMB LinCmb (combination of the rest)'	LinCmb $\geq$ the rest
[12]	KNN, LS, BPCA, SVD	BPCA, SVD, KNN>LS, 3 measurements
[4]	B-Spline, KNN, linear Interp., simple splines	B-Spline beats the rest
[40]	NN, LVCF, Random, Mean, EM	NN(correlated)>> the rest (cost-result)
[39]	NN, LVCF, Random, Mean, EM	NN(correlated)>> the rest (cost-result)

Table 3.2: Complementary table to the information shown in Table 3.1.The table relates references where the work was published, IMs used, and findings.

Schoellhamer [62] performs a particular instance of the Singular Spectrum Analysis (SSA). SSA is basically the Principal Component Analysis equivalent for TSs. One of the requirements of this technique is a complete TS. The author proposes a SSA modification which is able to deal with MD, to prove that it can be a possible solution when the goal is exploiting an incomplete TS.

Besides, a clustering algorithm that estimates the latent values while the clustering is performed is presented by Kim et al. in [33]. This alternative reduces the impact (possible bias) of an IM towards 0. Its working methodology is similar to an iterative imputation method. Each time an observation is assigned to a cluster, imputed values are recalculated.

However, the most common solution to the MD problem is, once again, imputation. Two research fields monopolize the research performed in this area. The two authors who have developed a considerable part of the literature on this MD in TSs area, work in the Social Science field. But the research field in which the major part of literature has been built is DNA microarray gene expression data analysis. Also, some works considering data collected by sensors over time have treated this MD in TSs issue. In the following, some of the most relevant papers involving the TS imputation problem are discussed.

3.2.1 Social Science

Honaker and King, are the main authors of possibly the most relevant literature about the application of IMs to TS databases in the field of Social Science. They have studied multiple political situations in many different countries. This type of studies is usually based on incomplente and heterogenous information. Although their main goal is not the imputation performance, but the inferences they obtain from imputed data, they have used a considerable amount of IMs. In [27], they propose a multiple imputation model that allows time trends in the TS, thus making the imputation much more accurate in each of them. Still, they propose another variant, an EM algorithm, which they claim it to produce more accurate imputations, increased efficiency, and reduced bias. Also, they conclude that the effect of the algorithm will depend on the characteristics of the data.

In [35], in collaboration with other authors, they propose another alternative to the classical multiple imputation algorithm, the EM with importance resampling. They emphasize that their method uses up to 50% more information than the MI, and is considerably faster. Also, it clearly outperforms a simple EM algorithm.

Also, Honeker and King, together with other authors, participated in the conception of Amelia [23], and its second version Amelia II [25]. This is software for EM imputation that is freely available and has been used in various fields, such as TSs in eHealth research [7]

3.2.2 DNA microarray gene expression data analysis

The DNA microarray gene expression data analysis allows monitoring the expressions levels of thousands of genes under a variety of conditions. The microarrays have been used to study gene expression in human tumors, among other applications. Due to insufficient resolution, dust or scratches on the slides, they may contain MD [65].

Troyanskaya et al. [65] consider three imputation methods in three DBs (of which two are multivariate TSs): Singular Value Decomposition (SVD) based method, weighted K-NN, and row average. The authors conclude that K-NN imputation provides the best results, followed by SVD, relegating mean imputation to the last place.

Kim et al developed another imputation approach in [34], in which they propose two similar methods based on Local Least Squares. They considered 5 multivariate TSs that originally contained no missing values. Next, MD was introduced in varying percentages (1, 5, 10, 15 and 20). Then, they compared the performance of the method proposed in the paper to K-NN imputation. Apart from the MD percentage variation, the authors also experimented with different DB sizes. Results showed constant improvement over the Root Mean Square Deviation between original and imputed values considering variations in both mentioned aspects.

Oba et al. [53] develop a model based IM. They use a probabilistic version of the principal component analysis algorithm. This strategy follows a Bayesian approach, and first computes the posterior distribution of the parameter set. Then, the method imputes the missing values using the model. This whole process is run iteratively, resulting in an algorithm that would be cataloged as EM. The authors carry out several experiments with different types and amounts of missing data, over data obtained from two different sources. Their results show that the proposed Bayesian Principal Component Analysis (BPCA) obtained a better outcome than Single Value Decomposition and KNN IMs.

Gan et al. [20] produce an ad-hoc iterative procedure named Projection Onto Convex Sets (POCS). Roughly, this strategy forms convex sets with every piece of information available, and searches for a solution, i.e., an intersection of the sets. The authors test the proposed POCS against other three IMs; Mean K-NN and Least Squares imputation. They selected seven existing and available TSs, and introduced different amounts of MD on them (1, 5, 10 and 15%). Their algorithm consistently obtains better results than KNN and Mean imputation, and slightly beats Least Squares Imputations in most of the DBs.

In [55], Ouyang et al. use a Gaussian Mixture Based (GMB) clustering algorithm as an IM. In this algorithm each component is modeled as a multivariate model to later produce a mixture. Then it is used combined with an EM strategy, to estimate the missing parameters. They compared it to other two IMs, i.e. KNN and SVD, using two different DBs with no natural MD. Their results showed a

big improvement, as the introduced algorithm produced much more consistent outcome, along with a better error towards the original values in the DB.

Jörnsten et. al [31] develop and test a new method, *LinCmb*, a combination of different IMs, whose weight will vary depending on the MD characteristics. This strategy involves mean, KNN imputation, Single Value Decomposition, BPCA (cited above [53]) and GMB (also already cited [55]). Depending on the amount of MD, the algorithm would apply more weight to global methods (last two) rather than local ones (first two). Two liver statistic DBs were selected as benchmark, and MD was introduced in 1, 4 and 7 %. Then imputed values were compared to original ones, with a regularized t-test to find significant differences. The authors show that the proposed method performs at least as good as the best of the submethods it is composed of. This work is one of few articles that combine different methods for a kind-of multiple imputation scheme.

Chiu et al. [12] made a recollection of four existing IMs (making slight variations on 2 of them, which results in total 9 IMs). They studied KNN (and its iterative and sequential versions), Least Squares (its Adaptive, Local, Iterative and Sequential versions), and previously introduced BPCA and SVD. They used 13 DBs, from which 10 contained a temporal component. In order to have a total control over the MD in the DB, the observations (genes) with missing entries were deleted. Then, missing values were introduced, in varying percentages (1, 5, 10, 15 and 20 %). BPCA, SVD and (simple) KNN obtained the best results, respectively, monopolizing the top three spots in all experiments involving a temporal component. Three measure types were considered to compute this ranking: normalized root mean squared error, a clustering quality index and the Biomarker list concordance index, an index that determines how accurate values are in gene data.

Bar-Joseph et al. [4] use an interpolation approach to solve the problem as they produce a B-spline algorithm to impute latent values. They tested their proposal in a DB with non-uniformly sampled data, with a posterior model based clustering to measure the results. This method was tested against other 3 IMs (KNN, linear interpolation and simple splines), and outperformed all of them in almost all of the scenarios (KNN was able to beat the proposed algorithm once).

3.2.3 Sensor Data analysis

Information gathered by sensors over time can also contain missing values due to a communication malfunction or unreliable sensors. When it comes to a machine learning process application, MD can be a problem that needs to be solved.

In [40], Li and Parker propose a Nearest-Neighbor imputation method in a sensor data context. This method considers extractable informations from both sources, spatial (multivariate TS) and temporal. They first check for correlated sensor values that will posteriorly be used to deduce which the Nearest Neighbor is.

They use *wide* DBs, as they contain 6500 TS and 1500 observations. The experimentation consists on comparing the proposed method to other four (LVCF, constant imputing, mean imputing and EM), contrasting imputed to real values. The final conclusion states that Nearest Neighbor computed by correlated values lightly beats EM in imputation accuracy, and by a considerable difference in computational cost.

Also authored by the same researchers, and in the same action field, the work in [39] reports similar strategies but with a different target in mind; SC. In the paper, a particular class of Neural Networks based on Adaptive Resonance Theory is used for classification. This time LVCF almost matched the performance of the EM algorithm, but once again, the Nearest Neighbor strategy stood tall. Li and Parker are among the few researchers that directly use information from both sources (spatial and temporal) to perform imputation, without creating a model.

3.3 Study of different Imputation Methods for Time Series with Missing Data

The previous section has covered recent literature on IMs for TSs and emphasized the limitations of this previous work. In this section we attempt to explore over these limitations.

As it has been stated, MD is a critical problem for TSs, specially, when the goal is the classification of the data. For this reason, the necessity of an IM with a contrasted good performance is needed. To fulfill this goal, an in-deph investigation of several IMs is carried out in this section. Different algorithms are tested against each other considering different types of MD and evaluation criteria, so that all the variants are covered.

We start with the application of four simple IMs. Then other more elaborated strategies based on the simpler ones are proposed to result in a total of 11 IMs. These methods are then tested against each other in order to know what kind of methodology suits best the problematic of TS imputation. In the following, the developed methods (and the rationale behind them) are presented.

In the analysis of the related work we identified only one paper describing a combination of two methods that exploit both dimensions of the information directly, [39]. This methodology seems a quite promising strategy, since it possibly will reduce the time consumption compared to other algorithms. This approach also will allow the user to replace the strategies that form the combination depending on the characteristics of the MD. For example, if the TS has a regular shape and its missing values are isolated, Interpolation could be chosen as the component that uses the temporal factor. If the characteristics of either the TS or the MD are different, another more complex method could be used to ex-

ploit the temporal component. For these reasons, the methods proposed in this chapter follow a similar direction to the one described in [39].

With the goal of selecting IMs that offer good results to be considered as base methods in mind, the findings obtained in Chapter 2 have been used. In terms of performance, the complex IMs produced the best results, and for this reason, they will also be considered for the imputation of TSs with MD.

Several types of methods to estimate missing values are discussed in the following sections to obtain the best possible performance in terms of the quality/time consumption ratio of the IMs.

3.3.1 Baseline Imputation Methods

The following four IMs are considered simple methods in this section and will be used later to integrate more elaborated algorithms for imputation. The first two strategies exploit the temporal component that characterizes the TSs, while the other two strategies are selected because of the good performance that they showed in the experimental part of Chapter 2.

3.3.1.1 Interpolation This interpolation method has been implemented using a Multiple Imputation approach. First, all the missing segments of each TS are detected. Then, a certain set of points is selected, in order to control which points are used to interpolate, as it is explained later. Next, three interpolations are performed over the missing values in each TS; Spline based, cubic, and 4 grade polynomial interpolation. Finally, the values estimated by the three interpolation methods are combined to obtain the final result, computing the average values generated by the three interpolations.

With respect to the selection of the timestamps the interpolation of the MD will based upon, we opted for the points immediately previous and posterior to the MD segment. The length of the selected previous and posterior segments is equal to the length of the missing sequence. For example, if we have a missing segment of three adjacent latent values, the points used for the interpolation would be the previous and subsequent three points.

In case a TS begins or ends with a missing value, then LVCF (or its reversed version) is applied. Also, when two segments are too near, they are combined and regarded as a single missing segment. Two segments are considered to be too close to each other when there are less observed points between the two missing segments than half the length of the longest missing segment surrounding the known points. For example, if there are three observed points between two missing segments of length four and seven, the segment would be merged in one, since $7 > 4$ and $7/2 > 3$. But if the length of the segments would be four and five, $5 > 4$ but $5/2 \not> 3$, and therefore the sequences would be considered as two distinct segments.

As previously (Section 2.3.3), the Pandas implementation was used for Interpolation. The *method* argument receives in this case three different values: "quadratic", "cubic" and "spline".

Algorithm 3.1 shows the pseudocode of the Interpolation method. It uses the following functions and notation:

- detectSeq(TS): Given a TS, this function returns the starting and ending indexes of all the missing sequences.
- joinSeq(seq): Given the missing sequences of a TS in the format produced by detectSeq(TS), this function joins sequences when the criteria described before in this section are met.
- pointSel(seq): Given one single sequence, this function returns the points that shall be used to perform interpolation. The points are selected using the method explained before in this section.
- interpolate(DB, s, pt): Given the complete DB with MD, the sequence being treated, and the points selected for interpolation, this function performs the Multiple Imputation process, combining the three Interpolations mentioned previously in this section. Then, the function returns the DB with the specified sequence imputed.
- DB[:,i]: This notation returns the full 'i'-est column of the DB.

Algorithm 3.1: Interpolation algorithm.

```
1  Input: DB with MD
2  Output: Imputed DB
3  begin
4      for i ∈ range(0, nCol(DB)) do
5          seq = detectSeq(DB[:,i])
6          seq = joinSeq(seq)
7          for s ∈ seq do
8              pt = pointSel(s)
9              DB = interpolate(DB, s, pt)
10     return (DB)
```

3.3.1.2 Seasonally Splitted Imputation Method This IM is based on the concepts Seasonality, Kalman Smoothing and State Space Models. In a TS, seasonality refers to a specific pattern being repeated cyclically over the TS [50]. The Kalman filter is an iterative algorithm that uses a multivariate TS to build a model that allows discovering unknown variables. It is composed by a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance. State Space Models are a notational convenience for estimation and control

problems, developed to make tractable what would otherwise be a notationally-intractable analysis. In this case, it represents an ARIMA model of the multivariate TS [68]. This strategy consists on first splitting seasonally each univariate TS, then building the mentioned models and imputing the TSs using the models.

This strategy has been implemented using the **R** package *imputeTS*[4].

3.3.1.3 *Standard* Imputation Method Due to the good results obtained by MICE in Chapter 2, this method was selected to be part of the imputation approach for TSs. The specifications are the same as the ones described in Section 2.3.3. It was applied in the same manner.

3.3.1.4 Regression This method considers the TS with MD as the variable to be regressed and use the rest of the TSs as information to perform a regression on them. This method needs a fully complete DB, except for the target variable, for which the complete DB was obtained using the *standard* IM method. MICE and Regression combine together for a two-stage algorithm.

The implementation of this method is imported from sklearn, via the *Ridge()*[5] function, a linear least squares with l2 regularization.

Figure 3.2 contains a visual representation of the process. The first stage shows the MICE regression. For the final stage, in order to reimpute the variables, the information generated by MICE in the first stage is used. In this example, to obtain the values of the fourth and sixth observation of the variable *X1* in the final stage, the values of variables *X2* and *X3*, that had been imputed in the first stage are used. Then, the process is repeated for each variable.

3.3.1.5 *Polished* Regression This method consists on constructing small regression models, in order to minimize the computing complexity. This is managed by selecting those observations that are the most similar to the ones to be imputed. To achieve this goal, first the pairs of variables with the highest correlation are identified. Then, based on a previous low-medium-high value discretization for each TS, the most similar observations are chosen to build the model that will produce new imputable values.

Finally this method performs regression as the strategy described in Section 3.3.1.4.

To compute the related variables, the mutual information was used, via normalized_mutual_info_score()[6] function, by *sklearn*.

[4] https://cran.r-project.org/web/packages/imputeTS/imputeTS.pdf
[5] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html
[6] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html

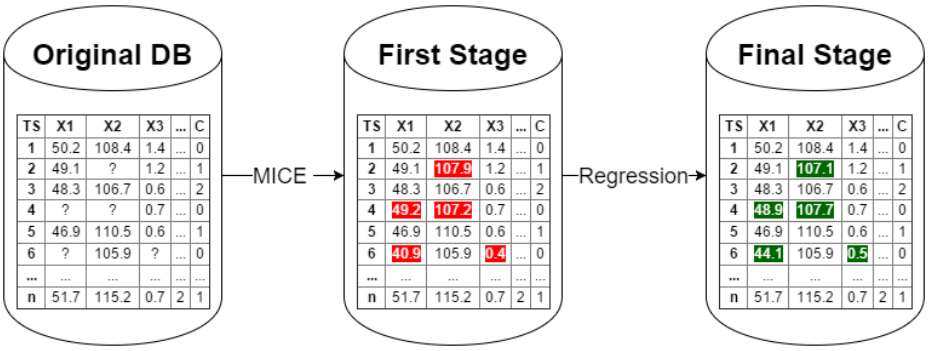


Fig. 3.2: Two stage Regression Imputation example. In the first step MICE is applied. For the second step, each TS is reimputed. To reimpute the values highlighted in green, the values highlighted in red are considered as observed.

All the steps of the process are described Algorithm 3.2. This strategy uses a number of subfunctions as building blocks. Their description follow:

- baseImp(DB): Given a DB with MD this function returns the DB imputed by a base method. In this case MICE is used.

- discretize(DB): This function takes a full DB as an argument, and it returns its discretized version. For this task, each TS gets its values divided in three sets, lowest, highest and medium values, ignoring the timestamp they were recorded in. Then the values are replaced by 1 (low values), 2 (medium values) and 3 (high values). Also, 0 is used for MD.

- mutualInf(DB): This function takes a DB in matrix form, and returns another matrix, with as much columns and rows as columns the original DB has. The returned data is an upper diagonal matrix containing the mutual information for each pair of variables.

- observations(Ddb, DB, MI): This function takes the original DB (with MD) and the discretized versions of the same database along with the mutual information matrix associated to the imputed DB, and returns a matrix with two columns. For each missing entry in the original DB, this function adds a new row to the matrix that it returns, containing the index of the latent value in one column, and the observations that most likely will create an appropriate regression model to impute it. To achieve this, the function treats each TS individually. For each missing value in each TS, the function looks at the discrete values that the two most correlated variables to the one being treated take, and find similar ones, in which the value for the variable being treated is known.

– groupRows(obs): taking a matrix produced by observations(), this function groups observations that for the same TS will use the same observations to build a model, in order to minimize the amount of models created.

– range(x,y): this function produces a list of the integers between x (included) and y (not included).

– numRows(x): being x a matrix, this function returns the amount of rows of x.

– regression(obs, DB): Given one (or more) missing value index and the observations that need to be used to build the model (in the format produced by observations()) and the original DB with latent values, this function computes the regression and imputes the value.

Algorithm 3.2: Polished Regression algorithm.

```
1  Input: DB with MD
2  Output: Imputed DB
3  begin
4      Idb = baseImp(DB)
5      Ddb = discretize(DB)
6      MI = mutualInf(DB)
7      obs = observations(Ddb, DB, MI)
8      obs = groupRows(obs)
9      for i ∈ range(0, numRows(obs)) do
10         regression(DB, obs[i])
11     return (DB)
```

Figure 3.3 provides a graphical representation of the algorithm.

3.3.2 Advanced Imputation Methods

The methods presented in this section are more *complex*, since they have been designed performing different combinations of the *simple* methods described in Section 3.3.1.

3.3.2.1 Interpolation edge smoothed *polished* regression This method combines the Interpolation and Polished Regression IMs (3.3.1.1 and 3.3.1.5), so that the transition from observed TS values to the imputed ones is as smooth as possible. Regression as an IM lacks of a temporal component that can be exploited in TSs. For this reason, in this case it is combined with interpolation. Interpolation usually produces *good* results near the observed TS segments, due to the temporal component, and tends to struggle more in the central parts of the missing sections (specially when the missing segment is long). This is why

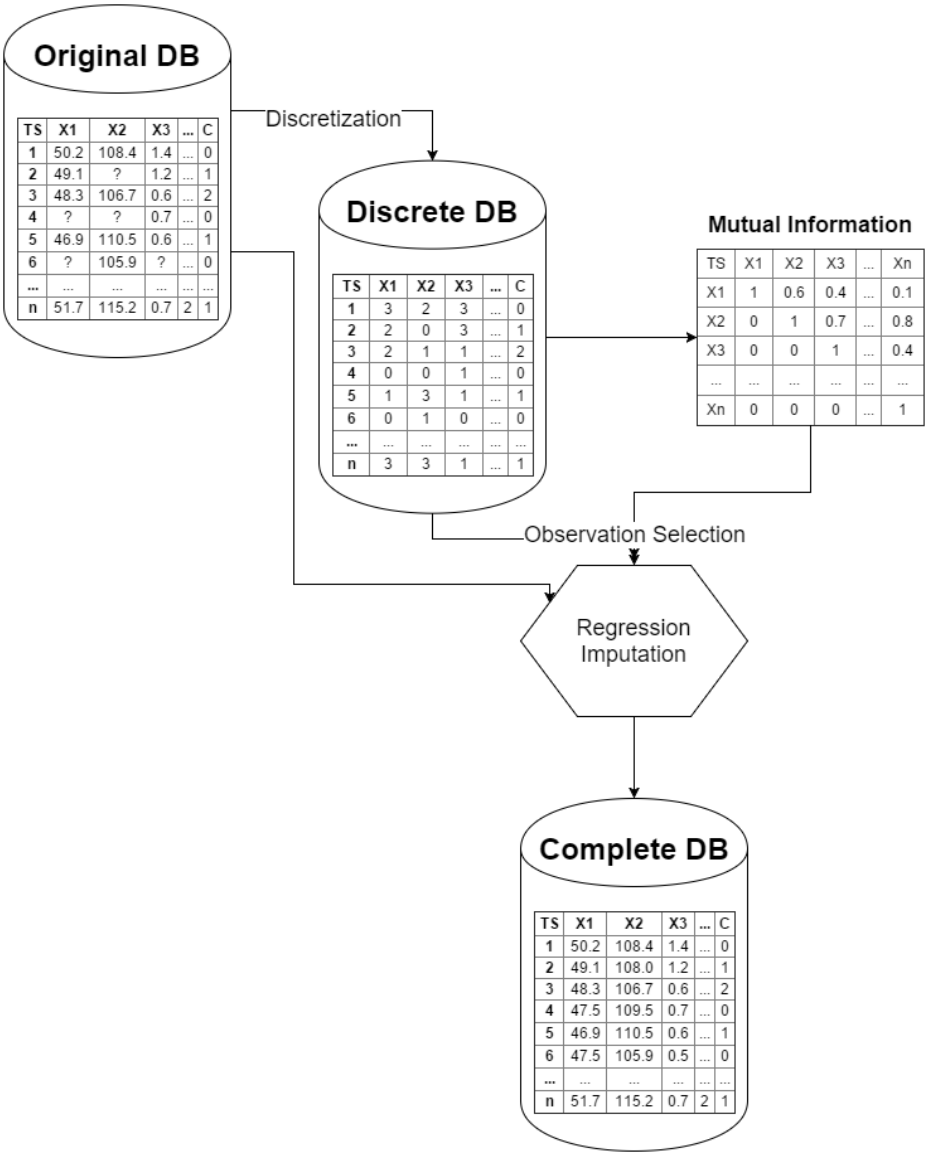


Fig. 3.3: Polished Regression Imputation visual representation. This diagram shows graphically the information shown in Algorithm 3.2. The Regression hexagon performs Regression for each TS with MD.

this method proposes a combination in which Interpolation is weighted higher in the edges of the missing segments, and Regression gains consideration in central parts.

As said before, this method uses the implementations developed in 3.3.1.1 and 3.3.1.4. The formula for the combination of two methods is given by Equation 1:

$$x \cdot InterpolationValue + (1 - x) \cdot RegressionValue \tag{1}$$

x is calculated for each missing position (k) as in Equation 2, in which *start* and *end* represent the indexes of the starting and ending points of the missing segment, and l is the total length of the segment.

$$max((end - k)/l, (k - start)/l)) \tag{2}$$

The procedure is formally explained in Algorithm 3.3, which uses the following functions and notation:

- Interpolation(DB): Performs Multiple Interpolation, as explained in Section 3.3.1.1 to fill in missing values.
- polishReg(DB): Performs Polished Regression as explained in Section 3.3.1.5 to return a DB with no MD.
- detectMissSeq(TS): Given a TS, this function returns a two-column matrix. Each row of the matrix represents a long missing sequence, and the two values on it are the starting and ending points of the sequence.
- ncol(DB): Given a DB, this function returns the amount of columns of the matrix.
- range(x,y): Returns the list of ordered integers in the [x,y) interval.
- max(x,y): Returns the number with the higher value between x and y.
- DB[:,i]: This notation returns the full 'i'-est column of the DB.

Algorithm 3.3: Interpolation Edge Smoothed Polished Regression algorithm.

```
1  Input: DB with MD
2  Output: Imputed DB
3  begin
4      IntDB = Interpolation(DB)
5      PolDB = polishReg(DB)
6      for i ∈ ncol(DB) do
7          seq = detectMissSeq(DB[:, i])
8          for j ∈ range(0, numRows(seq)) do
9              length = seq[j,1] - seq[j,0]
10             start = seq[j,0]
11             end = seq[j,1]
12             for k ∈ range(start, end) do
13                 x = max((end-k)/l, (k-start)/l))
14                 DB[k,i] = x · InterpolationValue + (1 − x) · RegressionValue
15     return (DB)
```

3.3.2.2 Interpolation intermittently smoothed *polished* regression This method also combines Interpolation and Polished Regression (Sections (3.3.1.1 and 3.3.1.5)). First of all, the data is imputed as described in the Polished Regression method. Then it is combined with Interpolation. In this case, for each missing segment of length l, the section is divided in subsegments of length $ln(l)$. Next, half of the segments (interleaved) are reimputed via Interpolation, considering the other half of the imputed values as observed. Then, the process is run again, but this time the roles of the subsegments are changed. The subsegments that were reimputed are now considered observed (their Polished Regression values), and the subsegments considered observed, are reimputed. Finally, the values obtained via Regression and via Interpolation are combined by computing their mean.

To implement the Interpolation intermittently smoothed *polished* regression algorithm, the Interpolation and Polished Regression processes described before were used as building blocks.

Figure 3.4 shows a fictional example of how this IM works, in a step-by-step explanation. The figures include the process from the complete, original TS to the imputed TS, passing through the MD introduction, and the imputation combination.

The first step described in Figure 3.4a contains the original TS, and in the second (Figure 3.4b) the MD introduction is performed. For this example, a long missing segment was introduced, to ease the understanding of the methodology. Step three (Figure 3.4c) runs the Regression imputation (green part). Steps 4.1 and 4.2 (Figures 3.4d and 3.4e, respectively) divide the now imputed sections in two

(Note the interleaved shorter missing sections). Steps 5.1 and 5.2 (Figures 3.4f and 3.4g, respectively) perform Interpolation over the missing values produced in steps 4.1 and 4.2 respectively (yellow segments). Finally, step 6 (Figure 3.4h) computes the averages between the Polished Regression and Interpolation values (green and yellow ones) to produce the final result.

The process is also described in Algorithm 3.4, which uses the following functions and notation:

- polishReg(DB): Takes a DB with MD as an argument, and returns the DB imputed using MICE Polished Regression (Section 3.3.1.5).

- detectMissSeq(TS): Given a TS, returns a two-column matrix. Each row of the matrix represents a long missing sequence, and the two values on it are the starting and ending points of the sequence.

- numRows(x): Being x a matrix, this function returns the amount of rows of x.

- int(x): Returns the integer part of (presumably float) x.

- ln(x): Returns the napierian logarithm of x.

- Interpolation(DB): Performs Multiple Interpolation, as explained in Section 3.3.1.1 to fill in missing values.

- mean(DB1, DB2): Returns the mean of two matrices, performed element wise.

- db[x:y,z]: This notation selects all the cells between indexes db[x,z] and db[y,z].

Algorithm 3.4: Interpolation intermittently smoothed *polished* regression algorithm.

```
1  Input: DB with MD
2  Output: Imputed DB
3  begin
4      PolDB = polishReg(DB)
5      PolDB1 = PolDB
6      for i ∈ ncol(DB) do
7          seq = detectMissSeq(DB[:, i])
8          for j ∈ range(0, numRows(seq)) do
9              l1 = int(ln(l))
10             k = 0
11             start = seq[j,0]
12             end = seq[j,1]
13             while k · l1 < l do
14                 startIndex = start + k · l1
15                 medIndex = min(start + (k + 1) · l1, end)
16                 endIndex = min(start + (k + 2) · l1, end)
17                 PolDB[startIndex:medIndex, i] = "NaN"
18                 PolDB1[medIndex:endIndex, i] = "NaN"
19                 k = k + 2
20     PolDB = Interpolation(PolDB)
21     PolBD1 = Interpolation(PolDB1)
22     return (mean(PolDB, PolDB1))
```

3.3.2.3 Random based *Polished* Regression This method also follows the same strategy described in Algorithm 3.2 with the same function specification except in *baseImp(DB)*. In this case, instead of using MICE as the step to obtain the necessary full DB in the first stage, random imputation is used. Random imputation takes a random value between the maximum and minimum values in the TS, and uses it for imputation. More formally:

- baseImp(DB): Given a DB with missing values, this function returns the imputed DB. For each TS, the function computes the maximum and minimum values, and imputes a random value between them in each latent value.

The rationale behind defining the Random Imputation is that using it we can contrast whether there is a need of a *good* imputation (in this case, MICE) as a base for subsequent Regression application.

3.3.2.4 Random based Interpolation intermittently smoothed *polished* regression This method uses the same strategy followed in Interpolation

intermittently smoothed polished regression (Section 3.3.2.2) in Figure 3.4, and Algorithm 3.4. The only difference is that, in this case, instead of basing the Regression on MICE, it is based on Random Imputation;

– polishReg(DB): Takes a DB with MD as an argument, and returns the DB imputed using Random based Polished Regression.

3.3.2.5 Seasonally Splited-Kalman model-based Regression

This method is also similar to the strategy explained in Section 3.3.1.4, but instead of using MICE as a base method, Kalman model (Section 3.3.1.2) is used. Following the example in Figure 3.2, the change will only affect the process before Stage 1, in which MICE would be replaced by Kalman.

3.3.2.6 Seasonally Splited-Kalman model-based Polished Regression

This strategy is also implemented as in Algorithm 3.2, but *baseImp(DB)* produces a different result compared to previous descriptions. In this case:

– baseImp(DB): Takes a DB with missing values on it, and returns its imputed version, via Seasonally Splitted IM (Section 3.3.1.2).

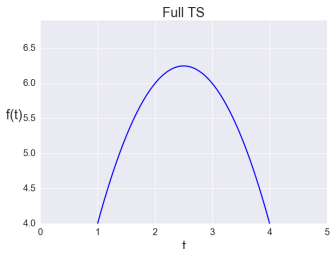
3.4 Experiments

In order to build our own experience apart from the knowledge obtained from the work developed by other authors, some experiments have been designed. The goal of the analysis presented in this section is to find out what is the performance of some of the usually applied TMs for multi-variate TSs and compare them to the newly proposed methods. As a result of the analysis, we will we expect to rank the IMs according to their behavior for TMs. In addition, we expect to identify those IMs that are ineffective in terms of their performance.

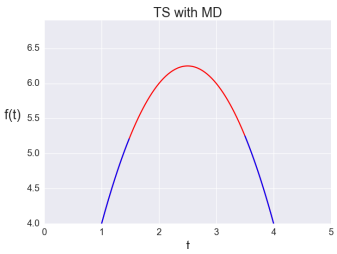
More specifically, this section has three main objectives:

1. Introducing the algorithms that will serve us to insert MD into the complete TSs.
2. Compare four *basic* IMs: Interpolation, Kalman-based, MICE and Regression
3. Evaluate the new proposed algorithms using themselves as a benchmark, along the basic methods.

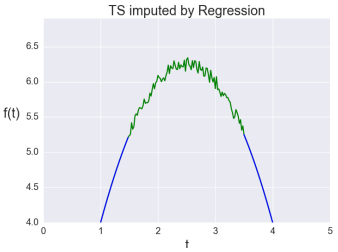
With this goal in mind, a set of experiments similar to those developed developed in Chapter 2 were designed and executed.



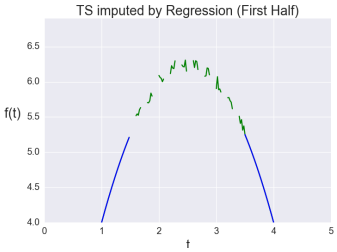
(a) Step 1; full univariate TS.



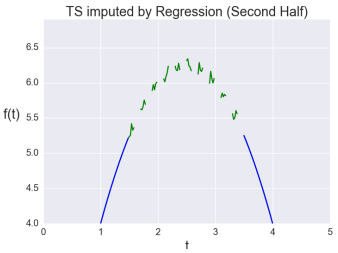
(b) Step 2; TS gets MD inserted (in this case, a long segment).



(c) Step 3; TS is imputed using Regression.



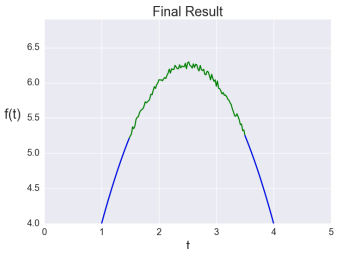
(d) Step 4.1; Regression results are divided (first half).



(e) Step 4.2; Regression results are divided (second half).

(f) Step 5.1; First half gets imputed using Interpolation.

(g) Step 5.2; Second half gets imputed using Interpolation.



(h) Final step 6; Both results in steps 5.1 and 5.2 are combined.

Fig. 3.4: Interpolation intermittently smoothed *polished* regression step by step example.

3.4.1 Experimental Settings

The general experiment consisted on the selection (and creation) of multivariate TSs, which will lose some of their values for a posterior imputation. Then, the distance between the imputed and the original TSs is computed, and used as a metric to determine the quality of the IM. In this section the complete experimental process is detailed.

3.4.2 Data Base benchmark

First, we need to decide on a set of multivariate TSs to test the IMs. To achieve this first goal, two strategies were developed to generate the artificial benchmarks. The first one consisted on creating a synthetic DB containing 5 TSs. All of them were composed by 100 timestamps (indexed with integers from 0 to 99).

$$y = x \tag{3.1}$$
$$y = 2x \tag{3.2}$$
$$y = sin(x) \tag{3.3}$$
$$y = cos(x) \tag{3.4}$$
$$y = ln(x) \tag{3.5}$$

The functions used to compute the values of the five TSs are shown in Equations 3.1 to 3.5 The first TS corresponds to the timestamp index. The second variable is also a linear function. The third and fourth TSs contain sinusoidal functions, while the last TS is a logarithmic function. This set of 5 TSs provides a DB from which missing values are *easy* to predict, since an polynomial Interpolation with a not necessarily high order or a linear Regression could predict them quite accurately. This way, the DB contains two linearly ascending functions, other two TSs with periodic behaviors, and finally, slowly increasing TS.

Other more realistic multivariate TSs were obtained from the **R** package *TSDist*[7]. This package offers multiple methods to measure the distance between two TSs, methods that also consider their temporal aspect. It also provides three example Multivariate TSs, with varying dimensions. These DBs are more irregular and *realistic* than the artificial DBs generated by Equations 3.1 to 3.5, thus it will be a good benchmark to test the developed IMs.

The first of these three multivariate TS consists of a numerical matrix conformed by six $ARMA$ series of coefficients ($ARMA(3, 2)$). $ARMA$ is a model that can be computed from a TS, and is usually used to describe a TS or predict future values. It is composed by two polynomial models that serve the same purpose;

[7] https://cran.r-project.org/web/packages/TSdist/TSdist.pdf

Autoregressive and Moving Average. Their parameters $AR = (1, -0.24, 0.1)$ and $MA = (1, 1.2)$ and length 100. The second DB is formed by 100 series of length 100 obtained from 6 different classes. The final example is a DB conformed by 50 series of length 100 obtained from 5 different classes. Each class is obtained from a different initializations of an $ARMA(3, 2)$ process of coefficients $AR = (1, -0.24, 0.1)$ and $MA = (1, 1.2)$ [49].

Table 3.3 shows a summary description of the TS characteristics as well as the references to four graphical representations of the TSs. These figures contain all the information in the complete TSs, therefore, the relative development of the individual TSs can be observed. Examining these images, it is clear that the *TSdist* multivariate TSs are drawn from the same origin.

Index	DB	Obs.	TSs	Description	Fig.
1	Synthetically created	100	5	x, $2x$, $sin(x)$, $cos(x)$, $ln(x)$	3.5a, 3.5b
2	1 TSdist	100	6	$ARMA(3, 2)$ ($AR = (1, -0.24, 0.1)$ & $MA = (1, 1.2)$)	3.6
3	2 TSdist	100	100	6 classes	3.7
4	3 TSdist	100	50	$ARMA(3, 2)$ ($AR = (1, -0.24, 0.1)$ & $MA = (1, 1.2)$)	3.8

Table 3.3: DBs used to investigate the behavior of the different IMs proposed in this chapter.

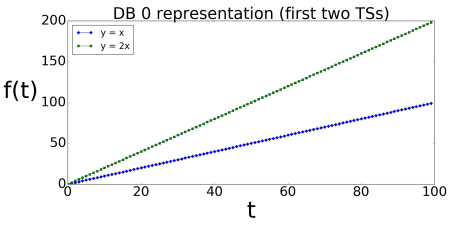
3.4.3 Algorithm to add missing data to the time series

The knowledge gained on the previous chapter stated that MCAR was the MD type that offered the most significant differences in regards of IMs performance. Also, MCAR is a realistic MDT for a TS, since a machine could miss recording a value one specific time, or a doctor could forget to ask certain information to a patient to write it down, only once. As a result, from the four MDTs considered in the previous chapter, only MCAR is contemplated in this second problem.

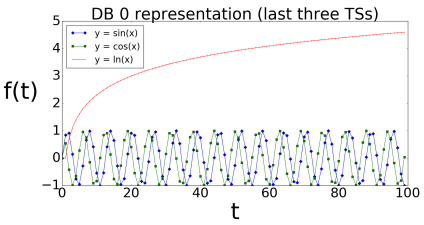
The problem specification has changed from Chapter 2 to this section, and so has some of the terminology. The MDT detectable in a TS database differs from the types found in discrete DBs. For this reason, the MDT concept changes from now on. In the previous discrete problem, MDT referred to four different types (MAR, MCAR, MIV and MuOV), but from now on it will make reference to two types of MD exclusive of TSs. The first one, long missing segments. For example, a malfunctioning sensor could produce various adjacent missing values in the same variable.

Summarizing, we have two new types of MDT, long and short segments.

For introducing MCAR, the structure of the algorithm is the same as the one described in Algorithm 2.1.



(a) First two TSs.



(b) Last three TSs.

Fig. 3.5: Synthetic Multivariate TS (5 TSs).

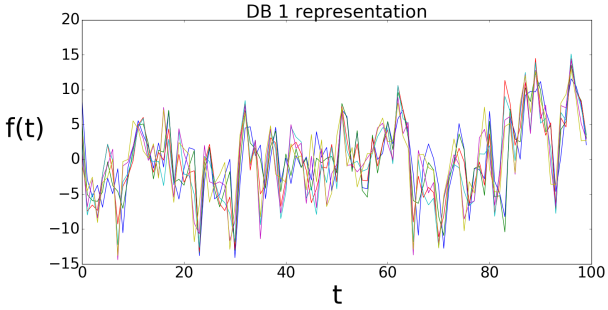


Fig. 3.6: First Multivariate TS in TSdist package (6 TSs).

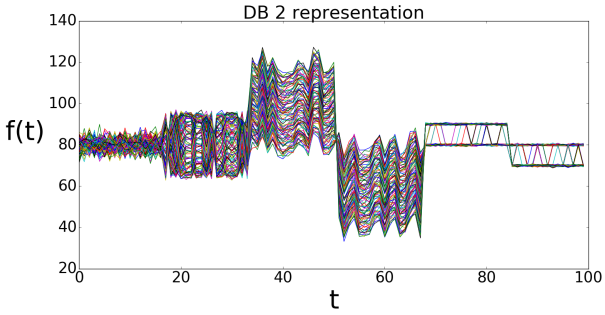


Fig. 3.7: Second Multivariate TS in TSdist package (100 TSs).

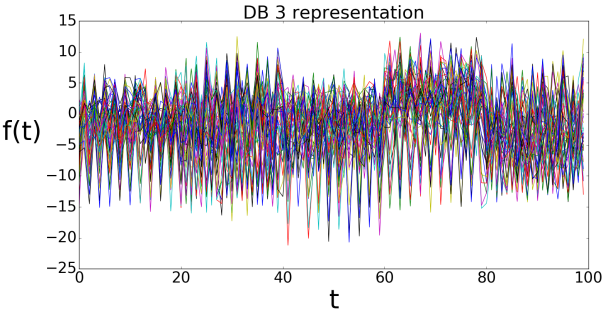


Fig. 3.8: Third Multivariate TS in TSdist package (50 TSs).

As for the long missing sequence introducing algorithm, it is defined as in Algorithm 3.5.

Note that this algorithm uses the functions random(x,y), which produces a random integer between x and y, and numTS(db), which returns the amount of TSs in a DB. Also, it uses the notation db[x:y,z], that selects all the *cells* with index between (x,z) and (y,z). Finally, the parameter *lengths*, is a list in which the dimensions of the missing chunks will be introduced. For example, if we want to erase one segment of 10 timestamps, $lengths = [10]$, or if the aim is to introduce 5 chunks of MD of 5 positions each, then $lengths = [5, 5, 5, 5, 5]$.

In the experiments described in this section, this variable was set as $lengths = [13, 14, 15, 16, 17, 18, 19]$. With these values, the missing entries will be significantly different from MCAR. Also, the introduction of this amount of MD will produce a large percentage of unobserved entries in the smallest DBs, about 20%, the MD percentage commonly introduced in the papers commented in Related Work (Sections 2.2 and 3.2).

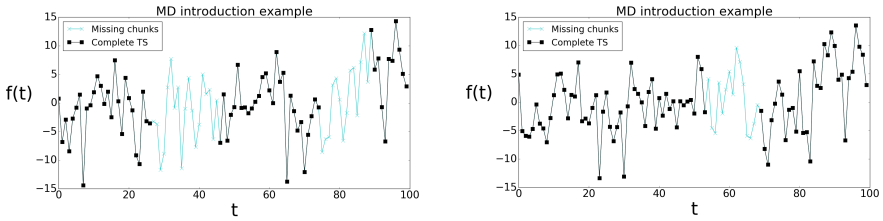
Algorithm 3.5: Long sequence MD generating algorithm.

```
1  Input: DB
2  Output: DB with MD
3  begin
4      y = numTS(data)
5      x = numObs(data)
6      for i ∈ lengths do
7          obs = random(0,y-i)
8          ts = random(0,y)
9          data[obs:obs+i, ts] = "NaN"
10     return (data)
```

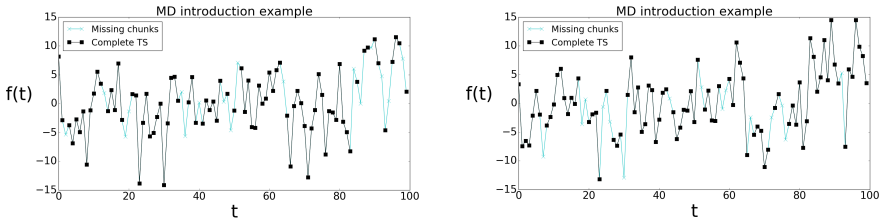
This parameter (lengths) had the same value for all four DBs. In this way, different MD percentages were used, since the dimensions of the TSs were all different. For the synthetic DB, this process supposed introducing 22.4% of MD, while for the TSs obtained from the **TSdist** package, these values were 18.667%, 1.12% and 2.24%.

These long strings of missing values generated by Algorithm 3.5 will be considered as such if the missing gap occupies three *cells* or more, as recommended by Bar-Joseph et al. in [4]

Figures in 3.9 show different examples of how MD is is introduced in some of the univariate TSs. Each figure shows the original TS (black, with square markers) with the introduced MD (cyan, with X markers). Figure 3.9a presents examples where long MD segments have been introduced, while Figure 3.9b shows examples of the introduction of short MD segments.



(a) Two examples of long missing segment introduction.



(b) Two examples of isolated MD introduction.

Fig. 3.9: Four examples of MD introduction in TSs.

3.4.4 Metrics

Considering the question of MD types and IMs for the particular case of multivariate TSs, other performance measurements of the IMs can be introduced. In the previous chapter, the accuracy value obtained via 5-fold cross-validation was used as a way to evaluate the performance of the IM. This method misses the temporal component of the TS, as a general SC algorithm classifies regarding simply the information available in a single observation. For this reason, a new methodology is proposed. In this case, distances between TSs will be used to contrast the effectiveness of the IMs. We will compute the distance between the original TS and that same TS after MD has been introduced and imputed. The distance will be used as a criterion to determine the imputation quality. Three distance computing methods have been selected;

– STSDistance: The Short Time Series Distance [47], computed as:

$$STS = \sqrt{\sum_{k=0}^{length(TS)} \left(\frac{y_{k+1}-y_k}{tx_{k+1}-tx_k} - \frac{x_{k+1}+x_k}{ty_{k+1}-ty_k}\right)^2}$$

Where tx_k and ty_k represent the k-th timestamp, and x_k and y_k stand for the respective values of the two TSs in that same k-th timestamp [49].

– ARLPCCepsDistance [32]: Computes the dissimilarity between two numeric time series in terms of their Linear Predictive Coding (LPC) ARIMA pro-

cesses [48]. ARIMA is a similar, generalized model to the one explained in previous Section 3.4.2, ARMA [51].

– IntPerDistance [11]: Calculates the dissimilarity between two numerical series of the same length based on the distance between their integrated periodograms. It is computed as:

$$d(x,y) = \int_{-\pi}^{+\pi} |F_x(\lambda) - F_y(\lambda)| d\lambda$$

where $F_x(\lambda) = C_x^{-1} \sum_{i=1}^{j} I_x(\lambda_i)$ and $F_y(\lambda_j) = C_y^{-1} \sum_{i=1}^{j} I_y(\lambda_i)$ with $C_x = \sum_i I_x(\lambda_i)$ and $C_y = \sum_i I_y(\lambda_i)$ in the normalized version. $C_x = 1$ and $C_Y = 1$ in the non-normalized version. $I_x(\lambda_k)$ and $I_x(\lambda_k)$ denote the periodograms of x and y, respectively [48].

These three functions were selected because the totally different methods they utilize to compute distances, which will produce three perspectives of the influence of the IM in the imputed TS.

All three distances were computed using the same **R** package as for the generation of multivariate TSs as described in Section 3.4.2, *TSDist*.

3.4.5 Experimental Results

In order to measure the effectiveness of the methods presented in Section 3.3, an experiment involving IMs and the components mentioned previously (MD introduction strategies 3.4.3 and metrics 3.4.4) was designed. The steps of the experiment follow:

1. Introduce MD in the original database, DB_{Orig} with a selected MD introduction strategy, to create DB_{MD}
2. Apply an IM to the DB_{MD} to create an imputed database, DB_{IM}.
3. Compute the distance between the DB_{Orig} and $DBIM$ using a determined distance, for each TS pair in the DBs.

This process is executed for each possible combination of MD introduction strategy, IM, and distance. Then, all the results are subjected to a statistical test that consists on the Kruskal-Wallis/Dunn statistical test combination. Using this test, all the different components are examined to detect which factor produces the major impact in imputation quality. Each time the test found that a combination of components obtained significantly better results than other one, the first one was awarded one point and another point was subtracted from the one with worse result. The points won and lost are shown in tables in the following sections.

The last row of each table shows the number of times each algorithm was outperformed by the others (o^-). The last column shows the number of times each algorithm outperforms the rest (o^+). The final score of the algorithms was obtained by subtracting the points lost to the points won compared to other strategies.

To improve the readability of the tables, the names of the IMs are substituted by their index:

IM1 Interpolation.

IM2 Seasonally Splitted Imputation Method (Kalman).

IM3 Standard IM (MICE).

IM4 Regression (based on MICE).

IM5 Polished Regression.

IM6 Interpolation edge smoothed polished regression.

IM7 Interpolation intermittently smoothed polished regression.

IM8 Random based Polished Regression.

IM9 Random based Interpolation intermittently smoothed polished regression.

IM10 Seasonally Splited-Kallman model-based Regression.

IM11 Seasonally Splited-Kallman model-based Polished Regression.

3.4.5.1 Short Missing Segments First, we present the results obtained from the DBs that had MCAR MDT introduced on them. The following Tables 3.4, 3.5 and 3.6 show the statistical differences between the dissimilarities computed by each one of the three distances used for these experiments, while the last Table 3.7 shows the combined result.

Table 3.4 refers to the results of the comparisons between the IMs using STS distance. This table ranks MICE based Regression (IM4) in first place, with an overall score of 29. Only Kalman based Polished Regression (IM11) can barely keep up to it, scoring 21. The rest of methods obtained 12 or less. It seems like Regression makes the difference, as both (Polished and Regular) outscore clearly their base methods (MICE (IM3) and Kalman (IM2)). Also, interpolation does not provide acceptable results, being, by far, the worst method. The Random based Regressions also fail by a large margin.

Table 3.5 refers to the results using ARIMA-LCP distance. This dissimilarity measure shows less differences than the previous metric. Again, the same two IMs top the overall ranking (recording scores of 17 and 14) over the rest (8 or lower). However, this time, the top two have their roles changed, as Polished Kalman Regression beats Mice Regular Regression. The Interpolation score worsens with respect to the previous result. Once again, the Random based Regressions do not offer good results either.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	1	1	1	1	1	1	1	1	1	9
IM 2	3	0	1	1	1	2	1	4	4	1	1	19
IM 3	3	3	0	0	0	2	0	3	3	0	0	14
IM 4	3	3	4	0	3	2	3	4	4	3	3	32
IM 5	3	3	2	0	0	2	1	4	4	0	0	19
IM 6	3	1	1	0	0	0	0	3	4	0	0	12
IM 7	3	3	2	0	0	3	0	4	4	0	0	19
IM 8	3	0	0	0	0	1	0	0	0	0	0	4
IM 9	3	0	1	0	0	0	0	3	0	0	0	7
IM 10	3	3	2	0	0	2	1	4	4	0	0	19
IM 11	3	2	2	1	2	4	2	4	4	2	0	26
Total	30	18	16	3	7	19	9	34	32	7	5	180

Table 3.4: Results of the statistical tests on the difference between the performance of the IMs in the TS with short missing segments using STS distance.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	0	0	0	0	1	0	0	0	0	1
IM 2	3	0	1	0	1	0	1	2	2	1	0	11
IM 3	4	1	0	0	0	1	0	3	3	0	0	12
IM 4	4	2	1	0	0	2	1	3	3	0	0	16
IM 5	4	1	0	0	0	0	0	3	3	0	0	11
IM 6	4	1	0	0	0	0	0	2	1	0	0	8
IM 7	3	1	0	0	0	0	0	3	3	0	0	10
IM 8	1	0	0	1	1	2	1	0	0	1	1	8
IM 9	2	0	0	0	0	2	0	0	0	0	1	5
IM 10	4	1	0	0	0	0	0	3	3	0	0	11
IM 11	3	2	2	1	1	2	1	3	3	1	0	19
Total	32	9	4	2	3	9	5	22	21	3	2	112

Table 3.5: Results of the statistical tests on the difference between the performance of the IMs in the TS with short missing segments using the ARIMA-LCP process distances.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	1	1	1	1	1	2	2	1	1	11
IM 2	3	0	1	1	1	1	1	3	3	1	1	16
IM 3	3	2	0	0	0	1	0	4	3	0	1	14
IM 4	3	2	2	0	1	1	1	4	4	1	2	21
IM 5	3	2	1	0	0	1	0	4	4	0	1	16
IM 6	3	2	1	1	1	0	1	4	4	1	1	19
IM 7	3	2	1	0	0	1	0	4	4	0	1	16
IM 8	1	1	0	0	0	0	0	0	0	0	1	3
IM 9	1	0	0	0	0	0	0	2	0	0	1	4
IM 10	3	2	1	0	0	1	0	4	4	0	1	16
IM 11	3	1	1	1	1	1	1	3	3	1	0	16
Total	26	14	9	4	5	8	5	34	31	5	11	152

Table 3.6: Results of the statistical tests on the difference between the performance of the IMs in the TS with short missing segments using Integrated Periodogram based distance.

This last individual Table 3.6 shows the results using the Integrated Periodogram based distance. This measurement puts MICE based Regular Regression in the top of the rank by a clear difference over the rest (17 overall, against 11 or lower obtained by the others). In this case, Kalman Polished Regression is unable to score better than its Regular version, (11 and 5 overall). Interpolation and Random based Regressions still fail to perform well.

Finally, the combined Table 3.7 shows the addition of all three previous tables (different metrics for the same MDT) and summarizes the observations made up to now. MICE Regular Regression tops the ranking, as it obtains a 60 overall score. Considerably far away stands Kalman Polished Regression, with 43. The other IMs are unable to do better than 31. The bad performance shown by Interpolation and Random Regressions is noteworthy.

3.4.5.2 Long Missing Segments The same experimental structure was used for the long missing segment imputation performance measure. Three tables of results obtained using the three different distance methods are shown first, and a final one that considers all the distances at the same time concludes.

Table 3.8 is based on the STS distance and shows nothing but equality in the top of the ranking. As in the short segment experimentation, the Kalman Polished Regression and the MICE Regular Regression continue to outperform the other IMs. This time it is the Kalman based IM that just beats the MICE based one (25 to 24 overall). The other methods fail to obtain more than 15.

Table 3.9 presents the results of the comparison among the IMs using ARIMA-LCP distance for long missing segments. The scores based on this measurement again differ from the others, as there are four methods with ratings of 15 and 14.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	2	2	2	2	3	3	3	2	2	21
IM 2	9	0	3	2	3	3	3	9	9	3	2	46
IM 3	10	6	0	0	0	4	0	10	9	0	1	40
IM 4	10	7	7	0	4	5	5	11	11	4	5	69
IM 5	10	6	3	0	0	3	1	11	11	0	1	46
IM 6	10	4	2	1	1	0	1	9	9	1	1	39
IM 7	9	6	3	0	0	4	0	11	11	0	1	45
IM 8	5	1	0	1	1	3	1	0	0	1	2	15
IM 9	6	0	1	0	0	2	0	5	0	0	2	16
IM 10	10	6	3	0	0	3	1	11	11	0	1	46
IM 11	9	5	5	3	4	7	4	10	10	4	0	61
Total	88	41	29	9	15	36	19	90	84	15	18	444

Table 3.7: Results of the statistical tests on the difference between the performance of the IMs in the TS with short missing segments considering all three dissimilarity measures.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	1	1	1	1	1	1	1	1	0	8
IM 2	3	0	1	1	1	3	4	4	4	1	0	22
IM 3	3	1	0	0	0	3	3	3	3	0	0	16
IM 4	3	3	3	0	2	3	3	4	3	2	1	27
IM 5	3	2	2	0	0	3	3	4	3	0	0	20
IM 6	3	0	1	0	0	0	0	2	2	0	0	8
IM 7	3	0	1	0	0	0	0	0	2	0	0	6
IM 8	3	0	1	0	0	0	1	0	2	0	0	7
IM 9	3	0	1	0	0	0	1	0	0	0	0	5
IM 10	3	2	2	0	0	3	3	4	3	0	0	20
IM 11	3	2	2	1	1	4	4	4	4	1	0	26
Total	30	10	15	3	5	20	23	26	27	5	1	165

Table 3.8: Results of the statistical tests on the difference between the performance of the IMs in the TS with long missing segments using STS distance.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	0	1	0	0	0	0	0	0	0	1
IM 2	3	0	0	0	0	1	2	1	1	0	0	8
IM 3	4	1	0	0	1	1	2	2	3	1	1	16
IM 4	3	2	1	0	0	2	2	2	3	0	1	16
IM 5	4	1	0	0	0	2	2	2	3	0	1	15
IM 6	4	1	0	0	0	0	0	0	0	0	1	6
IM 7	4	2	0	0	0	2	0	1	2	0	1	12
IM 8	3	0	0	0	0	1	1	0	1	0	0	6
IM 9	3	0	0	0	0	0	0	0	0	0	0	3
IM 10	4	1	0	0	0	2	2	2	3	0	1	15
IM 11	3	1	0	0	0	2	2	2	2	0	0	12
Total	35	9	1	1	1	13	13	12	18	1	6	110

Table 3.9: Results of the statistical tests on the difference between the performance of the IMs in the TS with long missing segments using the ARIMA-LCP process distances.

These are MICE, MICE Regular and Polished Regression, and Kalman Polished Regression. The other IMs do not perform better than 6.

Table 3.10 shows the outcome produced by Integrated Periodogram Based distance, over long missing segments. This time MICE Regular Regression once again stands tall in first position with a score of 29, as Kalman Regular Regression reaches second with 23.

This last Table 3.11 considers all the dissimilarity measures. As expected, MICE Regular Regression holds to the first place (68), followed by Kalman Regular Regression and MICE Polished Regression (52).

3.4.5.3 Short Time Series Distance Table 3.12 contains the results of the statistical tests for both lengths of missing segments and the STS distance. It shows a slight advantage of MICE Regular Regression (55) over Kalman Polished Regression (46), and once again, the other methods stay far from these results.

3.4.5.4 ARIMA-LCP distance Table 3.13, shows results for ARIMA-LCP distance and reaffirms the dominance of the MICE Regular Regression (29) over the other IMs; MICE and Kalman Regular Regression (23).

3.4.5.5 Integrated Periodogram based distance Table 3.14 shows results for Integrated Periodogram based distance. This table also confirms the supremacy that MICE Regular Regression has over the other IMs, as it scores 46, edging the second best, Kalman Regular Regular Regression, by 12 points.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	0	0	0	0	0	0	0	0	0	0
IM 2	3	0	0	0	0	3	3	2	3	0	0	14
IM 3	3	3	0	0	0	3	3	3	3	0	2	20
IM 4	4	4	2	0	1	3	3	4	4	1	3	29
IM 5	4	4	2	0	0	3	3	3	3	0	2	24
IM 6	4	1	1	0	0	0	0	0	0	0	0	6
IM 7	4	1	1	0	0	3	0	1	3	0	0	13
IM 8	4	1	1	0	0	2	2	0	2	0	0	12
IM 9	4	1	0	0	0	0	0	0	0	0	0	5
IM 10	4	4	2	0	0	3	3	3	3	0	2	24
IM 11	4	3	1	0	0	3	3	3	3	0	0	20
Total	38	22	10	0	1	23	20	19	24	1	9	167

Table 3.10: Results of the statistical tests on the difference between the performance of the IMs in the TS with long missing segments using Integrated Periodogram based distance.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	1	2	1	1	1	1	1	1	0	9
IM 2	9	0	1	1	1	7	9	7	8	1	0	44
IM 3	10	5	0	0	1	7	8	8	9	1	3	52
IM 4	10	9	6	0	3	8	8	10	10	3	5	72
IM 5	11	7	4	0	0	8	8	9	9	0	3	59
IM 6	11	2	2	0	0	0	0	2	2	0	1	20
IM 7	11	3	2	0	0	5	0	2	7	0	1	31
IM 8	10	1	2	0	0	3	4	0	5	0	0	25
IM 9	10	1	1	0	0	0	1	0	0	0	0	13
IM 10	11	7	4	0	0	8	8	9	9	0	3	59
IM 11	10	6	3	1	1	9	9	9	9	1	0	58
Total	103	41	26	4	7	56	56	57	69	7	16	442

Table 3.11: Results of the statistical tests on the difference between the performance of the IMs in the TS with long missing segments considering all three dissimilarity measures.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	2	2	2	2	2	2	2	2	1	17
IM 2	6	0	2	2	2	5	5	8	8	2	1	41
IM 3	6	4	0	0	0	5	3	6	6	0	0	30
IM 4	6	6	7	0	5	5	6	8	7	5	4	59
IM 5	6	5	4	0	0	5	4	8	7	0	0	39
IM 6	6	1	2	0	0	0	0	5	6	0	0	20
IM 7	6	3	3	0	0	3	0	4	6	0	0	25
IM 8	6	0	1	0	0	1	1	0	2	0	0	11
IM 9	6	0	2	0	0	0	1	3	0	0	0	12
IM 10	6	5	4	0	0	5	4	8	7	0	0	39
IM 11	6	4	4	2	3	8	6	8	8	3	0	52
Total	60	28	31	6	12	39	32	60	59	12	6	345

Table 3.12: Results of the statistical tests on the difference between the performance of the IMs in the TS with long and short missing segments using STS Distance.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	0	1	0	0	1	0	0	0	0	2
IM 2	6	0	1	0	1	1	3	3	3	1	0	19
IM 3	8	2	0	0	1	2	2	5	6	1	1	28
IM 4	7	4	2	0	0	4	3	5	6	0	1	32
IM 5	8	2	0	0	0	2	2	5	6	0	1	26
IM 6	8	2	0	0	0	0	0	2	1	0	1	14
IM 7	7	3	0	0	0	2	0	4	5	0	1	22
IM 8	4	0	0	1	1	3	2	0	1	1	1	14
IM 9	5	0	0	0	0	2	0	0	0	0	1	8
IM 10	8	2	0	0	0	2	2	5	6	0	1	26
IM 11	6	3	2	1	1	4	3	5	5	1	0	31
Total	67	18	5	3	4	22	18	34	39	4	8	222

Table 3.13: Results of the statistical tests on the difference between the performance of the IMs in the TS with long and short missing segments using ARIMA-LCP Distance.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 0	1	2	3	4	5	6	7	8	9	10	11	12
IM 1	0	0	1	1	1	1	1	2	2	1	1	11
IM 2	6	0	1	1	1	4	4	5	6	1	1	30
IM 3	6	5	0	0	0	4	3	7	6	0	3	34
IM 4	7	6	4	0	2	4	4	8	8	2	5	50
IM 5	7	6	3	0	0	4	3	7	7	0	3	40
IM 6	7	3	2	1	1	0	1	4	4	1	1	25
IM 7	7	3	2	0	0	4	0	5	7	0	1	29
IM 8	5	2	1	0	0	2	2	0	2	0	1	15
IM 9	5	1	0	0	0	0	0	2	0	0	1	9
IM 10	7	6	3	0	0	4	3	7	7	0	3	40
IM 11	7	4	2	1	1	4	4	6	6	1	0	36
Total	64	36	19	4	6	31	25	53	55	6	20	319

Table 3.14: Results of the statistical tests on the difference between the performance of the IMs in the TS with long and short missing segments using Integrated Periodogram based distance.

Method	IM 1	IM 2	IM 3	IM 4	IM 5	IM 6	IM 7	IM 8	IM 9	IM 10	IM 11	Total
IM 1	0	0	3	4	3	3	4	4	4	3	2	30
IM 2	18	0	4	3	4	10	12	16	17	4	2	90
IM 3	20	11	0	0	1	11	8	18	18	1	4	92
IM 4	20	16	13	0	7	13	13	21	21	7	10	141
IM 5	21	13	7	0	0	11	9	20	20	0	4	105
IM 6	21	6	4	1	1	0	1	11	11	1	2	59
IM 7	20	9	5	0	0	9	0	13	18	0	2	76
IM 8	15	2	2	1	1	6	5	0	5	1	2	40
IM 9	16	1	2	0	0	2	1	5	0	0	2	29
IM 10	21	13	7	0	0	11	9	20	20	0	4	105
IM 11	19	11	8	4	5	16	13	19	19	5	0	119
Total	191	82	55	13	22	92	75	147	153	22	34	886

Table 3.15: Results of the statistical tests for all TS distances and MD lengths.

3.4.5.6 Full Table Table 3.15 contains the summation of all distances and lengths. As it could have been expected, MICE Regular Regression obtains the best score of all methods (128) over Kalman Polished Regression (85), MICE Polished Regression, and Kalman Regular Regression (both 83).

3.4.6 Computational time

Figure 3.10 shows the total results obtained by each IM, differentiated by the MDT they were covering. Note that all the results in the right part of the vertical blue line correspond to Regressions based on the methods written below the chart.

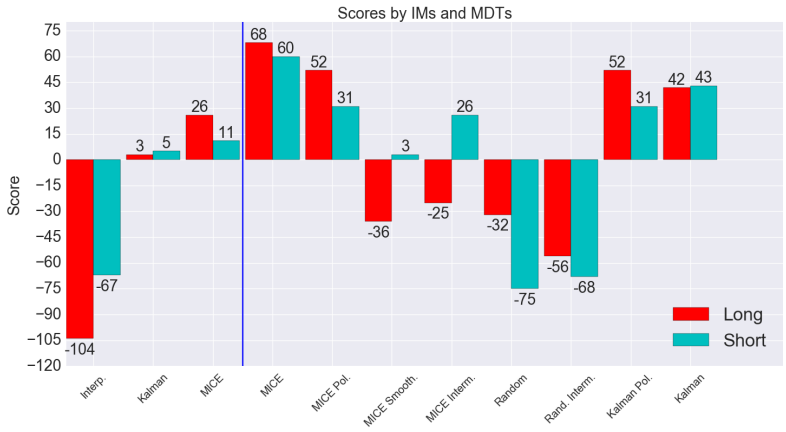


Fig. 3.10: Scores obtained by IMs on both long and short MDTs introduced.

Figure 3.11 represents the Table 3.15 in a heat map. Note that the column and row representing the *Total* show the mean of all the results, in order to make it possible to distinguish all the color differences. Also as in the previous Figure 3.10, all the results in the right part of the vertical blue line, and in this case, below the horizontal green line refer to results obtained by Regressions based on the methods represented in the margins of the chart.

3.4.7 Global analysis of the IMs

Figure 3.12 shows the computational complexity of the different IMs in terms of processing time. The largest multivariate TS was considered for this experiment (the one containing 100 univariate TSs in 100 different timestamps). It had it length multiplied by 2,3,4 and 5 in order to roughly approximate a scenario

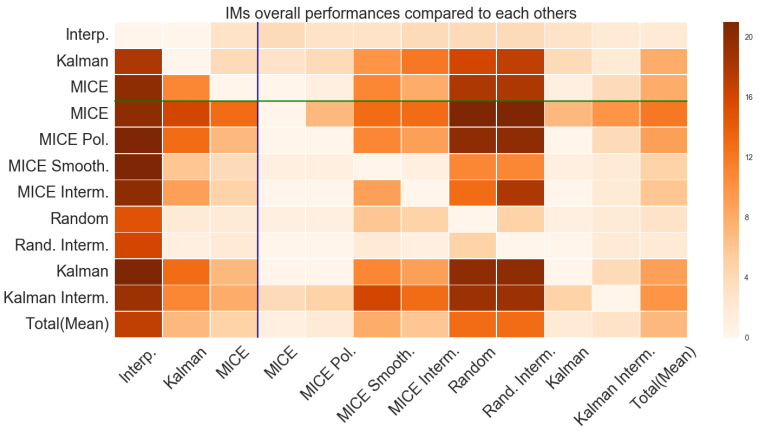


Fig. 3.11: Heatmap origined from Table 3.15. A dark color represents a high number in the table. An IM with a dark row and a light column would have produced good results.

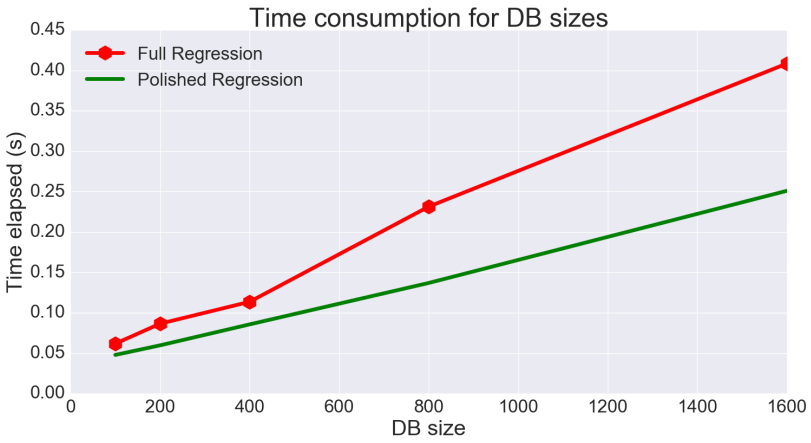
in which fast imputation is needed. This resulted in other four DBs all of them with 100 univariate TSs, but with 200, 400, 800 and 1600 timestamps. These DBs underwent the same procedure to introduce MD, scaled to the corresponding DB dimension, and were finally imputed. In this case, the time consumed by each process was measured. Figure 3.12 is divided into two charts to differentiate the time consumption of the processes run in different environments, **R** (Figure 3.12b) and Python (Figure 3.12a).

This whole process was executed 30 times, and the means obtained by all the values are shown in the chart.

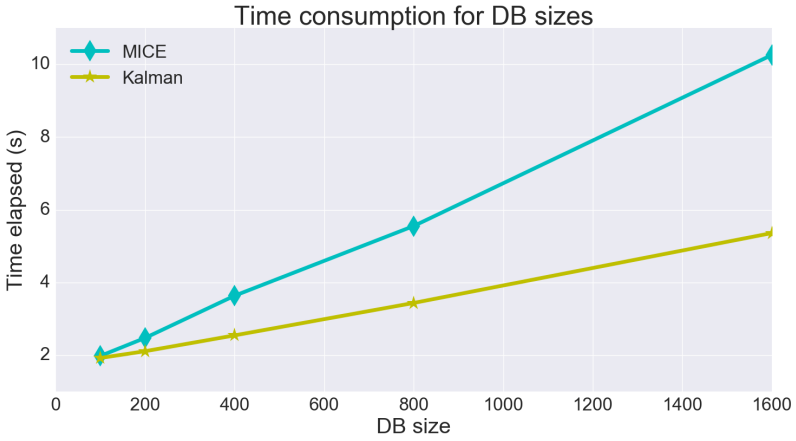
3.4.8 Analysis of the results

Starting with the tables, it seems like the different metrics used in the experiments produced the results desired, as they offered different outcomes. Particularly, ARIMA-LCP is the only dissimilarity that put Kalman Polished Regression over MICE Regular Regression. Also, this metric showed less differences among the IMs than the other two. Despite this fact, the rest of the tables converge to the same argument, manifesting the superiority of the performance offered by MICE Regular Regression over the other IMs.

This time, the different procedures to introduce MD did not produce a very diverse outcome, as it could have been expected. The long missing segments accentuated the dominance of MICE Regular Regression over the other IMs more than MCAR, but the ranking of top IMs did not change.



(a) Time elapsed during the Python processes.



(b) Time elapsed during the **R** processes.

Fig. 3.12: Time consumed by different components involved in top scoring IMs.

It is noteworthy the bad performance offered by Interpolation, and all the methods that involve this technique, as only Kalman Polished Regression was sometimes present in the higher positions, while other times it was surpassed by simpler methods like MICE. The Edge Smoothing procedure did not make it to the top of the ranking in any consideration.

Also, the poor behavior showed by the Random Regular Regression demonstrates the fact that the usage of a *good* IM as a background for Regression is absolutely necessary.

Regarding the time consumption, Figure 3.12 shows that the task that holds the best performing position, also requires a long processing time, in this case, the base method MICE is the bottleneck. It is worthy remembering that both base imputations (MICE and Kalman) have been implemented using **R**, and that the time appearing in the chart includes input and output costs, which possibly represent a large part of the total. However, this cost should be the same for both applications, and their increasing rates are different. Even though both are linear-like, MICE grows (considerably) faster than Kalman. Also Figure 3.12 shows the insignificant amount of elapsed time while running the Python parts , for which they shall not be considered in this instance.

3.5 Conclusions

The main objective of this chapter was to introduce and compare new complex IMs, specific for the MD in TS problem. With this target in mind, some artificial DBs had some values erased following two MD patterns, and were imputed by several simple and newly proposed IMs. Finally, their performance was measured using different metrics.

Starting with the simple methods, Interpolation provided incredibly bad results, being by far the worst IM among the considered ones. It was MICE the method that obtained the best result among the three most basic methods, leaving Kalman in second place with an acceptable performance.

About the utility of Polished Regression, it did not produce a sufficiently good result results to be paired with the Full Regression when MICE was used as a base method. However, when Kalman Imputation was performed before reimputing with a regression, the results were similar considering overall performance.

It is demonstrated that the application of a good imputation before performing regression is absolutely necessary. The random based regressions produced very poor results, while the regressions performed on DBs that had been imputed by *good* IMs were the best strategies in terms of data quality.

The two combination attempts of Interpolation and Regression (Intermittently and Edge Smoothed) were also clearly outperformed by the rest of the regressions. This was unmistakably due to the bad results provided by Interpolation.

Regarding differences between the performances discriminating results by the MD introducing strategies, the top method, MICE based Regression did not have a considerable variation on its results. Neither did Kalman-based Regression, the second one on the imputation quality ranking. However, the MICE and Kalman based Polished Regressions did, performing considerably better in when the MD was introduced in long segments, rather than short ones.

Interpolation also experienced a considerable difference between its performances in different MDTs. As it could have been expected, it offered much better results for short segments of missing values. Nevertheless, the outcome is bad in both contexts.

In terms of time consumption, the biggest difference was found between Kalman-model Imputation and MICE. MICE almost doubled the time growth rate of Kalman.

Overall, it was determined that MICE based Regression is the best IM of all those considered. If there is no urge of fast imputation, its usage is absolutely recommended. However, in cases that need a considerably fast imputation, Kalman based Regression could be the correct choice. Also, it was able to be the top IM in some occasions.

Chapter 4

Advanced Imputation Methods for a real-world time series prediction problem

4.1 Objectives

In previous Chapters 2 and 3 we have investigated the application of IMs on DBs from the literature. In this section we investigate the MD issue in a real-world problem of event-detection in multivariate TSs.

These TSs in question find their origin in an analysis of the quality of fluvial environments. The TSs contain different parameters related to the water condition. The sensors could fail at the time of recording or sending the information in question, and this behavior leads to the introduction of MD in the TSs.

The main objective of this chapter is to evaluate the effect of imputation in this real-world problem. We will exploit the knowledge gathered and generated in the previous two chapters of this work.

4.2 SIRENE® Project

The SIRENE®[8] project continuously measures a number of parameters from water resources. The analysis of these indicators can be used to conserve, or detect the necessity to improve, the quality of the water bodies [57]. The stations that the SIRENE® project deploys perform a real time recording of diverse high frequency parameters from surface waters. These stations are divided into two types. Autonomous stations, which may be found in buoys and are battery-powered, and stationary ones, which work in riverbanks and need continuous power supply.

These machines record data every 10 minutes. This information is sent to a platform, with the main objective of determining water's quality in various aspects and automatic detection of condition degrading pollution events. The sensors can record up to 13 different parameters. In Table 4.1, these measurements can be found, as well as their minimum and maximum feasible values.

[8] The SIRENE® project is developed by the Rivages Pro Tech company.

Indicator	Min. feasible value	Max. feasible value	Used in this Work
Water level (m)	0		✓
Temperature (C)	0	40	✓
Conductivity (µ/cm)	100	5000	✓
Turbidity (NTU)	0	2000	✓
pH	0	14	✓
Dissolved oxygen (mg/L or %)	0	20 or 200	✓
Ammonia (mg/L)	0	100	✓
Orthophosphates (mg/L)	0	10	✗
Hydrocarbon			✗
Nitrates (mg/L)	0	100	✗
Redox potential (mV)	-1000	1000	✓
Chlorophyll a (µg/L)	0	100	✗
Phycocyanin (µg /L)	0	100	✗

Table 4.1: All the parameters that SIRENE® stations can record along with their maximum and minimum feasible values.

4.2.1 Event detection and Missing Data problem in SIRENE® database

As it was previously mentioned, this enormous data quantity is recorded, among other things, in order to be able to detect anomalies in water. With this objective, an expert tagged some records of the data in two classes. It determined whether an event happened (or was happening) or not. An event is recognized by the change in the parameters of the water with respect to normal values. It can be due to a natural phenomenon (e.g. rain) or be the result of external intervention (e.g. water pollution due to waste from an industry). The task here is to use these annotated DBs to try to perform automatic detection of events in unannotated multivariate TSs.

This dataset can also suffer from the MD problem. The most common missing values are caused by two reasons:

- The first pattern to be observed is that values for the same variable are missing for contiguous measurements, which is probably caused by a sensor not working properly.

- The second main pattern is a full timestamp being lost, which probably is caused by a connection error between the platform and the SIRENE® deployments.

Nevertheless, these are not the only MD patterns in the data, as isolated missing values are also present. Not responding to a definable pattern.

4.2.2 Database Description

DBs generated by the deployed stations can contain up to 13 TSs (recording the 13 parameters enumerated in Table 4.1). However, this is not necessarily the case for all the DBs as some can contain a subset of the whole group.

In this case, three examples provided by SIRENE® have been used, named Example 1, Example 2 and Example 3. Each one is defined by a measurement of certain subset of parameters over a defined period of time. Table 4.2 shows the characteristics of each one.

DB	TSs	Timestamps	Annotated
Example 1	8	8,782	✓
Example 2	7	29,142	✓
Example 3	7	32563	✗

Table 4.2: Characteristics of the databases available and used for our analysis.

The TSs databases considered for our analysis contain 8 or 7 TSs, which allows us to perform an analysis from a multivariate approach (Section 1.2.2).

4.2.3 Missing Data Description

Here we illustrate the characteristics of MD in the SIRENE® DB examples. This analysis is supported by graphical visualization (Figures 4.1-4.6) of the three different multivariate TS examples generated by SIRENE®. This step is required to identify which, among the types of MD discussed before (Section 3.4.3), is present in the data.

The first consignment of plots includes Figures 4.1 to 4.6. The axes of these graphics represent the number of variables contained in a particular dataset (horizontal axis), and all the possible MD combinations (vertical axis). The color represents the amount of times that the combination was found; the warmer the color, the more times the combination was present. To obtain these combinations, DBs are converted into a binary table, which contain "0" when a value is missing and "1" when it's not. This binary matrix could be interpreted as a decimal list, representing each decimal (binary row) a different missing value combination. For example, if in a DB with eight variables, there is a timestamp containing a missing value in the second column, then it will be transformed to (1,0,1,1,1,1,1,1), and interpreted as 253. Some examples can be found in Figures 4.1, 4.3, 4.5. Also, as it is very difficult to identify a clear pattern from them,

summarized graphics containing only combinations suffering from missing values are shown in Figures 4.2, 4.4, 4.6.

Figures 4.7 and 4.8 summarize the lengths of the missing segments in the three examples considered. These figures contain the histograms describing the lengths of the missing contiguous pieces of information, i.e. missing segments for all the examples combined and separately, respectively. Note that this histogram only considers segments up to 50 unities long, despite the fact that in the data there may be found sequences as long as 1479. To compress this information, the frequencies of all sequences longer than 50 were considered together with those equal to 50.

As it can be seen in the chart, the majority of the missing segments can be considered as long sequences since their length is greater than two [4]. Also, there are some enormous missing gaps, that can reach to 1479 consecutive latent entries, omitted in the figure for the sake of simplicity. This probably would have a considerable effect in the performance of some IMs, according to the information obtained in Section 3.4.5. For example, observing the results offered by Interpolation and any other strategy involving it, it would be highly inadvisable for this type of MD. However, the DBs contain a small portion of short missing entries.

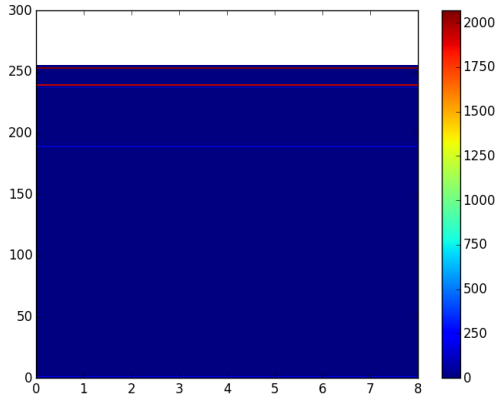


Fig. 4.1: Example 1 MD distribution.

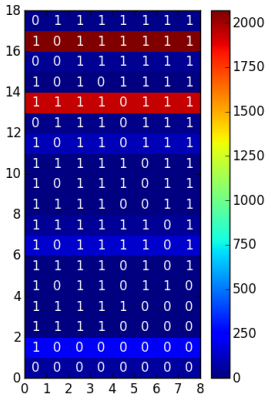


Fig. 4.2: Example 1 MD distribution (condensed).

Figures 4.1 - 4.6 illustrate the existence of three main types of MD. The most obvious MDT is the one that causes a full observation to go missing. This type can be found in Example 2 and 3. Example 2 shows over 2400 observations fully

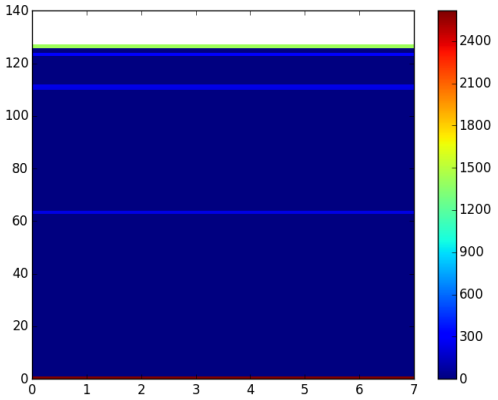


Fig. 4.3: Example 2 MD distribution.

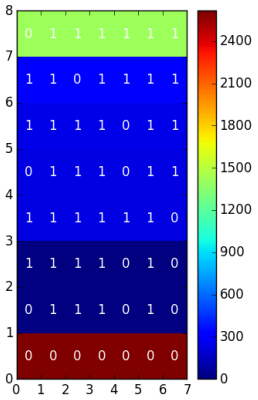


Fig. 4.4: Example 2 MD distribution (condensed).

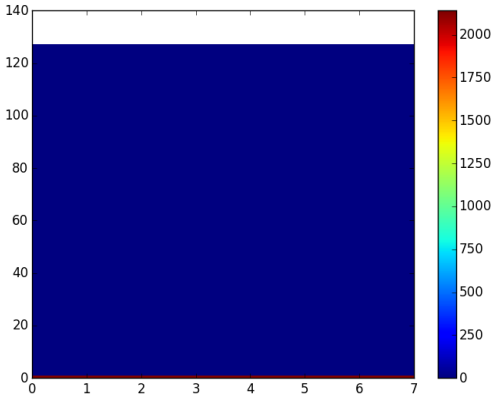


Fig. 4.5: Example 3 MD distribution.

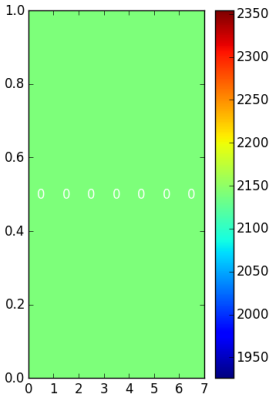


Fig. 4.6: Example 3 MD distribution (condensed).

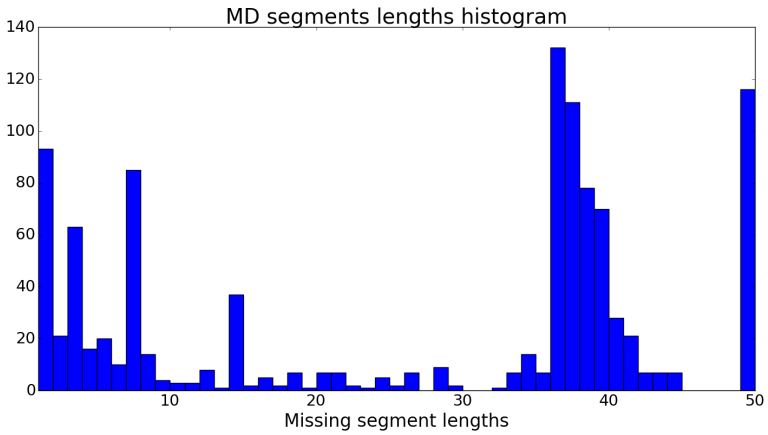
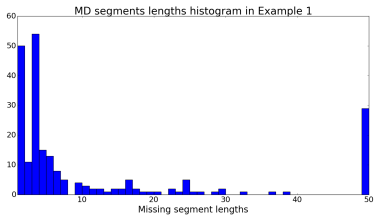
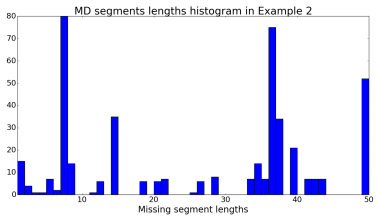


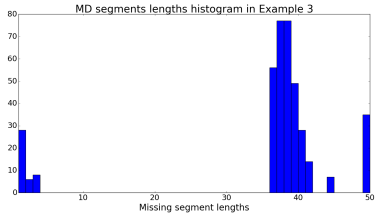
Fig. 4.7: MD segment length distribution in all the three examples exposed in this work, grouped.



(a) MD segment length distribution in Example 1.



(b) MD segment length distribution in Example 2.



(c) MD segment length distribution in Example 3.

Fig. 4.8: MD segment length distribution in all the three example exposed in this work, individually.

lost, while in Example 3, there are over 2000 observations with MD, and in all cases the full timestamp was lost.

The second most common type of MD is the adjacent missing values, which is similar to the missing blocks explained above, but for a single TS instead of the full observation. This can be seen in Examples 1 and 2, as there are considerable amount of observations with the same pattern. Example 1 reflects this situation in the (1, 0, 1, 1, 1, 1, 1) and (1, 1, 1, 1, 0, 1, 1, 1) combination, which represent almost all of the MD in the DB. Example 2 finds this same scenario with the (0, 1, 1, 1, 1, 1, 1) combination. These timestamps with MD will presumably be adjacent.

Finally, there are some other observations that lose values randomly, producing MCAR-like type of MD. Examples 1 and 2 show this type of problem in their data.

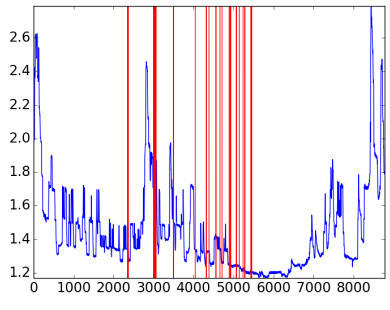
Figure 4.7 confirms that MCAR-like MDT exists, as we can find short lengths of missing adjacent values (one or two). However, it seems like the predominant type is the long missing segment form. Figure 4.8 contains the same information, but separated by DB, i.e., each multivariate TS has its own chart. The first Figure 4.8a represents the MD segment length distribution for the Example 1 DB, and it shows that the predominant MDT is MCAR, as most of the segments are relatively short. Figure 4.8b corresponds to Example 2, and it shows more distributed segment lengths. The most common lengths are positioned around 10 and 40, which makes this DB a long MD segment container. Finally, Figure 4.8c shows the same information for Example 3, and we can see that it contains little amount of MCAR combined with more frequent long missing segments.

These figures demonstrate the variability that the MD problem can experience, even in the same research area. Furthermore, Example 2 and Example 3 have been recorded in the same place, in different periods of time, and the MD distribution varies enormously.

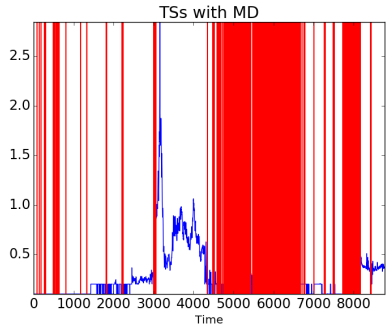
Figures 4.9, 4.10 and 4.11 show the behavior of all TSs in each of the DBs, representing MD with red blocks. The x axis represents the index of the record entry and the y axis the measurement for the variable.

As a general conclusion of this study, we have been able to determine that three types of MD are present in these DBs. The first one would correspond to the fully missing, contiguous observations (Example 2, Figures 4.5 and 4.6). This MDT could be cataloged as a data transmission failure. The second would be the long lost segments, majority in the data, as shown in Figure 4.7. Finally, the last (and least common) MDT in these DBs would be the MCAR. The TSs do contain this pattern of MD, but as it can be seen in Figure 4.7, it represents a small portion of the total missing values.

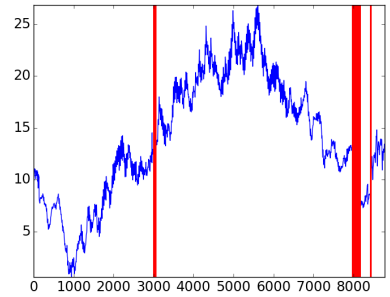
Example 1 contains 9.625% of MD. Example 2, 10.253%. But the last 85 observations of this second DB are completely missing. These timestamps are absolutely useless, as the only extractable information would be the time they were



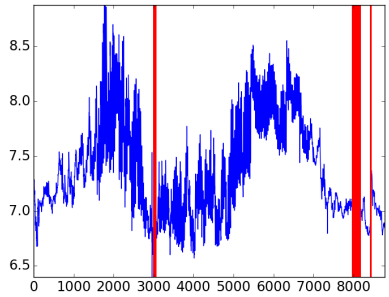
(a) Water level TS in Example 1.



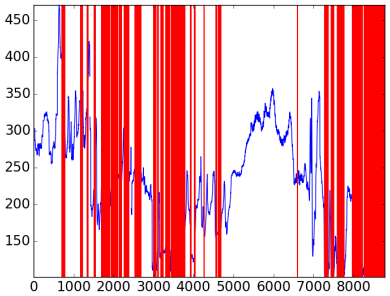
(b) Ammonium TS in Example 1.



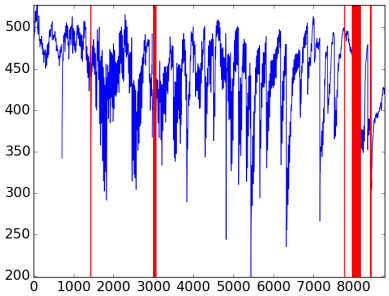
(c) Temperature TS in Example 1.



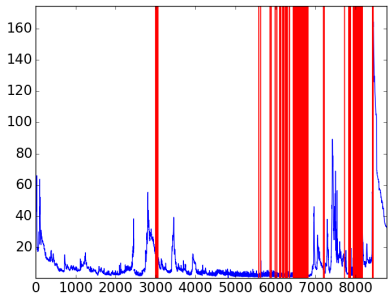
(d) pH TS in Example 1.



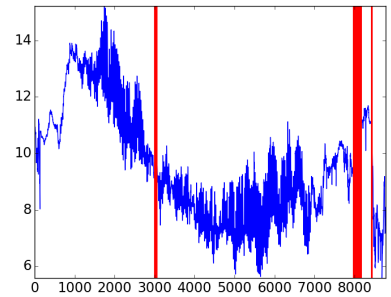
(e) Conductivity TS in Example 1.



(f) Redox TS in Example 1.

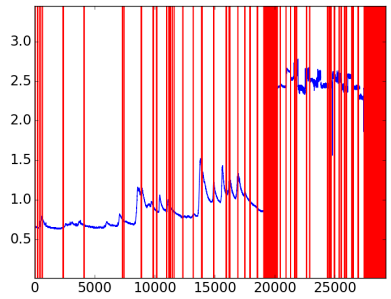


(g) Turbidity TS in Example 1.

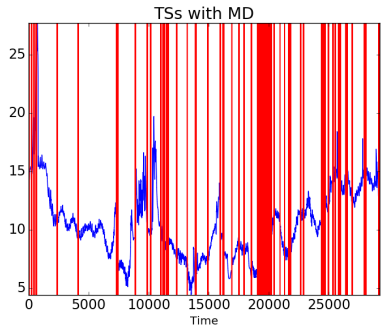


(h) Oxigen TS in Example 1.

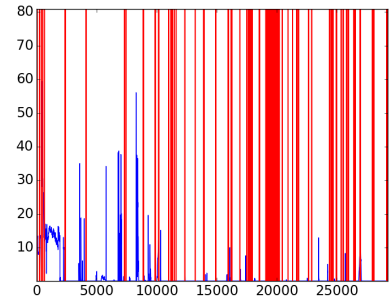
Fig. 4.9: Time series of all variables in Example 1. Missing data is shown with red lines.



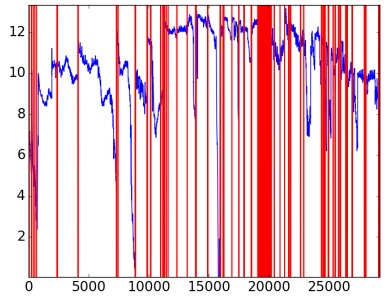
(a) Water level TS in Example 2.



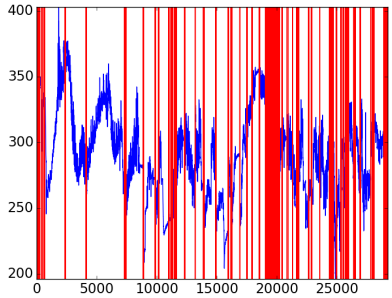
(b) Temperature TS in Example 2.



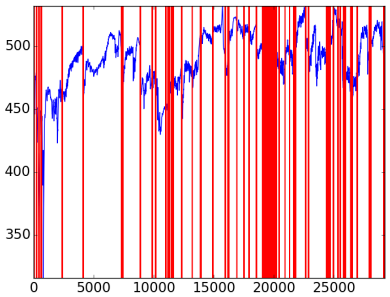
(c) Ammonium TS in Example 2.



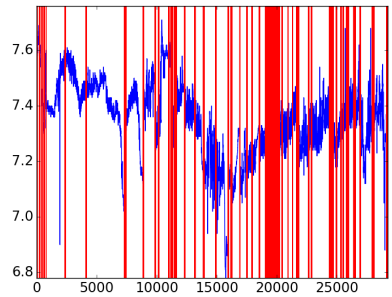
(d) Oxigen TS in Example 2.



(e) Conductivity TS in Example 2.

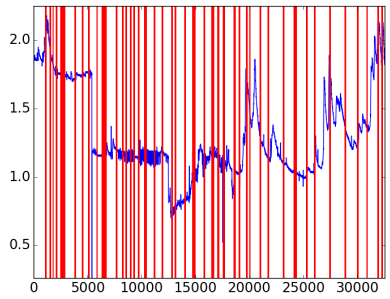


(f) Redox TS in Example 2.

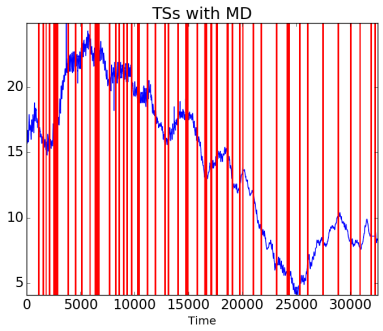


(g) pH TS in Example 2.

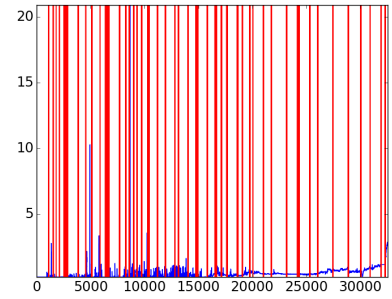
Fig. 4.10: Time series of all variables in Example 2. Missing data is shown with red lines.



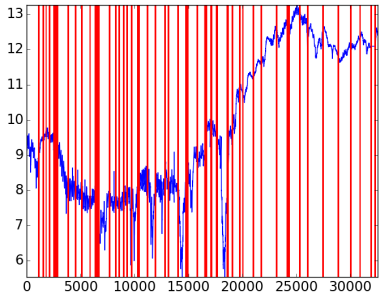
(a) Water level TS in Example 3.



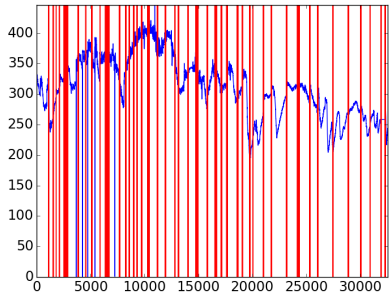
(b) Temperature TS in Example 3.



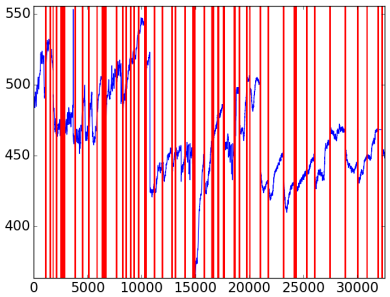
(c) Ammonium TS in Example 3.



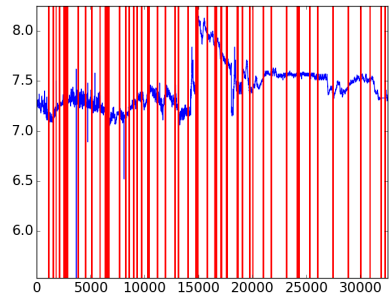
(d) Oxigen TS in Example 3.



(e) Conductivity TS in Example 3.



(f) Redox TS in Example 3.



(g) pH TS in Example 3.

Fig. 4.11: Time series of all variables in Example 3. Missing data is shown with red lines.

obtained in. For this reason, with these observations ignored, the MD percentage decreases to 9.962%. Finally the last DB has 6.52% of missing values.

4.3 Imputation Method selection

Once the MD distribution is known, the next step to solve the problem would be choosing a suitable IM. The results obtained in Chapter 3 should be the starting point. The best two methods in terms of imputation accuracy were: 1)Mice Regular Regression and 2) Kalman Polished Regression. In the experiments that obtained the ranking of IMs, the TSs were generated from the same ARIMA model, which possibly causes that the variables are strongly related to each other. This would mean that the Polished versions of the Regressions could have been virtually improved. Considering the SIRENE® DBs, with much more independent TSs, these strategies will likely offer poorer results. This is the reason why we do not recommend the usage of methods involving Polished Regression.

Regarding the behavior of the two complete regression methods investigated in the previous chapter, the final conclusion stated that using MICE as a basis for subsequently performing regression produced better results than Kalman, in terms of imputation value accuracy. However, Kalman offered much better results when computational time consumption is the main concern.

The task of imputing the SIRENE® DBs involves real time series, but since there is a considerable amount of time between timestamp recording, this problem does not require fast imputation algorithms. In consequence, the MICE based regression is the most advisable IM for this problem.

4.4 Imputation

Once a good IM candidate has been selected for this problem, it is time to proceed to fill the missing gaps.

4.4.1 Goal and methodology

As stated before, the objective of the SIRENE® project is to detect events that lead to quality changes in the water, as expressed in the measurements for the different descriptors. The target of this concrete task is to do so automatically, thus, a SC problem. For this reason, two of the examples provided will be used as a benchmark for this imputation problem. These two DBs are appropriate because they have been manually tagged (event occurring or not) by expert knowledge.

This task is treated as a binary SC problem. Whether an event has occurred (represented by 1) or not (represented by 0). Therefore, after imputing an annotated, incomplete DB, the data is put through a 5-fold cross-validation process to determine the effectiveness of the imputation. The result of the process at the end is leveled to other standard IM to get to know whether the *complex* imputation really pays off. Then, the same procedure is applied to the other annotated DB.

4.4.2 Experimental Framework

This is the methodology followed to determine the effectiveness of IMs; First, a multivariate TS is imputed by two methods. One of the methods was the strategy chosen in previous Section 4.4.1, the MICE Regular Regression. The other approach, a Random Imputation. This second method produces a value between the maximum an minimum value of a TS to impute values of that same TS. Therefore, a simple IM is available in order to compare the performance of the chosen method. The behavior of the IMs is tested through a 5-fold cross validation using a Logistic Regression classifier with l1 penalization (for more details, see Section 2.3.4).

Since both MICE and random imputation are stochastic methods, 30 executions of the same program are run, then compared to each other using the same statistical test used in previous experiments in this work, the Kruskal-Wallis test. Also, the average accuracy of all the experiments are computed.

4.4.3 Results

The means of the 30 executions for the 2 TSs, for the two methods are shown in Table 4.3:

Example 2		Example 3	
Random Imputation	MICE Regression	Random Imputation	MICE Regression
0.651	0.696	0.808	0.812

Table 4.3: Results obtained by both Random and MICE Regression Imputation in the two Example DBs.

As it can be seen, only one of the problems experienced a classification accuracy improvement when imputed by a complex method. This result is supported by the mentioned statistical test, as it showed significant differences between the 30 results obtained in the Example 2 DB (p-value ≈ 0).

On the other hand, Example 3 showed almost no difference on classification accuracies when imputed by different methods, but the statistical test once again indicated considerable differences (p-value ≈ 0).

4.5 Conclusions

This chapter has focused on treating a real-world DB with its original MD problem. These DB contained records of the same variables over time, which catalogs them as multivariate TSs. We found three types of MD, each one potentially produced by different causes. These DBs are useful to address an event detection problem, in which the variable values determine whether a particular event has been produced. To treat this problem, the knowledge built in the previous chapters has been used, in order to identify, from a characterization of the MDT, which are the most suitable IMs.

The results obtained in this chapter back up the findings deduced in previous chapters. We have corroborated that the effectiveness of the imputation procedure strongly depends on the problem.

Both problems in this real-world DB with MD scenario contained around 10% of missing values (9.625% and 9.962%), for a disparate amount of timestamps (8.782 and 29.142).

Even though the MD percentage and the information (variables) gathered are almost equal, the results the IMs produced were considerably different. One DB experienced a large increase in the classification accuracy (from 65% to almost 70%), while the accuracy of the other DB remained almost the same (improved from 80% to 81%). Part of the large improvement found in the first DB can be caused by the fact that the better the existing information is for SC purposes, the harder is improving via imputation.

In general, it seems clear that the dependence of the performance of an IM on the problem is a strong factor in this task, and this dependence may come in different ways, i.e., MD percentage, MDT, observation and variable amounts, etc. But as an overall conclusion, it seems safe to affirm that imputation does not harm the data, but does not necessarily improve it.

Chapter 5

Conclusions

5.6 Summary

In this thesis we have treated the missing data problem and one general strategy to minimize the effect this issue has over the data quality, the use of imputation methods. To achieve this goal, the interactions between these two factors among other secondary components, have been investigated.

First a review of the basic concepts, related to the topics treated in the thesis, was set as an introduction.

Following the related work research, a concrete problem was chosen, imputation in discrete DBs (DBs in which independence between observations is assumed). To start with this concrete problem specification involving discrete DBs, the literature on this topic was revised in order to understand where the limits of the investigation in this area laid. Then, a series of experiments were set to make a first exploration of the mechanics behind the treatment of MD in this domain. As a result of this research, we searched, found, and described some patterns of interactions between MDTs, IMs and SC algorithms.

After investigating discrete DBs, the problem specification was narrowed to another type of DB, the multivariate TSs. Again, a review of the most important literature in this area was performed.Based on the findings of the review, we propose a number of IMs based on the combination of regression and temporality. In a new set of experiments, we compared different variants of IMs with introduced new strategies, and studied their shortcomings. The background obtained in the discrete part was applied to this experimental section. In this chapter, the performances of various simple and complex IMs were tested, so as to have an idea of what IMs can provide promising results.

Finally, a problem of water quality control DBs with MD was treated. This is a real-world problem, with real-world DB, and MD. While addressing this problem the information gathered in the previous parts was employed once again, concluding in a final experiment. We tested the IMs with contrasted good performance in the previous two chapters.

5.7 Conclusions

Two main ideas can be concluded generally from the whole work described in this thesis.

The first one is that the results of missing values imputation is strongly problem dependent. In all three chapters the results obtained offered more contrasts between different problem specifications than distinct IMs or metrics.

However, some other factors have stood out. The quality of the IMs has also proven to be determinant in some cases. This can be considered as the second main component in this problem, as in some problems the complex methods could not perform better than other simpler strategies. But in other cases, they do make a difference. In a similar manner, some SC algorithms can also be determinant in the final outcome of the process.

The other main conclusion extractable from this work is that *good* imputation does not harm the data. In both experimental parts in this thesis *bad* imputation was deliberately performed so that other IMs could have their performance measured. In all the cases, the strategies that were expected to improve the outcome of any of the metrics used along this work, they delivered. The *good* IMs outscored the *bad* ones.

In the discrete domain, some interactions between the MDTs, IMs and SCs were detected. However, these were not found in the combinations that had the best accuracy at the end of the process. The largest relation in this subject was detected between MDT and IM.

In the TSs field, the imputation methods that involved the temporal component in their estimations failed to outperform those which ignore it. This fact suggests that the two problem specifications are not very different, despite the fact that the TSs could contain more information to be exploited when imputing. After all, the TS problem is a concrete instance of the general discrete problem. Nevertheless, the temporal component has proven to benefit the time consumption of the imputation methods that exploit this aspect, possibly due to seasonality.

Finally, the SIRENE® DB proved the effectiveness of the complex method in a real-world problem, but above all, confirmed the complete procedure strongly depends on the problem, and its characteristics.

5.8 Future Work

This work has considered all the factors involved in a classical MDT-IM-SC problem. In the MDT section, this thesis regarded four different types, while the majority of the literature does not differentiate more than three. The SC part was also widely covered, as the three common types of classification algorithms (tree, model building and lazy) were represented with, at least, two methods.

However, some imputation techniques were not taken into account in this work. The IMs used in this work cover a wide range of approaches (i.e. simple, complex, iterative, based on classification algorithms, specific for TS, different combinations of IMs...) but this research could be complemented in this aspect.

First of all, all the IMs used were third party implementations, had different origins and were developed in distinct environments. This complicates the understanding of the mechanics underlaying the application in terms of optimality regarding results and time consumption, among other issues. This could be solved by implementing the needed software on our own, which, on the other side, would result in an incredible expense of time.

Also, the analysis could be extended in the multivariate TS part, as not many DBs were considered. In addition, the data could not be an accurate representation of the real-world. In this aspect, the usage of more real-world databases is recommended

Finally, a more in depth analysis of new IMs specific for TSs should be addressed. The strategies utilized in this work did not take advantage of the temporal component, as they offered poorer results than traditional IMs. With this aim, the recommendation is to look for or conceive new methods for this specific type of DB.

References

1. E. Acuña and C. Rodríguez. The treatment of missing values and its effect on classifier accuracy. In *Classification, clustering, and data mining applications*, pages 639–647. Springer, 2004.

2. D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

3. R. R. Andridge and R. J. Little. A review of hot deck imputation for survey non-response. *International statistical review*, 78(1):40–64, 2010.

4. Z. Bar-Joseph, G. K. Gerber, D. K. Gifford, T. S. Jaakkola, and I. Simon. Continuous representations of time-series gene expression data. *Journal of Computational Biology*, 10(3-4):341–356, 2003.

5. G. E. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6):519–533, 2003.

6. C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

7. M. Blankers, M. W. Koeter, and G. M. Schippers. Missing data approaches in ehealth research: simulation study and a tutorial for nonmathematically inclined researchers. *Journal of medical Internet research*, 12(5):e54, 2010.

8. L. C. Blomberg and D. D. A. Ruiz. Evaluating the influence of missing data on classification algorithms in data mining applications. *SBSI 2013: Simpósio Brasileiro de Sistemas de Informação*, pages 734–743, 2013.

9. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

10. S. Buuren and K. Groothuis-Oudshoorn. MICE: Multivariate imputation by chained equations in R. *Journal of statistical software*, 45(3), 2011.

11. D. Casado de Lucas. Classification techniques for time series and functional data. 2010.

12. C.-C. Chiu, S.-Y. Chan, C.-C. Wang, and W.-S. Wu. Missing value imputation for microarray data: a comprehensive comparison study and a web tool. *BMC systems biology*, 7(Suppl 6):S12, 2013.

13. Y. Ding and J. S. Simonoff. An investigation of missing data methods for classification trees applied to binary response data. *The Journal of Machine Learning Research*, 11:131–170, 2010.

14. R. Elmasri. *Fundamentals of database systems*. Pearson Education India, 2008.

15. P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

16. A. Farhangfar, L. Kurgan, and J. Dy. Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, 41(12):3692–3705, 2008.

17. R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

18. J. H. Friedman. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989.

19. J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

20. X. Gan, A. W.-C. Liew, and H. Yan. Microarray missing data imputation based on a set theoretic framework and biological knowledge. *Nucleic Acids Research*, 34(5):1608–1619, 2006.

21. P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, 2010.

22. A. Gelman and J. Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, 2006.

23. I. A. Gheyas and L. S. Smith. A neural network-based framework for the reconstruction of incomplete data sets. *Neurocomputing*, 73(16):3039–3065, 2010.

24. E. M. Hernández-Pereira, D. Álvarez-Estévez, and V. Moret-Bonillo. Automatic classification of respiratory patterns involving missing data imputation techniques. *Biosystems Engineering*, 138:65–76, 2015.

25. M. Hilbert and P. López. The worlds technological capacity to store, communicate, and compute information. *science*, 332(6025):60–65, 2011.

26. J. Honaker, A. Joseph, G. King, K. Scheve, and N. Singh. Amelia: A program for missing data (windows version) Cambridge, MA: Harvard university, 2001.

27. J. Honaker and G. King. What to do about missing values in time-series cross-section data. *American Journal of Political Science*, 54(2):561–581, 2010.

28. J. Honaker, G. King, M. Blackwell, et al. Amelia II: A program for missing data. *Journal of statistical software*, 45(7):1–47, 2011.

29. E. R. Hruschka Jr, E. R. Hruschka, and N. F. Ebecken. Bayesian networks for imputation in classification problems. *Journal of Intelligent Information Systems*, 29(3):231–252, 2007.

30. T. Joachims. *Text categorization with support vector machines: Learning with many relevant features*. Springer, 1998.

31. R. Jörnsten, H.-Y. Wang, W. J. Welsh, and M. Ouyang. DNA microarray data imputation and significance analysis of differential expression. *Bioinformatics*, 21(22):4155–4161, 2005.

32. K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 273–280. IEEE, 2001.

33. D.-W. Kim, K.-Y. Lee, K. H. Lee, and D. Lee. Towards clustering of incomplete microarray data without the use of imputation. *Bioinformatics*, 23(1):107–113, 2007.

34. H. Kim, G. H. Golub, and H. Park. Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, 21(2):187–198, 2005.

35. G. King, J. Honaker, A. Joseph, and K. Scheve. Analyzing incomplete political science data: An alternative algorithm for multiple imputation. In *American Political Science Association*, volume 95, pages 49–69. Cambridge Univ Press, 2001.

36. O. Kramer. Scikit-learn. In *Machine Learning for Evolution Strategies*, pages 45–53. Springer, 2016.

37. K. Lakshminarayan, S. A. Harp, R. P. Goldman, T. Samad, et al. Imputation of missing data using machine learning techniques. In *KDD*, pages 140–145, 1996.

38. K. Lakshminarayan, S. A. Harp, and T. Samad. Imputation of missing data in industrial databases. *Applied Intelligence*, 11(3):259–275, 1999.

39. Y. Li and L. E. Parker. A spatial-temporal imputation technique for classification with missing data in a wireless sensor network. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3272–3279. IEEE, 2008.

40. Y. Li and L. E. Parker. Nearest neighbor imputation using spatial–temporal correlations in wireless sensor networks. *Information Fusion*, 15:64–79, 2014.

41. M. Lichman. UCI machine learning repository, 2013.

42. Y. Liu and S. D. Brown. Comparison of five iterative imputation methods for multivariate classification. *Chemometrics and Intelligent Laboratory Systems*, 120:106–115, 2013.

43. J. Luengo, S. García, and F. Herrera. A study on the use of imputation methods for experimentation with radial basis function network classifiers handling missing attribute values: the good synergy between RBFNs and EventCovering method. *Neural Networks*, 23(3):406–418, 2010.

44. J. Luengo, S. García, and F. Herrera. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and information systems*, 32(1):77–108, 2012.

45. E. T. Matsubara, R. C. Prati, G. E. Batista, and M. C. Monard. Missing value imputation using a semi-supervised rank aggregation approach. In *Advances in Artificial Intelligence-SBIA 2008*, pages 217–226. Springer, 2008.

46. W. McKinney et al. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56, 2010.

47. C. S. Möller-Levet, F. Klawonn, K.-H. Cho, and O. Wolkenhauer. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *International Symposium on Intelligent Data Analysis*, pages 330–340. Springer, 2003.

48. P. Montero and J. A. Vilar. TSclust: An R package for time series clustering. *Journal of Statistical Software*, 62(1):1–43, 2014.

49. U. Mori, A. Mendiburu, and J. Lozano. *TSdist: Distance Measures for Time Series Data*, 2016. R package version 3.3.

50. U. Mori Carrascal. *Contributions to time series data mining departing from the problem of road travel time modeling.* PhD thesis, UPV/EHU, 7 2015.

51. R. Nau. Introduction to ARIMA: nonseasonal models. `http://people.duke.edu/~rnau/411arim.htm`. Accessed: 2016-09-09.

52. B. M. Nogueira, T. R. Santos, and L. E. Zárate. Comparison of classifiers efficiency on missing values recovering: application in a marketing database with massive missing data. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 66–72. IEEE, 2007.

53. S. Oba, M.-a. Sato, I. Takemasa, M. Monden, K.-i. Matsubara, and S. Ishii. A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

54. L. Olshen, C. J. Stone, et al. Classification and regression trees. *Wadsworth International Group*, 93(99):101, 1984.

55. M. Ouyang, W. J. Welsh, and P. Georgopoulos. Gaussian mixture clustering and imputation of microarray data. *Bioinformatics*, 20(6):917–923, 2004.

56. J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

57. C. Paroissin, L. Penalva, A. Pétrau, and G. Verdier. New control chart for monitoring and classification of environmental data. *Environmetrics*, 2016.

58. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

59. D. B. Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

60. S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

61. M. Saar-Tsechansky and F. Provost. Handling missing values when applying classification models. *The Journal of Machine Learning Research*, 8:1623–1657, 2007.

62. D. H. Schoellhamer. Singular spectrum analysis for time series with missing data. *Geophysical Research Letters*, 28(16):3187–3190, 2001.

63. Q. Song, M. Shepperd, X. Chen, and J. Liu. Can k-NN imputation improve the performance of c4. 5 with small software project data sets? a comparative evaluation. *Journal of Systems and software*, 81(12):2361–2370, 2008.

64. P.-N. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*, volume 1. Pearson Addison Wesley Boston, 2006.

65. O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

66. B. Twala. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23(5):373–405, 2009.

67. S. Van Buuren. *Flexible imputation of missing data*. CRC press, 2012.

68. G. Welch and G. Bishop. An introduction to the Kalman filter. department of computer science, university of north carolina, 2006.

69. Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 12(1):40–48, 2010.

70. H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, 2011.

71. Y. C. Yuan. Multiple imputation for missing data: Concepts and new development (version 9.0). *SAS Institute Inc, Rockville, MD*, 49, 2010.

72. H. Zhang. The optimality of naive Bayes. *A A*, 1(2):3, 2004.