Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series [E & A]

Mourad Khayati, Alberto Lerner, Zakhar Tymchenko, and Philippe Cudré-Mauroux
University of Fribourg
Switzerland
{firstname.lastname}@unifr.ch

ABSTRACT

Recording sensor data is seldom a perfect process. Failures in power, communication or storage can leave occasional blocks of data missing, affecting not only real-time monitoring but also compromising the quality of near- and on-line data analysis. To handle missing values, several recovery (imputation) algorithms have been proposed. Unfortunately, little is known about their relative performance, as existing comparisons are limited to either a small subset of relevant algorithms or to synthetic datasets, making it hard to draw general conclusions.

In this paper, we empirically compare twelve recovery algorithms, most of which were re-implemented on a uniform test environment. We run each algorithm over ten different datasets that collectively represent a broad range of applications. Our methodology allows us to fairly evaluate the relative strengths and weaknesses of each approach and to give recommendations for selecting the best technique on a use-case basis. Furthermore, we pinpoint opportunities not covered by any algorithm, suggesting that future research directions in this area still exist.

PVLDB Reference Format:
Mourad Khayati, Alberto Lerner, Zakhar Tymchencko, and Philippe Cudré-Mauroux. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series. *PVLDB*, 12(xxx): xxxx-yyyy, 2019.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

With the emergence of the Internet of Things (IoT), time series data became ubiquitous in a range of domains such as Astronomy [17, 55], Climate [24, 34], Energy [17], Environment [48], Finance [28, 52], Medicine [46], Neuroscience [59, 61], and Traffic [41, 65]. When sensors' time series are recorded, missing values very often occur. For instance, the Intel-Berkeley Research Lab dataset is missing about 50% of the expected measurements [6]; the University of California Irvine's repository of time series, 20% [11]. Missing values often occur consecutively, forming a block in a time series. These blocks can be rather large, as it can take arbitrarily long to fix a faulty sensor.

Usually, data management and analysis systems assume no such gaps exist in the data. Even if a system can work with incomplete data (e.g., NULLs in Relational Databases), leaving missing values untreated can cause incorrect or ill-defined results [8]. Recently, some systems started to incorporate missing-values imputation as a native feature [2, 8, 37, 49]. We expect other systems to follow – but the number of alternative algorithms available complicates this choice.

The main source of variation among recovery techniques comes from different ways to achieve accuracy. Some algorithms assume that the time series present temporal continuity, as the nature of the phenomena they measure is not discrete (e.g., quantity of rainfall). They recover missing blocks by looking at an entire set of series at once and applying on them dimensionality reduction-based techniques – matrix completion being the common one. Other algorithms consider that sensors which are at close proximity can present trend similarity (e.g., movement sensors in a same body). They apply pattern matching techniques to search for replaceable values. Unsurprisingly, different algorithms' accuracy will be sensitive to different peculiarities.

Naturally, the efficiency of these algorithms varies as well. Matrix completion and pattern matching techniques involve expensive computations, but many techniques are known to mitigate this. For instance, one can speed-up matrix completion algorithms by using a large dimensionality reduction step – and compensate for it by repeating the same step until a chosen error metric reaches a threshold. While this process brings efficiency, it creates the need to set the parameters for the algorithm properly, i.e., the dimensionality to use. If the dimensionality reduction is poorly chosen, all performance benefits are lost.

In this paper, we aim to provide a guide to the state-of-the-art algorithms for the recovery of missing values. To the best of our knowledge, this is the first comparative study to provide in-depth analysis of *accuracy, efficiency, and parameterization* across these algorithms under a representative body of datasets. The chosen algorithms, summarized in Table 1, cover the full gamut of techniques currently available to recover large missing blocks. As these algorithms were suggested across a wide period of time and range of domains, we re-implement a large number of them using a single programming language (C++) and a common, modern underlying library for data manipulation.[1]

[1]Source code and datasets available: `https://github.com/eXascaleInfolab/bench-vldb19.git`

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

Table 1: Recovery Techniques, described in Section 3.

		Recovery			Technique		Implementation		
		Initialization	Multiple TS	Type	Matrix D/F	Termination	Original	LoC	Speedup
Matrix comp.	CDRec [25, 24]	interpolation	✓	batch	CD	dynamic	JAVA	196	27
	GROUSE [3, 66]	N/A	✓	batch	PCA	static	Python	94	10
	ROSL [53]	interpolation	✓	batch	PCA	dynamic	C++	330	-
	SoftImp. [38]	zero	✓	batch	SVD	dynamic	Python	92	5
	SVDImp. [58]	zero	✓	batch	SVD	dynamic	Python	91	9
	TeNMF [39]	zero	✓	batch	NMF	dynamic	Python	78	2
	TRMF [64]	random	✓	batch	MF	static	Matlab/C++	-	-
	SPIRIT [44, 45]	N/A		online	PCA	static	Matlab	214	110
	SVT [7]	zero	✓	batch	SVD	dynamic	dynamic	158	21
Pattern	DynaMMo [29]	interpolation	✓	batch		dynamic	Matlab	208	3
	STMVL [62]	N/A	✓	batch		static	C#	768	2
	TKCM [60]	N/A		online		static	C	-	-

Results. By re-implementing the algorithms in a uniform way and running a comprehensive benchmark, we not only reproduced results from original papers but also uncovered new findings. Some of the most salient are:

There is no single-best accurate algorithm. Five distinct algorithms stand out. SoftImpute and TRMF are the most accurate on datasets with repeating trends; CDRec, on time series exhibiting high variations in correlation; STMVL, in highly correlated time series; and, lastly, DynaMMo is particularly adapted to datasets with irregular fluctuations.

Larger missing blocks may sometimes yield higher recovery accuracy. This is due to the iterative nature of some algorithms. Large missing blocks require a larger number of iterations which, in turn, yield better recovered values.

Blackouts pose an accuracy challenge. Blackouts refer to episodes where all sensors go quiet at the same time causing widespread and aligned missing blocks. All the evaluated techniques' accuracy suffer when blackouts are present. Only CDRec and DynaMMo show reasonable recovery, as they estimate missing values with an approximation of linear interpolation.

There is a wide runtime difference across solutions. The most efficient algorithms, SVDImpute and CDRec, are three orders of magnitude faster than the slowest, DynaMMo.

Small dimensionality yields best results. Techniques relying on truncation (i.e., dimensionality reduction) achieve the best performance with a relatively small truncation value.

Contributions. We summarize the contributions as follows:

- We curate a comprehensive set of large-block recovery algorithms for time series data. Some algorithms were chosen after we adopted improvements to their original definitions;
- We re-implement nine recovery algorithms in C++ that were originally implemented in different programming languages and provide the resulting source code, along with the datasets and the scripts necessary to reproduce our results. This creates, for the first time, a unified comparison test-bed for the field;
- We evaluate and discuss the accuracy, efficiency, and the proper parameterization of these recovery techniques under a large variety of scenarios;
- We provide a comprehensive guide to navigating the choice of available algorithms, identifying the scenarios in which they perform well. We also discuss scenarios where no acceptable results exist, which points to potential future research areas.

Related Work. The closest work to compare large missing blocks recovery techniques for time series data is Balzano, et al. [3]. The authors evaluate the efficiency and accuracy of subspace-tracking techniques, but focus exclusively on algorithms based on Principal Component Analysis (PCA). Xiaoping Zhu [68] evaluates four different techniques to handle missing data. Only basic statistical methods to recover missing values are compared, i.e., Mean Imputation, kNN Imputation, Deletion and Multiple Imputation. Moritz, et al. [43] compare different recovery techniques for time series in R language. The authors discuss statistical methods solely, but adding a recovery technique that uses Kalman Filters [18]. In contrast to these surveys, ours includes a more extensive set of recovery techniques and real-world time series covering a wider range of characteristics.

In [35, 36, 40], missing values recovery techniques are also compared. However, the focus of those studies is in classification/clustering tasks rather than in the quality of the recovery.

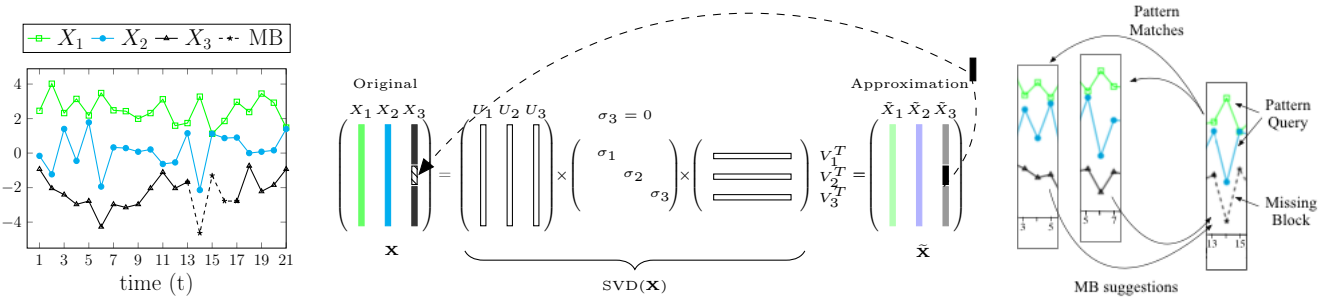
There are a number of relevant works designed specifically for very small missing blocks (i.e., single missing values or only a handful of consecutive missing values) such as kNNImpute [67] or Mean Impute [15]. Similarly, some techniques aim at recovering only a (test) subset of the dataset, e.g., [1, 63]. These algorithms form a body of work of their own, different than our focus here.

Similarly to time series data, other types of data provide unique opportunities for recovery techniques. For instance, there are works that focus on graph recovery techniques [9, 12, 20, 30, 42, 51], image reconstruction [16, 21, 57], and recovery of categorical data [4, 5], to mention a few. None of these algorithms made it into our study, as they are highly dependent on the specific type of data for which they are originally designed – and thus not applicable for time series.

Outline. The rest of this paper is organized as follows. Section 2 provides background information on the families of algorithms we study here. Section 3 surveys all algorithms in our test-bed, analyzes their properties, and discusses our implementation choices. Section 4 reports on the experimental results. Section 5 discusses our findings and makes suggestions for future work. Lastly, Section 6 concludes this paper.

2. BACKGROUND

Missing-values recovery algorithms can be divided into *matrix-based* and *pattern-based* ones, according to the underlying method they employ.



(a) Input time series. (b) Matrix completion recovery. (c) Pattern-matching recovery.

Figure 1: Recovery Illustration of the Missing Block (MB) in X_3.

Matrix-based recovery techniques use data analysis methods to infer the missing block. To illustrate this process, assume a set of m time series, each having n points, and that one of these time series has a missing block of values. We show such scenario in Figure 1a, without loss of generality, for $m = 3$ and $n = 20$. These time series can be represented as an $n \times m$ matrix where each time series occupies a single column, shown as the left-most matrix in Figure 1b.

A matrix-based algorithm finds an equivalent representation of the data in such a way that simplifies dimensionality reduction, or *truncation*. We depict such process using the *Singular Value Decomposition* (SVD) [54]. SVD decomposes the input matrix $\mathbf{X}$ into three matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$, such that $\mathbf{X} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^T$. The $\mathbf{\Sigma}$ matrix exposes not only the rank of the original matrix, i.e., number of linearly independent dimensions, but also the relative importance of each of the dimensions, as $\sigma_1 > \sigma_2 > \sigma_3$. Truncating the original matrix can be done by nullifying some dimensions (cells in the diagonal) in $\mathbf{\Sigma}$, σ_3 in our example. An approximation of the original dataset is recalculated by multiplying back the component matrices – filling the missing block in the process. The algorithm would further plug those values to the original matrix and would iterate over this process, trying to minimize a given error metric.

SVD is one among many applicable matrix decomposition techniques. We will introduce algorithms that rely on *Principal Components Analysis* (PCA) [19], *Centroid Decomposition* (CD) [10], *Matrix Factorization* (MF) [27], and *Non-Negative Matrix Factorization* (NMF) [26]. We will also see that the error metric and the stopping conditions vary among algorithms. The error metric can be based on the distance between the input and the approximated series and can be computed in different ways, e.g., Frobenius norm [25], nuclear norm [7], rank minimization [53], root mean square error minimization [22, 23], etc. The decision to terminate the iterations could be dynamic, via a threshold error, or static, through a certain number of steps.

In turn, pattern-based recovery techniques aim at identifying patterns in the historical data that often repeat, and apply the knowledge derived from them to recover missing values. We depict a pattern-based imputation in Figure 1c. These techniques assume that some degree of correlation exists between the *base* time series that contains the missing block and the *reference* time series used to detect the patterns. For instance, in Figure 1c, X_3 is a base series whereas X_1 and X_2 are reference ones. When a block of values is missing, one can look back into the reference series for the pattern that occurred on the missing site. Each of the matches will represent a candidate block for recovery.

Similarly to matrix-based algorithms, pattern-based ones also require calibration (i.e., their similar mechanism to truncation). The size of the pattern to look for has a big impact on the accuracy/efficiency trade-off. If the pattern is too small, the technique loses accuracy, especially if the time series are not cyclic. If too big, the computational time involved in pattern manipulation primitives (e.g., comparison) becomes too costly. Moreover, pattern-based algorithms are also iterative. They go over the candidate recovery blocks until they reach a given error metric threshold.

Notations. In the following, bold upper-case letters refer to matrices, regular font upper-case letters refer to vectors (rows and columns of matrices) and lower-case letters refer to elements of vectors/matrices. The symbol $\|\|_F$ refers to the Frobenius norm of a matrix, while $\|\|$ refers to the l-2 norm of a vector. Assume $\mathbf{X}$ is an $n \times m$ matrix where each column is $X = [x_1, \ldots, x_n]$, then $\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij})^2}$ and $\|X\| = \sqrt{\sum_{i=1}^{n} (x_i)^2}$.

A *time series* $X = \{(t_1, v_1), \ldots, (t_n, v_n)\}$ is an ordered set of n temporal values v_i that are ordered according to their timestamps t_i. We write $\mathbf{X} = [X_1 | \ldots | X_m]$ (or $\mathbf{X}_{n \times m}$) to denote an $n \times m$ matrix having m time series X_j as columns and n values for each time series as rows. Time series can be *univariate* (also called 2-dimensional) or *multivariate* (multi-dimensional). In univariate series, each temporal value represents a scalar that refers to one specific phenomenon, e.g., temperature. In multivariate series, each value is a vector that represents multiple phenomena, e.g., temperature, precipitation, humidity, etc.

To measure the recovery accuracy, we use the two most commonly used measures in this field: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) between the original block and the recovered one, i.e.,

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{t \in T} (x_t - \tilde{x}_t)^2}, \quad MAE = \frac{\sum_{t \in T} |x_t - \tilde{x}_t|}{|T|}$$

where T is the set of missing values, x_t is the original value and $\tilde{x}_t$ is the recovered value. Unlike MAE, RMSE is computed using the square root of the average squared errors and thus, penalizes more large errors even if they are few.

3. SELECTED ALGORITHMS

Collectively, the algorithms presented in this study cover a wide area of the design space. The "Recovery" and "Technique" portions in Table 1 summarize the main features of these algorithms (type of initialization, support of multiple recoveries, type of the recovery, the underlying method they employ and the type of termination). Some of the algorithms can be quite sophisticated. Due to space constraints, we have omitted details that are not immediately relevant to our analysis.

3.1 Matrix Completion Algorithms

SVDImpute [58] is a recovery algorithm similar to the one depicted in Figure 1b. This algorithm was originally introduced to recover missing genes in DNA micro-arrays where missing blocks are usually small. SVDImpute first initializes the missing values as zeroes. Then, it selects the k most significant columns of $\mathbf{V}$ (obtained through SVD) and uses a linear combination of these columns to estimate the missing values. We adopted a common improvement that allows SVDImpute to scale better: we calculate the SVD in a randomized [14] and a faster [13] way. This change required only a marginal sacrifice in precision, as compared to the full SVD.

SoftImpute [38] is an extension of the SVDImpute technique with an Expectation Maximization (EM) algorithm. This extension results in improvements to the recovery precision while only slightly sacrificing the efficiency, when compared to SVDImpute. The EM method alternates between recovering the missing values and updating the SVD using the observed values. This technique uses a soft-thresholded version of SVD and, unlike SVDImpute, it uses the product of all three matrices produced by the decomposition.

SVT [7] is another SVD-based algorithm designed with one notable feature: it automatically finds the degree at which rank truncation should occur, avoiding the parameterization that any other SVD-based algorithm requires. It does so by applying a thresholding technique to reduce the number of the singular values obtained from the decomposition. The preserved singular values are rescaled using only the observed values, and the recovery is obtained by iteratively multiplying back the three matrices of the decomposition.

CDRec [24, 25] is a technique aimed at recovering time series with mixed correlation. It is based on a matrix decomposition technique called *Centroid Decomposition* (CD) [10]. Similarly to the SVD technique, CD decomposes an $n \times m$ input matrix $\mathbf{X}$ into an $n \times m$ loading matrix $\mathbf{L}$ and an $m \times m$ relevance matrix $\mathbf{R}$, such that $\mathbf{X} = \mathbf{L} \cdot \mathbf{R}^T$. CD performs the decomposition by finding a maximizing sign vector, Z, that contains 1s or -1s and that maximizes the centroid value, i.e., $\|\mathbf{X}^T \cdot Z\|$. CDRec performs recovery by first using interpolation/extrapolation to initialize the missing values. It then computes and truncates CD by keeping only the first k columns of $\mathbf{L}$ and $\mathbf{R}$ to produce $\mathbf{L}_k$ and $\mathbf{R}_k$ respectively. Finally, it replaces the interpolated values by the corresponding elements in $\widetilde{\mathbf{X}} = \mathbf{L}_k \cdot \mathbf{R}^T$. This process is performed iteratively until the normalized Frobenius norm between the matrix before and the one after the update reaches a small threshold.

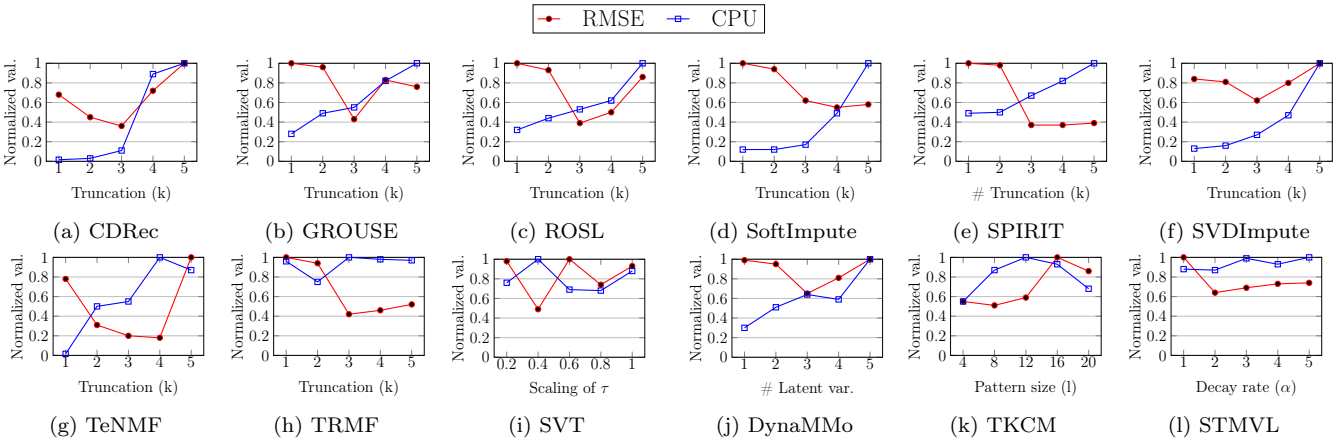
GROUSE [3, 66] is a technique that is agnostic to the initialization of missing values. It gets this property from a decomposition technique we mentioned before, PCA [19]. PCA takes an $n \times m$ input matrix $\mathbf{X}$ and finds n eigenvectors (vectors of $\mathbf{U}$ from SVD) each of size m that correspond to the loadings of the principal components. In other words, $\mathbf{U}$ is a new subspace that approximates the dimensions of the initial data. The approximation is performed by applying an incremental gradient descent procedure to minimize an objective function, and subsequently derive the missing values. We experimented with two different objective functions: one with a distance-based step-size and one with a greedy step-size. We kept the distance-based implementation as it yielded more stable results. The gradient-descent approach best accounts for the variance in a data set, as compared to other decomposition techniques.

SPIRIT [44, 45] is also a PCA-based algorithm but one designed to perform streaming recovery (i.e., the missing block is at the tip of the series). As streaming algorithms require, SPIRIT presents low runtime because it operates only on the observed values. This technique uses PCA to reduce a set of m co-evolving and correlated streams to a small number of k hidden variables. These variables summarize the most important features of the original data. For each variable, SPIRIT fits one auto-regressive (AR) model on historical values, which is incrementally updated as new data arrives. Then, these models are used to predict the value of each variable, from which an estimate of the missing value is derived. The recovered value, along with the non-missing values, are then used to update the AR coefficients and subsequently recover the missing values.

ROSL [53] is also a PCA-based algorithm, but one that specializes in denoising corrupted data. It assumes the input matrix contains corrupted data and uses its orthonormal subspace to find a better estimation. This subspace uses a rank measure to identify the rank representation of the data. We slightly modified the original algorithm to consider only (the initialized) missing values as corrupted values and thus, to better estimate them.

TRMF [64] is an algorithm that learns from different types of data dependencies, making it suitable to time series exhibiting varied characteristics. It is based on temporal *Matrix Factorization* (MF) [27] which takes an $n \times m$ input matrix $\mathbf{X}$ and seeks to approximate it using two factor matrices, $\mathbf{W}$ and $\mathbf{H}$, respectively of size $n \times r$ and $r \times m$ (with $r \leq min(n, m)$) such that $\mathbf{X} \approx \mathbf{WH}$. The input time series are factorized into a so called latent temporal embeddings and an auto-regressive (AR) model is applied to derive the temporal structure of the embeddings.

TeNMF [39] is a particularly suited algorithm for recovering correlated time series. It does so by combining temporal aggregation techniques with yet another matrix decomposition technique, the *Non Negative Matrix Factorization* (NMF) [26]. NMF is similar to the MF technique described above, but it constrains $\mathbf{W}$ and $\mathbf{H}$ to contain only non-negative elements. More specifically, TeNMF first applies NMF to obtain temporal aggregates and uses them to define a loss function. The latter is then modified by adding a so called penalization parameter to capture the correlation across time series. The original implementation uses multiplicative divergence-based update to compute NMF [56], which makes the recovery unstable (varies from one run to



(a) CDRec (b) GROUSE (c) ROSL (d) SoftImpute (e) SPIRIT (f) SVDImpute

(g) TeNMF (h) TRMF (i) SVT (j) DynaMMo (k) TKCM (l) STMVL

Figure 2: Parameterization of the techniques.

another). We improved its stability by using instead alternating least squares update (ALS) to compute NMF [32].

3.2 Pattern-based Recovery Algorithms

TKCM [60] is similar to the algorithm we depicted in Figure 1c in that it leverages the dependency across time series to identify repeating patterns (seasonality) in the time series history. In TKCM's case, the pattern search is performed in a dynamic warping fashion allowing time lags and thus, recovering time-shifted time series. Similarly to SPIRIT, TKCM is a streaming technique that operates on batches of the input data, rendering the technique efficient.

DynaMMo [29] is an algorithm that can leverage the similarities across few time series only. The underlying assumption used by this technique is that some time series present co-evolving patterns. Internally, the algorithm relies on Kalman Filters and iterates using an Expectation Maximization (EM) process. The Kalman Filter uses the data that contains missing blocks together with a reference time series to estimate the current state of the missing blocks. This estimation is performed as a multi-step process that uses two different estimators. For every step in the process, the EM method predicts the value of the current state and then the two estimators are used to refine the predicted values of the current state and to maximize their likelihood.

STMVL [62] is yet another algorithm that can recover missing values in highly correlated time series. STMVL focuses on two types of correlation between time series: the spatial correlation (geographical distance between sensors) and the temporal correlation (closeness of the values in time). Internally, STMVL combines a data smoothing technique with collaborative filtering to derive models out of the historical data. These models are then used to estimate the missing values based on the closeness of the sensors represented using a decay rate (α).

3.3 Algorithms Implementation

In order to provide a fair comparison, we rewrote all the algorithms described in the previous section in C++ , except for TRMF (inextricable from Matlab) and TKCM (efficient original implementation). As expected, the implementations sizes varied according to the sophistication of the algorithms, as we show in the "Implementation" portion of Table 1. The latter describes the original programming language, the number of lines of code (LoC) and the speedup.

We use the same advanced linear algebra operations across the techniques, thanks to a modern library called Armadillo [50]. Thus, we eliminate any sources of disparities that would otherwise exist if each algorithm re-implemented common primitives.

All of the algorithms' runtimes showed improvements when compared to their original implementations – in one case becoming 110x faster. Table 1 shows each speedup. The re-implementation brought insights that allowed us to re-engineer the original implementations, as opposed to simply translate them from one language to the other. We selected some notable examples of these re-engineering techniques and discuss them in more detail in Appendix A. We also repeat core experiments from previous work. We describe the reproducibility of the results in Appendix B.

4. EXPERIMENTAL EVALUATION

In this section, we submit the algorithms to different benchmarks, each designed to exercise a specific aspect.

We conduct our experiments on a 32GB RAM machine with an Intel i7-4770 that consists of 8 cores with an 8MB L3 cache, running at 3.4 GHz. The code was compiled with g++ 7.3.0. We use eight real-world datasets from different domains and exhibiting different characteristics, and two synthetic datasets. We will describe each of the datasets shortly, as they appear in our experiments.

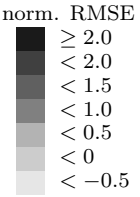
4.1 Parameterization

We begin our evaluation by fine tuning each algorithm. Parameterization is the process of calibrating a technique to work with the best accuracy/efficiency trade-off. We will show that this step is not straightforward, as accuracy and efficiency do not interact as one would expect.

The most critical parameter for all the matrix-completion techniques is the truncation factor (or rank). It affects both the accuracy and the scalability of an algorithm. SVT is a special case, as the threshold parameter (τ) is more critical than the truncation factor. For the pattern-based techniques, i.e., DynaMMo, STMVL and TCKM, we vary respectively the number of latent dimensions, the decay rate

Table 2: Description of time series and accuracy of each technique.

Name	Length	# TS	CDRec	DynaMMo	GROUSE	TeNMF	ROSL	SoftImpute	SPIRIT	STMVL	SVDImpute	SVT	TKCM	TRMF
Air	1k	10												
Chlorine	1k	50												
Climate	5k	10												
Gas	1k	100												
Electricity	5k	20												
Meteo.	10k	10												
Temp.	5k	50												
BAFU	50k	10												



(α) and the pattern size (l). Figure 2 reports the recovery RMSE and the execution time each normalized by the largest value of each technique (the lower the better).

We use the Chlorine dataset for this process as it is the most commonly used by the techniques we evaluate (e.g., in DynaMMo, GROUSE, SPIRIT, and TKCM). This dataset simulates a drinking water distribution system and describes the concentration of chlorine at multiple junctions.

For matrix completion techniques, we found out that the runtime of all techniques increases along with the truncation value (k), except for TRMF. This result is expected, since the higher k is, the higher the number of dimensions used to produce the recovery (yielding more time- and space-intensive computations). For TRMF, since its runtime is dominated by computation of (expensive) auto-regressive coefficients, varying k has little impact on the efficiency.

Surprisingly, increasing k did not always improve accuracy. We observe that for all the matrix-based techniques, the optimal truncation value showed to be $k \in \{2, 3\}$ as it presents the best compromise between runtime and accuracy. Note that some of the matrix-completion techniques use two additional parameters, i.e., the maximum number of iterations and the convergence threshold. We ran experiments varying these parameters but found that certain fixed values provided the best performance. Therefore, we kept them constant for the rest of the experiments. We describe these parameters' calibration in detail in Appendix C.

For the pattern-based techniques, different parameters impact the performance of the recovery. For DynaMMo, the critical parameter is the number of latent variables. The optimal number of variables was found to be 3. TKCM is dependent on the length of the pattern. We vary it from 4 and 70 and set the value to 4 for the subsequent experiments as it produces the best result. Lastly, we found that STMVL depends on many parameters. The decay rate is the parameter that impacts the performance of the technique the most. The optimal value of the decay rate is 2.

4.2 Accuracy

After having properly calibrated the algorithms, we evaluate the accuracy of all techniques under varying sizes of missing blocks (from 20% to 80%). We set the size as the percentage missing in a single time series. We then compute the average RMSE which we normalize using the z-score technique (the lower the better). Table 2 reports on these results. As the table shows, we use several other datasets we describe below:

Air brings air quality measures collected from 36 monitoring stations in China from 2014 to 2015 (appeared in STMVL).

Gas shows gas concentration collected between 2007 and 2011 from a gas delivery platform situated at the ChemoSignals Laboratory at University of California in San Diego (appeared in [47]).

Climate presents monthly aggregated data for climate collected from 18 climate agents over 125 locations in North America between 1990 and 2002 (appeared in [34, 33]).

Electricity has data on household energy consumption collected every minute between 2006 and 2010 in France (obtained from the UCI repository and used by TRMF and TeNMF).

Temperature contains temperature time series collected from climate stations in China from 1960 to 2012[2] (appeared in STMVL and CDRec).

MeteoSwiss is a weather time series provided by the Swiss Federal Office of Meteorology and Climatology[3] collected from different Swiss cities from 1980 to 2018 (appeared in CDRec).

BAFU consists of water discharge time series provided by the BundesAmt Für Umwelt (BAFU)[4], the Swiss Federal Office for the Environment, collected from different Swiss rivers from 1974 to 2015 (appeared in [2]).

Chlorine (introduced earlier) simulates a drinking water distribution system and describes the concentration of chlorine in 166 junctions over a time frame of 15 days with a sample rate of 5 minutes.

As Table 2 shows, TeNMF and TKCM have a low recovery accuracy on most of the datasets, while GROUSE performs similarly on half of them. We also observe that, on large datasets (with either long or many time series), SPIRIT, SVT and ROSL fail to achieve a good recovery. Hence, we keep our focus on the most accurate algorithms: CDRec, DynaMMo, SoftImpute, SVDImpute, STMVL, and TRMF.

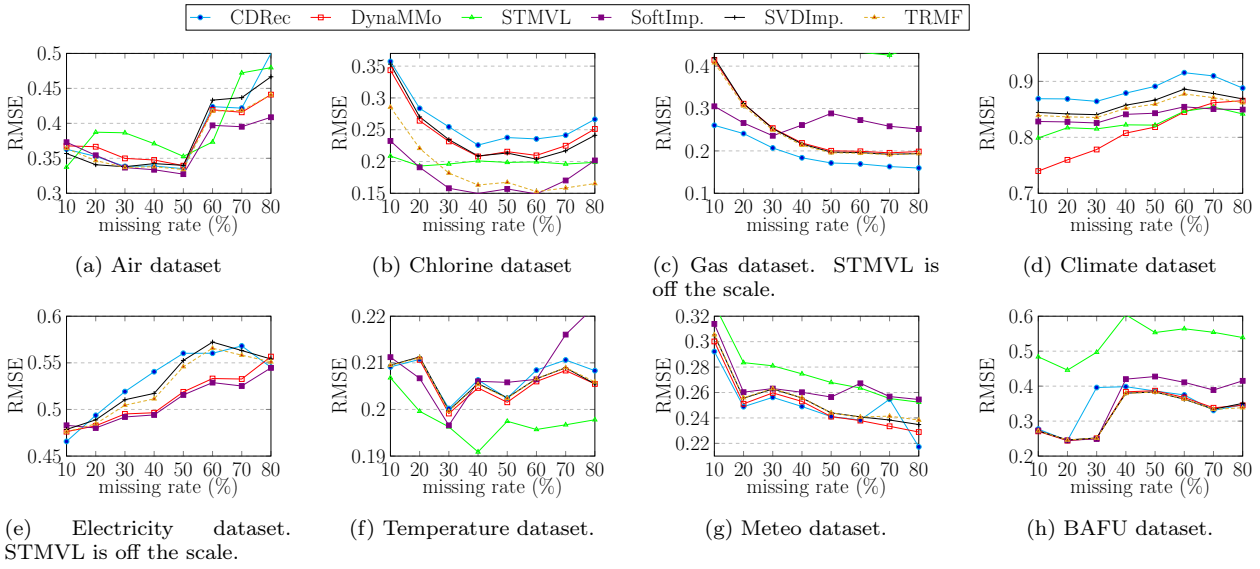
4.2.1 Impact of Missing Block Size

In this set of experiments, we evaluate the recovery accuracy when increasing the percentage of missing values in one time series (This is a breakdown of the values we see in Table 2). We show the accuracy results using RMSE only, as we obtained similar results with MAE. When varying the percentage of missing values, the length and the number of time series are kept to their maximum per dataset. We set the missing block to appear arbitrarily in the middle of the chosen time series. Figure 3 shows the results.

[2]http://www.cma.gov.cn/en2014/

[3]meteoswiss.admin.ch

[4]https://www.hydrodaten.admin.ch/en



(a) Air dataset (b) Chlorine dataset (c) Gas dataset. STMVL is off the scale. (d) Climate dataset

(e) Electricity dataset. STMVL is off the scale. (f) Temperature dataset. (g) Meteo dataset. (h) BAFU dataset.

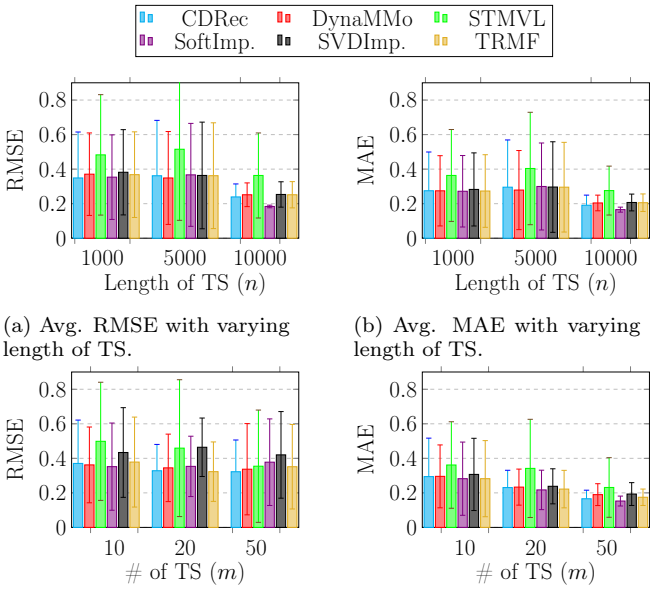
Figure 3: Accuracy comparison with increasing missing block size.

We can separate the datasets/results into two types. There are those where a same group of algorithms perform equivalently well. In the Air, Meteo, and BAFU datasets (cf. Figures 3a, 3g, 3h, resp.), all of CDRec, DynaMMo, SoftImpute, and TRMF present somewhat indistinguishable low RMSE. These datasets contain similar features (weather-related time series), which pose no significant challenge.

In contrast, the other datasets have each a prominent feature that exposes the differences among the algorithms. The Chlorine dataset (cf. Figure 3b) has repeating trends, to which TRMF and SoftImpute respond particularly well. The AR and EM models used respectively by the two techniques serve the purpose of capturing the regularity inside the data and thus, accurately detect the trend repetition. The Gas dataset (cf. Figure 3c) presents mixed correlation (positive/negative and high/low), better handled by CDRec's use of a weight vector capable of capturing such varied correlation. The Climate (cf. Figure 3d) and the Electricity datasets (cf. Figure 3e) both present irregularities – fluctuations and shifted time series, respectively – which DynaMMo and SoftImpute handle well because of their attention to co-evolution of time series. The Temperature dataset (cf. Figure 3f) stands out by its very high correlation. This is why STMVL, which captures such models by design, performs so well. Note though that the advantage that STMVL has is localized to this scenario.

We conclude from this experiment that the absolute best accuracy can only be currently achieved by specialization (the careful pairing of data features with the algorithm design). We also observe that specialization can very well have the opposite effect, if chosen wrongly. For instance, in the case of STMVL, the accuracy is particularly poor in large datasets that contain either a high number of time series, such as Gas ($m = 100$), or long time series, such as BAFU ($n = 50k$) –Figure 3c and 3h, respectively. STMVL is not the only such example; SoftImpute and CDRec, which did well in other scenarios, also become among the least accurate techniques in these datasets.

We also conclude from this experiment that, counter-intuitively, the error does not always increase along with the size of the missing block. In three datasets, Chlorine (cf. Figure 3b), Gas (cf. Figure 3c) and Meteo (cf. Figure 3g) the trend is the opposite. As we stated earlier, using larger blocks of missing values avoids early termination – recall, these algorithms are iterative – and thus, increasing iterations further improves the refinement of the initial values.

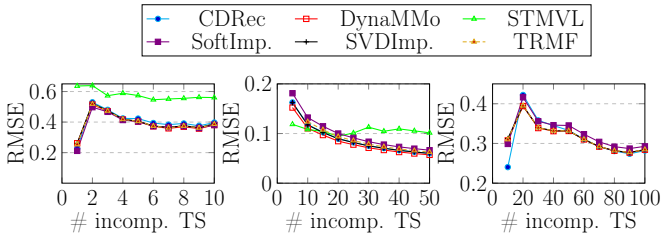


(a) Avg. RMSE with varying length of TS. (b) Avg. MAE with varying length of TS.

(c) Avg. RMSE with varying # of TS. (d) Avg. MAE with varying # of TS.

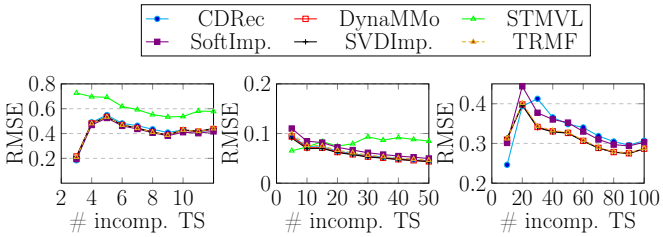
Figure 4: Accuracy comparison with increasing time series length and number.

4.2.2 *Impact of Sequence Length and Number*



(a) BAFU dataset (m=10). (b) Chlorine dataset (m=50). (c) Gas dataset (m=100). STMVL is off the scale.

Figure 5: Accuracy comparison with increasing number of incomplete TS (overlapping).



(a) BAFU dataset (m=10). (b) Chlorine dataset (m=50). (c) Gas dataset (m=100). STMVL is off the scale.

Figure 6: Accuracy comparison with increasing number of incomplete TS (disjoint).

Figure 4 depicts the recovery accuracy on different datasets when increasing the sequence length and number. We use the RMSE and MAE values as geometric means across different datasets, with the confidence bands representing the standard deviation. Both metrics appear here as they highlight different aspects in some cases. When the sequence length varies, the number of time series is set to 10, whereas the sequence length is set to 1k when the number of time series varies. We set the size of the missing block to 10% of one time series.

This experiment shows that, in general, the algorithms take advantage of having longer time series to produce better accuracy (cf. Figures 4a and 4b). The improvement is more noticeable when we vary the length from $5k$ to $10k$. This is an expected result, because using more data should help to better capture the main features of the time series and thus, yields a better estimate the missing blocks. We also observe that STMVL presents higher errors compared to the rest of the algorithms as it performs poorly in several datasets (Gas, Electricity and BAFU).

When it comes to varying the number of time series, the RMSE accuracy of the algorithms remained largely unaffected (cf. Figures 4c). This was an unexpected result, as using more time series from the same dataset should help the techniques to better compute the dimensionality reduction (for matrix completion), the spatio-temporal model (for STMVL) and the Kalman Filer (for DynaMMo). The MAE results (cf. Figure 4d), however, show the expected trend where the accuracy increases with the number of the used time series, i.e, the curve goes down. The reason behind this discrepancy is that all the techniques perform on average a better recovery, but some outliers are introduced by the addition of the new time series. Since MAE reflects the average recovery by giving less weight to the outliers, then it is more indicative of the expected behavior.

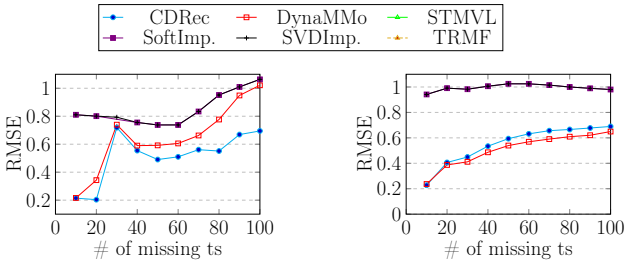
4.2.3 *Impact of Number of Affected Series*

We compare the recovery RMSE of the best methods when varying the number of incomplete time series. We use datasets with different numbers of time series and we consider two scenarios: overlapping missing blocks (as multiple sensors can break during the same time interval) and disjoint missing blocks.

Figure 5 shows the recovery accuracy for multiple incomplete time series where each of them has 10% of missing values. The missing blocks are partially overlapping, i.e., the first half of the block is overlapping with the previous time series while the second half is overlapping with the next time series. We keep the length and number of time series to their maximum per dataset and we increase the number of affected time series.

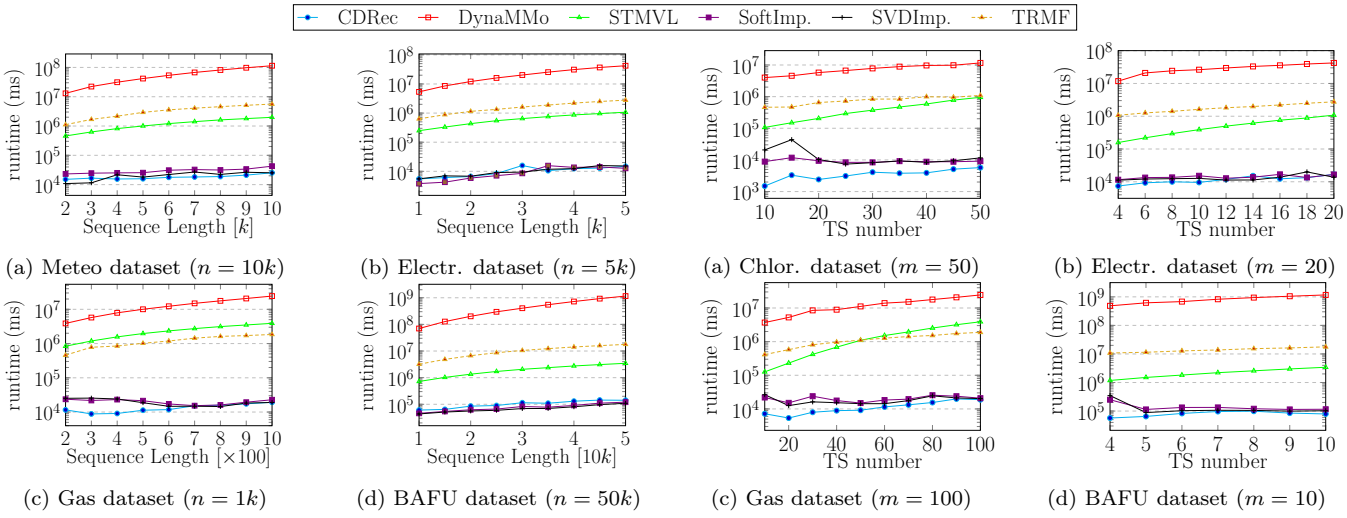
We observe two trends in the results. For the datasets with a small number of time series (cf. Figure 5a) and high number of time series (cf. Figure 5c), the error generated by all techniques, except STMVL, first increases with the number of affected series, then it decreases. The increase in the RMSE occurs as expected, since adding more incomplete time series increases the number of missing values and subsequently the RMSE. However, the decrease is unexpected and is explained by the fact that having more incomplete time series avoids early termination, thus improving the recovery. For the datasets with medium number of time series(cf. Figure 5b), we observe a decrease in the RMSE caused by the same reason previously explained. Interestingly, there is not much differentiation among algorithms for those expect for STMVL. The accuracy of the latter is barely affected, as the spatio-temporal produces similar recovery for all incomplete time series.

Figure 6 shows the recovery accuracy when the incomplete time series have disjoint missing blocks. Similarly to the overlapping case, the important factor differentiating the algorithms in this experiment is the number of time series per dataset. For the dataset with a large number of time series (cf. Figure 6c) we observe that the recovery TRMF, SVDImpute and DynaMMo outperforms the rest of the techniques.



(a) Air dataset. (b) BAFU dataset.

Figure 7: Accuracy comparison in case of blackout by varying the number of timestamps (ts) lost. STMVL and TRMF are off the scale. SVDImpute and SoftImpute achieve the same recovery accuracy.



(a) Meteo dataset ($n = 10k$) (b) Electr. dataset ($n = 5k$)

(c) Gas dataset ($n = 1k$) (d) BAFU dataset ($n = 50k$)

Figure 8: Efficiency with increasing sequence length.

(a) Chlor. dataset ($m = 50$) (b) Electr. dataset ($m = 20$)

(c) Gas dataset ($m = 100$) (d) BAFU dataset ($m = 10$)

Figure 9: Efficiency with increasing sequence number.

We also evaluate the techniques in the case of a blackout, i.e, an event in which all time series lose data at the same time period. In the experiment in Figure 7, we compare the recovery RMSE of the best techniques when varying the size of the missing block. We choose the Air dataset which has the highest number of time series and the BAFU dataset which has the longest time series. We observe that all techniques incur a very high RMSE compared to the size of the missing portion (less than %10 per time series). On the dataset with long time series (cf. Figure 7b), CDRec and DynaMMo achieve the lowest error as both techniques return an approximation of the initialized interpolated values. On the Air dataset (cf. Figure 7a), using more time series helps CDRec to iterate more, thus achieving a better recovery compared to DynaMMo.

4.3 Efficiency

We now evaluate the efficiency of the recovery process by measuring the elapsed runtime while varying the time series length and number, and the percentage of missing values per time series. Elapsed times are shown using a log scale as the runtime difference between the algorithms is extremely high (up to three orders of magnitude).

4.3.1 *Impact of Sequence Length and Number*

In the experiments in Figure 8 we increase the sequence length while keeping the number to its maximum per dataset. We set the size of the missing block to 10% of the maximum length of one time series. The results show that the efficiency is clearly an aspect that divides the accurate algorithms chosen before in two groups, fast and slow. The algorithms from the first class, i.e., CDRec, SoftImpute and SVDImpute are matrix decomposition-based. They rely on truncation to operate on few dimensions only, thus achieving a near-linear time complexity.

The algorithms from the second group, on the other hand, rely on building expensive models to capture the specific features of the data. Take the example of DynaMMo which is able to capture the co-evolution in the data and uses it to estimate missing values. This technique relies mainly on building Kalman Filters which are very expensive to compute. In the case of TRMF, we observe a high runtime even though it is a matrix decomposition-based technique. This is explained by the fact that this technique relies on the computation of AR coefficients from all time series, which requires quadratic time complexity. The algorithms from the second class are more than two orders of magnitude slower than the ones from the first class, and this difference grows with the sequence length.

In the experiments in Figure 9, we increase the number of time series while keeping the length to their maximum per dataset. We observe similar results as in the experiments in Figure 8. We also observe that CDRec is faster than the rest of the algorithms from the first class on datasets with a smaller number of time series. However, the runtime difference becomes indistinguishable as the number of time series per dataset increases.

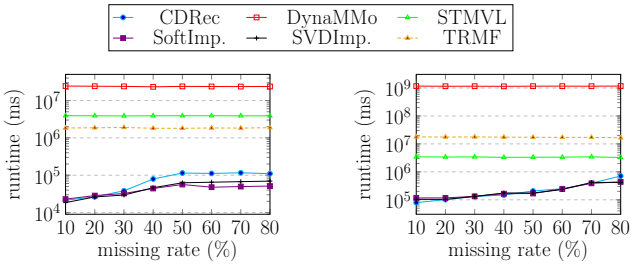
4.3.2 *Impact of Missing Block Size*

In the experiments in Figure 10, we compare the efficiency by increasing the size of the missing block while keeping the dataset length and size to their respective maximum per dataset. We choose the Gas and the BAFU datasets for the same reason as before.

The results show the same trends as the ones obtained when increasing the time series length and number: they confirm that the efficiency is still a differentiator among the algorithms. We can observe that the runtime of the fastest algorithms increases with the percentage of missing values, while it remains almost constant for the slowest algorithms. This is because the computation of the models used by the pattern-bases techniques depends only on the size of the input matrix. Thus, the efficiency of these techniques is independent from the size of the missing block.

4.3.3 *Synthetic Data*

In this section, we evaluate the efficiency of the fastest techniques on large datasets where thousands of sensors could be used to record millions of observations. We evaluate the algorithms able to perform the recovery in less than 1 minute. We use synthetic time series, as the real-world time



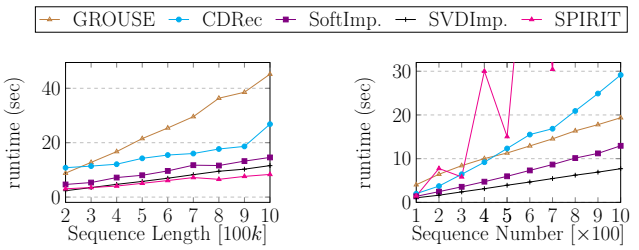
(a) Gas dataset (b) BAFU dataset

Figure 10: Efficiency comparison with increasing block size.

series we have chosen are limited in length and in number of time series.

We generate synthetic time series through a random walk. The steps are given by triangular distribution to obtain a flexible bias towards zero. Unlike normal distributions, our generation method guarantees that the values have upper and lower limits. We generate two datasets: the first dataset has $n = 1M$ and $m = 100$ while the second has $n = 10k$ and $m = 1k$. We set the size of the missing block to 10% of the maximum length of one time series.

The results show that SPIRIT outperforms the rest of the algorithms when varying the sequence length (cf. Figure 11a). The reason for this is that SPIRIT learns the AR coefficients and applies them only to a small window of the time series, rendering the technique very efficient. However, the performance of SPIRIT deteriorates as the number of time series grows (cf. Figure 11b). This is explained by the fact that this technique needs to orthogonalize the principal components – recall, SPIRIT is PCA-based – using Gram Schmidt process. This process has a quadratic complexity with the number of time series.



(a) Runtime with sequence length variation.

(b) Runtime with sequence number variation.

Figure 11: Efficiency on Synthetic Datasets.

5. RECOMMENDATIONS

In the previous section, we have shown hints that point to specialization – knowing the properties of the data series and finding a match in terms of recovery algorithm – as the key to accuracy and efficiency. In this section, we propose a number of dimensions that are useful to quantify specialization. We note that we will continue to consider only the algorithms that presented acceptable accuracy.

However, we have just seen that previously discarded algorithms can perform well in certain cases. SPIRIT shows very good efficiency compared to the rest of the techniques if the scenario in question had less than a hundred very long time series. TKCM is another example. It is an online algorithm and thus assumes that the series end at the missing block. (Incidentally, so does SPIRIT.) A missing block at the middle of the series has the effect of shortening that series for those algorithms. This explains their poor performance in Table 2 – and strengthens the argument for specialization.

Back to our high-accuracy algorithms, the main discriminant feature among them is efficiency. It clearly divides the algorithms in two classes, as we show in Figure 12. The algorithms also respond differently to certain features of the data. For instance, certain algorithms perform well if the series present irregularities, such as those in household power consumption. Other algorithms react well to high degrees of correlation, as those in temperature series. Yet other algorithms can handle mixed features better, such as those appearing in greenhouse gas series. Therefore, certain characteristics of the data end up being discriminant features themselves. Two further discriminants are the size of the time series and whether the missing blocks we see are the effect of a blackout.

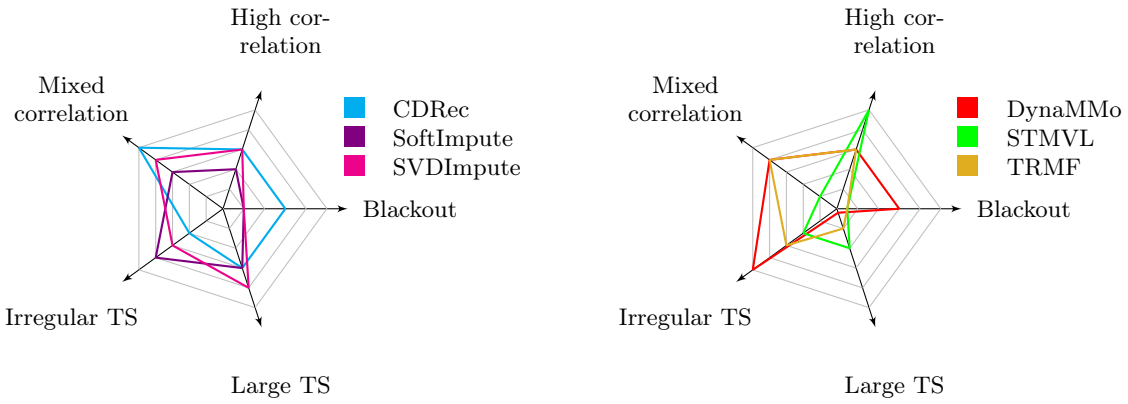
The comparison shows that none of the studied algorithms outperforms the rest in all dimensions. Although, in certain dimensions, there are clearly better algorithms. For instance, STMVL outperforms the rest in highly correlated time series, CDRec is the best contender in time series with mixed correlation (high and low), while DynaMMo stands out in irregular time series (fluctuations, spikes, outliers, etc.).

The blackout dimension stands out as none of the algorithms is able to properly handle them. Blackout remains an open dimension for the future imputation techniques – but that is not the only aspect that deserves more attention. We believe the at least the following topics can produce new algorithms.

Missing-block Initialization. Neither linear interpolation nor zero-initialization are ideal for matrix completion algorithms. Interpolation and zero-init introduce changes to the real rank of the matrix, which can penalize the convergence of these techniques. The initialization can be improved by "masking-away" the missing values which can be achieved by sampling from the observed values using Stochastic Gradient Descent.

Automatic Parameterization. The parameters introduced in the original paper of the respective techniques are tuned for datasets with specific properties. These parameters were updated in our implementation to produce the best results on most of the datasets. We observed that changing the parameters has a noticeable impact on the quality of the recovery. Utilizing an algorithm to automatically pick reasonable parameters based on the size of the time series (similarly to the rank auto-detection performed by SVT) could mitigate this impact.

Scalable Recovery. The studied algorithms are not scalable enough in a massive time series collection. These time series appear in applications where sensors with very high frequency are used, generating billions or even trillions of time series. The SENTINEL-2 mission conducted by the European Space Agency (ESA) represents such an example [31]. In such applications, GPU-based implementation could speed-up the matrix operations and thus, the scalability of these techniques. Alternatively, distributed recovery



(a) Most time-efficient techniques.

(b) Most time-consuming techniques.

Figure 12: Recommendation graph.

techniques, based on the MapReduce paradigm, for example, could also improve the scalability.

Implementation Techniques. Because the algorithms in this area are so computationally intensive, the difference between an efficiently designed implementation and one that is not can result in runtime differences of orders of magnitude. We describe a number of techniques which greatly affected performance in Appendix A. Yet we believe that there are still a number of areas that can be improved. For instance, some algorithms use a training model to learn the spatial and temporal correlations of the time series used for the recovery. In such cases, caching parts of the training values in main memory can largely improve the runtime.

6. CONCLUSIONS

In this paper, we conducted an experimental study that investigates the relative strengths and weaknesses of twelve algorithms for the recovery of large missing blocks in time series. In order to provide a test-bed for accurate comparisons, we re-implemented all but two of them. We then subjected the algorithms to an extensive benchmark containing a mix of real-world and synthetic time series. Not only were we capable of reproducing the results published in the algorithms' original papers, but we also documented behavior previously unknown.

The results of this benchmark helped us to uncover new findings, some of which are quite counter-intuitive. For instance, there are cases where the larger the missing block, the better the accuracy in recovering them. We substantiate all these findings and provided a method to systematically select the most suitable algorithms on a use-case basis. Lastly, we have also identified and described areas that can use further improvement, laying ground for further research topics. In terms of future work, it would be interesting to extend our framework to evaluate windowing techniques for the recovery of missing values.

ACKNOWLEDGMENTS

The first and third authors received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732328 (FashionBrain). The second and fourth authors received funding from the European Research Council (ERC) under the European Union Horizon 2020 Research and Innovation Programme (grant agreement 683253/Graphint).

APPENDIX

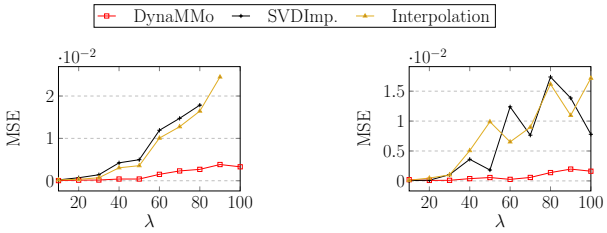
A. High-Impact Implementation Techniques

As we re-implemented a large number of algorithms, we noticed that some improvements applied to several of them. As described below, these improvements fostered a significant gain in runtime.

Improved memory management. In certain algorithms such as DynaMMo, SPIRIT and SVT, the original implementation fully materialized a number of intermediate results/matrices before the main matrix decomposition/factorization was done. We avoided such full materialization by adopting an *on-demand* approach to computing intermediate results, where we would produce the cells of those matrices as needed. Besides the space savings, this also brought better use of CPU caches, as the data in hot loops could be placed close together in memory.

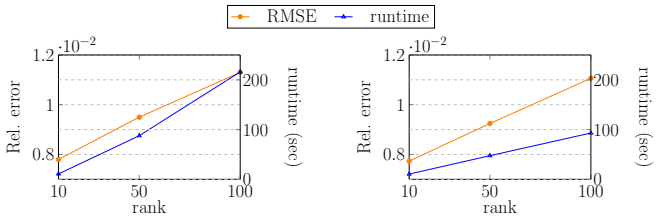
Support for sparse matrices. These kinds of matrices can use a hybrid representation that switches between different storage formats depending on the executed operation [50]. Such representation not only takes less space but also improves speed as the initialized zero values do not need to be processed. Support for sparse matrices had a significant impact because such matrices appear in the truncation step of every single matrix decomposition technique. As such, every one of the algorithms in that class benefited.

Efficient computation of algebraic operations. Some algorithms such as SPIRIT, DynaMMo, and TRMF perform very specific algebraic operations that are not readily available in many linear algebra libraries. In the original algorithms, Matlab provides high-level abstraction with which these operations could be easily expressed. In some cases, Matlab abstractions are also fast. In the case of DynaMMo and SPIRIT, though, we ported the Matlab operations to work with the (efficient) data structures and primitives provided by Armadillo. For example, we can vastly speed up algorithms that solve linear equations because we can parallelize their underlying matrix operations (e.g., multiplication, transpose, and QR decomposition).



(a) Original plot [29]. (b) Our result.

Figure 14: Reproducing results of DynaMMo [29].



(a) Original plot [7]. (b) Our result.

Figure 13: Reproducing results of SVT [7].

B. Reproducibility of Original Results

We repeat core experiments of two major works on missing values recovery: Cai et al. [7] (SVT) and Li et al. [29] (DynaMMo). We reproduce the runtime and accuracy experiments when varying the size of the missing blocks. For most of the remaining techniques, it was not possible to reproduce both the runtime and accuracy experiments as the respective papers do not study both dimensions together. The overall outcome is the following: i) we are able to reproduce the results of the experiments, ii) our implementation yields the same accuracy but better runtime results and iii) we identify the truncation value as the main factor that impacts the runtime and accuracy.

SVT. Cai et al. [7] introduced SVT. We repeat two core experiments on a synthetic dataset of size 1000×1000, which evaluate the runtime and accuracy of SVT by varying the rank factor and the missing rate. Figure 13a shows the results from the original paper while our results are shown in 13b. The recovery error is identical which confirms the correctness of our new implementation. However, the absolute runtimes differ, which is expected as unlike our C++ implementation, the original Matlab implementation does not support parallelization.

DynaMMo. Li et al. [29] introduced DynaMMo and compared it against SVDImpute [58] and linear interpolation. We repeat two core experiments on Mocap and Chlorine datasets, which evaluate the accuracy and runtime of DynaMMo by varying the average missing rate (λ) and the sequence length, respectively. Figure 14a shows the accuracy results from the original paper while our results are shown in 14b.

The results show that the difference between DynaMMo and the two competitors is smaller compared to what is reported in the paper. This is because we apply the interpolation within each time series containing missing values while the authors apply it across different time series. However, by adopting the latter interpolation setting, we achieve the same results. We also note that the runtime results reported in the paper are around $2\times$ higher than the ones we obtain by running the Matlab code provided by the authors. This result is expected as we use in our experiments a faster processor.

C. Complete Parameterization of Selected Algorithms

Some of the matrix completion techniques use two additional parameters, i.e., the max number of iterations and the convergence threshold. We ran experiments varying the max number of iterations from 50 to 500 and the convergence from 10^{-2} to 10^{-8}. The results show that tuning these two parameters generally only affects whether or not the algorithm returns a value; it doesn't however affect their accuracy or their efficiency. Thus, we set the max_iter and the conv_thre to 100 and 10^{-6} respectively for all algorithms except for ROSL whose optimal max_iter is equal to 500. For SVT, the optimal value of the scaling factor is 2. TRMF takes two other parameters: lag index set ($\mathcal{L}$) and the learning rates (λ). The default values provided by the authors, *i.e.*, $\mathcal{L} = [1, \ldots, 10]$ and $\lambda = [0.75, 0.75, 0.75]$, give the best performance and thus, were kept for the rest of experiments.

REFERENCES

[1] A. Agarwal, M. J. Amjad, D. Shah, and D. Shen. Time series analysis via matrix estimation. *CoRR*, abs/1802.09064, 2018.

[2] I. Arous, M. Khayati, P. Cudré-Mauroux, Y. Zhang, M. L. Kersten, and S. Stalinlov. Recovdb: Accurate and efficient missing blocks recovery for large time series. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, pages 1976–1979, 2019.

[3] L. Balzano, Y. Chi, and Y. M. Lu. Streaming PCA and subspace tracking: The missing data case. *Proceedings of the IEEE*, 106(8):1293–1310, 2018.

[4] D. Bertsimas, C. Pawlowski, and Y. D. Zhuo. From predictive methods to missing data imputation: An optimization approach. *Journal of Machine Learning Research*, 18:196:1–196:39, 2017.

[5] F. Biessmann, D. Salinas, S. Schelter, P. Schmidt, and D. Lange. "deep" learning for missing value imputationin tables with non-numerical data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 2017–2025, New York, NY, USA, 2018. ACM.

[6] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux. Intel berkeley research lab dataset, homepage: `http://db.csail.mit.edu/labdata/labdata.html`, 2004.

[7] J. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[8] J. Cambronero, J. K. Feser, M. J. Smith, and S. Madden. Query optimization for dynamic imputation. *PVLDB*, 10(11):1310–1321, 2017.

[9] S. Chouvardas, M. A. Abdullah, L. Claude, and M. Draief. Robust online matrix completion on graphs. In *2017 IEEE International Conference on*

Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017, pages 4019–4023, 2017.

[10] M. T. Chu and R. Funderlic. The centroid decomposition: Relationships between discrete variational decompositions and svds. *SIAM J. Matrix Analysis Applications*, 23(4):1025–1044, 2002.

[11] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[12] U. Dick, P. Haider, and T. Scheffer. Learning from incomplete data with infinite imputations. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 232–239, New York, NY, USA, 2008. ACM.

[13] X. Feng, W. Yu, and Y. Li. Faster matrix completion using randomized SVD. In *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI 2018, 5-7 November 2018, Volos, Greece.*, pages 608–615, 2018.

[14] N. Halko, P. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[15] D. Hening and D. A. Koonce. Missing data imputation method comparison in ohio university student retention database. In *Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management, Bali, Indonesia, January 7 – 9, 2014*, 2014.

[16] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(9):2117–2130, 2013.

[17] P. Huijse, P. A. Estévez, P. Protopapas, J. C. Príncipe, and P. Zegers. Computational intelligence challenges and applications on large-scale astronomical time series databases. *IEEE Comp. Int. Mag.*, 9(3):27–39, 2014.

[18] A. Jain, E. Y. Chang, and Y.-F. Wang. Adaptive stream resource management using kalman filters. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 11–22, New York, NY, USA, 2004. ACM.

[19] I. Jolliffe. *Principal component analysis*. Springer Verlag, New York, 2002.

[20] V. Kalofolias, X. Bresson, M. M. Bronstein, and P. Vandergheynst. Matrix completion on graphs. *CoRR*, abs/1408.1717, 2014.

[21] R. Kennedy, L. Balzano, S. J. Wright, and C. J. Taylor. Online algorithms for factorization-based structure from motion. *Computer Vision and Image Understanding*, 150:139–152, 2016.

[22] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Trans. Information Theory*, 56(6):2980–2998, 2010.

[23] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11:2057–2078, 2010.

[24] M. Khayati, M. H. Böhlen, and P. Cudré-Mauroux. Using lowly correlated time series to recover missing values in time series: A comparison between SVD and CD. In *Advances in Spatial and Temporal Databases - 14th International Symposium, SSTD 2015, Hong Kong, China, August 26-28, 2015. Proceedings*, pages 237–254, 2015.

[25] M. Khayati, M. H. Böhlen, and J. Gamper. Memory-efficient centroid decomposition for long time series. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4*, pages 100–111, 2014.

[26] H. Kim, J. Choo, J. Kim, C. K. Reddy, and H. Park. Simultaneous discovery of common and discriminative topics via joint nonnegative matrix factorization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 567–576, 2015.

[27] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[28] A. Lerner, D. E. Shasha, Z. Wang, X. Zhao, and Y. Zhu. Fast algorithms for time series with applications to finance, physics, music, biology, and other suspects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 965–968, 2004.

[29] L. Li, J. McCann, N. S. Pollard, and C. Faloutsos. Dynammo: mining and summarization of coevolving sequences with missing values. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 507–516, 2009.

[30] Y. Li, T. Yang, J. Zhou, and J. Ye. Multi-task learning based survival analysis for predicting alzheimer's disease progression with multi-source block-wise missing data. In *Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018, May 3-5, 2018, San Diego Marriott Mission Valley, San Diego, CA, USA.*, pages 288–296, 2018.

[31] M. Linardi and T. Palpanas. Scalable, variable-length similarity search in data series: The ULISSE approach. *PVLDB*, 11(13):2236–2248, 2018.

[32] H. Liu, X. Li, and X. Zheng. Solving non-negative matrix factorization by alternating least squares with a modified strategy. *Data Min. Knowl. Discov.*, 26(3):435–451, 2013.

[33] Y. Liu, A. Niculescu-Mizil, A. C. Lozano, and Y. Lu. Learning temporal causal graphs for relational time-series analysis. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 687–694, 2010.

[34] A. C. Lozano, H. Li, A. Niculescu-Mizil, Y. Liu, C. Perlich, J. Hosking, and N. Abe. Spatial-temporal causal modeling for climate change attribution. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 587–596, New York, NY, USA, 2009. ACM.

[35] J. Luengo, S. García, and F. Herrera. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowl. Inf. Syst.*, 32(1):77–108, 2012.

[36] Q. Ma, Y. Gu, W. Lee, and G. Yu. Order-sensitive

imputation for clustered missing values (extended abstract). In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, pages 2147–2148, 2019.

[37] C. Mayfield, J. Neville, and S. Prabhakar. Eracer: A database approach for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 75–86, New York, NY, USA, 2010. ACM.

[38] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010.

[39] J. Mei, Y. de Castro, Y. Goude, and G. Hébrail. Nonnegative matrix factorization for time series recovery from a few temporal aggregates. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2382–2390, 2017.

[40] P. Merlin, A. Sorjamaa, B. Maillet, and A. Lendasse. X-som and l-som: A double classification approach for missing value imputation. *Neurocomputing*, 73(7):1103 – 1108, 2010. Advances in Computational Intelligence and Learning.

[41] K. Mirylenka, V. Christophides, T. Palpanas, I. Pefkianakis, and M. May. Characterizing home device usage from wireless traffic time series. In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.*, pages 539–550, 2016.

[42] F. Monti, M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3697–3707. Curran Associates, Inc., 2017.

[43] S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer, and J. Stork. Comparison of different methods for univariate time series imputation in R. *CoRR*, abs/1510.03924, 2015.

[44] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 697–708, 2005.

[45] S. Papadimitriou, J. Sun, C. Faloutsos, and P. S. Yu. Dimensionality reduction and filtering on time series sensor streams. In *Managing and Mining Sensor Data*, pages 103–141. 2013.

[46] N. Radhakrishnan and B. Gangadhar. Estimating regularity in epileptic seizure time-series data. *IEEE engineering in medicine and biology magazine*, 17(3):89–94, 1998.

[47] I. Rodriguez-Lujan, J. Fonollosa, A. Vergara, M. Homer, and R. Huerta. On the calibration of sensor arrays for pattern recognition using the minimal number of experiments. *Chemometrics and Intelligent Laboratory Systems*, 130:123 – 134, 2014.

[48] K. Rong, C. E. Yoon, K. J. Bergen, H. Elezabi, P. Bailis, P. Levis, and G. C. Beroza. Locality-sensitive hashing for earthquake detection: A case study scaling data-driven science. *PVLDB*, 11(11):1674–1687, 2018.

[49] F. Saad and V. K. Mansinghka. A probabilistic programming approach to probabilistic data analysis. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2011–2019. Curran Associates, Inc., 2016.

[50] C. Sanderson and R. R. Curtin. A user-friendly hybrid sparse matrix class in C++. In *Mathematical Software - ICMS 2018 - 6th International Conference, South Bend, IN, USA, July 24-27, 2018, Proceedings*, pages 422–430, 2018.

[51] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Robust principal component analysis on graphs. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[52] D. E. Shasha. Tuning time series queries in finance: Case studies and recommendations. *IEEE Data Eng. Bull.*, 22(2):40–46, 1999.

[53] X. Shu, F. Porikli, and N. Ahuja. Robust orthonormal subspace learning: Efficient recovery of corrupted low-rank matrices. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 3874–3881, 2014.

[54] D. Skillicorn. *Understanding Complex Datasets: Data Mining with Matrix Decompositions (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007.

[55] Soldi, S., Beckmann, V., Baumgartner, W. H., Ponti, G., Shrader, C. R., Lubi´nski, P., Krimm, H. A., Mattana, F., and Tueller, J. Long-term variability of agn at hard x-rays. *Astronomy & Astrophysics*, 563:A57, 2014.

[56] D. L. Sun and C. Févotte. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 6201–6205, 2014.

[57] Q. Tan, G. Yu, C. Domeniconi, J. Wang, and Z. Zhang. Multi-view weak-label learning based on matrix completion. In *Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018, May 3-5, 2018, San Diego Marriott Mission Valley, San Diego, CA, USA.*, pages 450–458, 2018.

[58] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, David, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays . *Bioinformatics*, 17(6):520–525, 06 2001.

[59] X. Wang, Y. Chen, S. L. Bressler, and M. Ding. Granger causality between multiple interdependent neurobiological time series: Blockwise versus pairwise methods. *Int. J. Neural Syst.*, 17(2):71–78, 2007.

[60] K. Wellenzohn, M. H. Böhlen, A. Dignös, J. Gamper, and H. Mitterer. Continuous imputation of missing values in streams of pattern-determining time series. In *Proceedings of the 20th International Conference on*

Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017., pages 330–341, 2017.

[61] C. M. Yeh, N. Kavantzas, and E. J. Keogh. Matrix profile IV: using weakly labeled time series to predict outcomes. *PVLDB*, 10(12):1802–1812, 2017.

[62] X. Yi, Y. Zheng, J. Zhang, and T. Li. ST-MVL: filling missing values in geo-sensory time series data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2704–2710, 2016.

[63] J. Yoon, W. R. Zame, and M. van der Schaar. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Trans. Biomed. Engineering*, 66(5):1477–1490, 2019.

[64] H. Yu, N. Rao, and I. S. Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 847–855, 2016.

[65] M. Yue, L. Fan, and C. Shahabi. Inferring traffic incident start time with loop sensor data. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 2481–2484, 2016.

[66] D. Zhang and L. Balzano. Global convergence of a grassmannian gradient descent algorithm for subspace estimation. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, pages 1460–1468, 2016.

[67] S. Zhang. Nearest neighbor selection for iteratively knn imputation. *J. Syst. Softw.*, 85(11):2541–2552, Nov. 2012.

[68] X. Zhu. Comparison of four methods for handing missing data in longitudinal data analysis through a simulation study. *Open Journal of Statistics*, 4:933–944, 2014.