UCLouvain

École polytechnique de Louvain

Automatic correction and completion of weather and climate time series

Author: **Benoît Loucheur**
Supervisors: **Pierre-Antoine Absil, Michel Journée**
Reader: **Laurent Jacques**
Academic year 2020–2021
Master [120] in Mathematical Engineering

Abstract

Scientific observations show that Earth's climate has rapidly changed in the last 100 years. The human-induced global warming and natural variations from year to year and decade to decade, shape our climate. Therefore, the World Meteorological Organization has defined rules for creating climate normals that represent the average over a period of 30 years and are updated every 10 years. These climate normals are then used to compare shorter-term data at local, national or global levels.

According to the official recommendations of the WMO, from the year 2021, all the meteorological services in Europe and in the world will launch a recalculation of the climate normals over the period from January 1, 1990 to December 31, 2020. However, unexpected events for example, data logger overload, renovation of the weather station will cause missing values in time series, making the data much harder to be utilized. Errors in the datasets can also occur, such as incorrect calibration of the measuring instruments, overloading of the thermometer shelter, etc.

In this master's thesis, we compare the performance of several matrix completion methods for the problem of missing data imputation, the error detection and correction using some own-made benchmark. We compare two state of the arts methods with new matrix completion methods, among them graph regularized matrix factorization techniques. We are also proud to have found methods giving better results than the state of the art for the data completion problem. Although the results are good, there are still areas for improvement that can be considered for the future.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisors, Pierre-Antoine Absil and Michel Journée.

I have to thank Pierre-Antoine Absil for all his advice in the use and analysis of matrix completion methods. I thank him for all his comments which helped me to improve this text.

I am very grateful to Michel Journée for all the time he dedicated to the supervision of this work. I thank him for making this work possible by providing the necessary data from the Royal Belgian Meteorological Institute. For all his information about climate measurements in Belgium and its challenges.

I would also like to thank Laurent Jacques for accepting to spend time in reading my thesis.

My 5 years at university were a totally exciting time in my life, I would like to thank all the people I interacted with during those years, my classmates, my friends and my family. In particular, I would like to thank my parents and my sister for their unconditional support.

Contents

List of Figures

List of Tables

Notation

Sets

$\mathbb{R}$	The set of real numbers
Ω	The set of observed entries
$\bar{\Omega}$	The set of unknown entries
π	The set of noisy entries

Operations

$\|\cdot\|_{\ell_p}$	The ℓ_p norm : $\|\mathbf{X}\|_{\ell_p} = (\sum_{i=1}^{m}\sum_{j=1}^{n}\|\mathbf{X}_{ij}\|^p)^{1/p}$
$\|\cdot\|_{\ell_2} = \|\cdot\|_F$	The ℓ_2 norm or Frobenius norm: $\|\mathbf{X}\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}\|\mathbf{X}_{ij}\|^2\right)^{1/2}$
$\mathcal{P}_\Omega$	The orthogonal projector that replaces a matrix's entries in $\bar{\Omega}$ with zero.

$$\mathcal{P}_\Omega(\mathbf{X})_{ij} = \begin{cases} \mathbf{X}_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

Abbreviation

RMI **R**oyal **M**eteorological **I**nstitute of Belgium

TX Daily Ma**X**imum **T**emperature (in French : **T**empérature Ma**X**imale Journalière)

TN Daily Mi**N**imum **T**emperature (in French : **T**empérature Mi**N**imale Journalière)

IDW **I**nverse **D**istance **W**eighting

PCA **P**rincipal **C**omponent **A**nalysis

LRMC **L**ow **R**ank **M**atrix **C**ompletion

SVD **S**ingular **V**alue **D**ecomposition

LMaFit **L**ow-Rank **Ma**trix **Fit**ting

RTRMC **R**imannian **T**rust-**R**egion **M**atrix **C**ompletion

GRMC **G**raph-**R**egularized **M**atrix **C**ompletion

TRMF **T**emporal **R**egularized **M**atrix **F**actorization

GRALS **G**raph-**R**egularized **A**lternating **L**east **S**quares

RMSE **R**oot **M**ean **S**quare **E**rror

ROC **R**eceiver **O**perating **C**haracteristics

AUC **A**rea **U**nder **C**urve

Introduction

Context

Meteorological data

The meteorological data are, like any other measurements, subject to data loss or corruption.

Errors can occur when taking the measurement, due to a reading error on the measuring instrument, an omission of the sign for the temperature, an instrumental error, etc. Errors and losses can also occur during the transmission of the measurement to the RMI database, during an overload of the date logger or a malfunction of the electrical network.

The climate data must be complete and as reliable as possible because they are essential for the monitoring of the climate.

For example, Figure 1 shows the percentage of missing data for about 100 stations located in Belgium for the period from January 1, 1991 to December 31, 2020. It is not a matter of a few days of missing data! Some stations have up to 15 years of data to complete!

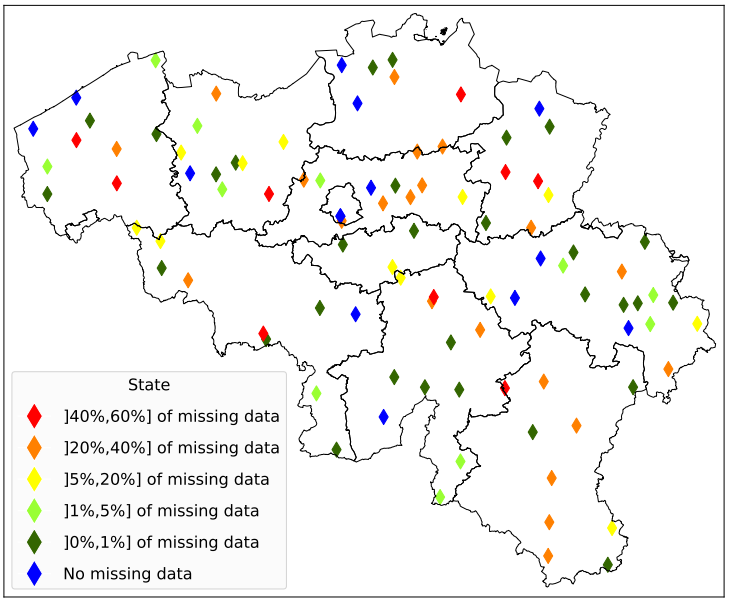


Figure 1: Data availability for ≈ 100 stations for the period 1991-2020

Almost all of the missing data is often due to the fact that the station was not in operation for the

entire period. It may have been set up well after 1991 or it may have stopped operating before 2020. Appendix A.1 shows the pattern of missing data in ≈ 150 Belgian stations. However, even if the stations are now completely inactive, it is interesting to try to complete the missing data to determine the climate normals.

The importance of climate normals.

The climate normals are intended to represent the average climate over a long period of time (i.e. 30 years). Since January 1, 2021, the reference period for the actual climate is from 1991 to 2020. This period is reviewed every 10 years due to climate change. Therefore, until December 30, 2030, the climate normals will always represent the climate over the interval 1991-2020.

The climate normals concerns all types of variables: temperature, precipitation, wind, sunlight, $\cdots$ but also a lot of other useful indicators illustrating the statistical distribution like mean, variance, quintile, records, numbers of days below/above a certain threshold.

It is very useful to compare the climate normals every 10 years because it allows to establish on a large scale the evolution of the climate at a given location. They are widely used as decision-making tools for public authorities and climate-sensitive industries. For example, sectors impacted by energy, water resources, agriculture and tourism needs climate normals.

It is possible to visualize the evolution of the normal values to observe the long term trend, as shown in Figure 2 which illustrates the normal of the monthly average temperature. Graphs like this one are very interesting to figure out the global warming.

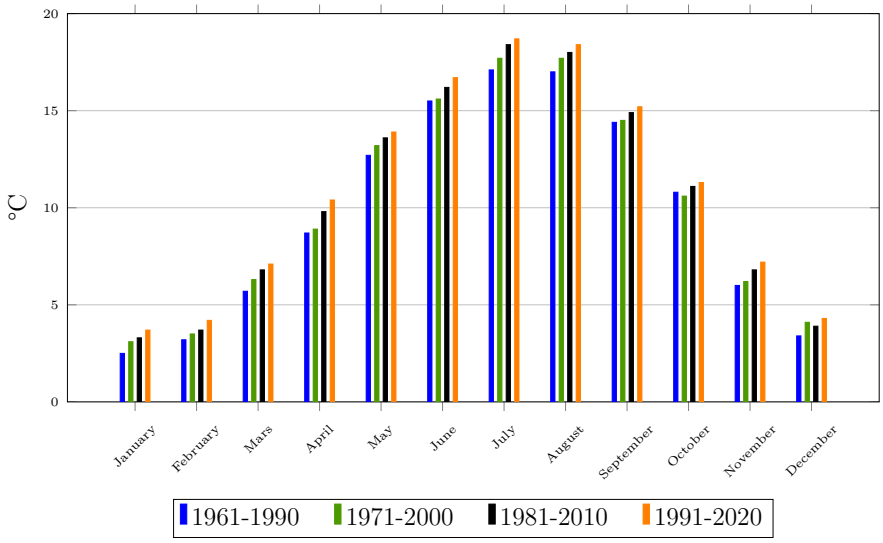


Figure 2: Climate normals in Uccle of average monthly temperature

The Royal Belgian Meteorological Institute frequently updates its climate atlas [1]. A climate atlas illustrates the geographical distribution of normals of several meteorological variables. One can find there as shown in Figure 3 a map of the annual average temperature normals, but also the average number of frost days per year [2], the average number of thunderstorms days per year

[1] https://www.meteo.be/fr/climat/climat-de-la-belgique/atlas-climatique

[2] https://www.meteo.be/fr/climat/climat-de-la-belgique/atlas-climatique/cartes-climatiques/temperature-de-lair/indices-de-temperature/jours-de-gel

[3] , ⋯ All maps made available are defined with respect to the 1991-2020 reference period, as recommended by the WMO, in order to be representative of the current climate.

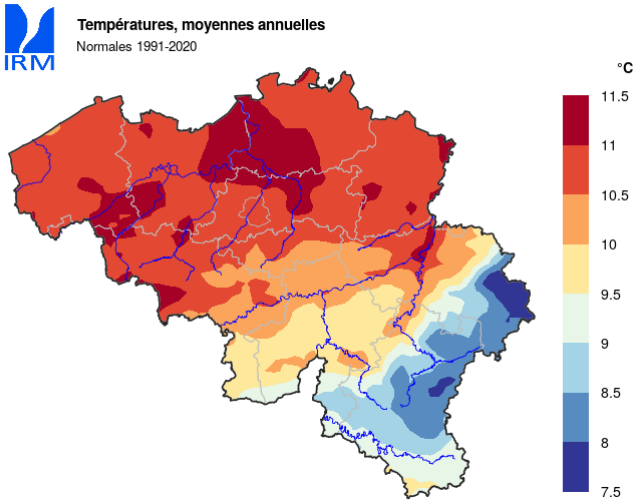


Figure 3: Annual average temperature normals for 1991-2020

Matrix Completion.

In October 2006, the famous company Netflix launched its competition: the Netflix Prize. It challenged anyone to achieve a higher accuracy than their recommandation system, which was called CineMatch. The winner was declared in June 2009, the "BellKor's Pragmatic Chaos" [3] having obtained better results than Netflix own's algorithm. They obtained a root mean square error (RMSE) lower by 10% and therefore won the $1M price! The Netflix problem is now categorized as a recommandation problem: the goal is to predict the rating an user would give to a film they have never rated. In order to allow Netflix to make good movies recommendations to its users.

The recommandations system is one of the possible applications of matrix completion. Indeed, the data can be stored as matrices. For instance, when someone watches a movie, it could rate it by filling an entry in a matrix, where each row correspond to a user and each column to a film. Clearly, this matrix would be really *incomplete* since not everyone has time to watch all the movies on Netflix. The aim of a Low-Rank Matrix Completion method is to complete this incomplete matrix. The idea of this type of method is that *often, data live in low-dimensional spaces*.

Our contributions

The aim of this thesis is to highlight the usefulness of matrix completion methods for the problem of data completion, error detection and correction in the field of time series.

[3] `https://www.meteo.be/fr/climat/climat-de-la-belgique/atlas-climatique/cartes-climatiques/orages/jours-dorage`

To do so, we have developed different benchmarks to deal with accuracy and parameterization. We will not discuss the efficiency because all the methods have not been implemented in the same programming language.

We will use both very standard matrix completion methods as well as more complex ones that are regularized by graphs. We will explain how we have adapted these methods to work in the time series field.

In our analysis, we have treated separately the problem of missing data completion from the problem of error detection and correction. For the data completion problem, we will compare the matrix completion methods with two classical methods currently used by Belgium and Norway. The error detection and correction problem is much more difficult to analyze, we will then explain the tools we used to evaluate the performance of the matrix completion methods.

Structure of the thesis

This master's thesis is structured as a report.

Chapter 1 will explain how climate data are measured and retrieved. It will also explain how temperature measurements are completed and how errors are currently detected and corrected in some meteorological services including the Royal Meteorological Institute of Belgium and the Norwegian Meteorological Institute. This chapter will also explain the basic principle behind matrix completion methods and how to apply these methods to complete time series.

Chapter 2 will briefly present the dataset provided by the RMI and the tools we used to generate our maps.

Chapter 3 will review in detail all the matrix completion methods used in the thesis. The parameters of these algorithms will be discussed since it is one of the major challenge of the thesis: *what is the best method and its parameters configuration to complete in the best way the missing entries in the meteorological data?*

Chapter 4 will first present the dataset provided by the RMI. As the completion problem and the error detection and correction problem are treated separately. First, the proposed benchmark for missing data completion will be presented and then the benchmark for error detection and correction. Both benchmarks will be explained in detail as well as the evaluation methods used to evaluate the performance of the matrix completion methods and the state of the art.

Chapter 5 will present all our results obtained on the dataset provided by the RMI.

The final chapter proposes a conclusion to this master thesis and lists potential directions for further research.

Chapter 1

State of the Art

1.1 Daily temperature measurement

In this thesis, we are exclusively interested in the daily minimum and maximum temperatures. The notation TN for daily minimum temperature and TX for daily maximum temperature will be used throughout the thesis. The purpose of this section is to show how the extreme daily temperature measurements are measured.

The following information is a rewrite from [1].

First, the way that thermometers were placed and protected has changed over time. The Belgian standard of the Stevenson screen was introduced in the 1950s. Before this, different types of exposures and stands were used in the network for protecting thermometers.

From 1880 to about 1910, the thermometers were placed in a large screen constructed of wood with no northern wall, no floor under the thermometer and single louvered wall on the other sides (see screen A on 1.1). This type of screen was quite basic, it allowed a strong natural ventilation, but a poor protection against soil radiation.

From 1911 to 1920, a small prism-shaped double sided screen with no floor constructed of zinc was used (see screen B on 1.1). This screen had no floor and was fixed on the north side of a mast (or a tree).

From 1920 to 1950, a small wooden screen with single louvered walls on all sides, a floor built with tree boards with the central one slightly shifted to the top and a double roof was in use (see screen C on 1.1).

Finally, the current version in place since 1950 is a large wooden Stevenson screen in a double-louvered design (see screen D on 1.1).

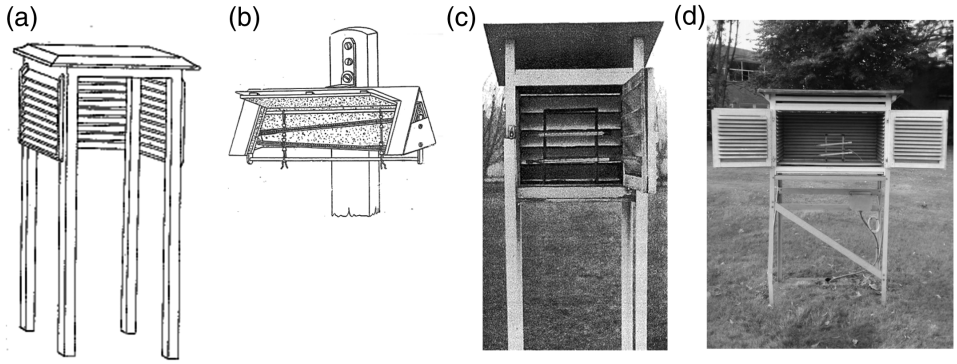
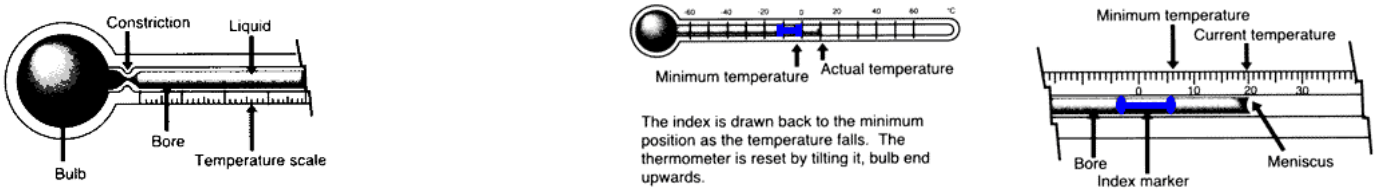


Figure 1.1: The different types of exposures and stands used in the network for protecting thermometers [1]

The thermometer used is a liquid-in-glass thermometer, consisting of a reservoir of liquid, called the bulb, and a very thin capillary tube through which the liquid rises when warming and descends when cooling. This type of thermometer is used in 80%[1] of the RMI network.



(a) Thermometer measuring the maximum daily temperature

(b) Thermometer measuring the minimum daily temperature

Figure 1.2: Measuring instruments [2]

The maximum temperature is measured with a mercury-in-glass thermometer which has the particularity of having a constriction in the neck of the tube as shown in 1.2a. As the temperature in the air increases, the mercury is forced past the constriction. However, as the temperature in the air falls, the constriction prevents the mercury from returning to the bulb. In this way, the mercury height then represents the maximum temperature of the day. Once the maximum temperature measurement has been taken, to reset this thermometer, the observer shake it gently.

The minimum temperature is measured with an alcohol-in-glass thermometer that contain a moveable index 1.2b. This moveable index plays the same role as the constriction in the mercury-in-glass thermometer. When the temperature falls, the alcohol and index move down the tube, but when the temperature rise the moveable index remains in the lowest position while the liquid expands up the tube. To determine the minimum temperature of the day, simply read the position of the moveable index. Once the minimum temperature measurement has been taken, to reset this thermometer, the observer tilt it bulb end upwards.

In Belgium, these measurements must be taken every day at 8h00 a.m. local time, the minimum temperature (TN) measured is then that of the day in question, on the other hand the maximum temperature (TX) measured is that of the preceding day.

When measurements are taken at 8h00 a.m., the minimum temperature indicated is indeed that of the current day, it is unlikely that there will be a new minimum temperature after 8:00 am. On the other hand, the maximum temperature indicated is not for the current day but for the day before. Indeed, the peak temperature of the day will occur in the afternoon after the measurement at 8h00 a.m.

1.2 Performing completion of missing data

The climate data are very often incomplete and this can be due to a multitude of reasons, for example the measuring device breaks down, renovations have taken place, the human observer is absent and does not measure, the data logger which is the device collecting the measurements is overloaded and loses data.

It is therefore necessary to estimate the missing data in order to perform a robust analysis of climate variability and change.

The official document of the WMO [4] explains in detail the procedure to follow for the update of the climate normals every 10 years. For example, the guide recommends that for a climate normal to be calculated for a specific month, data should be available for at least 80% of the year. In other words, this means that we need to have data for this month for at least 24 years of the 30 years to be able to establish a climate normal. Further information recommends not to calculate climate normals, if data are missing for at least 3 consecutive years.

To estimate the missing data, the WMO indicates some possible methods to use, such as spatial interpolation, temporal interpolation or use ancillary measurements : for example, using cloud amounts to estimate a missing daily sunshine value.

We will present two methods currently used to complete missing data, which will then be compared with our matrix completion methods.

The first one is the Inverse Distance Weighting method which is one of the methods used by the Royal Meteorological Institute of Belgium. The second is the Principal Component Analysis which is the method used by the Norwegian Meteorological Institute [5].

1.2.1 Inverse Distance Weighting

The Inverse Distance Weighting is a spatial interpolation method that explicitly makes the assumption that things that are closer to one another are more alike than those who are farther appart. This methods is also widely used in image interpolation[6] and algorithm optimization[7]. IDW assumes that each station has a local influence that decreases with distance.

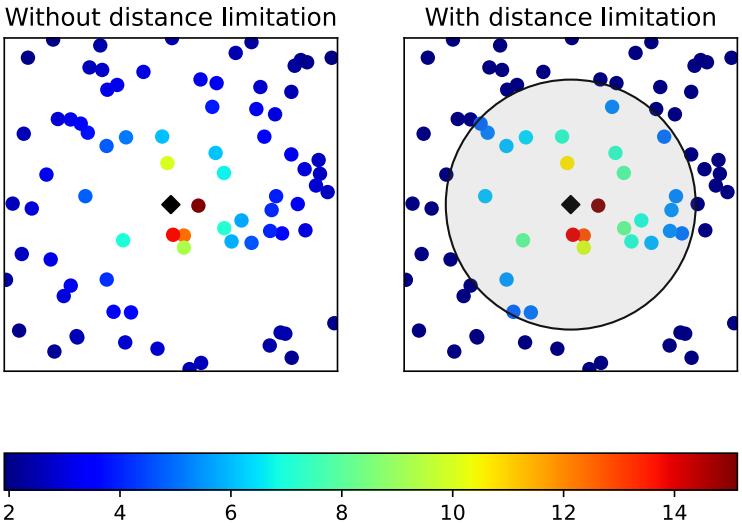


Figure 1.3: Two common forms of IDW

Figure 1.3 represents the two most used forms of IDW

The color of each point represents the weight if we wanted to make a prediction for the black symbol in the middle of the graph.

In the right graph in Figure 1.3, we restrict the action area according to a circle of a certain radius. Therefore, all points outside this circle automatically have a weight of 0 and will not influence the prediction.

Mathematical Formulation

The common formulation of IDW is the following :

$$\hat{x}_j(t) = \frac{\sum_{i=1}^{n} w_{i,j}^p x_i(t)}{\sum_{i=1}^{n} w_{i,j}^p} \tag{1.1}$$

As we can see in equation (1.1), the weights are raised to a power p. Then the rate at which the weights decrease is dependent on the value of p. In the trivial case where $p = 0$, there is no decrease with distance, and because each weight is the same, the prediction will be the mean of all the data values in the search neighborhood. As p increases, the weight for distant points decreases more and more. If the value of p is very high, only the immediate surrounding points will have an influence on the prediction.

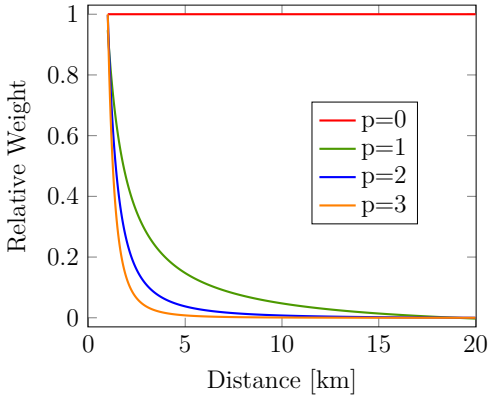


Figure 1.4: Comparison of differents power values

In our analysis, we compare two different weight functions

1. The *Shepard* function : $w_{i,j} = \frac{1}{dist(x_j,x_i)}$: $w_{i,j}$ represents the inverse of the distance between the station i and j. The distance is calculated according to the Haversine formula taking into account the latitude and longitude of both stations. This is the formulation that was used to generate the left graph in Figure 1.3.
2. The *Frank-Little* function : $w_{i,j} = \max\left(1 - \frac{dist(x_i,x_j)}{R}, 0\right)$ with R a parameter representing the radius of the circle of influence, maximum distance beyond which the interpolation points have no effect on the interpolated value. This function has the same purpose as the graph on the right in Figure 1.3.

Parameters

The parameters defining the IDW method are :

- **p** : the power parameter
- The weight function
  - The **Shepard** functions
  - The **Frank-Little** function
    - **R** : the radius of influence

1.2.2 Principal Component Analysis

The Principal Component Analysis (PCA) is a dimensionality-reduction technique that is generally used to reduce the dimensionality of a large datasets, by transforming a large set of variables into a smaller one that contains most of the information in the large set. This smaller set of uncorrelated vairables are called principal components.

PCA has been widely used for data visualisation and machine learning. Because smaller dataset are easier to explore, visualize and analyze.

The PCA was introduced by Karl Pearson in 1901 [8], explaining simply as a method of dimension reduction of many interrelated variable, while keeping a maximum variation present in the dataset.

PCA for data completion

In a time series dataset, the Principal Component Analysis is in theory a data reduction method used to find temporal and spatial patterns. However, inspired by linear regression, PCA can be used as a method to complete data.

The methods is as follows :

1. Select the submatrix representing the **complete** time series : $\mathbf{N} \in \mathbb{R}^{x \times n}$ with v the numbers of stations with complete information (stations without any missing data).
2. Centered this matrix : $\mathbf{F} = \mathbf{N} - \mathbf{1}_x \bar{\mathbf{N}}$. In order that every series is centered (i.e., zero mean).
3. Compute the covariance of the centered matrix : $\mathbf{C} = \mathbf{F}^\top \mathbf{F}$.
4. Perform the SVD decomposition of the covariance matrix : $\mathbf{C} = \mathbf{U} \Lambda \mathbf{U}^\top$.
5. Calculate the r first Principal Components (PC) : $\mathbf{p}_i = \mathbf{F}\mathbf{U}_i$ with $0 \leq i \leq r \leq v$.
6. Let $\mathbf{y}_t$ be the vector of known values of a station t for which we want to complete the missing data : $|\mathbf{y}_t| = |\Omega_t|$.
7. Centered the centered vector : $\mathbf{x}_t = \mathbf{y}_t - \bar{\mathbf{y}}_t$.
8. Finally, make the prediction of missing values with a weighted sum of the Principal Components : $\hat{\mathbf{y}}_t = \left(\sum_{k=1}^{r} \mathbf{h}_{t,k} \mathbf{p}_k \right)$ where the weights coefficients can be estimated [9] as: $\mathbf{h}_{t,k} = \dfrac{\mathrm{corr}(\mathbf{x}_t, \mathbf{p}_k)\mathrm{sd}(\mathbf{x}_t)}{\mathrm{sd}(\mathbf{p}_k)}$. with corr and sd denote the correlation coefficient and the standard deviation respectively.

The main components $\mathbf{p}_i$ describe the temporal variation, when the coefficients $\mathbf{h}_{t,k}$ represent the spatial pattern.

It is important to note that this method needs to have at least one complete time series without any missing data to be usable.

Parameter

The parameter defining the PCA method is :

- **r** : the total number of Principal Components

1.3 Performing error detection & correction of temperature data

Climate data are also subject to errors that may come from different sources:

- Observation errors
  - A bad reading of the thermometer or the rain gauge for instance.
- Recording / Digitisation errors
  - *Fat Finger Errors* : typing the wrong key when the measurement is encoded on a computer.
  - For the temperature, it is possible to forget the sign.
  - Precision (rounding) errors
- Systematic errors
  - The data is not in the right units.
  - The data are not what they are suppose to be : for example humidity data instead of the precipitations.
  - A data was declared as missing when it was just a 0.
  - A bad calibration of the measuring device.
  - An overload of the thermometer shelter.

The sources of errors are so numerous that the meteorological institutes set up a quality control for the data they receive.

We will analyze the quality procedure implemented by the RMI [10] which is based on a sequential test.

1.3.1 RMI Methods

The RMI quality control process for daily temperature data is therefore a sequential test. Each measurement will go through a series of tests, starting with simple tests to more complex ones. This process will give each measurement a confidence index ranging from -1 for missing data, 0 for incorrect data to 6 for validated data. The measures with confidence indexes between 1 and 5 included are therefore measures that are not certain to be wrong or correct. They may be probably correct or suspect. In this case, a human intervention is necessary to classify the measurement as totally wrong or totally correct.

The complete test diagram currently used by RMI [10] is presented in Figure 1.5.

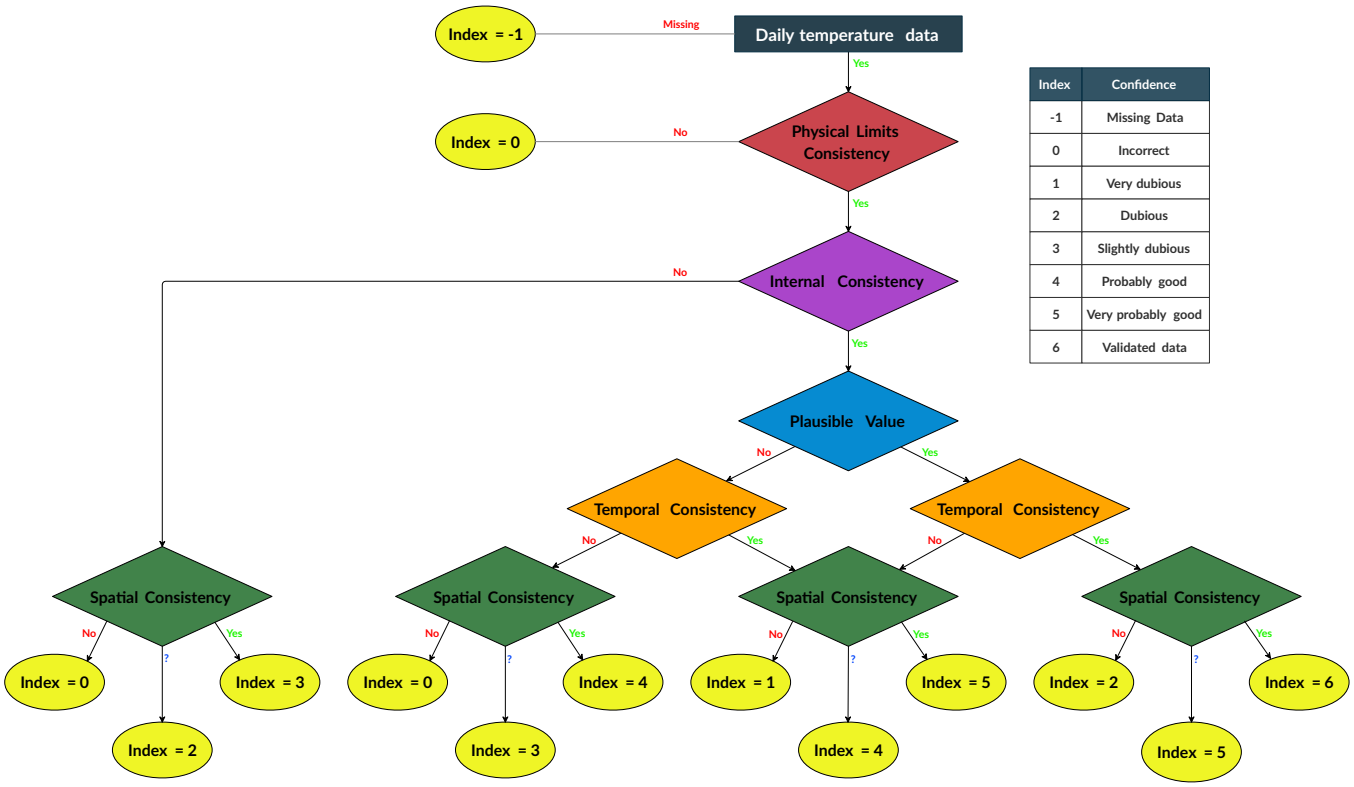


Figure 1.5: Automated Quality Control applied to temperature records and associated confidence

1) Physical Limits Consistency

The purpose of this test is to ensure that the temperatures are physically possible. Indeed, it is inconceivable to imagine temperatures of 30°C in winter in Belgium, at least for the moment! This type of test is mainly used to detect Fat Finger Errors. The boundary values that determine if a temperature is outside the acceptable physical range are based on the extreme data already obtained by the stations and adding a margin of 3°C.

The Belgian relief being rather particular, in fact in a few kilometers we pass from the maritime plain to the rocky escarpments of the Ardennes. Therefore, for this test there are temperature limits for the Flanders and the Ardenne zones. The upper and lower limits are presented in Table 1.1

	Lower limit TN [°C]	Upper limit TN [°C]	Lower limit TX [°C]	Upper limit TX [°C]
Winter	-26.5/-27.6	16.5/15.8	-15.2/-17.6	26.8/25.7
Spring	-12.8/-15.4	25.7/24	-2.8/-5.9	39.4/36
Summer	-1/-5.4	26/25.3	6.8/6	41.2/40
Autumn	-18.2/-23.2	21/18.2	-13.9/-14.2	31/29.9

Table 1.1: Extreme climatological boundaries for the Flanders/Ardenne zones per seasons used in the physical limit consistency test

2) Internal Consistency

This test verifies that for a given day, the minimum temperature of that day (TN) is not higher than the maximum temperature of that same day (TX). This test may seem unnecessary but it is actually consistent with the way TN and TX measurements are taken. Indeed, maximum and minimum temperatures, for the previous 24 h, are recorded at 08h00 a.m. LCT (local clock time). Minimum temperature is recorded against the day of the observation, and the maximum temperature against the previous day.

3) Plausible Value

This is one of the first tests that will need to use historical data to operate. To further refine the test, Belgium is not separated into two zones like the physical limits consistency test but into 4 zones presented in Figure 1.6 [10].

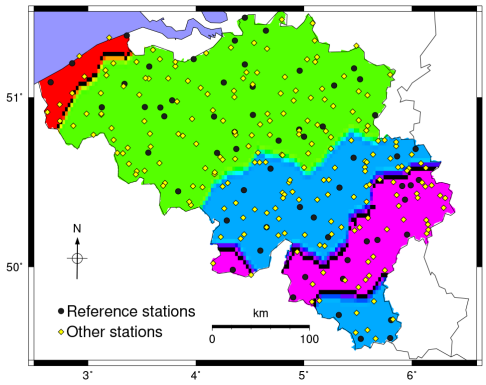


Figure 1.6: Division of the Belgium into 4 climate zones

In red is the Belgian coastline, a region with less than 5 meters of altitude. In green, the low and middle Belgium, region between 5 and 200m of altitude. In blue, you can find among others the Lorraine, the Condroz and the Fagne-Famenne, In purple, the Belgian Ardennes, an area of high plateau and valleys.

Assuming we are only interested in the TX measure, the following development is exactly the same for TN. So before performing the test, we must first perform some operations. First of all, for each zone, we need to retrieve the largest and smallest TX temperature value for each day of the 30 years of data. Assuming an annual seasonality of the temperature, we assume that the temperature follows a sinusoidal curve with a period of one year. So we make a fitting of 2 sinusoidal curves on the 30 years of data to obtain an upper curve representing the maximum temperature of TX, and a lower curve representing the minimum temperature of TX.

Once we have these curves, the test functional, it is then enough to look if our observation of the day d is between the two curves [11].

4) Temporal Consistency

The purpose of this test is to analyze the temperature variation from day to day.

To do this, for each zone in each season there is a maximum rate of temperature variation Δmax. To calculate this Δmax, we look at the percentile of the daily change for the 30 years data.

5) Spatial Consistency

The last test compares an observation at one station with neighboring stations on the same day. The result of this test can either be validated (Yes), failed (No) or suspect (?).

First, for each station, we look for three other closest stations whose difference in altitude is less than 150m. For each observation of a station, if the station record differs for more than 2.5 °C from each of its 3 closest neighboring stations or by more than 4 °C of the averaged values of the 3 closest neighboring stations then the observation is considered suspicious (?).

In the same way, if the station record differs for more than 5°C from each of its 3 closest neighboring stations or by more than 7°C of the averaged values of the 3 closest neighboring stations then the observation has failed the test (No). Otherwise the spatial consistency test is passed (Yes).

Resulting index

Once all the tests are passed, the observation finally gets its confidence index.

If the observation does not have a confidence index of -1,0 or 6, then a human intervention will be necessary to evaluate if the observation is erroneous or validated.

1.4 Matrix Completion

How to reconstruct a signal exactly by having only a few measurements? To answer this question, we have to look at a domain that is developing a lot these days: compressed sensing. It is said that it is possible to recover exactly the original signal, if the observed signal is sparse in some representation of the domain.

A similar question then arose for matrices. *Is it possible to reconstruct a matrix with only a few observations?* It was then shown that the exact recovery of the missing observations is possible if the matrix is low rank. This problem of recovering missing observations from a low rank matrix is then called the matrix completion problem.

This section aims at establishing a chronological follow-up of the different evolutions of the forms of the matrix completion problem.

1.4.1 General Concept

The Low-Rank Matrix Completion problem, also called LRMC problem, is the task of filling in the missing entries of a matrix from which only a subset of the entries are observed. The observed entries can of course contains noise or even outliers. If the original matrix was perfectly noiseless, the problem can be stated as recovering a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ knowing only a subset Ω of its entries, by finding a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ such that

$$\mathbf{X}_{ij} = \mathbf{M}_{ij} \quad \forall (i,j) \in \Omega \tag{1.2}$$

Without any constraint on $\mathbf{X}$, this problem has infinitely many solutions. Indeed, one could simply choose 0 as value of the unknown subset $\bar{\Omega}$ or take totally random numbers! It is then said that the problem in this state is ill-posed.

It is thus necessary to impose a condition. Generally speaking, we consider the hypothesis that the matrix $\mathbf{M}$ is of low rank $r \leq \min(m, n)$.

1.4.2 Rank minimization

Let's suppose that we want to retrieve the unknown entries of a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ of rank r. We know only p entries in $\Omega \subset \{1, \cdots, m\} \times \{1, \cdots, n\}$. The question we ask ourselves is the following: is it possible to recover exactly the entries of $\mathbf{M}$ if $p \ll mn$?

In a general case where the rank of the matrix is unknown, it is obviously impossible to solve this problem. However, if the rank r is small enough, there is hope. Indeed, the numbers of degree of freedom is $mr + r(n - r) \ll mn$: we first choose r linearly independent columns of size m and then choose r coefficients defining the linear combination that forms each of the remaining $(n - r)$ vectors. Since the actual dimensionality of this matrix is so much smaller than mn, perhaps a similarly small number of elements, sampled randomly, will be sufficient for recovering it.

We can now define the Rank Minimization problem (RM) that searches the lowest rank matrix that aggrees with our known entries:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \operatorname{rank}(\mathbf{X}) \\ \text{subject to} \quad & \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}) \end{aligned} \tag{1.3}$$

However, this optimization problem (1.3) is NP-hard to solve. This formulation tries to minimize the norm ℓ_0 of the singular values (i.e. the rank), Candès & Recht[12] then had the idea to relax the problem by minimizing instead the norm ℓ_1 of the singular values (i.e. the nuclear norm : $\|\cdot\|_*$).

The relaxed problem is called the nuclear norm minimization (NNM).

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}) \end{aligned} \tag{1.4}$$

The difference between the two formulations is that in (1.3), the objective function (i.e. the rank) counts the number of non-zero singular values, whereas in (1.4) the objective function (i.e. the nuclear norm) is the sums of the amplitude of the singular values.

From the field of optimization methods, we know that this type of problem is easy to solve with semidefinite optimization.

The methods we just described are dedicated to minimize the rank, or some relaxation of the rank, and the constraint ensures that the new matrix is close to the original data. Now, we will do the opposite: the rank is fixed *a priori*, and the errors between the new matrix and the original data on the set Ω is minimized. This new formulation is called the Frobenius Norm

Minimization (FNM) problem.

$$\begin{aligned} \min_{\mathbf{X}} \quad & \frac{1}{2} \|\mathcal{P}_\Omega(\mathbf{M} - \mathbf{X})\|_F^2 \\ \text{subject to} \quad & \text{rank}(\mathbf{X}) \leq r \end{aligned} \tag{1.5}$$

It's important to note that the cost function is differentiable, so techniques such as stochastic gradient descent, gradient descent and conjugate gradient can be used to solve this problem.

With this formulation, the problem is still NP-Hard due to the fact that the low-rank search space is non-convex. However, by using the truncated SVD, it's much more easier to compute the best projection of a matrix onto a rank-r subspace. It is based on this idea that the Principal Component Analysis works.

Example 1 *Consider the partially observed matrix* $\mathbf{M}$ *defined as*

$$\begin{bmatrix} 1 & 3 & 2 \\ \star & 9 & \star \\ 2 & \star & 4 \end{bmatrix}$$

where the unkown entries are denoted as $\star$*. We have* $\Omega = \{(1,1);(1,2);(1,3);(2,2);(3,1);(3,3)\}$*. There are an infinite number of possibilities for the completion of* $\mathbf{M}$*. Let's take for example these 2 examples of solutions:*

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 9 & 6 \\ 2 & 6 & 4 \end{bmatrix} \quad \rightarrow \text{rank}(\mathbf{X_1}) = 1$$

$$\mathbf{X}_2 = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 9 & 6 \\ 2 & 5 & 4 \end{bmatrix} \quad \rightarrow \text{rank}(\mathbf{X_2}) = 2$$

If the rank parameter r *is set to* 1 *in* (1.5)*, then* $\mathbf{X}_1$ *is a solution of the LRMC problem, but* $\mathbf{X}_2$ *is not.*

This example illustrate how LRMC works. However, in practice, the matrices that we want to complete are much larger and we have to face other unforeseen events like noisy observations, ...

A regularization term is often added to (1.5) in order to avoid overfitting. Hence, a term $\lambda \|\mathbf{X}\|_F^2$ would be added to the objective function, with some non-negative regularization parameter $\lambda \geq 0$. Since we are in a minimization problem, the algorithm will naturally try to minimize the term $\|\mathbf{X}\|_F^2$ as much as possible. One can think of the term $\|\mathbf{X}\|_F^2$ as a way to quantify the model complexity. The expression of the regularization term $\lambda \|\mathbf{X}\|_F^2$ is often accompagnied by a multiplicative factor $\frac{1}{2}$, this is done to make the expression of the gradient easier to write.

1.4.3 Low Rank Matrix Factorization

As explained above, the LRMC algorithm requires the calculation of the SVD to obtain the singular values. However, the calculation of the SVD is very expensive. Since every definite-dimensional matrix has a rank decomposition, we can write $\mathbf{X}$ as a product of two rectangular matrices, $\mathbf{U} \in \mathbb{R}^{m \times n}$ and $\mathbf{W} \in \mathbb{R}^{m \times n}$, such that $\mathbf{X} = \mathbf{UW}$

This property of rank decomposition is also non-unique. For all $\mathbf{S} \in \mathbb{R}^{n \times n}$ invertible, we can develop the previous expression :

$$\mathbf{X} = \mathbf{UW} = \mathbf{USS}^{-1}\mathbf{W} = (\mathbf{US})(\mathbf{S}^{-1}\mathbf{W}) = \tilde{\mathbf{U}}\tilde{\mathbf{W}}. \tag{1.6}$$

We have find another factorization of $\mathbf{X}$

It is then judicious to replace the low rank matrix $\mathbf{X}$ by its factorization: $\mathbf{X} = \mathbf{UW}$, where $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{U} \in \mathbb{R}^{m \times r}$ and $\mathbf{W} \in \mathbb{R}^{r \times n}$.

This new formulation also removes the constraint on the row. Indeed, it is now implicit in view of the dimensions of $\mathbf{U}$ and $\mathbf{W}$. There is now no more need to use the SVD. We finally obtain the Matrix Factorization problem [13].

$$\min_{\mathbf{U},\mathbf{W}} \quad \frac{1}{2} \left\| \mathcal{P}_\Omega(\mathbf{M} - \mathbf{UW}) \right\|_F^2 \tag{1.7}$$

For example, this problem can be easily solve using the power factorization method [14]. The idea is to find the solution by updating $\mathbf{U}$ and $\mathbf{W}$ alternately :

$$\mathbf{U}_{i+1} = \arg\min_{\mathbf{U}} \left\| \mathcal{P}_\Omega(\mathbf{M} - \mathbf{UW}_i) \right\|_F^2$$
$$\mathbf{W}_{i+1} = \arg\min_{\mathbf{W}} \left\| \mathcal{P}_\Omega(\mathbf{M} - \mathbf{U}_{i+1}\mathbf{W}) \right\|_F^2$$

To avoid over-fitting, it is useful to add a regularization term to the minimization problem (1.7)

$$\min_{\mathbf{U},\mathbf{W}} \quad \frac{1}{2} \left\| \mathcal{P}_\Omega(\mathbf{M} - \mathbf{UW}) \right\|_F^2 + \frac{\lambda_u}{2} \left\| \mathbf{U} \right\|_F^2 + \frac{\lambda_w}{2} \left\| \mathbf{W} \right\|_F^2 \tag{1.8}$$

1.5 Matrix Completion on Graph

The basic principle behind matrix completion is the low rank approximation. Low rank implies a linear dependence between rows and columns. However, this dependence is unstructured, so inverting rows and columns before performing the matrix completion should not impact the structure of the problem.

In some situations, it is interesting to add information about rows and columns that can be used in the completion problem as a regularization term.

Let's go back to the Netflix problem explained in the introduction. In every matrix completion formulations presented in section 1.4, the problem is always in an unorganized form. However, in the case of Netflix, we know that there is a relationship between the rows (the users) of the matrix (such as their age, gender, education, country of residence, $\cdots$) and a relationship between the columns (the movies) of the matrix (such as genre, main actors, country of origin, release date, $\cdots$).

It is possible that several people sharing the same taste for a fantasy movie, are likely to rate them similarly. In order to encode these relationships between users and movies, we rely on the structure of a graph.

The goal of this problem is to find a matrix of low rank which is structured by the proximities between the rows (users) and the columns (movies).

The first applications of the new models were mainly developed to solve the collaborative filtering problem (i.e. the Netflix problem). However, as explained in more detail in section 3.5, we will adapt this problem to the case of time series completion.

Let's take a standard case, consider a weighted undirected row graph $\mathcal{G}_r = (V_r, E_r, \mathbf{W}_r)$ with vertices $V_r = \{1, \cdots, m\}$ and edges $E_r \subseteq V_r \times V_r$ weighted with non-negative weights stored in $m \times m$ matrix $\mathbf{W}_r$. In the same way, a weighted undirected column graph $\mathcal{G}_c = (V_c, E_c, \mathbf{W}_c)$ with vertices $V_c = \{1, \cdots, n\}$ and edges $E_c \subseteq V_c \times V_c$ weighted with non-negative weights stored in a $n \times n$ matrix $\mathbf{W}_c$.

Let's always consider a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, which can be seen as a collection of of m-dimensional column vectors denoted by $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$, or n-dimensional row vectors denoted by $\mathbf{X} = ((\mathbf{x}^1)^\top, \cdots, (\mathbf{x}^m)^\top)^\top$.

The question that arises is the following, *how to convert each of our graphs into a regularization term?* One idea would be to impose that the solution of the optimization problem be smooth on the graphs, to implicitly force rows/columns proximities.

For the columns, the assumption of smoothness implies that $\mathbf{x}_j \approx \mathbf{x}_{j'}$ if $(j, j') \in E_c$. Stated differently, we want to minimize :

$$\sum_{j,j'} \mathbf{w}^c_{jj'} \left\| \mathbf{x}_j - \mathbf{x}_{j'} \right\|_2^2 = \mathrm{Tr}(\mathbf{X}\mathbf{L}_c\mathbf{X}^\top) = \left\| \mathbf{X} \right\|_{\mathcal{D},c} \tag{1.9}$$

where $\mathbf{L}_c = \mathbf{D}_c - \mathbf{W}_c$ is the Laplacian of the column graph $\mathcal{G}_c$, $\mathbf{D}_c = \mathrm{Diag}(\sum_{j'=1}^{n} \mathbf{w}^c_{jj'})$. The operator $\|\cdot\|_{\mathcal{D},c}$ is the graph Dirichlet semi-norm for columns [15, 16].

Similarly, for the rows, the assumption of smoothness implies that $\mathbf{x}^i \approx \mathbf{x}^{i'}$ if $(i, i') \in E_r$. So we want to minimize :

$$\sum_{i,i'} \mathbf{w}^r_{ii'} \left\| \mathbf{x}^i - \mathbf{x}^{i'} \right\|_2^2 = \mathrm{Tr}(\mathbf{X}^\top\mathbf{L}_r\mathbf{X}) = \left\| \mathbf{X} \right\|_{\mathcal{D},r} \tag{1.10}$$

with the Laplacian $\mathbf{L}_r$ of the row graph $\mathcal{G}_r$.

These smoothness terms (1.9) and (1.10) are then added to the matrix completion problem as regularization terms.

$$\min_{\mathbf{X}} \quad \frac{1}{2} \left\| \mathcal{P}_\Omega(\mathbf{M} - \mathbf{X}) \right\|_F^2 + \lambda \left\| \mathbf{X} \right\|_* + \frac{\lambda_r}{2} \left\| \mathbf{X} \right\|_{\mathcal{D},r} + \frac{\lambda_c}{2} \left\| \mathbf{X} \right\|_{\mathcal{D},c} \tag{1.11}$$

1.6 Matrix Completion of time series : basic concept

The matrix completion algorithms use data analysis methods to infer the missing block.

We will use as notation that we have n time series each having a length m. For example, let's take the scenario where we have $n = 5$ time series of length $m = 20$. This time series can be

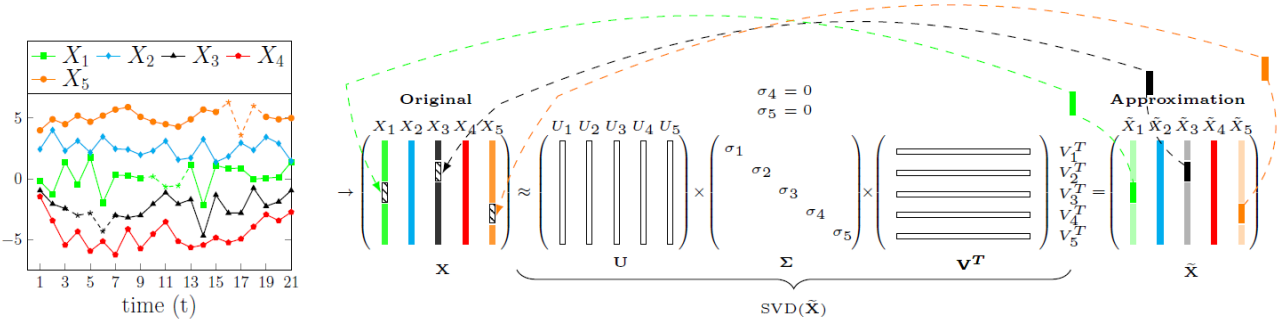


Figure 1.7: Completion of time series with SVD : simple example

represented as an $m \times n$ matrix : $\mathbf{X} \in \mathbb{R}^{m \times n}$ where each time series represents a single column. This scenario is presented on the left side of the Figure 1.7

Unfortunately, as can be seen in the time series graph, there is some missing data. Our $\mathbf{X}$ matrix therefore contains holes. A matrix-based completion algorithm transforms the data in a way that simplifies the application of dimensionality reduction. The SVD method is the most common technique that has been used to achieve this goal.

However, as our matrix contains missing data, we will not perform a classical SVD but a sparse SVD [17].

The sparse SVD decompose the input matrix that contains the time series $\mathbf{X}$ into three matrices $\mathbf{U}$,$\mathbf{\Sigma}$,$\mathbf{V}$. The $\mathbf{\Sigma}$ matrix exposes the number of linearly independant dimensions of the data and presents them in sorted order of importance, i.e, $\sigma_1 > \sigma_2 > \sigma_3 > \sigma_4 > \sigma_5$. Reducing the original matrix can be done by replacing by 0 the smallest value in $\mathbf{\Sigma}$, for example setting $\sigma_5 = 0$.

It is now sufficient to reduce the dimensionality by setting as shown on the image $\sigma_4 = \sigma_5 = 0$ to obtain a new $\hat{\mathbf{\Sigma}}$ matrix.

Finally, we determine the approximation of $\mathbf{X}$: $\tilde{\mathbf{X}} = \mathbf{U}\hat{\mathbf{\Sigma}}\mathbf{V}^\top$. The $\tilde{\mathbf{X}}$ matrix is represented as the matrix on the far right in the Figure 1.7.

The recovery process uses the results to fill the original missing block. The number of dimensions to reduce needs to be parametrized as it have a big impacts to the accuracy.

Chapter 2

Experimental Data

This chapter will present all the tools at our disposal. First we will discuss the tool that allows us to draw good quality maps of Belgium. Then we will see how the dataset provided by the RMI is structured.

2.1 Map visualisation

Basemap is a map creation tool running under Python. It is an extension of matplotlib, which allows to use all the tools available in the latter, so it has got all its features to create data visualizations, and adds the geographical projections and some datasets to be able to plot coast lines, countries, and so on directly from the library.

In order to create our geographical maps representing Belgium and its provinces, we had to use the shapefiles that fully characterize the Belgium.

The shapefile is a standard for representing geospatial vector data. It's a format that store the location, shape and attributes of geographic features.

The shapefiles of the Belgium used in the thesis are available for free on the website ArcGIS [1] created by the developers of the shapefiles format.

2.2 Dataset description

In the introduction, it was stated that the new climate normals are defined over the period 1991-2020. Even though the RMI has already completed the missing data in the period 1991-2020 by IDW, it is not possible to use their completed dataset as we cannot consider their estimates as real values in our benchmark.

To overcome this problem, the period has been reduced to 15 years over the interval 2005-2019. In this period the RMI has a large number of stations with only measured data. Thus, the dataset provided by the RMI contains daily minimum and maximum temperature data from January 1, 2005 until December 31, 2019, which makes exactly 5478 days of data.

[1] `https://hub.arcgis.com/datasets/esribeluxdata::belgium-provinces-1/about`

In order to realize reliable experiments, the RMI has tried to maximize as much as possible the number of stations in the dataset, so we still received stations with some missing values and therefore estimated beforehand by IDW.

For each day of the year, each station has a status which can be either :

- `obs`: The values in the dataset for this day and this station are actual measured values.
- `est` : The values in the dataset for this day and this station are values that were estimated by the RMI using IDW, the real measurements were missing.

The Figure 2.1 is a histogram showing the number of `obs` measurement days for all stations in the dataset. On the 97 stations available, 80 contains only observed data. Some stations have 5418 observations, so that about 1% of the 15 years of data had to be estimated for these stations.

How the `est` data is handled will be explained in detail in Chapter 4.

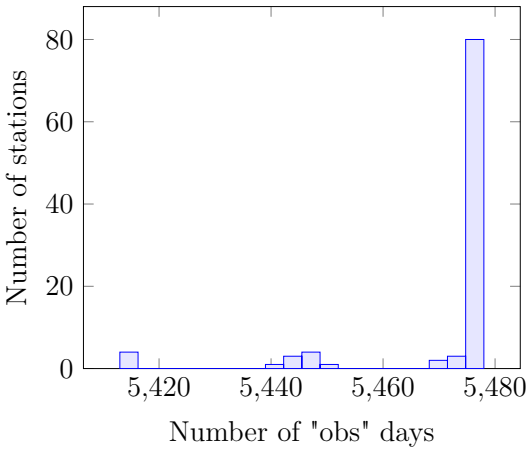


Figure 2.1: Histogram of the number of `obs` days of the stations in the dataset

The data available come from the RMI's climatological network [2], which is made up in part of volunteers, various companies, etc. The Figure 2.2 represents on a map the position of the stations at our disposal. In the appendix, there is a more detailed map A.2 containing the IDs of the stations as well as a Table A.3 with the IDs of the stations with their associated name, altitude and longitude.

[2] `https://www.meteo.be/fr/a-propos-irm/reseau-d-observation/reseau-climatologique`

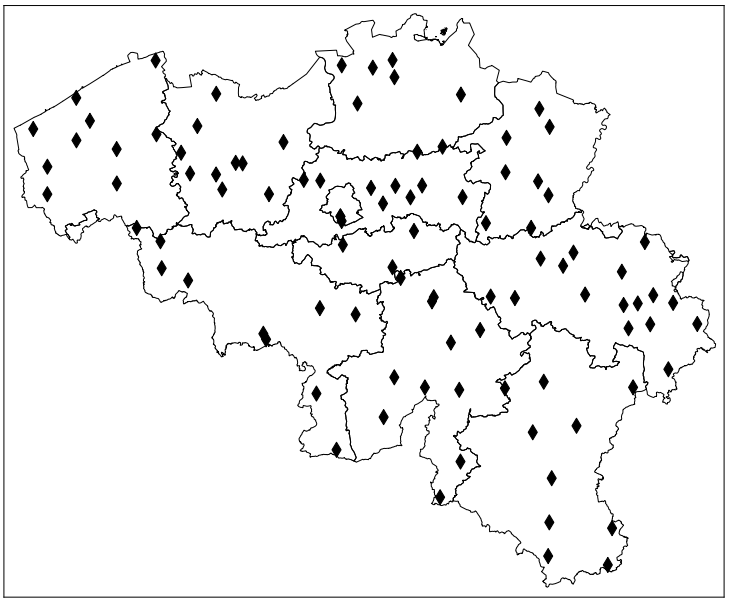


Figure 2.2: Map of the stations at our disposal

2.3 Source code

Following the idea of reproducible research, The source code including the implementation of most of the methods used, the generation of figures and tables, as well as our automated benchmark is available on my github `https://github.com/bloucheur/master-thesis`. Only the used dataset provided by the RMI is not available for confidentiality reasons.

Chapter 3

Matrix Completion methods

3.1 Low-rank Matrix Fitting

Wen et al. [18] suggested to solve the problem (1.7) by transforming it into another form without using a regularization term:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{W},\mathbf{Z}} \quad & \frac{1}{2}\|\mathbf{UW}-\mathbf{Z}\|_F^2 \\ \text{subject to} \quad & \mathcal{P}_\Omega(\mathbf{Z}) = \mathcal{P}_\Omega(\mathbf{M}) \end{aligned} \tag{3.1}$$

where $\mathbf{U} \in \mathbb{R}^{m\times r}$, $\mathbf{V} \in \mathbb{R}^{r\times n}$ and $\mathbf{Z} \in \mathbb{R}^{m\times n}$. The matrix $\mathbf{Z}$ is introducted in the problem for computational purpose.

A classical way to solve (3.1) is to fix the differential of the Lagrangian to 0. By this way, we obtain four systems of equations: one for each variable $\mathbf{U}, \mathbf{W}, \mathbf{Z}$ and one for the Lagrange Multiplier. We know [19] that these systems can be solved with the Gauss-Seidel elimination, giving and alternate optimization scheme.

To accelerate the convergence, Wen et al. [18] suggest minimizing separately over $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. The minimization of $\mathbf{Z}$ is also trivial. They used the successive over-relaxation (SOR) method which is known as a good variant of the Gauss-Seidel method.

First, the SOR algorithm start with arbitrary input $\mathbf{U}_0$, $\mathbf{W}_0$ and $\mathbf{Z}_0 = \mathcal{P}_\Omega(\mathbf{M})$, then the variables are updated in the i-th iterations as follow:

$$\begin{aligned} \mathbf{U}_{i+1} &= \arg\min_{\mathbf{U}} \|\mathbf{UW}_i - \mathbf{Z}_i\|_F^2 \\ \mathbf{W}_{i+1} &= \arg\min_{\mathbf{W}} \|\mathbf{U}_i\mathbf{W} - \mathbf{Z}_i\|_F^2 \\ \mathbf{Z}_{i+1} &= \mathbf{U}_{i+1}\mathbf{W}_{i+1} + \mathcal{P}_\Omega(\mathbf{M} - \mathbf{U}_{i+1}\mathbf{W}_{i+1}) \end{aligned} \tag{3.2}$$

The running time of this algorithm is very short due to the fact that the SVD computation is not necessary. However, this method may not converge to the global minimum but the authors proved convergence to a stationary point.

3.2 OptSpace

Keshavan et al. [20] introduced an algorithm known as OptSpace based on the minimization of the following cost function:

$$F(\mathbf{U}, \mathbf{V}) \equiv \min_{\mathbf{S}} \frac{1}{2} \|\mathcal{P}_{\Omega}(\mathbf{M} - \mathbf{USV})\|_F^2 \tag{3.3}$$

where $\mathbf{U} \in \mathbb{R}^{m \times r}$ and $\mathbf{V} \in \mathbb{R}^{r \times n}$ are orthogonal matrices, normalized such that $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}$. By definition, rank($\mathbf{USV}$)$\leq r$. Unfortunately, minimizing $F(\mathbf{U}, \mathbf{V})$ is not easy since it is a non-convex function but as explained in [20], the SVD of $\mathcal{P}_{\Omega}(\mathbf{M})$ gives a good initial guess.

3.3 Riemannian Trust-Region Matrix Completion

One of the disadvantages of the formulation (1.7), is that the factorization of the matrix $\mathbf{X}$ to $\mathbf{UW}$ is not unique. By this way, the optimal value of the inner optimization problem is a function of col($\mathbf{U}$) the column space of $\mathbf{U}$, rather than $\mathbf{U}$.

In this idea, Dai et al. [21] transformed the matrix factorization problem (1.7) on the Grassmann manifold $\mathcal{G}(m, n)$, i.e., the set of r-dimensional vector subspaces of $\mathbb{R}^m$, to obtain the following minimization problem:

$$\min_{\mathscr{U} \in \mathcal{G}(m,r)} \min_{\mathbf{W} \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} \Big((\mathbf{UW})_{ij} - \mathbf{M}_{ij} \Big)^2 \tag{3.4}$$

where $\mathbf{U} \in \mathbb{R}^{m \times r}$ is any matrix such that col($\mathbf{U}$)$= \mathscr{U}$.

The authors of RTRMC [22] (Boumal & Absil) have taken the formulation and improved it by adding a regularization term weighted by $\lambda > 0$, which yields to a smooth objectif function defined over an appropriate search space. They also add a confidence index $\mathbf{C}_{ij} > 0$ for each known observation $\mathbf{M}_{ij}$.

The RTRMC minimization function is the following:

$$\min_{\mathscr{U} \in \mathcal{G}(m,r)} \min_{\mathbf{W} \in \mathbb{R}^{r \times n}} \sum_{(i,j) \in \Omega} \mathbf{C}_{ij}^2 \Big((\mathbf{UW})_{ij} - \mathbf{M}_{ij} \Big)^2 + \frac{\lambda^2}{2} \sum_{(i,j) \in \bar{\Omega}} (\mathbf{UW})_{ij}^2 \tag{3.5}$$

3.4 SoftImpute

The idea of SoftImpute is to start from the nuclear norm minimization problem (1.4). The latter does not take into account the notion of measurement noise. So we adapt the problem in the following way:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & \|\mathcal{P}_{\Omega}(\mathbf{M} - \mathbf{X})\|_F^2 \leq \delta \end{aligned} \tag{3.6}$$

It is then possible to transform the problem (3.6) into the Lagrange form which gives:

$$\min_{\mathbf{X}} \quad \frac{1}{2} \|\mathcal{P}_\Omega(\mathbf{M} - \mathbf{X})\|_F^2 + \lambda \|\mathbf{X}\|_* \tag{3.7}$$

With $\lambda \geq 0$ the regularization parameter.

The basic ingredient behind SoftImpute is this lemma [23]:

Lemma 3.4.1 *Suppose the matrix* $\mathbf{M} \in \mathbb{R}^{m \times n}$ *has rank* r*. The solution of the optimization problem*

$$\min_{\mathbf{X}} \quad \frac{1}{2} \|\mathbf{M} - \mathbf{X}\|_F^2 + \lambda \|\mathbf{X}\|_* \tag{3.8}$$

is given by $\hat{\mathbf{X}} = \mathcal{S}_\lambda(\mathbf{M})$ *where*

$$\mathcal{S}_\lambda(\mathbf{M}) \equiv \mathbf{U}\mathbf{D}_\lambda\mathbf{V}^T \qquad \textit{with} \qquad \mathbf{D}_\lambda = \mathrm{Diag}[(d_1 - \lambda)_+, \cdots, (d_r - \lambda)_+] \tag{3.9}$$

$\mathbf{U}\mathbf{D}\mathbf{V}^T$ *is the SVD of* $\mathbf{M}$*,* $\mathbf{D} = \mathrm{Diag}[d_1, \cdots, d_r]$ *and* $t_+ = \max(t, 0)$

$\mathcal{S}_\lambda(\cdot)$ refers to *soft-thresholding* operation [24] with a threshold parameter λ, the proof uses the sub-gradient definition of the nuclear norm.

The SoftImpute algorithm uses the lemma 3.4.1 to solve the problem (3.7).

Algorithm 1: Soft-Impute for Matrix Completion

```
1 Initialize X^(0) = 0
2 for i ← 1 to max_iter do
3     X^(i) ← S_λ(P_Ω(M) + P_Ω̄(X^(i−1)))
4     if ||X^(i) − X^(i−1)||_F^2 / ||X^(i)||_F^2 < ε then return X^(i)
```

In other words, the algorithm iteratively replaces the missing entries with the current guess, and updates the guess by solving the problem (3.8) using the relation (3.9).

SoftImpute is a very interesting algorithm because it allows to require only a small number of known entries to recover the underlying low-rank matrix exactly [25],[12] or approximately [26],[27] from noisy entries.

3.5 Graph-Regularized Alternating Least Squares

The minimization problem that GRALS wants to solve is based on the matrix factorization problem with simple regularization (1.8). It is then necessary to add the two regularization terms of the row and column graphs as shown in (1.11).

However, in this case, the true matrix $\mathbf{X}$ is factorized as $\mathbf{X} = \mathbf{W}\mathbf{H}$, with a graph $\mathcal{G}_r = (V_r, E_r, \mathbf{W}_r)$ which still relates the rows of $\mathbf{W}$ and the graph $\mathcal{G}_c = (V_c, E_c, \mathbf{W}_c)$ which still relates the columns of $\mathbf{H}$.

The problem that GRALS wishes to solve is thus the following

$$\begin{aligned}\min_{\mathbf{U},\mathbf{W}} \quad & \frac{1}{2}\left\|\mathcal{P}_{\Omega}(\mathbf{M}-\mathbf{U}\mathbf{W})\right\|_F^2 + \frac{\lambda_u}{2}\left\|\mathbf{U}\right\|_F^2 + \frac{\lambda_w}{2}\left\|\mathbf{W}\right\|_F^2 \\ & + \frac{\lambda_L}{2}\{\mathrm{Tr}(\mathbf{U}^\top\mathbf{L}_r\mathbf{U}) + \mathrm{Tr}(\mathbf{W}\mathbf{L}_c\mathbf{W}^\top)\}\end{aligned} \tag{3.10}$$

Since this formulation use the Laplacian graph, it is then called the *graph-regularized matrix factorization* (GRMF).

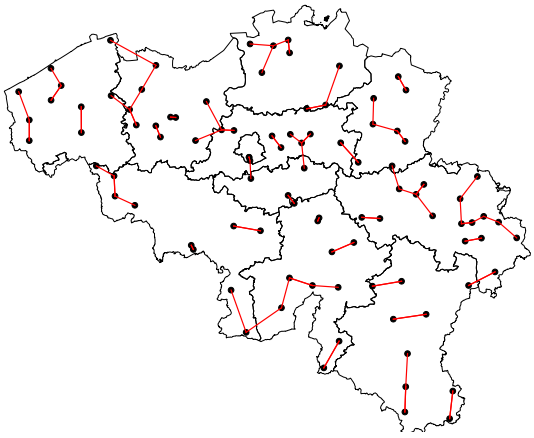
While this problem has a closed-form solution, the computational load quickly becomes prohibitive for larger and larger matrices. Then, Rao et al.[28] proposes an efficient implementation for solving (3.10) by applying the conjugate gradient method alternatively to $\mathbf{U}$ and $\mathbf{W}$, their algorithm named GRALS is explained in detail in their paper.

3.5.1 Spatial graph

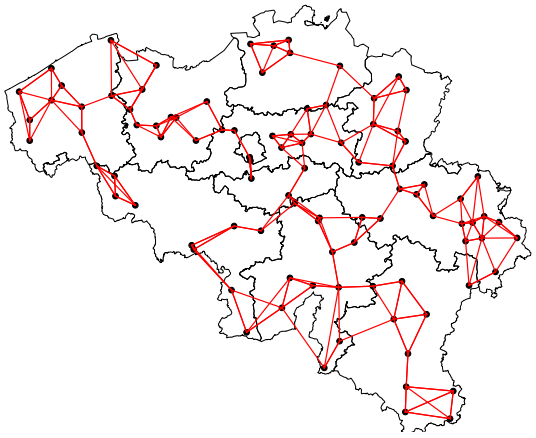
The first graph to provide to the algorithm is the graph for the rows. In our case, it is the spatial graph where each node represents a station and the weights enforce similarity between two stations.

The idea for the creation of this graph is to use the IDW method. Indeed, it seems obvious that the link between two very close stations is much stronger than the link between two very distant stations.

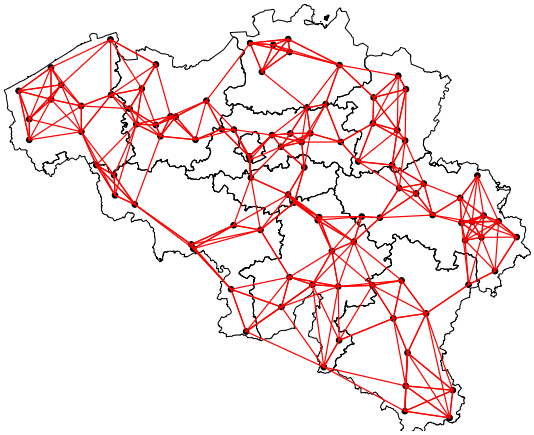
To create the spatial graph, we just have to choose the number of nearest neighbors K. Each station will then be linked by an edge to its K nearest neighbors and the weight of the edge between the two stations will be the inverse of the distance between these two stations.



(a) $K = 1$



(b) $K = 3$



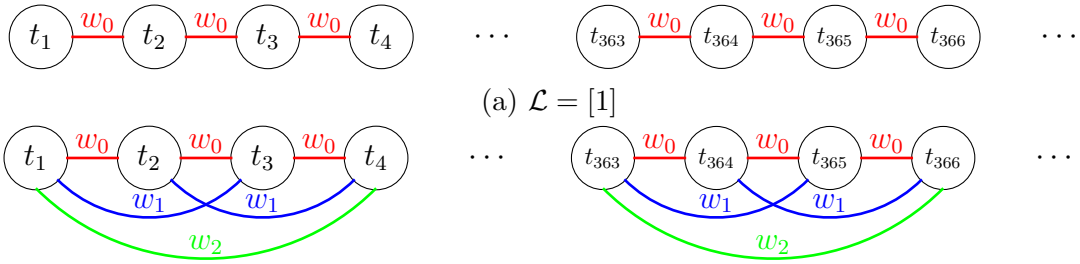
(c) $K = 5$

Figure 3.1: Spatial Graph with different nearest neighbors K

The Figure 3.1 shows for different values of K the shape of our graph. A red line between two stations indicates that there is an edge having as weight the inverse of the distance between these two stations.

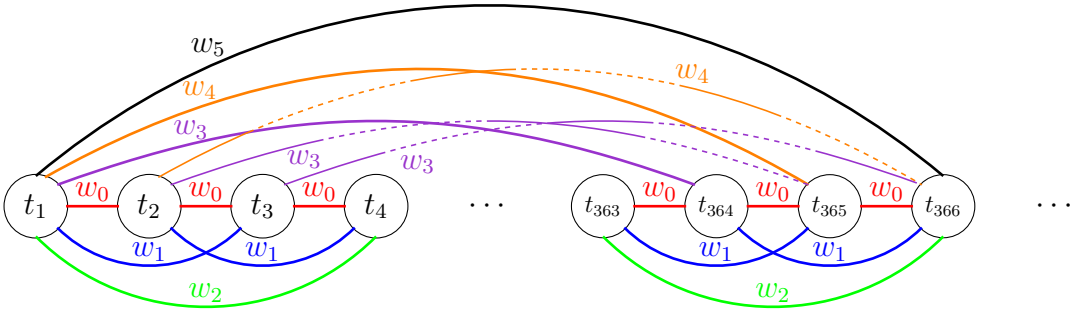
3.5.2 Temporal graph

The second graph to provide to the algorithm is the graph for the columns. In our case, it is the temporal graph where each node represents a day and the weights enforce similarity between two days.



(a) $\mathcal{L} = [1]$

(b) $\mathcal{L} = [1, 2, 3]$



(c) $\mathcal{L} = [1, 2, 3, 364, 365, 366]$

Figure 3.2: Temporal Graph with differents lag sets $\mathcal{L}$

Figure 3.2 shows the graph obtained with different types of lag sets.

First, the simplest lag set 3.2a, this one indicates that each day of the year is linked to the day before and after. In this case, all edges have a weight of 1.

Then , a more developed lag set allowing to see on several days 3.2b. In addition to the previous one, we allow each day to be linked to the two days after and before, as well as 3 days before and after. In this case, the red edges have a weight of 1, the blue edges have a weight of 0.66 and the green edges have a weight of 0.33.

Finally, the most complete lag set 3.2c. This one adds a seasonal aspect to the previous one. Since we have one measurement per day, the seasonality is in our case one year. In this case, the red edges have a weight of 1, the blue edges have a weight of 0.66, the green edges have a weight of 0.33 and the purple/orange/black edges have all a weight of 0.1.

3.6 Temporal Regularized Matrix Factorization

Contrary to GRALS which is a method that was basically built to answer the collaborative filtering problem (e.g. Netflix problem), TRMF is a method that is intended for the time series completion problem and developed by the same creators as GRALS.

TRMF is a much more complex model than GRALS, to understand in more detail how it works, it is recommended to read the official paper [29].

Like the GRALS method, TRMF also has a temporal graph defined by the lag set $\mathcal{L}$. However,

the difference is that in GRALS, we had to build the Laplacian ourselves and therefore we had to choose the weights of the edges. In TRMF, we only have to provide the lag set, which creates the structure of the graph, however the weights of the edges will be automatically learned by the algorithm. The last fundamental difference is that TRMF does not have a spatial graph.

TRMF propose to use well-known time series models (Auto Regressive) to describe temporal dependencies in $\mathbf{X}$ explicitly. The AR model is parametrized by the lag set $\mathcal{L}$ and weights $\mathcal{W} : \{\mathbf{W}^{(l)} \in \mathbb{R}^{k \times k} : l \in \mathcal{L}\}$. The lag set is known and fixed, however the weights are unknown and therefore to be determined in the minimization in addition to $\mathbf{F}$ and $\mathbf{X}$.

The minimization problem is as follows:

$$\min_{\mathbf{F},\mathbf{X},\mathcal{W}} \|\mathcal{P}_\Omega(\mathbf{Y} - \mathbf{F}\mathbf{X})\| + \lambda_f \underbrace{\frac{1}{2}\|\mathbf{F}\|_F^2}_{F\text{-regularizer}} + \lambda_x \underbrace{\mathcal{T}_{\text{AR}}\left(X \mid \mathcal{L}, \mathcal{W}, \eta\right)}_{\text{AR Regularizer}} + \lambda_w \underbrace{\frac{1}{2}\|\mathcal{W}\|_F^2}_{\mathcal{W}\text{-regularizer}} \tag{3.11}$$

Which can be rewritten as follows

$$\min_{\mathbf{F},\mathbf{X},\mathcal{W}} \|\mathcal{P}_\Omega(\mathbf{Y} - \mathbf{F}\mathbf{X})\| + \frac{\lambda_f}{2}\|\mathbf{F}\|_F^2 + \sum_{p=1}^{r} \lambda_x \mathcal{T}_{\text{AR}}\left(\overline{\boldsymbol{x}}_r \mid \mathcal{L}, \overline{\boldsymbol{w}}_r, \eta\right) + \frac{\lambda_w}{2}\|\mathcal{W}\|_F^2 \tag{3.12}$$

with

$$\mathcal{T}_{\text{AR}}(\overline{\boldsymbol{x}} \mid \mathcal{L}, \overline{\boldsymbol{w}}, \eta) = \frac{1}{2}\sum_{t=|\mathcal{L}|+1}^{n} \left(\mathbf{x}_t - \sum_{l \in \mathcal{L}} \mathbf{w}_l x_{t-l}\right)^2 + \frac{\eta}{2}\|\overline{\boldsymbol{x}}\|^2,$$

with $\mathbf{x}_t$ being the t-th element of $\bar{x}$, and $\mathbf{w}_l$ being the l -th element of $\overline{\boldsymbol{w}}$.

3.7 Summary of the methods

We have a total of 8 methods to evaluate in the completion problem and the error detection and correction problem. Among these are 6 matrix completion methods.

The Table 3.1 represents all the methods that we will use. Each method can be categorized according to the way it processes spatial and temporal information.

For example, a method in the global category in the spatial domain means that it does not use any spatial information. By spatial information, we refer to information such as latitude, longitude, altitude of stations. In our case, only IDW and GRALS can be considered as local because they use the altitude and longitude of the stations to operate. IDW uses it to calculate the weights of the interpolation and GRALS uses this information to generate the spatial graph.

In the same way, only TRMF and GRALS can have the temporal information since these two models can build a temporal graph created by the lag set $\mathcal{L}$.

		Temporal	
		Global	Local
Spatial	Global	LMaFit PCA OptSpace SoftImpute RTRMC	TRMF
	Local	IDW	GRALS

Table 3.1: Summary of the methods used organized to their temporal global/locality and spatial global/locality

Chapter 4

Methodology

This chapter aims to show the tools that we had to create in order to evaluate the algorithms in the data completion problem as well as the error detection and correction problem.

As explained in the introduction, the completion problem will be well separated from the error detection and correction problem.

At the level of the data completion module, the goal will be to find the algorithm and the value of its parameters allowing to complete the missing data in the best way. We will explain the path that was followed to choose the best choice of parameters for each algorithm and then choose the best algorithm. We will also explain the criterion that will allow us to define why one algorithm is better than another.

For the correction and detection of errors, we will explain the evaluation model that we have set up.

4.1 Missing Data Module

4.1.1 Parametrization

As shown in detail in chapter 3, all our methods have a number of parameters. The common point is that all matrix completion algorithms have one parameter in common, the rank r. But some of them have other parameters, like the coefficients λ for the regularization terms, $\cdots$

The question is then the following, *how to choose the optimal parameters for each method?* Often, at the level of the rank r, we choose $r = 5$ for a small problem and $r = 10$ for a big problem. This choice being arbitrary, we will present a more precise way to make this choice.

So for each method we will test a lot of parameter choices and we will have to choose the best combination. The first thing to define is the list of combinations of parameters we will test. To do this, we have based ourselves on the GridSearch approach.

The idea is the following: for each parameter, we define a set of values that we want to test. For example for RTRMC :

- The rank r : $\{1, 2, 3, 4, 5\}$

- The learning rate $\lambda : \{0.1, 0.2, 0.3, 0.4, 0.5\}$

The GridSearch will cross-reference each of the possible combinations of the parameters. In this case we have 25 possible parameter combinations. The GridSearch approach is very visual as shown in Figure 4.1. Each intersection point of the chosen parameter values forms a node and thus a parameter combination.

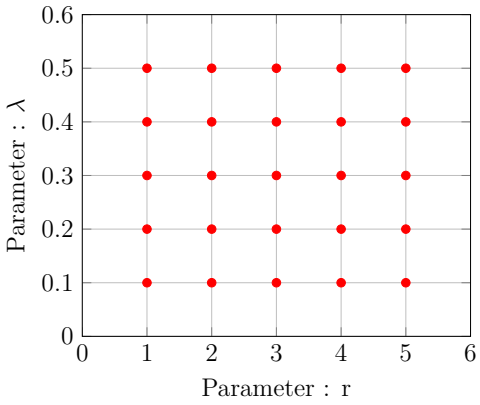


Figure 4.1: GridSearch example

Gridsearch has the advantage of being very simple to implement, but has the drawback that it can quickly become very heavy because the number of evaluations required (node on the grid) for this strategy increases exponentially with each additional parameter.

The Table 4.1 shows all possible value sets for each method. It is important to note that we have tried to test as many values as possible for the rank r, since this is the basic parameter for a matrix completion method. However for the algorithms, GRALS and TRMF, we had to strongly restrict the number of possible rank values. Indeed, these are models with a large number of parameters, respectively 5 and 3. As the number of combinations grows exponentially with the number of parameters, we could not afford to test as many rank values as the LMaFit method for example, which has only the rank as parameter.

Methods	Parameter	Definition	Values Tested
PCA [5]	r	The number of principal components	1 to 20
IDW	p	The power applied to the weight	1-6
	f	The weight function to use	Shepard(S) or Frank-Little (FL)
	R	The radius on influence (only if f = FL)	60,70,80, 90,100,125,150,200
LMaFit [18]	r	The rank of the model	1 to 20
SoftImpute [30]	r	The rank of the model	1 to 20
OptSpace [20]	r	The rank of the model	1,2,3,4,5,6,8,10,14,18,20
RTRMC [22]	r	The rank of the model	1 to 20
	λ	The learning rate	0.1,0.01,0.001
GRALS [28]	r	The rank of the model	1,10,15,20
	λ	The learning rate $[\lambda_l, \lambda_w, \lambda_u]$	[0.01,0.01,0.02],[1,0.01,1.01], [100,100,200]
	K	Number of nearest neighbors for the spatial graph	1,5,96
	$\mathcal{L}$	Lag set type for the temporal graph	[1],[1,2,3],[1,2,3,364,365,366]
	weight	Weighting of the spatial graph, if false, all edges have a weight of 1	true,false
TRMF [29]	r	The rank of the model	1,3,5,7,10,15,20
	λ	The learning rate $[\lambda_f, \lambda_x, \lambda_w]$	[0.75,0.75,0.75],[0.5,625,24], [0.2,1000,100]
	$\mathcal{L}$	Lag set type for the temporal graph	[1],[1,2,3],[1,2,3,364,365,366]

Table 4.1: Set of parameters for each methods

4.1.2 Monte Carlo cross-validation

The approach followed to evaluate our algorithms for the data completion problem is based on a machine learning method of model selection: Monte Carlo cross-validation. However, this model had to be modified in order to work in the case of data completion in matrix form.

The whole procedure we have created is illustrated in Figure 4.2.

First of all, we separate the procedure in 2 steps. The training part and the testing part. In order to work, it is absolutely necessary to separate our dataset into 2 parts. Since we work with time series, we have made the separation according to the years. The training part will have all data from 2005 to 2014 included, making a total of 10 years. The testing part will have all data from 2015 to 2019 included, making a total of 5 years.

Training Part

The goal of the training part is to find for each method, the optimal combination of parameters on the training set. This part has for each station the temperature data from 2005 to 2014.

The principle is to simulate missing data, as indicated by the big hollow arrow on the Figure 4.2. The known data are therefore represented by the set Ω.

Then, we give only the known data (indexed by Ω) to our matrix completion algorithms which will then have as objective to estimate the missing values (indexed by $\bar{\Omega}$). The way to generate the data as missing is quite particular. Indeed, we will generate holes in the same way as it is possible to see in reality, cfr. Appendix A.1. For some stations we remove a part at the beginning, at the end and in the middle. We also generate small data gaps here and there. We also make sure to have absolutely complete time series for PCA to work.

It only remains to evaluate the error committed by our algorithms. For this, we must establish a criterion, the one we use is the Root Mean Square Error (RMSE). Remember that some measures in the dataset have the status `est`, which means that they are not real measured values. For the calculation of the RMSE, it is thus necessary to remove from the set $\bar{\Omega}$, all the values having the status `est`. The set of the values having the status `est` is θ.

$$\mathrm{RMSE}_k = \sqrt{\frac{\sum_{(i,j)\in\bar{\Omega}\setminus\theta} (\mathbf{M}_{ij} - \mathbf{X}_{ij})^2}{|\bar{\Omega} \setminus \theta|}} \tag{4.1}$$

Once all RMSEs are calculated for each parameter combination of each algorithm. The experiment is repeated $K = 10$ times with each time a new Ω set.

When the 10 iterations are done, we calculate the average of each combination of parameters of each algorithm:

$$\overline{\mathrm{RMSE}} = \frac{\sum_{k=1}^{10} \mathrm{RMSE}_k}{10} \tag{4.2}$$

It only remains to select for each algorithm, its combination of parameters giving the smallest average RMSE.

Testing Part

Now, each algorithm has its own unique combination of parameters that give the best results in the training part.

It is then time to finally test all the algorithms in a new case. Just like the training part, we generate a set of known data but this time on the part of the dataset containing only the years 2015 to 2019.

We can then calculate the RMSE for each algorithm:

$$\text{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\bar{\Omega}\setminus\theta} \left(\mathbf{M}_{ij} - \mathbf{X}_{ij}\right)^2}{|\bar{\Omega} \setminus \theta|}} \tag{4.3}$$

It only remains to define the algorithm giving the smallest RMSE as the one having best answered the time series completion problem for our benchmark.

Summary

To summarize, the training part allows to determine for each algorithm its best combination of parameters on the training set (2005-2014). And the testing part, allows to determine the best algorithm, using the best parameters of the training part, on the testing set (2015-2019).

With the approach just explained, one could ask the question *why separate our data into two parts?* Why not directly select in the training part the best algorithm with the best parameters and not select the best parameter for each algorithm.

The answer is that the purpose of this two-part operation is to be able to test the methods on a brand new case. We should really think that the role of the training part is to train each of our models to find its best parameters and the testing part is the final competition between each method.

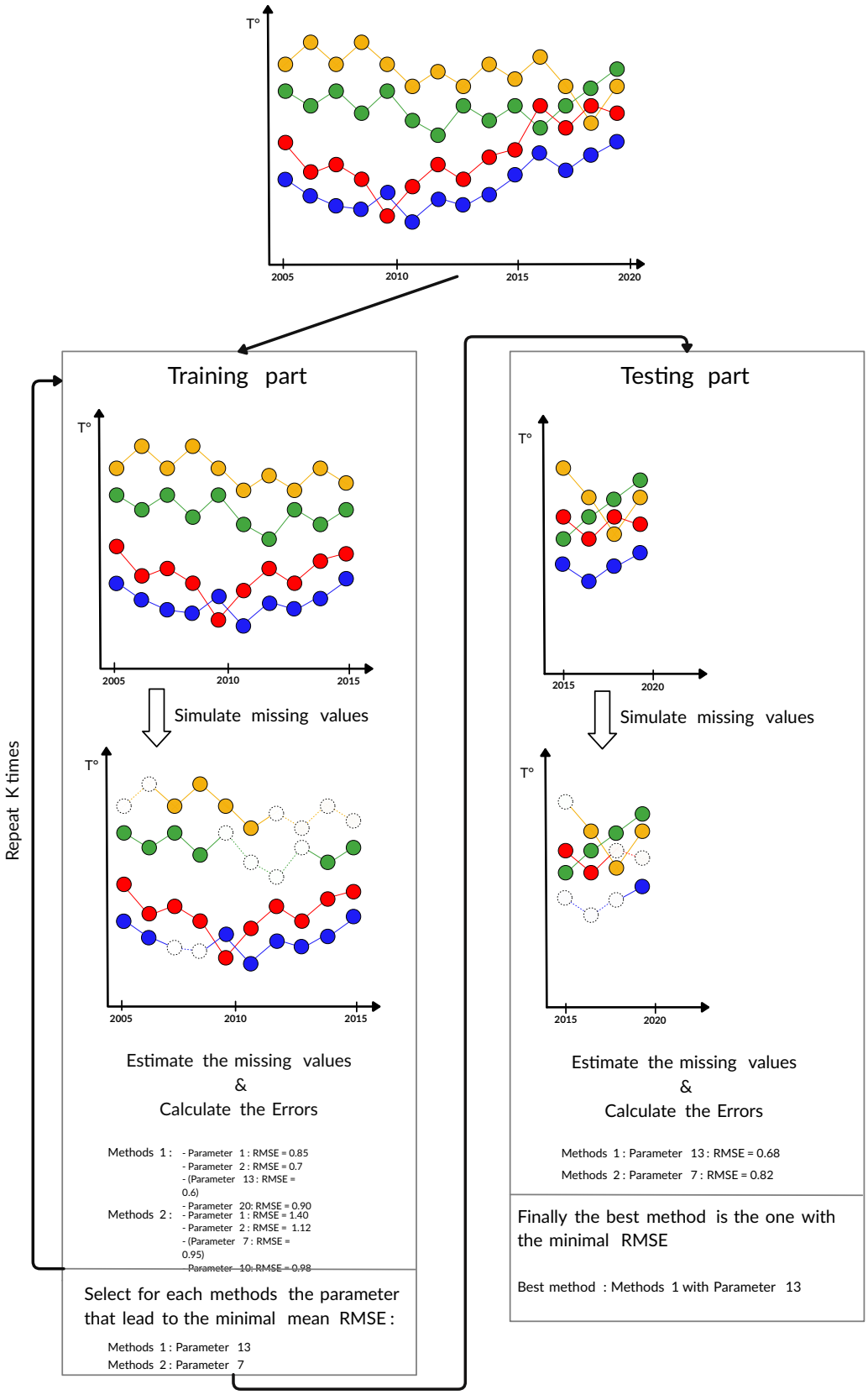


Figure 4.2: Missing Data Completion Module

4.2 Error Detection & Correction Module

In this module, we will work directly with the entire dataset, i.e. the 15 years of data. As said before, here we consider that all the data are known, we will not simulate missing data. However, we will have to simulate data as erroneous by adding noise.

Recall that even if the matrix has no missing data, the matrix completion methods will still factorize the input matrix into a new matrix of lower rank! The goal is then to investigate if this factorization allows to efficiently detect errors.

It is important to note that for this module, the LMaFit and SoftImpute algorithms cannot be used. Indeed, by definitions of these algorithms explained in section 3.1 and 3.4. They are only able to fill in missing data. For LMaFit, this comes from the constraint in its minimization problem 3.1. For SoftImpute, it is inherent to its way of working.

To evaluate the error detection performance, we will use the concept of ROC curves.

4.2.1 ROC curves

Historical

For the historical note, ROC curves were first used during World War II developed by electrical and radar engineers for detecting enemy objects in the battlefields. After the attack on Pearl Harbor, the army of the United States of America began new research to improve the rate of detection of Japanese aircraft from their radar signals. Their goal was necessarily to detect as many as possible while avoiding creating false alarms so as not to waste their resources. When the gain of the radar (comparable to the concept of volume in a radio) is fixed at zero, no signal (in this case an aircraft) is detected. By increasing the gain, more signal is detected but also more noise that can be interpreted as a real signal.

When the gain is low, the noise is very low and therefore the radar will make few false detections, but at the same time, only aircraft emitting a very strong signal (very close aircraft) will be detected.

When the gain increases, weaker signals will be detected but also more noise (the radar will then be able to detect bird noises like an airplane for example).

The principle of ROC curves is to find a gain value that allows the detection of a maximum number of aircraft while minimizing the false detection of noise.

After the war, the concept of ROC curves became generalized in many different fields, including medical decision making [31], laboratory testing [32],epidemiology [33] and bioinformatics [34]

Application of the ROC curves to our problem

To apply the concept of ROC curves to the detection of measurement errors, we need to define a binary classification test. In our case, we say that the test outcome is positive, i.e. the weather data is classified as an error/anomaly, or negative, i.e. the weather data is classified as correct.

The classification is performed as follows: each matrix completion method receives as input a complete matrix with some values that have been corrupted by adding noise. Obviously,

the information of which values have been corrupted is not revealed to the matrix completion method. Then for each value in this matrix, so for each row and for each column, we calculate the difference, in absolute value, between the input value and the output value. If the difference is greater than a threshold τ (similar to the gain for the radar), then this means that the method has detected an error for this measurement. Conversely, if the difference is less than the threshold then the method does not detect an error for this measurement.

There are four possible outcomes for this classification test:

- True Positive - TP(τ) : the measurement is classified as an error and it is indeed an error.
- False Positive - FP(τ) : the measurement is classified as correct when it was an error
- True Negative - TN(τ) : the measurement is classified as correct and it was indeed correct
- False Negative - FN(τ) : the measurement is classified as an error when it was correct

These 4 possibilities of classification are most commonly represented in a 2x2 confusion matrix as presented in Figure 4.3. Our goal is to maximize TP and TN and minimize FP and FN.

		Predicted Class	
		Erroneous	Correct
Actual Class	Erroneous	True Positive (TP)	False Positive (FP)
	Correct	False Negative (FN)	True Negative (TN)

Figure 4.3: Confusion Matrix

Figure 4.4 shows a concrete example of the classification. First, we have a signal represented by the very first line, *Original data*, in the figure. Then, we can manually add noise or not at some place in this signal, which corresponds to the second line and is therefore the input signal for our matrix completion algorithms. Finally, our matrix completions methods will output a signal represented by the line *Output data*. As shown, all values change, some more than others.

Then for different threshold values, we can display the different classification in the 4 possible categories. The choice of the threshold is very important because it influences the classification.

Here, for an easier representation, the example illustrates the case of a signal, represented by a vector. However, we do work with a matrix, the principle is of course the same.

The Table 4.2 illustrates how the 4 indicators evolve according to how the threshold evolves.

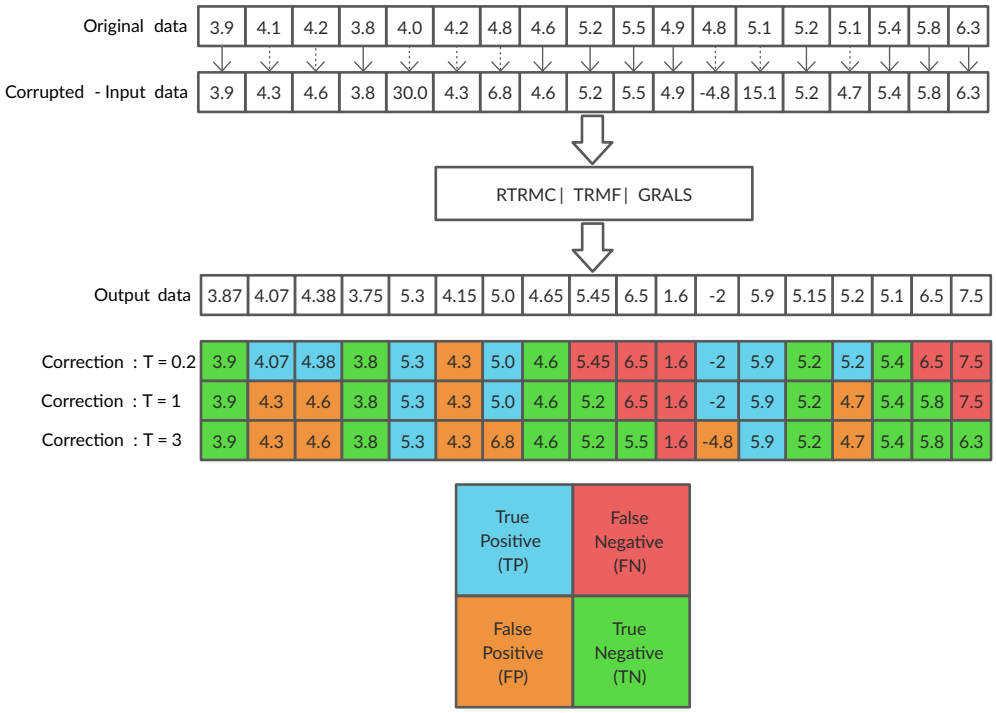


Figure 4.4: Error Detection & Correction Module

Decrease Thresholds		Increase Thresholds	
Increase #TP	☺	Decrease #TP	☹
Increase #FN	☹	Decrease #FN	☺
Decrease #FP	☺	Increase #FP	☹
Decrease #TN	☹	Increase #TN	☺

Table 4.2: Evolution of the distribution of the classification in 4 categories according to the threshold

As the table shows, increasing the threshold increases the number of true positives and decreases the number of false negatives which is a good thing. Unfortunately, it decreases the number of true positives and increases the number of false negatives. The choice of the threshold is therefore subject to compromises!

It is quite complicated to manage 4 variables at the same time. That's why the ROC curves are based only on 2 new variables obtained from the first 4. The new variables are therefore the following:

- The True Positive Rate (TPR), also called sensitivity, is the proportion of erroneous measurements that are correctly identified as erroneous by the algorithm.

$$\mathrm{TPR}(\tau) = \frac{\mathrm{TP}(\tau)}{\mathrm{TP}(\tau) + \mathrm{FN}(\tau)} \tag{4.4}$$

- The False Positive Rate (FPR), also called fall-out, is the proportion of non-erroneous measurement incorrectly identified as erroneous by the algorithm.

$$\mathrm{FPR}(\tau) = \frac{\mathrm{FP}(\tau)}{\mathrm{FP}(\tau) + \mathrm{TN}(\tau)} \tag{4.5}$$

In this new variable system, the False Positive Rate must always be maximized and the True Positive Rate minimized. In the same way as with our 4 classification categories, we can make a table describing the variation of the TPR and the FPR according to the evolution of the threshold.

Decrease Thresholds		Increase Thresholds	
Increase TPR	☺	Decrease TPR	☹
Increase FPR	☹	Decrease FPR	☺

Table 4.3: Evolution of the distribution of the classification in 2 categories according to the threshold

Finally, the ROC curve is a representation in a 2D graph with the TPR on the y-axis and the FPR on the x-axis. By varying the value of threshold, we obtain a series of couple (FPR,TPR) that we can put in the graph to draw a curve.

There are several ways to estimate the performance of a classifier, the simplest way is to calculate the area of the ROC curve, this integral is called the *Area Under the roc Curve* (AUC).

Figure 4.5 represents different theoretical cases, for example a perfect classifier, i.e. a classifier that can reach a TPR of 1 with an FPR of 0 is represented by the green curve in the graph.

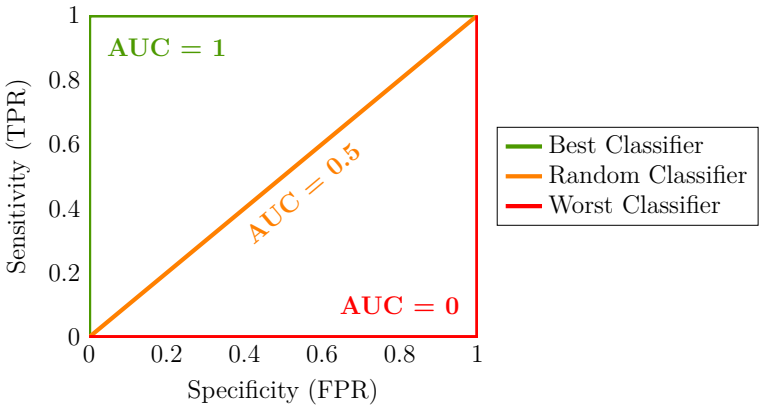


Figure 4.5: Theoretical ROC curves

AUC is a metric often used to evaluate the performance of a classifier. It has the following property:

- AUC is scale-invariant : It measures how well predictions are ranked, rather than their absolute values.

- AUC is classification-threshold-invariant : it measures the quality of the model's predictions irrespective of what classification threshold is chosen.

Unfortunately, in our case property 2 is not desirable. Indeed, for the moment we have only talked about the ability of the methods to detect errors. Now we have to evaluate the error correction which also depends on the threshold used. We cannot use the AUC as a metric because it only takes into account the error detection and not the correction.

4.2.2 Quality of the correction

To evaluate the error correction, we will always use the RMSE as a metric. However in this case, there is not one RMSE but two!

Indeed, depending on the value of the threshold, the algorithm will modify measurements that were perhaps errors, but it will also modify measurements that were not errors.

Let's consider the set π which represents the index of data where we have experimentally added noise. Obviously, the methods do not have access to this set. So we have the correction error and the corruption error.

- The correction error : $\text{RMSE}_{\text{Correction}}(\tau)$ allows to evaluate the performance of the method to correct the errors

$$\text{RMSE}_{\text{Correction}}(\tau) = \sqrt{\frac{\sum_{(i,j)\in\pi} (\mathbf{M}_{ij} - \mathbf{X}_{ij})^2}{|\pi|}} \tag{4.6}$$

- The corruption error : $\text{RMSE}_{\text{Corruption}}(\tau)$ allows to evaluate the error created by the method that modifies values that were not errors

$$\text{RMSE}_{\text{Corruption}}(\tau) = \sqrt{\frac{\sum_{(i,j)\in\bar{\pi}} (\mathbf{M}_{ij} - \mathbf{X}_{ij})^2}{|\bar{\pi}|}} \tag{4.7}$$

In all cases, the goal is to minimize these two values.

Summary

We therefore have a threshold value that impacts the error detection, which we observe via the TPR and FPR variables, and also impacts data correction and corruption, which we observe via the $\text{RMSE}_{\text{Correction}}$ and $\text{RMSE}_{\text{Corruption}}$ variables.

We need to find a threshold that maximizes the TPR and minimizes the FPR, $\text{RMSE}_{\text{Correction}}$ and the $\text{RMSE}_{\text{Corruption}}$.

In practice, we should create acceptance criteria. For example, I absolutely want an FPR lower than 0.1, a TPR higher than 0.9, an $\text{RMSE}_{\text{Correction}}$ lower than 0.5 and an $\text{RMSE}_{\text{Corruption}}$ lower than 0.5. If no choice of threshold respects all these conditions, then we should relax the bounds little by little until we find a threshold value that respects all the conditions. This constraint system allows to prevent the completion methods from modifying the dataset too much.

This way of doing things being very arbitrary, we have decided to perform just a more simplified analysis. We will only analyze how the 4 metrics evolve graphically for different rank values, by fixing the other parameters (like λ, $\mathcal{L}, \cdots$) using those obtained during the parameterization of the data completion problem.

Chapter 5

Experimental Results

5.1 Missing data completion

5.1.1 Training part analysis

The purpose of this subsection is to analyze the mean error of completions ($\overline{\text{RMSE}}$) of the methods by varying their parameters over the training set (2005-2014).

We will present here only the most interesting results, and thus what concerns the IDW, GRALS and TRMF methods.

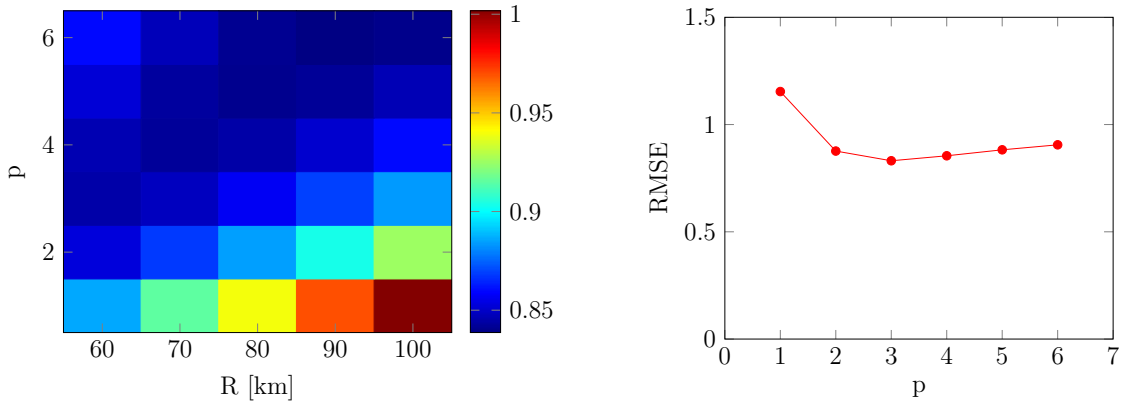
However, all the graphs for the other methods are available in the appendix. The RMSE graphs of the other methods for the TX dataset can be found in the appendix B.1 and for the TN dataset in the appendix B.2.

IDW

Figure 5.1 shows the results obtained for the IDW method on the TX dataset. This figure can be split in two. On the left, Figure 5.1a allows to observe the RMSE when we vary the power p and the influence radius R with the Frank-Little weighting function. By analyzing the RMSE values colored with the colorbar, we can see that when the radius of action R increases, the RMSE increases. Also, as the power p increases, the RMSE decreases slightly and then increases again.

The Figure 5.1b, considers that we use the Shepard function as a weighting function. We are almost in the same configuration as when we use the Frank-Little weighting function. When p increases, the RMSE decreases then increases when $p > 3$.

Among the two weighting functions, the lowest RMSE is obtained when the weighting function is Shepard's with $p = 3$.



(a) IDW with the Frank-Little weighting function (b) IDW with the Shepard weighting function

Figure 5.1: RMSE for IDW on the TX dataset

GRALS

Figure 5.2 illustrates some results obtained with the GRALS method. Indeed, this method has 5 parameters, so it is not possible to display everything in a graph. On the other hand, we are mainly interested in trying to see if the method improves when it is given more and more spatial information.

To do this, we displayed the RMSE with the simplest spatial graph $K = 1$. With also the spatial graphs with the 5 nearest neighbors with and without the weight information. Recall that when $w = false$, all edges have a weight of 1, so the stations do not know how to determine the order of proximity to their 5 nearest neighbors. When $w = true$, all edges have as weight the inverse of the distance between the two stations, so the more the weight is big, the more the stations are close.

The Figure 5.2 is particular because it indicates that the best RMSE is obtained when $K = 1$ thus with the lowest spatial information. Note also that when we give the weight information (from $w = false$ to $w = true$) for the same number of nearest neighbors K, the RMSE are clearly better (lower).

All curves indicate that the RMSE decreases as the rank increases until a certain point beyond which the RMSE increases again.

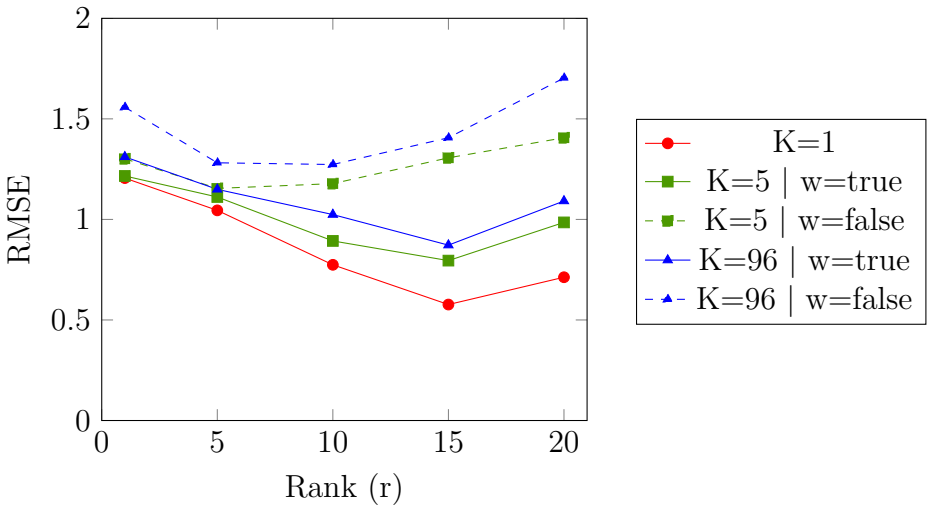


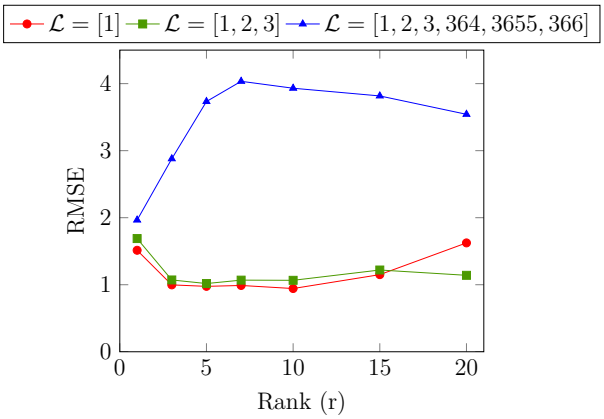
Figure 5.2: RMSE for GRALS on the TX dataset with $\lambda = [0.01, 0.01, 0.02]$ and $\mathcal{L} = [1]$

TRMF

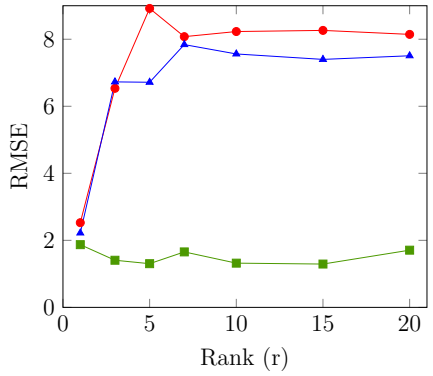
The Figure 5.3 illustrates some results obtained with the TRMF method. For this analysis, we wanted to analyze the influence of the choice of the regularization parameters.

The choices of λ represented in Figure 5.3a and 5.3b act almost in the same way. In Figure 5.3a, only the lag set $\mathcal{L} = [1, 2, 3, 364, 365, 366]$ explodes and gives bad results. In Figure 5.3b, the lag sets $\mathcal{L} = [1]$ and $\mathcal{L} = [1, 2, 3, 364, 365, 366]$ explode. It is not very surprising to see the lag set $\mathcal{L} = [1, 2, 3, 364, 365, 366]$ give poor results. Implementing seasonality may seem interesting, but in terms of meteorology it can't help. It is unlikely that creating a temporal link between the temperature of January 1, 2020 and January 1, 2021 will help the model. On the contrary, creating a link between the temperature of January 1st 2020 and January 2nd 2020 makes much more sense!

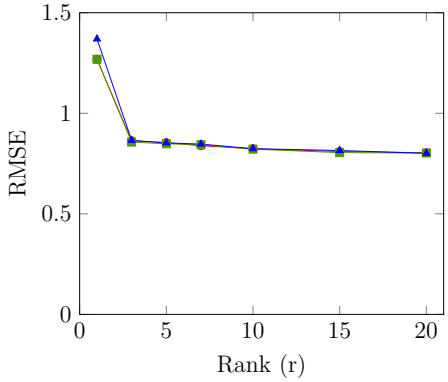
The Figure 5.3c is particular because it seems that in this configuration of the λ, the choice of the lag set $\mathcal{L}$ does not have much importance anymore. It is moreover this configuration which gives the best results for the training part in the TX dataset.



(a) TRMF with $\lambda = [0.5, 625, 24]$



(b) TRMF with $\lambda = [0.2, 1000, 24]$



(c) TRMF with $\lambda = [0.75, 0.75, 0.75]$

Figure 5.3: RMSE for TRMF on the TX dataset

Summary of the best parameters for TX and TN

The Table 5.1 summarizes for all methods the optimal parameter configuration resulting from the training part for TX and TN.

Methods	Parameter	Definition	Value for TX	Value for TN
PCA [5]	r	The number of principal components	5	6
IDW	p	The power applied to the weight	3	6
	f	The weight function to use	Shepard(S)	Frank-Little (FL)
	R	The radius on influence (only if f = FL)	/	100
LMaFiT [18]	r	The rank of the model	2	2
SoftImpute [30]	r	The rank of the model	15	15
OptSpace [20]	r	The rank of the model	1	5
RTRMC [22]	r	The rank of the model	11	13
	λ	The learning rate	0.001	0.001
GRALS [28]	r	The rank of the model	15	15
	λ	The learning rate $[\lambda_l, \lambda_w, \lambda_u]$	[0.01,0.01,0.02]	[0.01,0.01,0.02]
	K	Number of nearest neighbors for the spatial graph	1	5
	$\mathcal{L}$	Lag set type for the temporal graph	[1]	[1,2,3]
	weight	Weighting of the spatial graph, if false, all edges have a weight of 1	false	true
TRMF [29]	r	The rank of the model	20	20
	λ	The learning rate $[\lambda_f, \lambda_x, \lambda_w]$	[0.75,0.75,0.75]	[0.75,0.75,0.75]
	$\mathcal{L}$	Lag set type for the temporal graph	[1]	[1,2,3]

Table 5.1: Set of optimal parameters for each methods

Several interesting findings emerge from this table.

For the matrix completion methods with a regularization parameter (or learning rate). Whether it is with TX or TN, the optimal value is the same.

For LMaFit, SoftImpute, GRALS and TRMF, the optimal rank parameter does not change with TX or TN. On the other hand, for OptSpace and RTRMC, the optimal rank parameter increases when going from TX to TN.

At the IDW level, when we work with the minimum daily temperature, the optimal configuration is obtained with the Frank-Little weighting function, while with the maximum daily temperature, the optimal configuration is with the Shepard weighting function.

5.1.2 Testing part analysis

Here is the final test, the one that, using the optimal configurations presented in the Table 5.1, will allow to complete the missing data for a new case over 5 years of data.

First of all, the Appendix B.10 is a diagram that shows how the holes for this final test were generated.

The completed climate maps for the TX dataset via the methods and parameters in Table 5.1 are shown in Figure 5.4 and 5.5 with the RMSE written in the caption. At the top of these figures the original climate map constructed without the missing data are displayed as a reference. The RMSE values have also been compiled in the Table 5.2

Methods	RMSE
IDW	0.938
PCA	1.14
LMaFit	0.871
OptSpace	1.32
RTRMC	0.539
SoftImpute	0.563
GRALS	0.607
TRMF	0.813

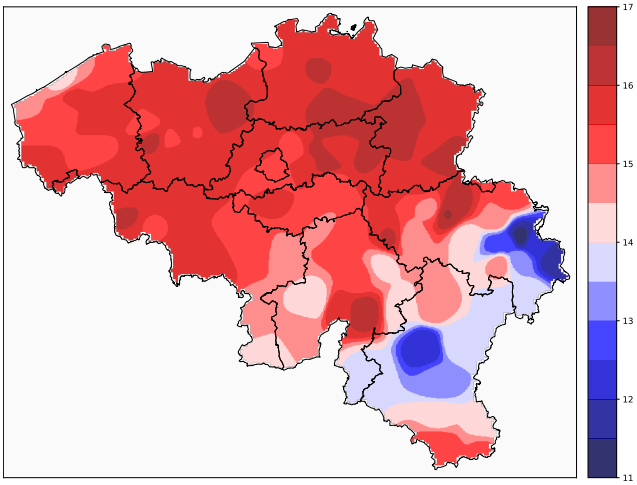
Table 5.2: RMSE value obtained in the TX dataset for each method

Similarly, the Figures 5.6 and 5.7 illustrate the results for the TN dataset. The RMSE values have also been compiled in the Table 5.3

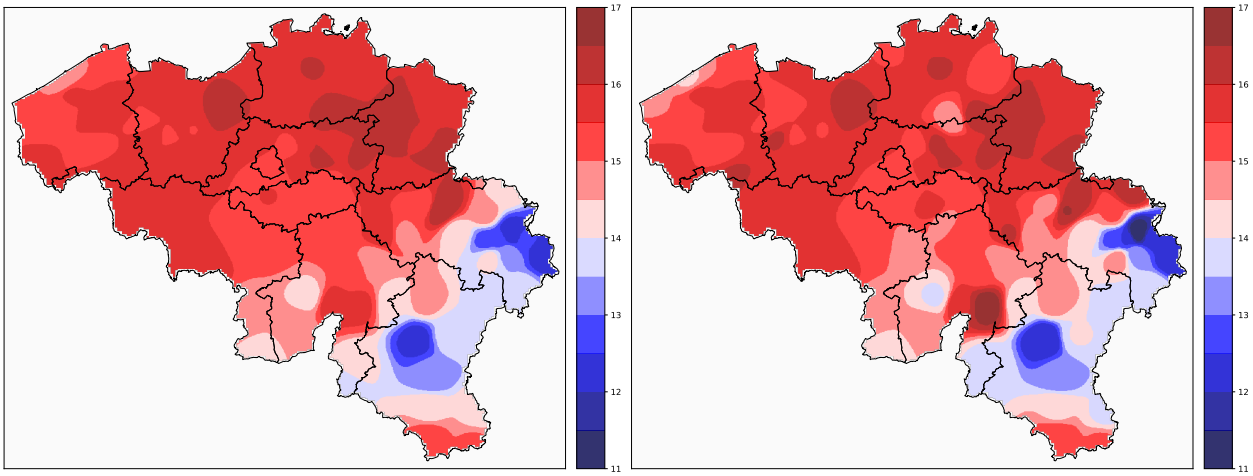
Methods	RMSE
IDW	1.138
PCA	1.214
LMaFit	1.126
OptSpace	1.378
RTRMC	0.932
SoftImpute	0.587
GRALS	0.828
TRMF	0.987

Table 5.3: RMSE value obtained in the TN dataset for each method

Comparing the results of the two tables, we observe that each algorithm has a larger RMSE for TN than for TX. Indeed, the maximum temperature is observed during the solar radiation period, while the minimum temperature is rather local, determined notably by the effects of valleys. These local effects for the minimum temperature are unpredictable and make the completion more complicated in this case.

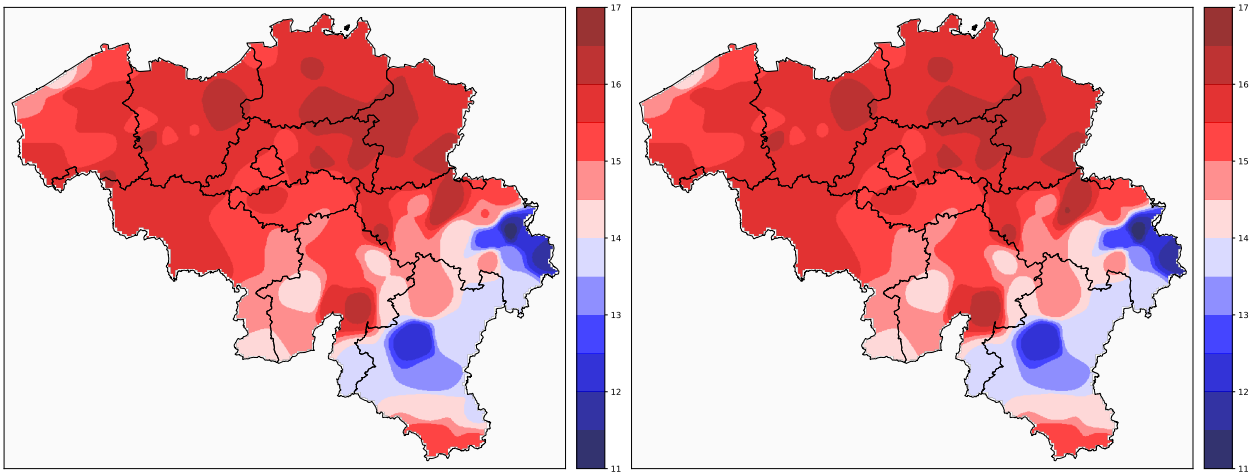


(a) Observed normals



(b) IDW : RMSE = 0.938

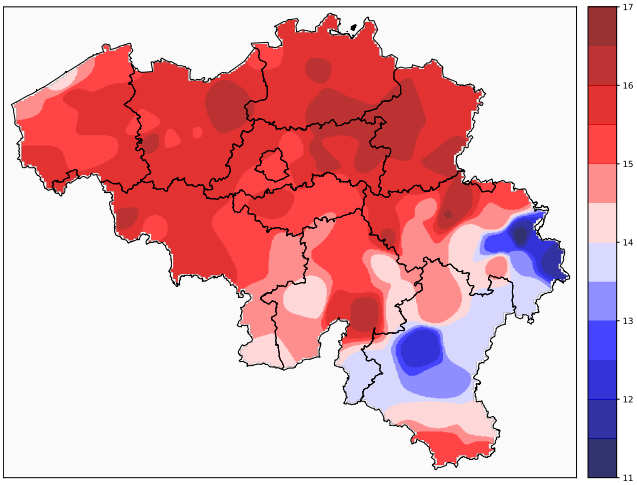
(c) PCA : RMSE = 1.14



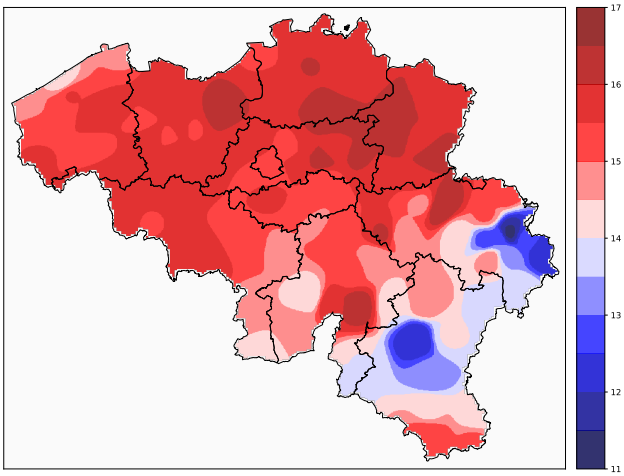
(d) LMaFiT : RMSE = 0.871

(e) SoftImpute : RMSE = 0.563

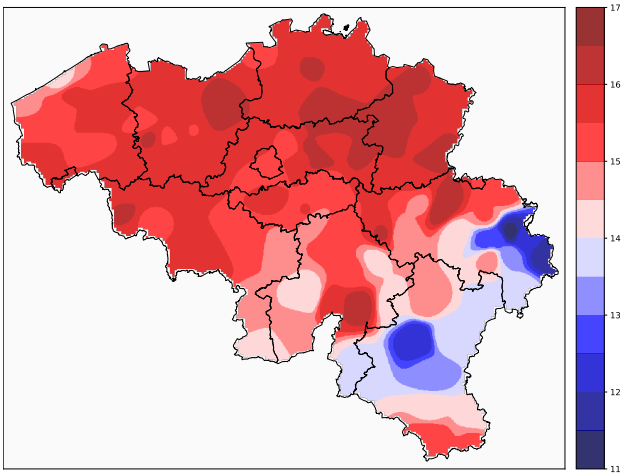
Figure 5.4: Annual TX Normals maps (1)



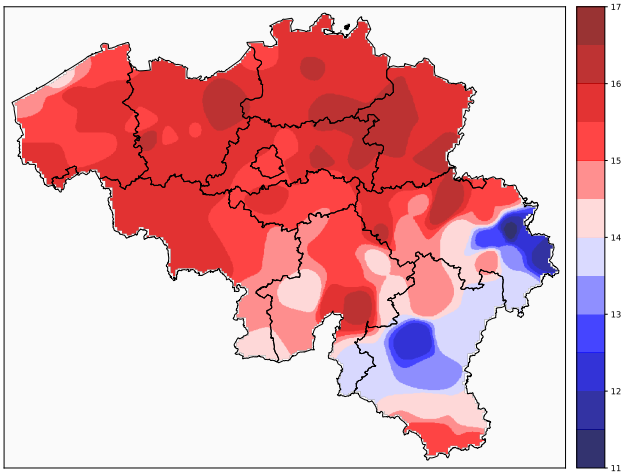
(a) Observed normals



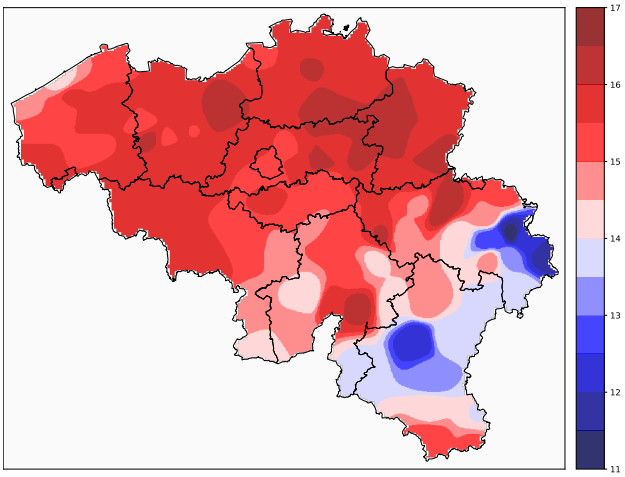
(b) OptSpace : RMSE = 1.32



(c) RTRMC : RMSE = 0.539

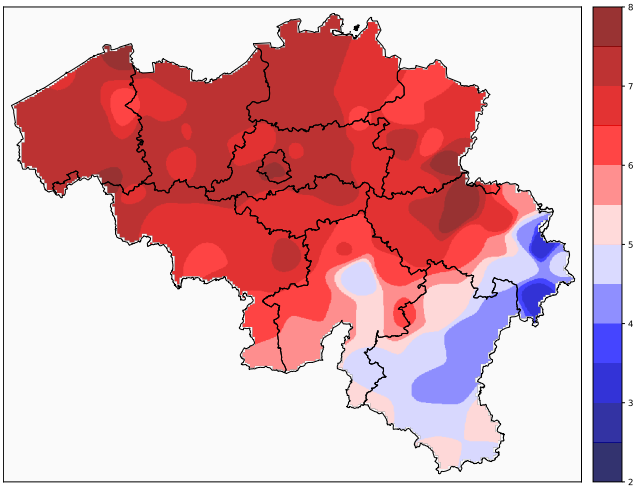


(d) GRALS : RMSE = 0.607

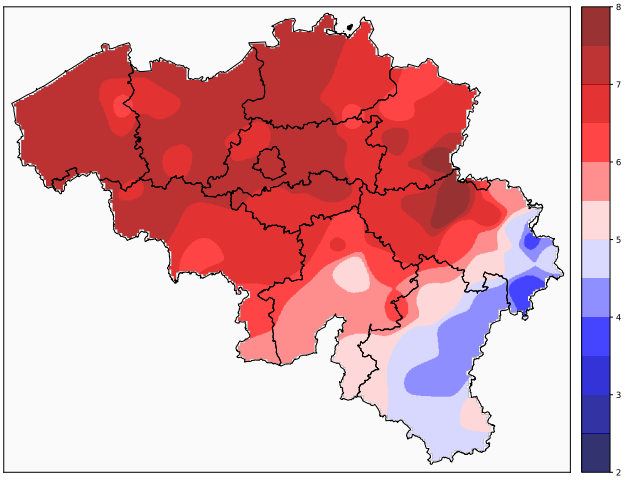


(e) TRMF : RMSE = 0.813

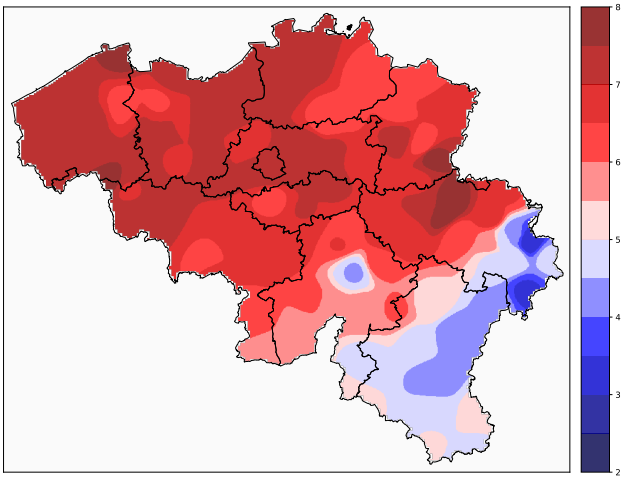
Figure 5.5: Annual TX Normals maps (2)



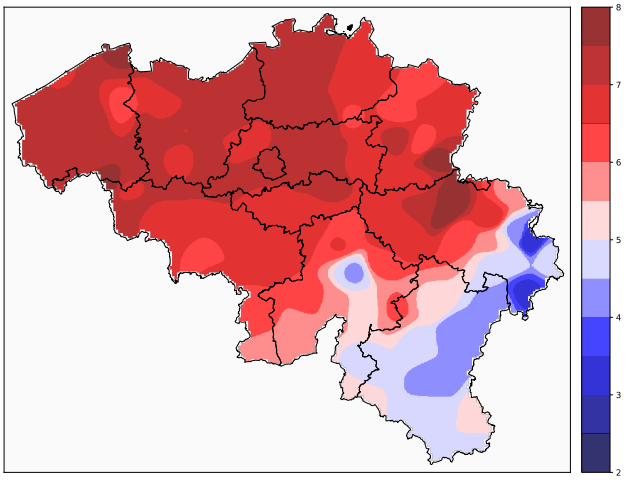
(a) Observed normals



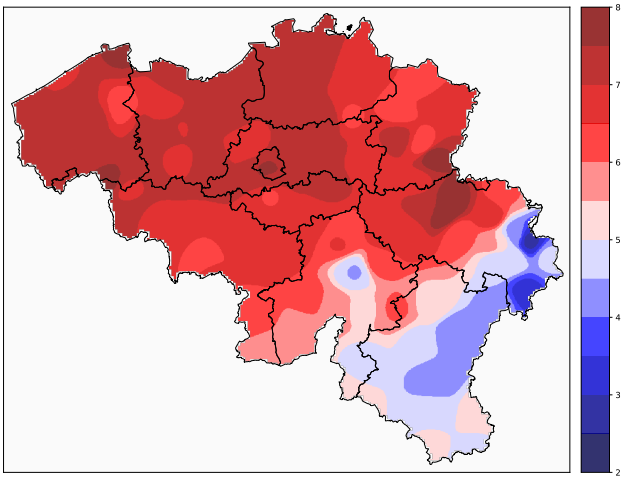
(b) IDW : RMSE = 1.138



(c) PCA : RMSE = 1.214

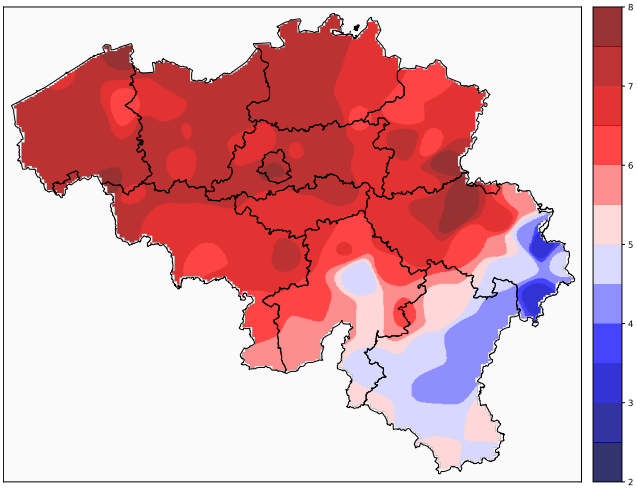


(d) LMaFiT : RMSE = 1.126

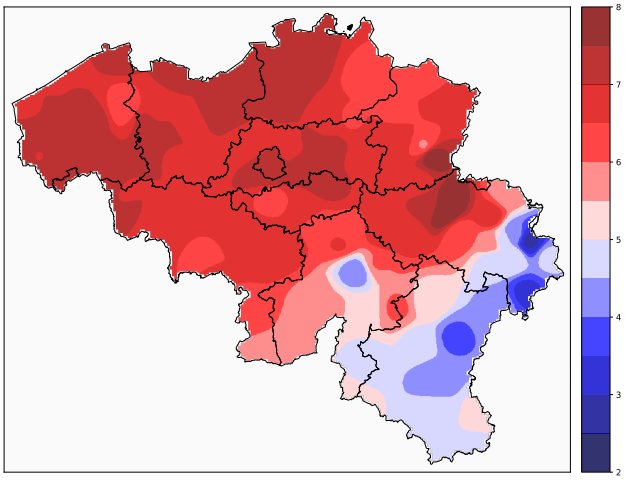


(e) SoftImpute : RMSE = 0.587

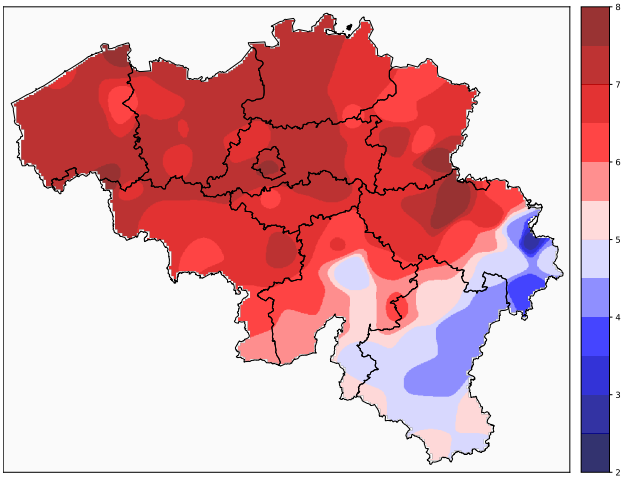
Figure 5.6: Annual TN Normals maps (1)



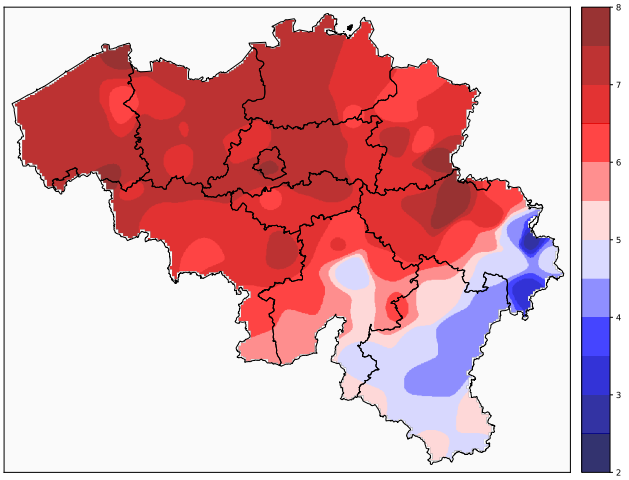
(a) Observed normals



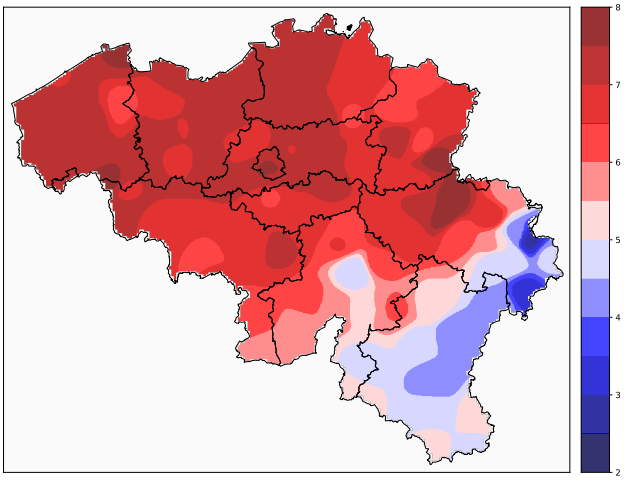
(b) OptSpace : RMSE = 1.378



(c) RTRMC : RMSE = 0.932



(d) GRALS : RMSE = 0.828



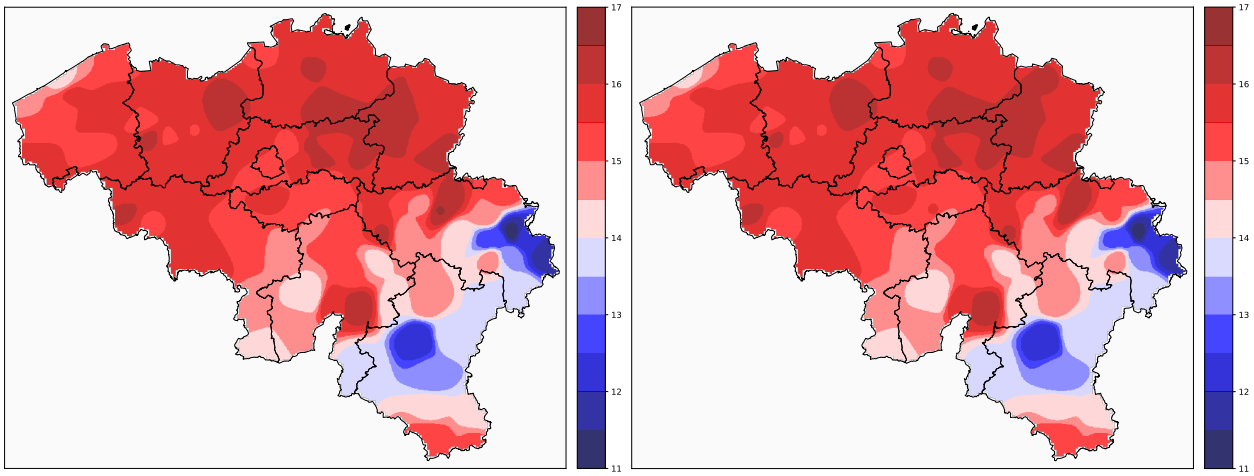
(e) TRMF : RMSE = 0.897

Figure 5.7: Annual TN Normals maps (2)

Summary

To conclude this data completion module, we have shown that matrix completion methods are effective! For both the TX and TN datasets, we have found at least one matrix completion method that performs better than the state of the arts.

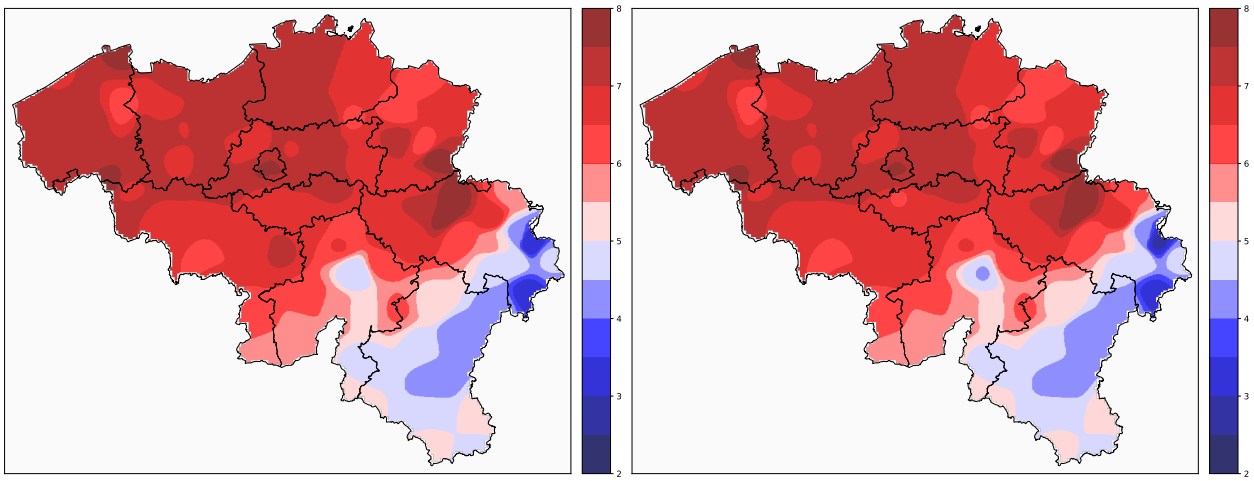
The most efficient method for the completion of the daily maximum temperature data is RTRMC with an RMSE almost twice as small as PCA and IDW.



(a) Original (b) RTRMC : RMSE = 0.539

Figure 5.8: Best method for the completion of the daily maximum temperature

For the completion of daily minimum temperature data, the best method is SoftImpute with an RMSE also almost twice as small as PCA and IDW.



(a) Original (b) SoftImpute : RMSE = 0.587

Figure 5.9: Best method for the completion of the daily minimum temperature

5.2 Error Detection & Error Correction

Among the 4 matrix completion methods tested, only RTRMC and GRALS gave us correct results. TRMF and OptSpace were not able to correct the errors at all. For this reason, only the results of these two first methods are presented here.

To generate the errors, we will add some non-Gaussian noise on a part of the entries. To do so, we add one realization of the following random variable

$$\mathcal{O} = \mathcal{S}_{\pm 1} \cdot \mathcal{U}(a, b) \tag{5.1}$$

where $\mathcal{S}_{\pm 1}$ is a random variable with equal probability to be equal to $+1$ or -1, while $\mathcal{U}(a, b)$ is the Uniform random variable with minimum value a and maximum value b.

In our experiments, we decided to fix $a = 0.5$ and $b = 10$ and the outlier are created uniformly at random with a probability of 5%.

The Figure 5.10 represents via several graphs all the results obtained for the detection and correction of errors. First of all, we have noticed that RTRMC and GRALS give exactly the same results, so the graphs concern both methods.

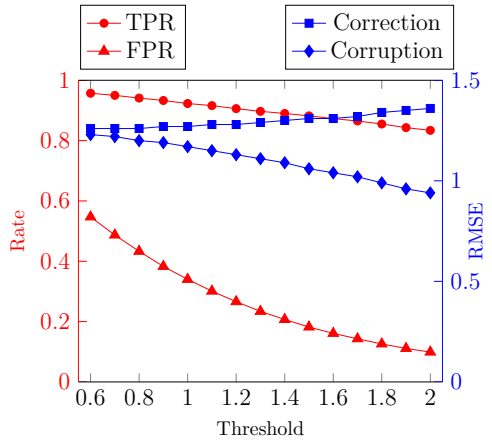
The graphs on the right represent for different values of rank r, the ROC curves obtained. The color of the curve indicates the threshold value. The graphs on the left allow to evaluate both the detection performance and the error correction. For these graphs, the left y-axis measures the detection by displaying the True Positive Rate and the False Positive Rate. The right y-axis measures correction by displaying the Correction RMSE and the Data Corruption RMSE.

As a reminder, we need to find the threshold that maximizes the TPR and minimizes the FPR, the correction RMSE and the corruption RMSE.

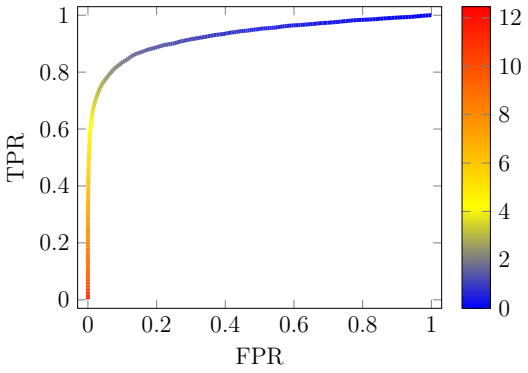
When we go from a very low rank value, like $r = 1$ to a rank value of $r = 5$, we can observe by comparing the ROC curves 5.10b and 5.10d, that this increase in rank has made the ROC curve go up, which is a very good thing! We can also observe differently this rise of the curve, in the graphs 5.10a and 5.10c, indeed, increasing the rank has made to considerably decrease the value of the FPR for the same threshold. Note also that in this case, increasing the rank has significantly decreased the RMSE of correction and corruption.

On the other hand, when we go from a rank value of $r = 5$ to $r = 10$, the ROC curves change very little, we can only note a decrease of the FPR for the same value of threshold. In this change of rank, the correction error has increased while the corruption error has decreased.

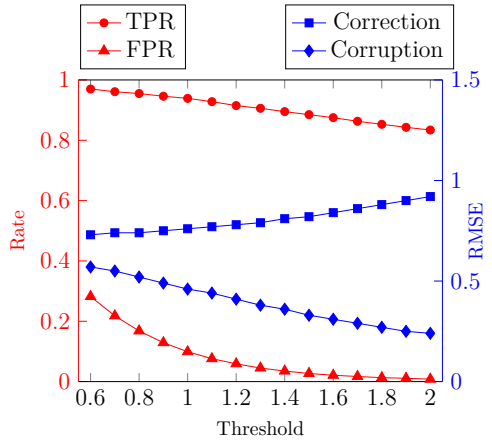
In view of these results, a rank value around 5 seems to give the best results. Even though, the FPR is smaller when the rank is bigger than 5, the error correction is clearly better. As explained in section 4.2.2, the choice of the threshold value is very arbitrary. It depends on the criterion imposed by the person who wants to correct his values. If this person wants to corrupt his data as little as possible even if it means less error correction, then he will tend to choose a larger threshold value. On the contrary, if he wants to take the risk of corrupting his data a little but correcting the errors better, then he will tend to choose a lower threshold value.



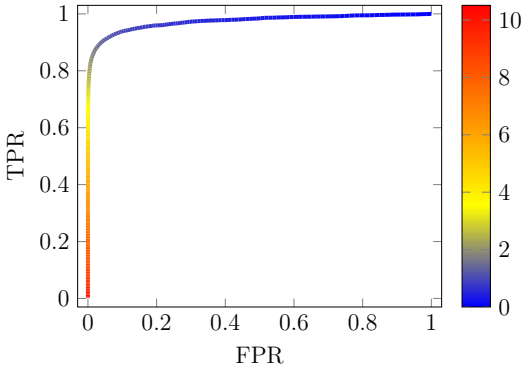
(a) Detection and Correction evaluation with $r = 1$



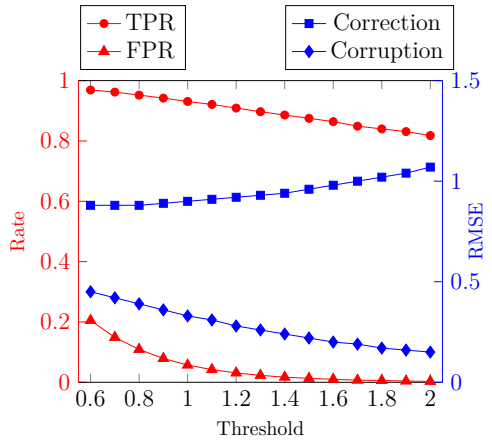
(b) ROC curve with $r = 1$



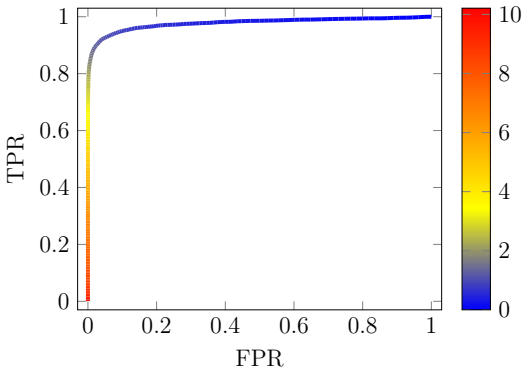
(c) Detection and Correction evaluation with $r = 5$



(d) ROC curve with $r = 5$



(e) Detection and Correction evaluation with $r = 10$



(f) ROC curve with $r = 10$

Figure 5.10: Correction and detection results for RTRMC/GRALS on the TX dataset with different rank values

Conclusion

Summary

In this master's thesis, we have proposed two benchmark models to evaluate the performance of matrix completion methods to deal with the completion problem as well as the error detection and correction problem.

The benchmark of the matrix completion method first allowed us to parameterize the algorithms by training them on 10 years of Belgian climate data. Then, we tested the methods in a completely new case spread over 5 years of data, we created for each method the map of climate normals that it generates after completing the missing data. Whether it is for the completion of the daily minimum or maximum temperature data, we succeeded in concluding that the matrix completion methods work very well and even better than the state of the art.

For error detection and correction, we have proposed a performance evaluation model. This model uses the concept of ROC curves to evaluate the error detection and the concept of RMSE to evaluate the error correction. Although it was not possible to compare the matrix completion methods with the state of the art, since the latter requires a human intervention that cannot be simulated. It is clearly conceivable that matrix completion methods are not the most efficient for this task.

Research Perspectives

- In our approach, we analyzed separately the minimum and maximum daily temperature dataset. Another approach would be to use a tensor to treat the problem in a multivariate way. The multivariate approach could contain temperature, precipitation, wind speed, pressure and more at the same time!
- The GRALS method was originally designed to address the collaborative filtering problem (i.e. Netflix problem). We then imagined a way to design graphs for time series. The possible improvements are in the weights of the edges. The choices we made are rather simple and surely improvable, considering the results we obtained this method has a lot of potential!
- For the parameterization of the methods in the data completion problem, the decision was made to use the GridSearch idea. This approach has the advantage of being simple but has the disadvantage of being restrictive on the choice of values. Another possible approach would have been the Random Search. This approach still uses the GridSearch notion of a grid but it creates a much more complete grid, with many more nodes. However, unlike

GridSearch, this one does not test all the nodes but only a part of the nodes is selected with a probability p. This alternative has the advantage of being able to handle a much larger value space than gridsearch but has the disadvantage of generating a large variance in the results, as the choices are random.
Another alternative called Bayesian Hyperparameter Optimization exists. This method aims to overcome one of the common defaults of gridsearch and random grid search. The latter are memoryless and therefore sometimes spend time evaluating "bad" parameters. The Bayesian approach keep track of past evolution results which they use to form a probabilistic model mapping parameters to a probability of a score on the objective function.

- In order not to complicate the analysis, the decision was made to separate the data completion problem from the error detection and correction problem. Obviously, in practice the two problems are strongly linked. Merging these two problems is not an easy task, because for the method parameterization task, a single metric is needed that allows at the same time to evaluate the completion of missing data and the detection and correction of measurement errors.

Appendix A

Dataset

A.1 Annual Availability 1991-2020

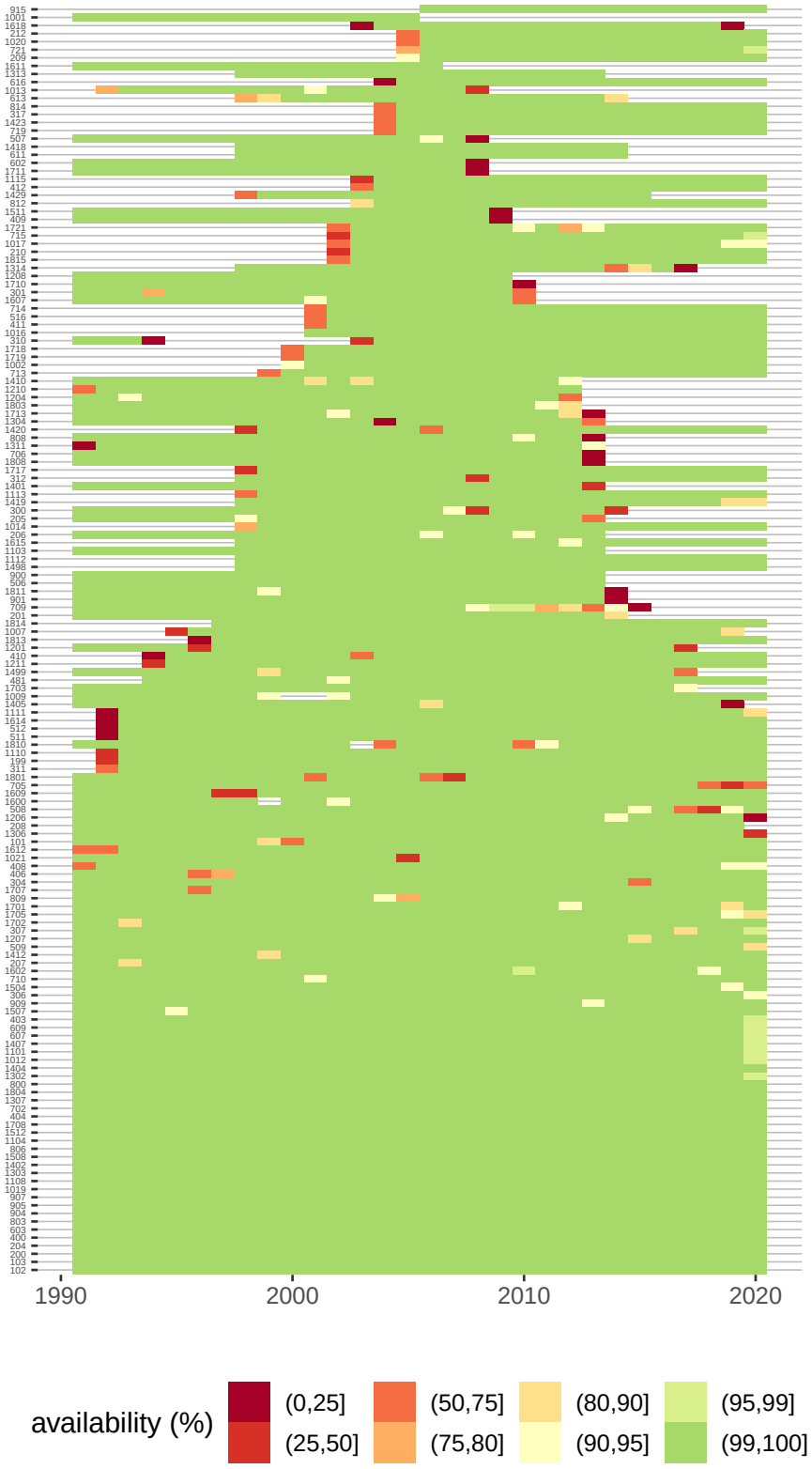


Figure A.1: Data availability for 157 stations over the period 1991-2020

A.2 Stations map

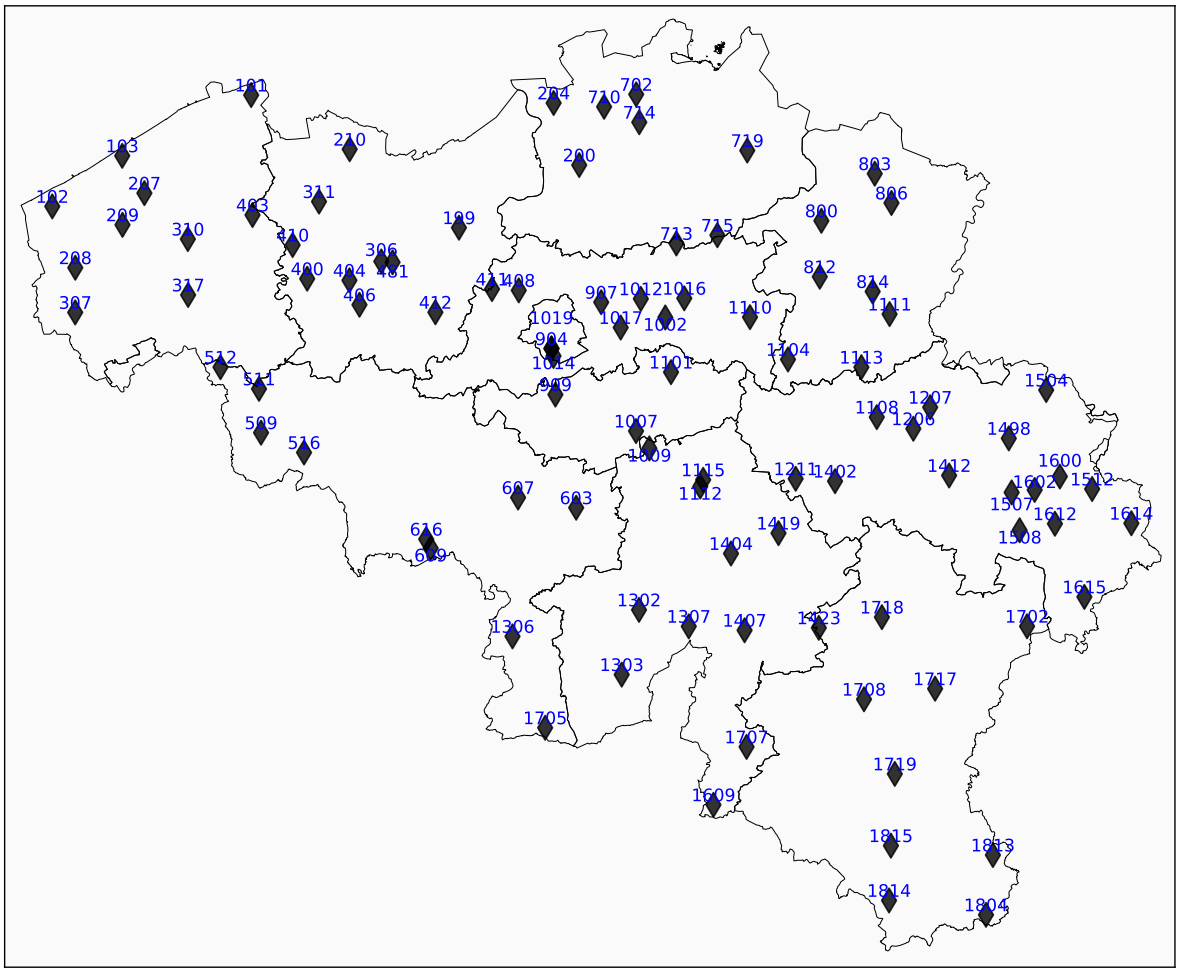


Figure A.2: Map of the stations with their associated code

A.3 Stations table

Name	Code	Lat (N)/Lon (E)
Flandre Occidentale		
KNOKKE-HEIST	101	51°20'10"/3°19'31"
KOKSIJDE	102	51°05'17"/2°39'08"
MIDDELKERKE	103	51°12'01"/2°53'14"
MOERE	207	51°07'15"/2°57'57"
POLLINKHOVE	208	50°57'26"/2°44'09"
BEERST	209	51°03'06"/2°53'36"
POPERINGE	307	50°51'40"/2°44'21"
LICHTERVELDE	310	51°01'26"/3°07'02"
BEITEM AWS	317	50°54'14"/3°07'18"
WINGENE	403	51°04'41"/3°20'11"
Flandre Orientale		
ZELE	199	51°03'21"/4°02'29"
BASSEVELDE	210	51°13'21"/3°39'56"
LEMBERGE	306	50°58'54"/3°46'39"
ZOMERGEM	311	51°06'32"/3°33'46"
KRUISHOUTEM	400	50°56'34"/3°31'33"
SEMMERZAKE	404	50°56'26"/3°40'10"
SINT-MARIA-LATEM	406	50°53'17"/3°42'17"
MEIGEM	410	51°00'53"/3°28'29"
HELDERGEM	412	50°52'24"/3°57'47"
MELLE AWS	481	50°58'49"/3°48'57"
Anvers		
DEURNE	200	51°11'29"/4°27'06"
Stabroek	204	51°19'29"/4°21'49"
BRECHT	702	51°20'35"/4°38'50"
BRASSCHAAT	710	51°19'00"/4°32'13"
WESTMALLE	714	51°17'00"/4°39'27"
BLAUBERG	715	51°02'23"/4°55'23"
RETIE AWS	719	51°13'17"/5°01'38"
Brabant Flamand		
BRUSSEGEM	408	50°55'17"/4°14'46"
ASSE-TER-HEIDE	411	50°55'26"/4°09'18"
BEGIJNENDIJK	713	51°01'21"/4°46'59"
ZAVENTEM	907	50°53'47"/4°31'36"
KORBEEK-LO	1002	50°51'48"/4°44'40"
HERENT	1012	50°54'13"/4°39'41"
SINT-PIETERS-RODE	1016	50°54'16"/4°48'33"
LEEFDAAL	1017	50°50'31"/4°35'34"
RANSBERG	1110	50°51'47"/5°01'56"

Name	Code	Lat (N)/Lon (E)
Brabant-Wallon		
BRAINE-L'ALLEUD	909	50°41'51" /4°22'17"
BLANMONT	1007	50°37'08" /4°38'39"
BEAUVECHAIN	1101	50°44'44" /4°45'47"
Namur		
ERNAGE AWS	1009	50°34'54" /4°41'21"
VEDRIN	1112	50°29'57" /4°51'37"
DAUSSOULX	1115	50°30'49" /4°52'13"
ANGLEUR	1206	50°37'09" /5°34'54"
FLORENNES	1302	50°14'04" /4°39'10"
DOURBES AWS	1303	50°05'43" /4°35'40"
HASTIERE	1307	50°11'55" /4°49'10"
CRUPET	1404	50°21'17" /4°57'45"
HOUYET	1407	50°11'22" /5°00'23"
SOREE	1419	50°23'52" /5°07'22"
BIEVRE	1707	49°56'19" /5°00'37"
Liège		
BIERSET	1108	50°35'43" /5°27'31"
LIEGE-MONSIN	1207	50°39'53" /5°38'23"
BEN-AHIN	1211	50°30'52" /5°10'56"
LOUVEIGNE	1412	50°31'03" /5°42'00"
STEMBERT	1498	50°35'42" /5°54'13"
WALHORN	1504	50°41'50" /6°02'00"
SPA-LA SAUVENIERE	1507	50°28'43" /5°54'35"
STAVELOT	1508	50°23'49" /5°56'03"
ELSENBORN	1512	50°28'56" /6°10'52"
MONT RIGI AWS	1600	50°30'38" /6°04'24"
HOCKAI	1602	50°28'59" /5°59'15"
SUGNY	1609	49°48'52" /4°53'56"
GEROMONT	1612	50°24'34" /6°03'12"
MURRINGEN	1614	50°24'24" /6°18'38"
NEIDINGEN	1615	50°15'02" /6°08'49"
Hainaut		
TOURNAI	509	50°36'37" /3°22'36"
HERINNES	511	50°42'16" /3°22'03"
MOUSCRON	512	50°44'59" /3°14'09"
VEZON	516	50°34'08" /3°31'22"
GOSSELIES	603	50°27'15" /4°26'31"
LA HESTRE	607	50°28'33" /4°14'46"
QUEVY-LE-PETIT	609	50°21'57" /3°57'10"
BOUGNIES	616	50°23'07" /3°56'15"

Limbourg			SIVRY	1306	50°10'38" /4°13'44"
KOERSEL	800	51°04'08"/5°16'42"	STREE	1402	50°30'30" /5°18'54"
KLEINE-BROGEL	803	51°10'07"/5°27'45"	FORGES	1705	49°58'51" /4°20'20"
MEEUWEN	806	51°06'19"/5°31'06"	GIVRY	1717	50°03'33" /5°38'29"
KURINGEN	812	50°56'55"/5°16'15"			
DIEPENBEEK	814	50°54'55"/5°27'01"	**Luxembourg**		
GORSEM	1104	50°46'17"/5°09'36"	BILZEN	1111	50°51'59" /5°30'24"
BILZEN	1111	50°51'59"/5°30'24"	HUMAIN AWS	1423	50°11'37" /5°15'18"
LAUW	1113	50°45'11"/5°24'31"	GOUVY	1702	50°11'24" /5°57'10"
			SAINT-HUBERT	1708	50°02'19" /5°24'16"
			CHEOUX	1718	50°12'53" /5°28'1"
			MASSUL	1719	49°52'38" /5°30'11"
Bruxelles-Capitale			AUBANGE	1804	49°34'16" /5°47'54"
UCCLE-UKKEL	904	50°47'47"/4°21'28"	FRASSEM	1813	49°41'58" /5°49'29"
UCCLE VIVAQUA	1014	50°46'50"/4°21'53"	MEIX-DEVANT-VIRTON	1814	49°36'19" /5°28'42"
UCCLE-UKKEL AWS	1019	50°47'48"/4°21'28"	ROSSIGNOL	1815	49°43'23" /5°29'13"

Table A.1: List of the available stations in Belgium

Appendix B

Experiments

B.1 Training results TX

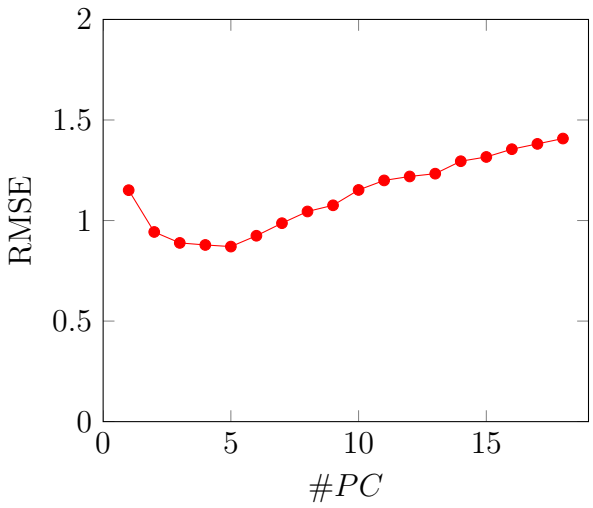
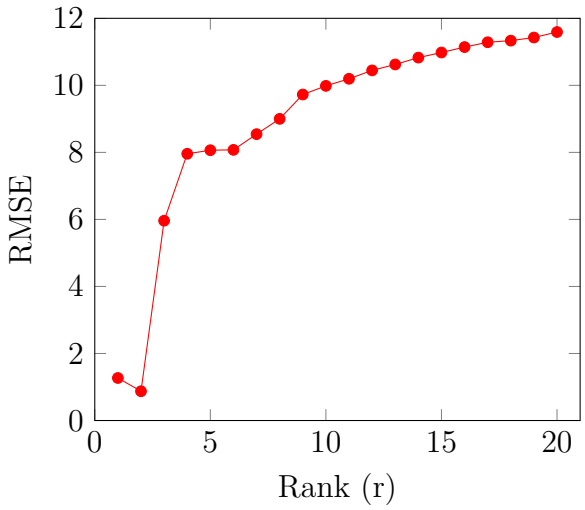
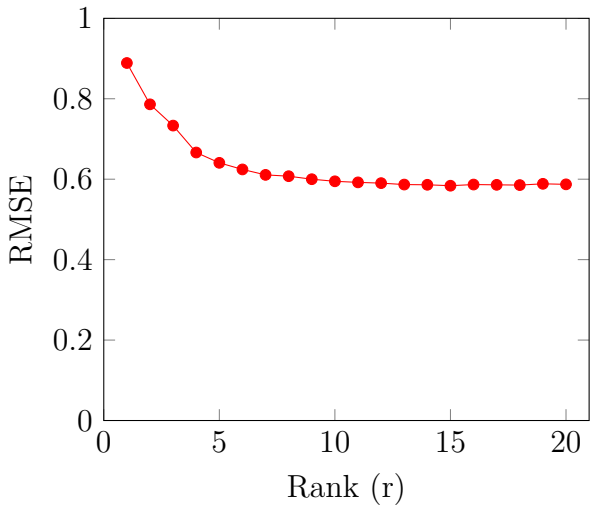


Figure B.1: RMSE for PCA on the TX dataset



(a) RMSE for LMaFit on the TX dataset



(b) RMSE for SoftImpute on the TX dataset

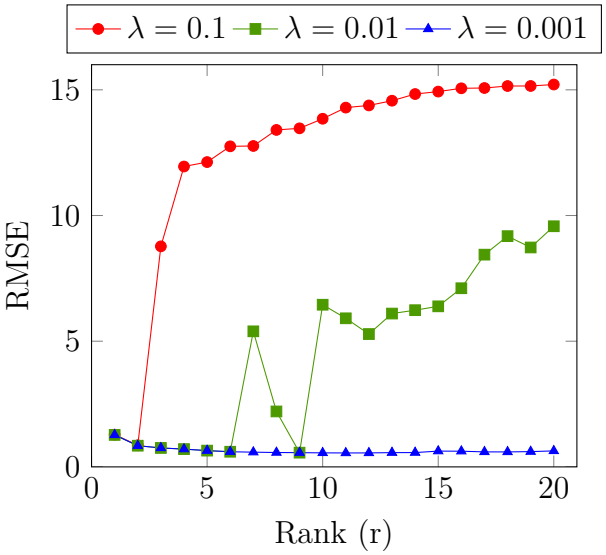
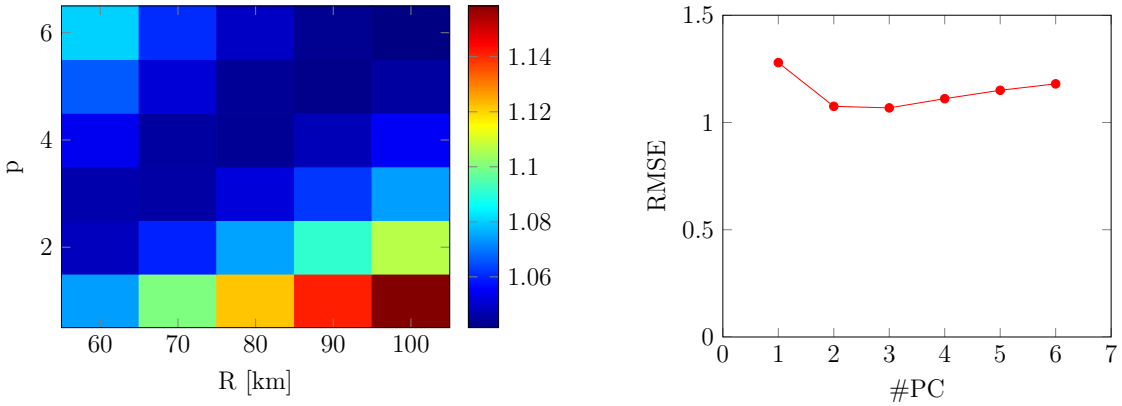


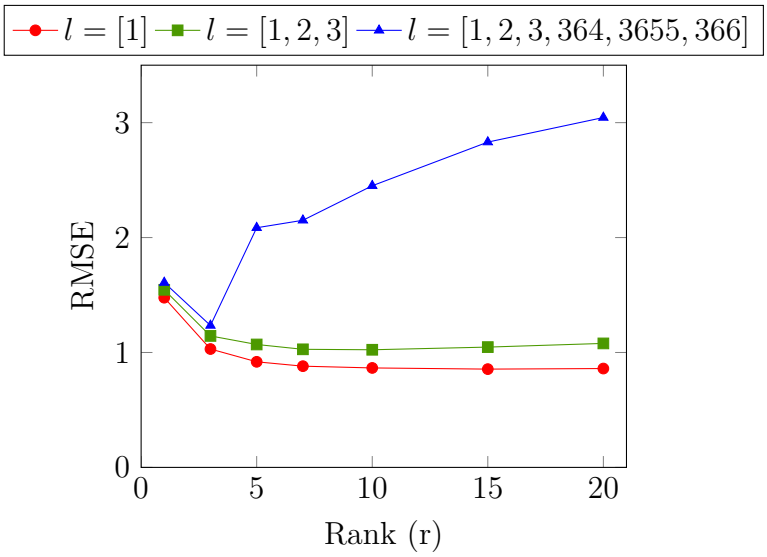
Figure B.3: RMSE for RTRMC on the TX dataset

B.2 Training results TN

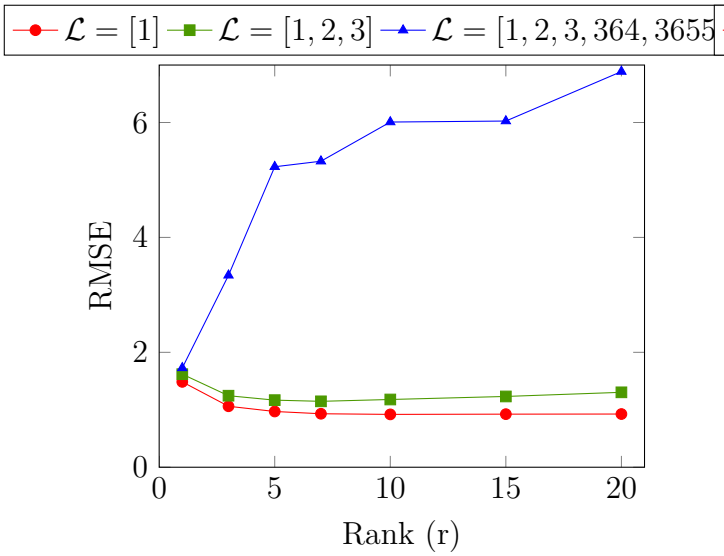


(a) IDW with the Frank-Little weighting function (b) IDW with the Shepard weighting function

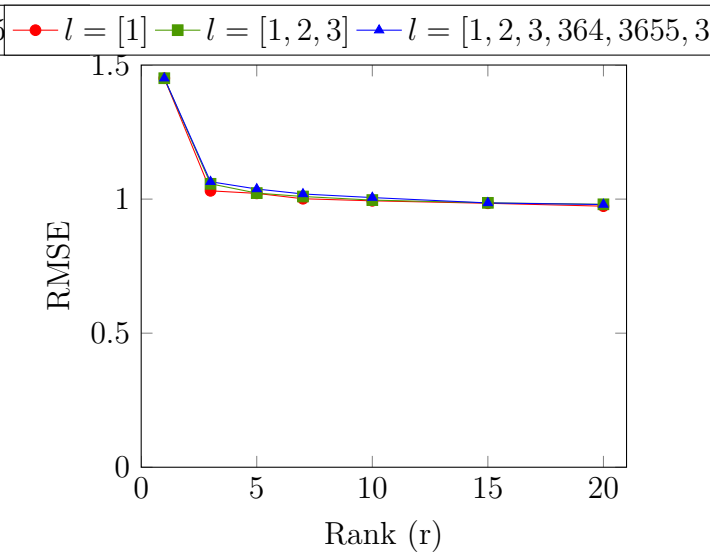
Figure B.4: RMSE for IDW on the TN dataset



(a) TRMF with $\lambda = [0.2, 1000, 100]$



(b) TRMF with $\lambda = [0.5, 625, 24]$



(c) TRMF with $\lambda = [0.75, 0.75, 0.75]$

Figure B.5: RMSE for TRMF on the TN dataset

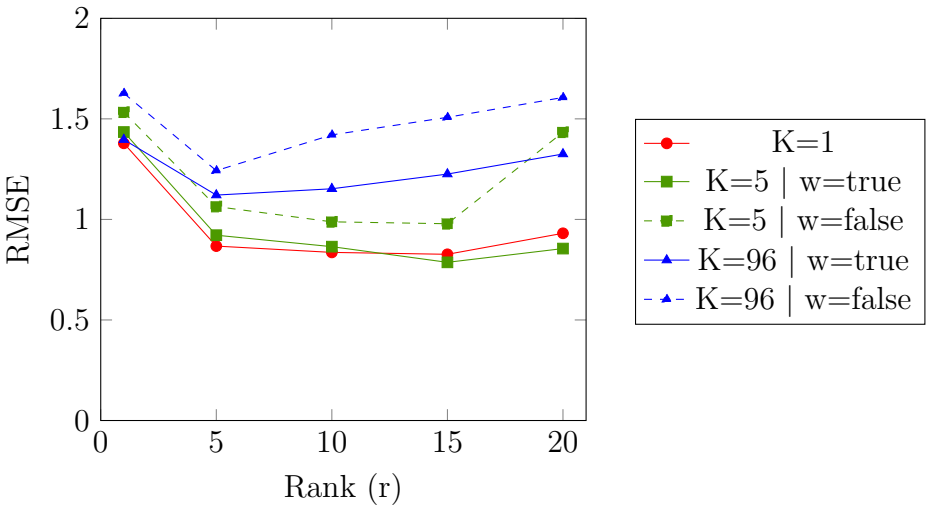


Figure B.6: RMSE for GRALS on the TN dataset

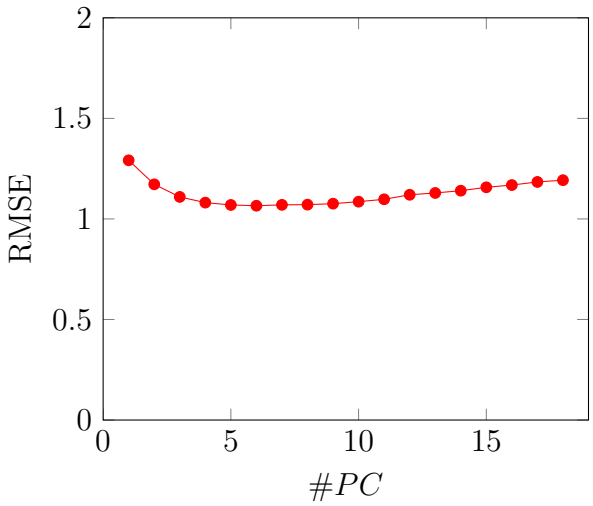
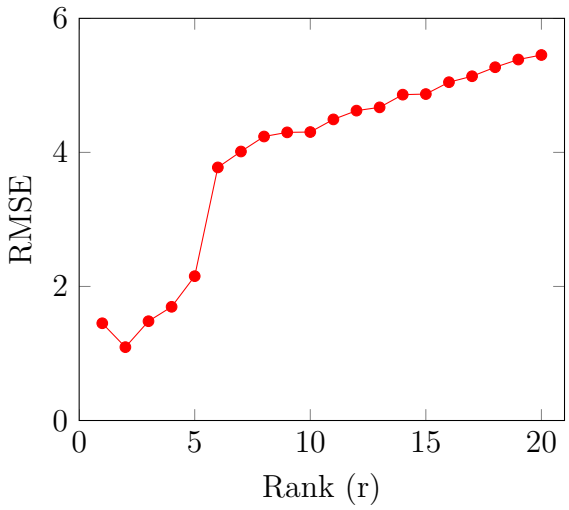
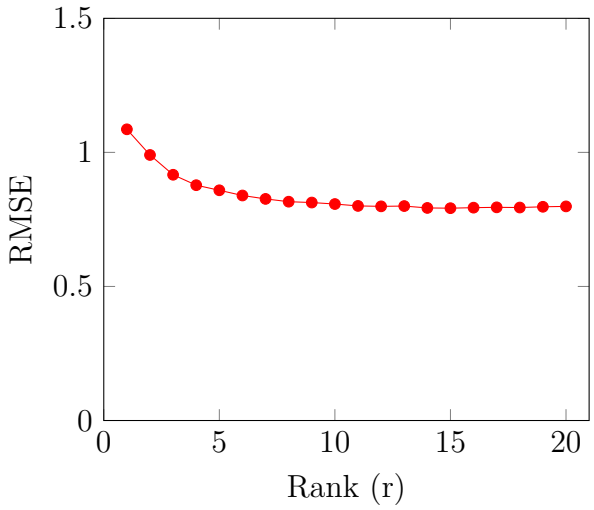


Figure B.7: RMSE for PCA on the TN dataset



(a) RMSE for LMaFit on the TN dataset



(b) RMSE for SoftImpute on the TN dataset

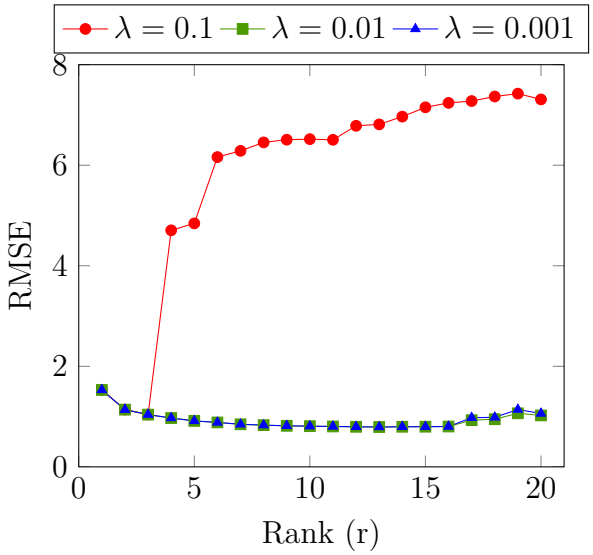


Figure B.9: RMSE for RTRMC on the TN dataset

B.3 Monthly Availability for the Test part

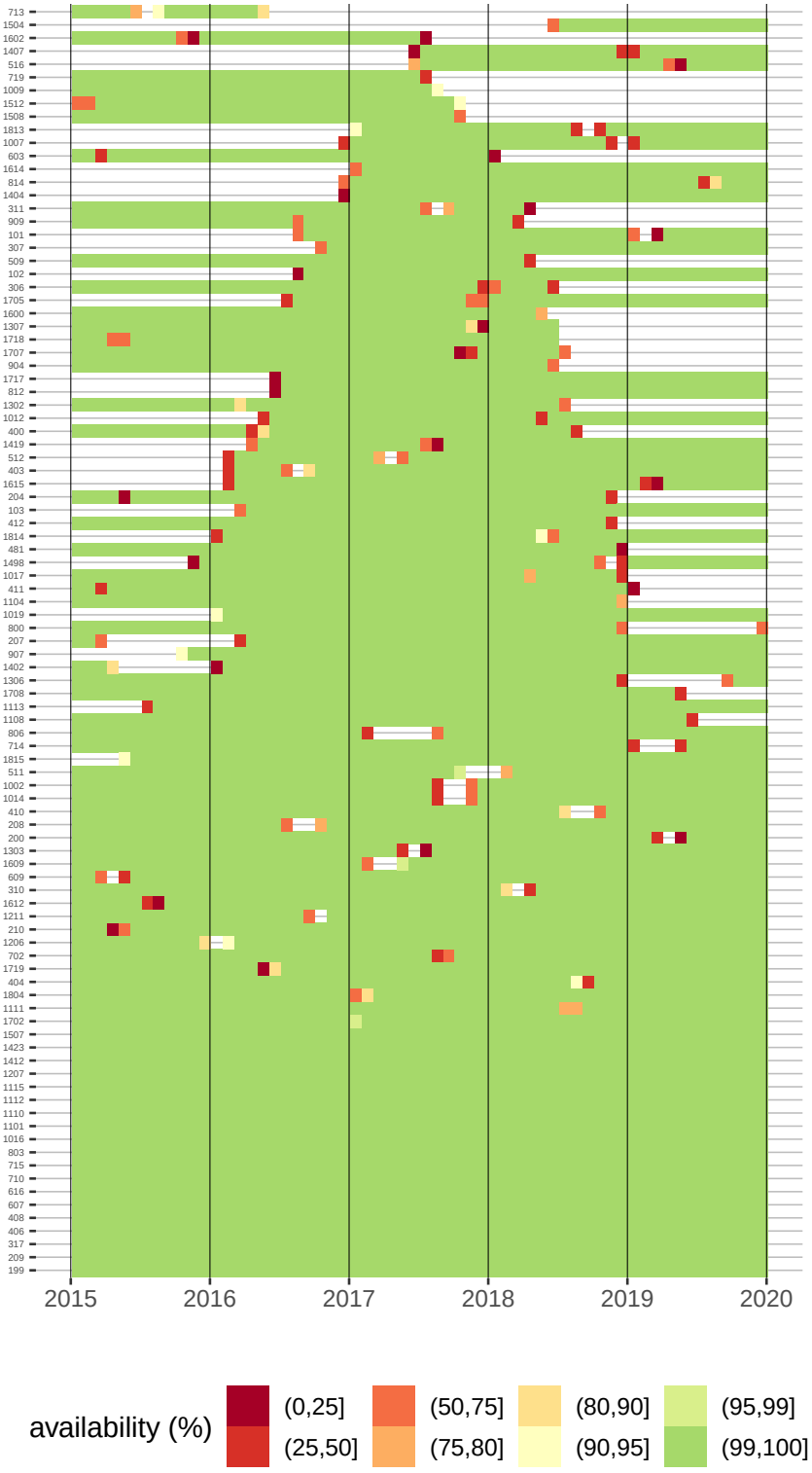


Figure B.10: Data availability for 97 stations over the period 2015-2019 for the final test of the completion module

Bibliography

[1] C. Delvaux, R. Ingels, V. Vrábeĺ, M. Journée, and C. Bertrand, "Quality control and homogenization of the belgian historical temperature data," *International Journal of Climatology*, vol. 39, 09 2018.

[2] A. B. of Meteorology, "Observation of air temperature," 2018. [Online]. Available: http://www.bom.gov.au/climate/cdo/about/airtemp-measure.shtml

[3] R. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," 11 2007, pp. 43–52.

[4] W. M. Organization, "WMO guidelines on the calculation of climate normals," 2017.

[5] O. E. Tveito, S. Aniskevica, J. Cappelen, E. Engström, H. M. Gjelten, C. D. Jensen, P. Jokinen, E. K. Kuya, C. Lussana, A. Mäkelä, K. Mändla, K. Vint, L. Wern, and V. Zandersons, "Climnorm - temperature data set, gap filling methods and regional analysis to prepare new climate normals," Tech. Rep., 2020.

[6] M. Jing and J. Wu, "Fast image interpolation using directional inverse distance weighting for real-time applications," *Optics Communications*, vol. 286, p. 111–116, 01 2013.

[7] Y. He, Z. Binwu, and E. Yao, "Weighted inverse minimum spanning tree problems under hamming distance," *J. Comb. Optim.*, vol. 9, pp. 91–100, 02 2005.

[8] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[9] H. Hisdal and O. E. Tveito, "Extension of runoff series using empirical orthogonal functions," *Hydrological Sciences Journal*, vol. 38, no. 1, pp. 33–49, 1993.

[10] C. Delvaux, M. Journée, and C. Bertrand, "The forbio climate data set for climate analyses," *Advances in Science and Research*, vol. 12, no. 1, pp. 103–109, 2015. [Online]. Available: https://asr.copernicus.org/articles/12/103/2015/

[11] G. Sciuto, B. Bonaccorso, A. Cancelliere, and G. Rossi, "Probabilistic quality control of daily temperature data," *International Journal of Climatology*, vol. 33, pp. 1211–1227, 04 2013.

[12] E. Candès and B. Recht, "Exact matrix completion via convex optimization," *Communications of the ACM*, vol. 9, pp. 717–772, 11 2008.

[13] Y. Lu and J. Yang, "Notes on low-rank matrix factorization," 2016.

[14] J. P. Haldar and D. Hernando, "Rank-constrained solutions to linear matrix equations using powerfactorization," *IEEE Signal Processing Letters*, vol. 16, no. 7, pp. 584–587, 2009.

[15] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in Neural Information Processing System*, vol. 14, 04 2002.

[16] ——, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, pp. 1373–, 06 2003.

[17] D. Yang, Z. Ma, and A. Buja, "A sparse singular value decomposition method for high-dimensional data," *Journal of Computational and Graphical Statistics*, vol. 23, no. 4, pp. 923–942, 2014.

[18] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, vol. 4, 12 2012.

[19] L. Grippo, "On the convergence of the block nonlinear gauss-seidel method under convex constraints," *Operations Research Letters*, vol. 26, pp. 127–136, 04 2000.

[20] R. Keshavan and S. Oh, "Optspace : A gradient descent algorithm on the grassman manifold for matrix completion," vol. 910, 10 2009.

[21] W. Dai, E. Kerman, and O. Milenkovic, "A geometric approach to low-rank matrix completion," *IEEE Transactions on Information Theory*, vol. 58, 06 2010.

[22] N. Boumal and P.-A. Absil, "RTRMC: A riemannian trust-region method for low-rank matrix completion," *NIPS*, 01 2011.

[23] J.-F. Cai, E. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, pp. 1956–1982, 03 2010.

[24] D. Donoho, I. Johnstone, and D. Picard, "Wavelet shrinkage: Asymptopia?" *Journal of the Royal Statistical Society, Series B*, vol. 57, 12 1999.

[25] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.

[26] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," *Journal of Machine Learning Research - JMLR*, vol. 11, 06 2009.

[27] V. Koltchinskii, A. Tsybakov, and K. Lounici, "Nuclear norm penalization and optimal rates for noisy low rank matrix completion," *Annals of Statistics - ANN STATIST*, vol. 39, 11 2010.

[28] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015, pp. 2107–2115. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/f4573fc71c731d5c362f0d7860945b88-Paper.pdf

[29] H. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization," 2015.

[30] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *Journal of machine learning research : JMLR*, vol. 11, pp. 2287–2322, 03 2010.

[31] L. Lusted, "Signal detectability and medical decision-making," *Science (New York, N.Y.)*, vol. 171, pp. 1217–9, 04 1971.

[32] G. Campbell, "Advances in statistical methodology for the evaluation of diagnostic and laboratory tests," *Statistics in Medicine*, vol. 13, no. 5-7, pp. 499–508, 1994. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4780130513

[33] D. Shapiro, "The interpretation of diagnostic tests," *Statistical methods in medical research*, vol. 8, pp. 113–34, 07 1999.

[34] T. A. Lasko, J. G. Bhagwat, K. H. Zou, and L. Ohno-Machado, "The use of receiver operating characteristic curves in biomedical informatics," *Journal of Biomedical Informatics*, vol. 38, no. 5, pp. 404–415, 2005, clinical Machine Learning. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046405000171

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**