MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation : Technologies de l'information et de la communication (TIC)

Using spatial and temporal correlations to assess the trustworthiness of IoT noise sensors

Fait par

Ludovic Gindre

Sous la direction de
Prof. Nabil Abdennadher
Dans l'institut inIT

Lausanne, HES-SO//Master, le 18.06.2020

Accepté par la HES-SO//Master (Suisse, Lausanne) sur proposition de

Prof. Nabil Abdennadher, conseiller du projet d'approfondissement

Lausanne, le 18.06.2020

Prof. Nabil Abdennadher	Prof. Nabil Abdennadher
Conseiller	Responsable de la filière

Contents

Glossary

API Application Programming Interface. 24, 25

EEA European Environment Agency. 7

FFT Fast Fourier Transform. 17–19

FT Fourier Transform. 6, 11, 19–21, 23, 24, 32

IoT Internet of Things. 7–9, 12

MAE Mean Absolute Error. 24

MSE Mean Squared Error. 21–24

OCEV cantonal environment office. 13, 32

REST Representational State Transfer. 24, 25

RFT Reference Fourier Transform. 18, 19, 21, 23

RTS Reference time-series. 15, 17–20, 23

SABRA service of air, noise, and non-ionizing radiations. 13, 32

SCD smart city device. 7, 8

TF trust factor. 6, 9–11, 32

TFS trust factor score. 9, 10

TM trust model. 8–10, 15, 32

TS time-series. 12, 15–17, 19, 20, 23, 25–28

UN United Nations. 6, 7, 32

WHO World Health Organization. 7

Abstract

Road traffic is the main source of environmental noise As it is the main source of environmental noise, people living in urban areas are much more exposed to these health risks. According to United Nations (UN)'s 2018 revision of world urbanization prospects [1], in 2018 the percentage of the world's population living in urban areas was 56%. By 2050, this percentage will be 68%. In order to reduce noise exposures, it is first necessary to understand how noise propagates in cities This paper proposes a partial solution to address the smart city noise sensors data integrity problem. Our solution is integrated into the generic trust model framework in [2]. This paper proposes a solution to build a signature trust factor (TF). Our solution starts with a filter using Fourier Transform (FT) to address the data integrity problem smart city devices suffer. Then we extract a signature from the cleaned data and run a clustering algorithm on these signatures to group them and extract knowledge about these contexts out of these signatures.

Introduction

Noise pollution

In the past few years, the impact of environmental noise on health has become a real concern. According to reports from World Health Organization (WHO) [3] and European Environment Agency (EEA) [4], noise pollution is a major environmental health problem in Europe that causes at least 10 000 cases of premature death in Europe each year. This report states that almost 20 million adults are annoyed and a further 8 million suffer sleep disturbance due to environmental noise. Environmental noise is also the source of 900'000 cases of hypertension each year. With the fact that road traffic is the main source of environmental noise. As it is the main source of environmental noise, people living in urban areas are much more exposed to these health risks. According to United Nations (UN)'s 2018 revision of world urbanization prospects [1], in 2018 the percentage of the world's population living in urban areas was 56%. By 2050, this percentage will be 68%.

The emergence of Internet of Things (IoT) brings a new opportunity to monitor noise pollution. It allows policymakers to take initiatives to better understand noise exposures in urban areas in order to minimize health risks to the population. IoT is a very broad term that includes all "things" connected to the internet. Among the different fields of application of IoT, smart cities are of particular interest to us. Smart cities are urban areas in which smart city devices (SCDs) are deployed at a city scale to collect data. SCDs are equipped with one or more sensors and/or actuators in order to act on the real world. By deploying this kind of smart city infrastructure, it is, therefore, possible to collect noise recordings and analyze them in order to better understand how noise affects the people living in a city. This precious information will then enable measures to be taken to reduce the health risks for these people.

SCD can take all forms and have a multitude of different tasks to perform depending on their field of application. What all these devices have in common is that they are very often subject to resource constraints such as a limited battery, small communication channels, or low computing capacities. These SCDs are most often composed of sensors such as microphones, thermometers, or any other type

of sensor that allow them to collect data. SCDs are also able to communicate via different technologies such as LoRa, 4G among others, but their computational resource limitations could prevent them from using encryption algorithms for these communications. In addition to all these intrinsic constraints, they might also be exposed to the physical world issues such as degradation due to meteorological risks, accidental alterations by humans, or tampering.

SCDs record data and may or may not, depending on the smart city architecture and the device computing capabilities, perform local pre-processing before sending data to the cloud. All these devices communicate wirelessly. These communications are also subject to the constraints mainly of batteries and small communication channels. Indeed, the more a device communicates, the more it will consume battery power, and the more it will occupy the communication channel that it can share with other devices. In addition, it is likely that communications will not be successful and data will be lost along the way. All these reasons, coupled with the fact that the devices are often low-cost, mass-produced low-end devices for which quality is never guaranteed, meaning that the integrity of the data collected can be affected.

In the context of smart cities, several applications could be deployed. These applications could, for example, run decision algorithms that would use the collected data to make a good decision based on a given context. These algorithms could, for example, decide to reduce the speed limit of a road in case of noise pollution exceeding a threshold. In order to make the right decisions for different contexts, decision algorithms need the collected data to accurately represent the context. However, all of the reasons mentioned above can lead to data that does not accurately represent reality. In summary, we have a data integrity problem.

The generic trust model

One of the solutions to this problem is to decide whether we trust or not the data collected by the device using a trust model (TM). Generally speaking, a TM is an agent that, under the hypothesis that a majority of devices behave well, is able to detect when a device is misbehaving and tag it as such. In this work, we will focus on the generic TM proposed in [2]. This TM operates in support layer of the IoT application stack of figure 1 and can, therefore, detect abnormalities in either the network layer or the perceptual layer.

IoT applications are often built upon a layered architecture stack as shown in figure 1 [5]. In this architecture, each layer has its purpose and needs to communicate with its upper and lower layers.

Perceptual layer interacts with the physical world. It can collect data or act on its environment by interacting with actuators.

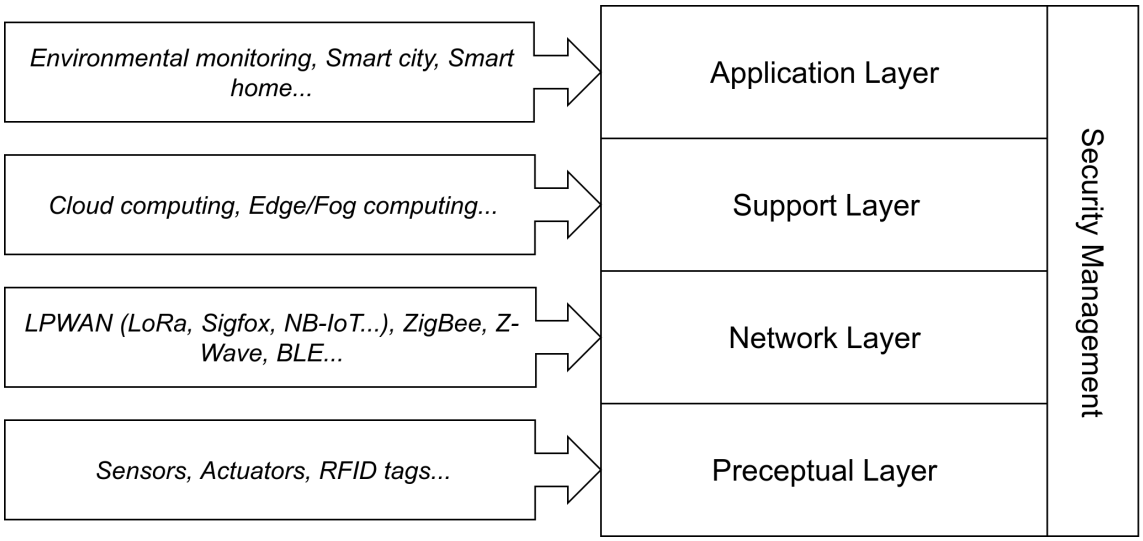


Figure 1. Architecture of IoT systems

Network layer deals with the communication between the perceptual layer and the support layer. Various means of communication can come into play in this layer such as LoRa, z-wave, or Bluetooth among others.

Support layer responsible for making calculations. This layer is in the cloud. Indeed, this layer being responsible for making calculations, it needs the large calculation capacities offered by the cloud. The cloud can also be assisted by edge devices. These devices, which are smarter than standard devices, are halfway between the devices and the cloud. They are able to pre-process data to avoid sending too much information to the cloud and thus limit the transfer of large volumes of data.

Application layer Finally, the application layer is responsible for linking stakeholders and data.

As shown in figure 2 the input of the TM is an interaction, *i.e.* a packet of data sent by a device. The content of a packet depends on the IoT deployment and on the nature of the devices that are deployed within it. For example, a device with a camera will send an image and its resolution, while a device with a microphone will send a sound recording and its sample rate. A packet also contains information about the transmission that occurred in the Network layer. In summary, each traversed layer, i.e. the perceptual layer and the Network layer, add elements to the packet. Anomalies can slip through these layers. The TM must, therefore, detect each of these anomalies. That's the role of trust factors (TFs).

Each TF analyses an element of the interaction in order to quantify its quality and thus detect any disturbance that could occur in the analyzed element. The quality is expressed as trust factor score (TFS) a value between [0, 1] (the higher

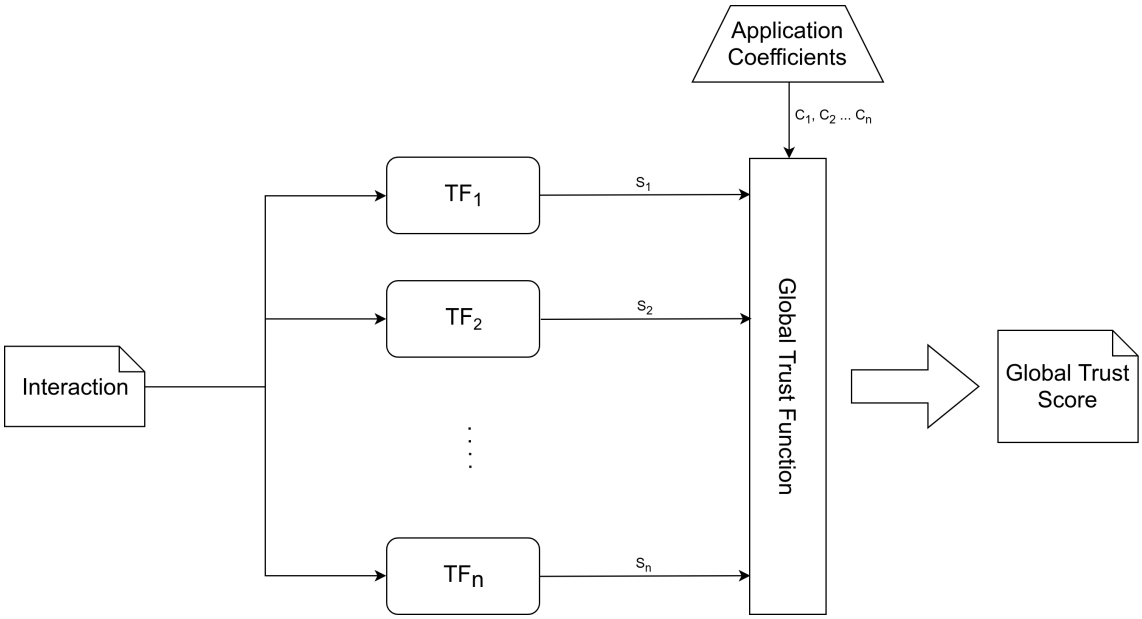


Figure 2. Generic trust model

the better). TFs are TM's input. They must be defined when the IoT architecture is deployed. Any disturbance that could occur must have a TF able to quantify it. These scores are then aggregated and weighted by application coefficients in the Global Trust Function (equation 1) to obtain a Global Trust Score.

$$GlobalTrustScore(I^D, A) = \frac{\sum_{i=1}^{n} C_i^A \times TFS_i^D}{\sum_{i=1}^{n} C_i^A} \in [0,1] \quad (1)$$

Where I^D is an interaction comming from device D, A is the application, $TFS_i^D \in [0,1]$ is the trust factor score (TFS) of the ith element of the interaction and C_i^A is the ith coefficient of the application A.

Different applications might have different tolerance to disturbance. Thus application coefficients are weights that are defined by the application to have power on which parameter is important to decide whether an interaction is trustful. They must, therefore, be defined by domain experts. Let's consider a smart city deployment of devices equipped with cameras. Above this deployment, two applications use the data provided by the devices. The first one is a car counting application. The second one is a license plate identification application. The second application needs a higher resolution than the first one. Thus the coefficient weight for the resolution will be higher in the first application.

The trust model has three generic TFs:

Precision. This device-specific factor expresses the closeness of a reported measurement to the value that a *good* sensor would give. Assuming that a majority

of sensors behave correctly, we expect that neighbor sensors report similar measurements. Thus, the good value can be estimated through a *spatial* correlation method.

Availability. This device-specific factor takes into account the ratio of the number of *received* reports to the total number of *expected* during an interval Δt. Thus, the availability score is defined as:

$$S_{Availability}(\Delta t) = \frac{\#\ reports\ received\ (\Delta t)}{\#\ reports\ expected\ (\Delta t)} \tag{2}$$

Packet Error Rate (PER). This network-specific factor gives a quality estimate of the communication channel between the sensor and the receiver gateway. It takes into account the ratio of the number of erroneously received packets to the total number of packets received in a time interval Δt. Thus, the PER score is defined as:

$$S_{PER}(\Delta t) = 1 - \frac{\#\ bad\ packets\ received\ (\Delta t)}{\#\ total\ packets\ received\ (\Delta t)} \tag{3}$$

This work will focus on creating the building blocks of a signature TF. The signature corresponds to the shape the signal has. It depends on the context, *i.e.* type of road (residential, main roads, secondary roads, etc.), in which the device is deployed. Indeed devices belonging to the same context, even in different roads, should provide data with matching signatures. This TF checks whether the signature of the device respects the signature of its context. This TF involves data analysis and machine learning techniques.

Before these techniques can be used, good quality data must be available. As explained above, the data collected are raw data that have not been processed by a trust model. They, therefore, suffer from an integrity problem. Given this problem, we need to filter out bad data. The main problem is that the very definition of good data is not clear. The only way to determine if the data is properly cleaned is to be an expert in the field of urban acoustics. We, therefore, need to develop a method to filter the data. We propose a filter solution using a Fourier Transform (FT) to filter the data that does not have a daily frequency. Once the data has been cleaned up, we can move on to the second step, which is to answer the questions: *what are the contexts and how many are there ?* We propose a solution that uses clustering techniques to group devices that have similar behaviors.

The rest of the paper is structured as follows. The related work and other studies on the subject will be presented in the state of the art part . Then, Section 1 presents our solution to filter out bad data and discuss our results. Section 2 presents the solution we propose to group devices of the same context and our results. Finally, we draw our conclusion and sketch some future developments in Section 2.2

State of the art

On the side of filtering techniques, there exist anomaly detection techniques using statistical analysis or machine learning. As this filtering problem is a very specific problem, we believe that a solution adapted to the problem must be developed.

On the side of clustering techniques, recent studies show that applying clustering techniques to noise time-series (TS) shows good results. The approaches diverge according to the context and the use cases but they confirm that it is a good approach. Studies such as ours are relatively recent given the recent advances in technology that have allowed the IoT to emerge. Indeed it was complicated, for operational reasons, a few years ago to deploy a large number of sensors. Prior to the emergence in the early 2010's of technologies such as LoRa, it was complicated to conserve device batteries. Consequently, deploying 1000 devices meant that batteries had to be changed regularly and thus drastically increased the maintenance costs of such an infrastructure.

[6] reviews different approaches for TS clustering. Approaches are depicted and compared with respect to the goal of the clustering. k-Means[7] and k-Medoids are very fast compared to other clustering methods which makes them very suitable in TS clustering.

In [8] a case study with similarities to ours was presented. In the city of Milan, Italy, 58 data collection points were used to achieve 24-hour road noise patterns. They used the K-means algorithm with euclidean distance on these 24h patterns to group roads with similarities.

In [9] still in the city of Milan, They kept 35 data collection points and re-used 24-hour road noise patterns. It was proposed to use these 24h patterns to classify roads by means of hierarchical clustering with Ward's algorithm. Their purpose was to compare whether the legal provisions in Italy regarding road type were really adapted to the reality. They discovered that the patterns only partially correspond to the legal road classification as this classification is mainly based on the geometrical characteristics of the road, rather than its noise emission.

Chapter 1

Filter

1.1 Methodology

The canton of Geneva, Switzerland, aims to reduce noise pollution. Thus, under the direction of the cantonal environment office (OCEV), an analysis campaign was launched to draw a noise map of the Carouge neighborhood. The service of air, noise, and non-ionizing radiations (SABRA), an OCEV body, deployed in the city around 1000 devices equipped with noise sensors and thermometers in the city. These devices communicate through the LoRa network deployed in the city of Geneva.

As shown in figure 1.1, devices are deployed along roads and the space between neighboring devices is relatively small. We can also see that in some hot spots different layers. These layers correspond to the level of the highest sensor. Layer 1 (blue) indicates that the devices are 3m above the ground. Layer 2 (green) indicates that there is a sensor 6m above the ground in addition to the sensor in the lower layer. Finally, layer 3 (yellow) indicates a sensor 9m above the ground in addition to the lower layers.

As shown in figure 1.2, a device records a sample of the sound pressure level in dB(A) every second for 15 minutes. Every 15 minutes, the device is supposed to send a report consisting of several pieces of information:

L_{min}**:** The minimum recorded level

L_{max}**:** The maximum recorded level

L_{eq}**:** The equivalent continuous recorded level also sometimes known as Average Sound Level

$L_{10}, L_{50}, L_{90}, L_{95}$**:** the level exceeded for 10%, 50%, 90% and 95% of the period (percentiles)

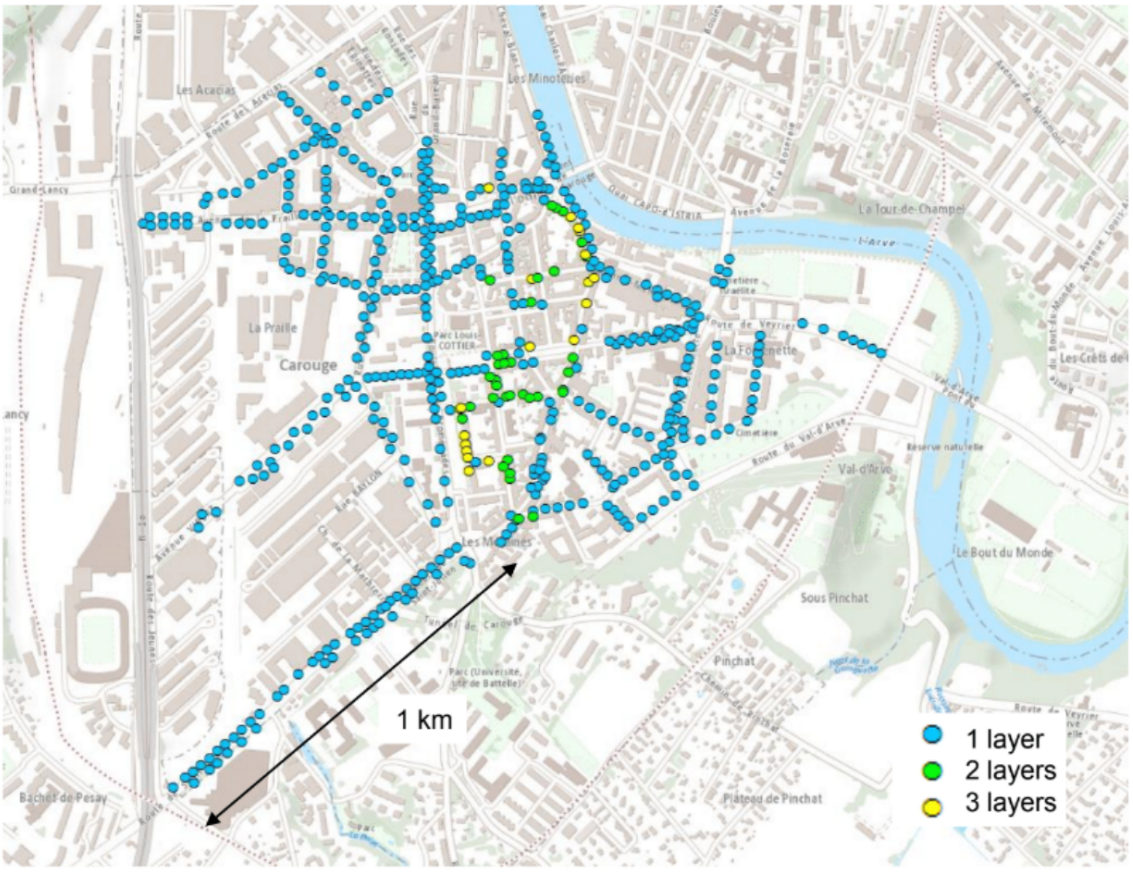


Figure 1.1. City of Carouge's noise sensor deployment

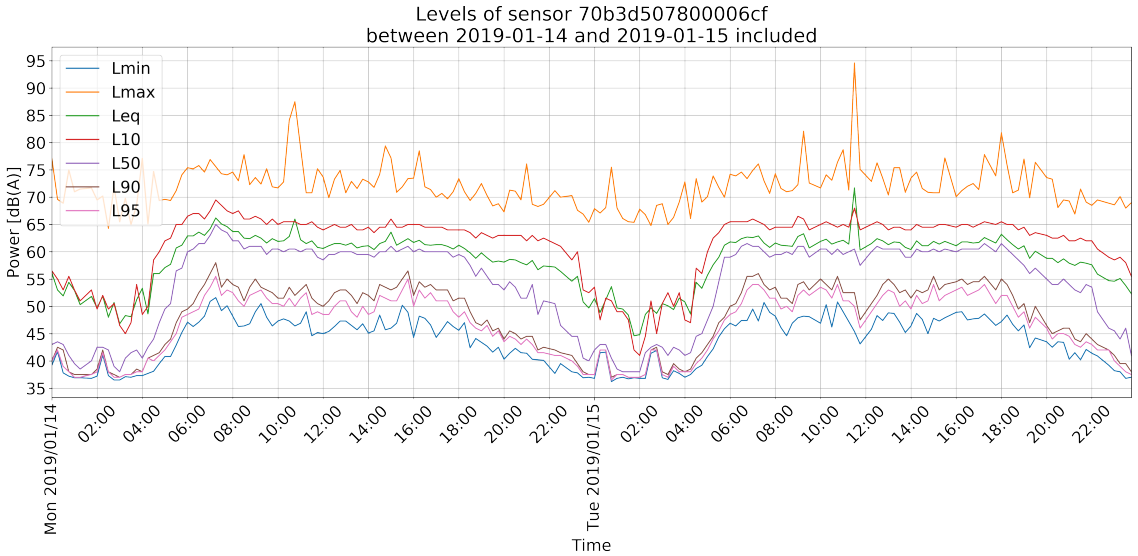


Figure 1.2. A 48-hours observation of one device's output

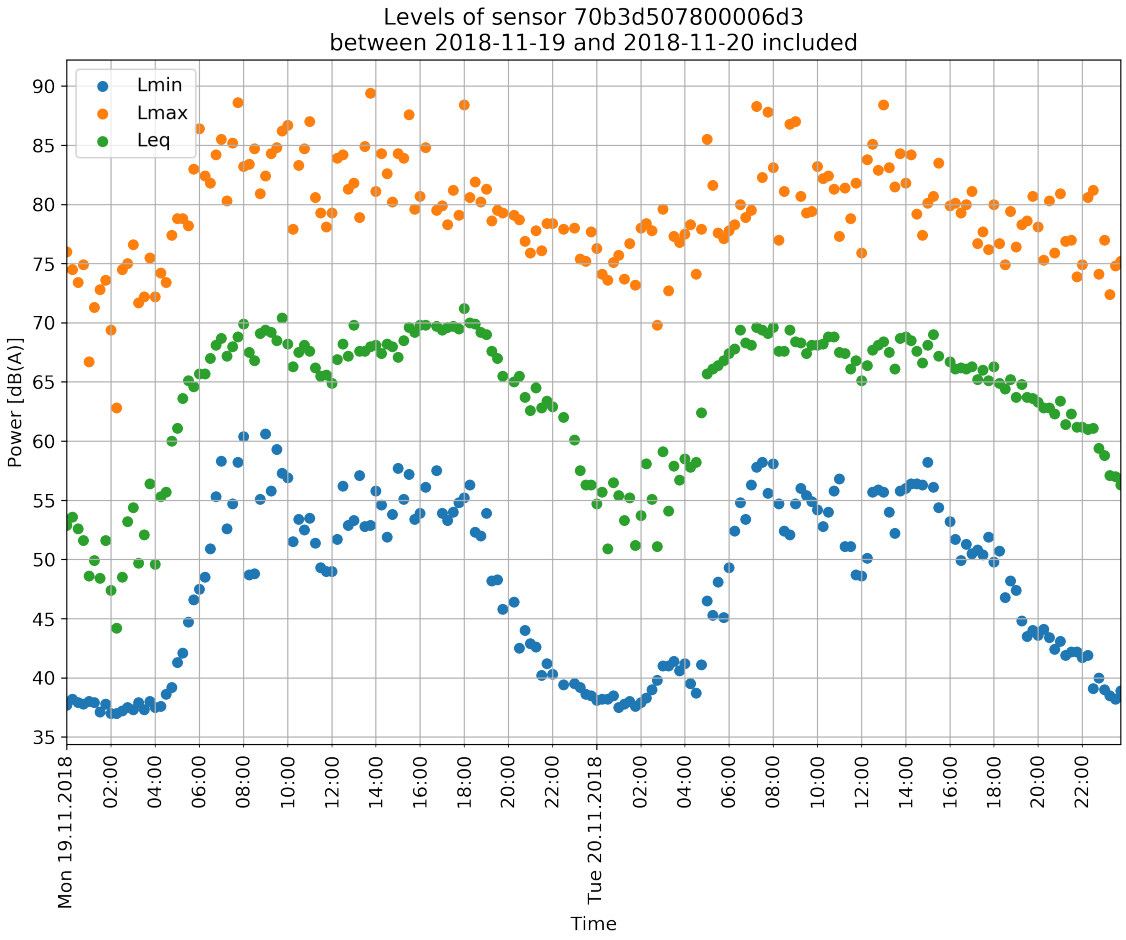
Between 2018 and 2019 the devices collected data without a trust model being integrated into the data pipeline. As SABRA found, this deployment of devices

suffers from the problems described above such as problems related to data transmission (lost data) and data integrity. In Figure 1.3a, we can see that the data follow a pattern in which noise levels are low at night, rise in the morning, remain high during rush hours, and then fall at the end of the day. As can be seen in figure 1.3b, there may be some missing data. These errors may occur either within a given period or in isolated events. We can also see in figure 1.3c that the pattern of the signal does not correspond to the pattern of the other days. In both cases, we do not know what caused the device to send these data. Should the application trust the data sent after a period without data? Should it trust data that doesn't follow the normal signal pattern? It was decided to apply the generic TM in order to answer these questions and thus avoid drawing an imprecise map of noise levels.

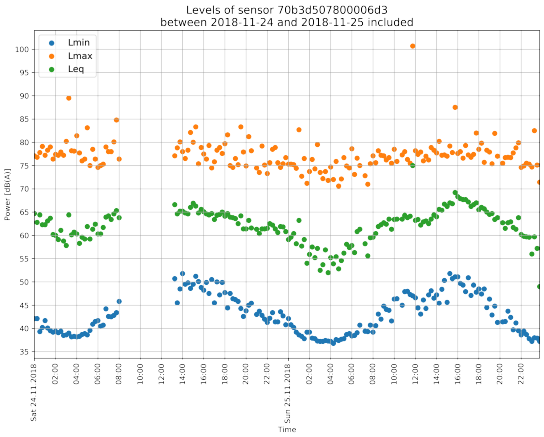
The difficulty in making this filter is that there is no definition of accurate data. Anomalies, which may come from several sources that are difficult to identify, make the validation of the filter difficult. Indeed, only an expert in the field of urban acoustics might be able to recognize noise levels representing reality. Since the collection period is long enough (about a year) the amount of data is large enough to allow the deletion of data. The main goal of this filter is to keep only good data. We, therefore, decided that the filtering policy should be: When in doubt, filter out. The application of this policy is as follows: if an anomaly is detected during one day, the whole day is tagged as bad by the filter. In order to keep a stable pattern, the data must not be affected by extreme values, we decided to discard the L_{eq} and L_{max} levels. Indeed, as we can see in figure 1.2 L_{eq} being an average, it can be attracted by extreme values. This left $L_{min}, L_{10}, L_{50}, L_{90}, L_{95}$. We chose L_{min} because it represents the minimum noise recorded by a sensor over the period. We thus think that an anomaly would be more easily identifiable on this value. This is why all the following analyses will be carried out on L_{min}.

As shows figure 1.3a, a 24h pattern occurs since urban noise is mainly affected by road noises[4]. Based on that observation an intuition was that anomalous data would not fit this pattern. Our solution, therefore, turned to data analysis in order to find this pattern and filter the data that did not respect this pattern. So we decided to analyze the spectrum of the signal in order to characterize a good day. In analyzing the data by weeks, we also found that the patterns were recurring from day to day. i.e. the Mondays of a device have a pattern that is different from the pattern of Tuesdays of the same device and so on. In figure 1.4, Saturdays and Sundays clearly have different patterns that other days.

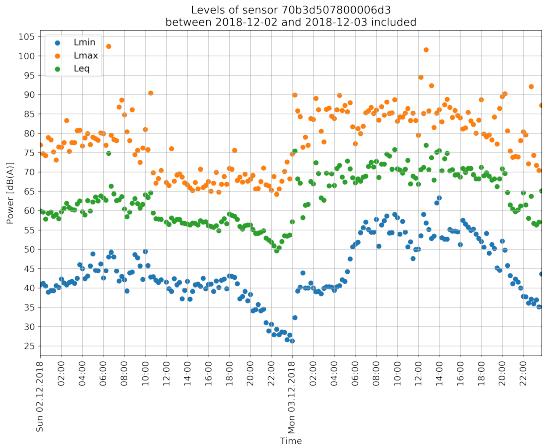
The filter answers the following question: *Does this day matches a good day pattern ?*. In order to answer this question, he needs to know what a *good day pattern* is, knowing that this definition could change depending on the day of the week. The filter itself will deduce what this pattern is by using all the days it has at its disposal. We will call this pattern the Reference TS (RTS) for a given day of the week and a given device. To do this, he will artificially create a TS by concatenating the days. given the TS S composed of noise levels samples S_i belonging to days of the week



(a) A 48-hours observation of one device's output considered as accurate



(b) A 48-hours observation of one device's output with missing data



(c) A 48-hours observation of one device's output containing inaccurate data

Figure 1.3. Three common observable scenarios in TS

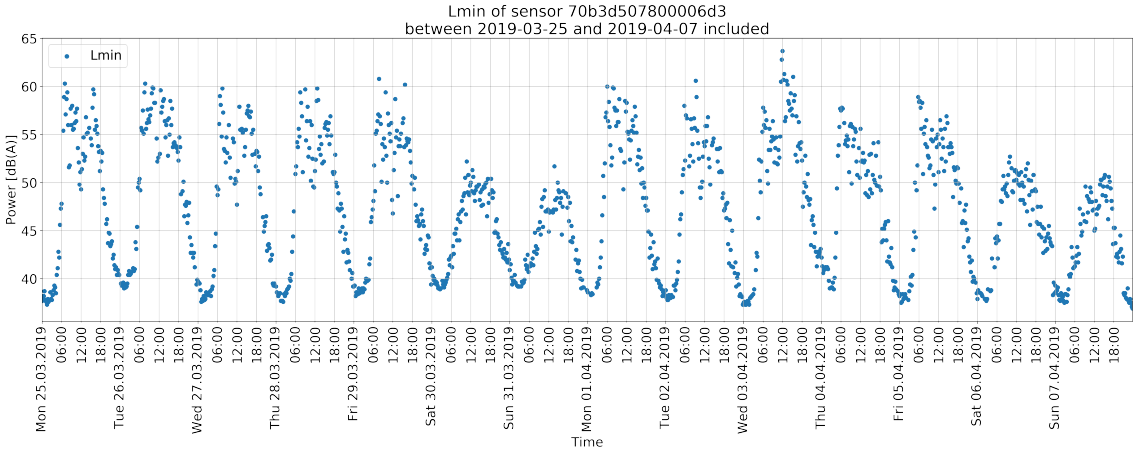


Figure 1.4. A two weeks observation of one device's output

$\{0, \ldots, 6\}$ (Monday to Sunday). There are 96 samples s per day ($4_{samples/hour} * 24$) For the sake of mathematical simplicity, let's say that the first sample of the TS S_0 is the first sample (midnight) of a Monday so S_{96} is the first sample of Tuesday. The function defining the RTS is the following:

$$Ref(S_{Device}, Day) = \forall i \in \{0, \ldots, |S| - 1\} | S_i \in ref \iff (i \div 96) \mod 7 = Day$$
$$\text{With } Day \in \{0, \ldots, 6\} \tag{1.1}$$

In other words, we take all the samples made on the given days and we compress them like they're consecutive. The resulting TS for Sunday is presented in figure 1.5

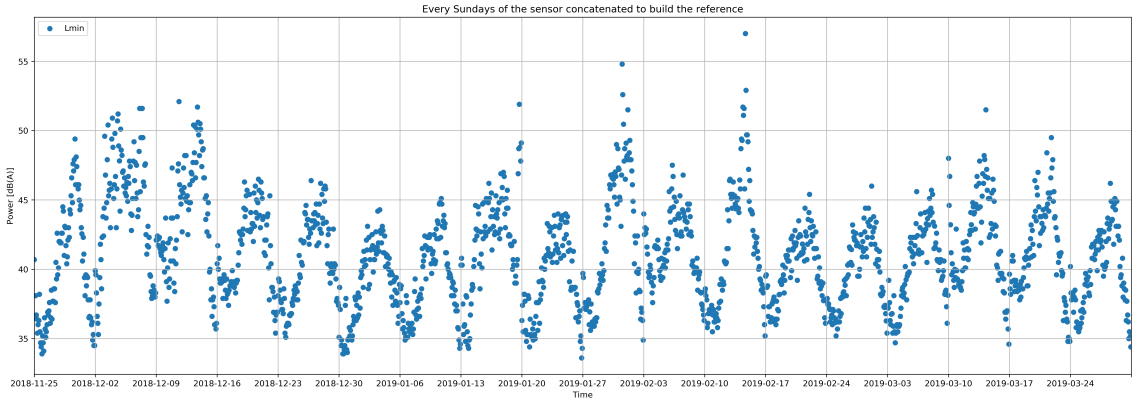


Figure 1.5. RTS for a given sensor and for Sundays

With the RTS we could now analyze the pattern with the Fast FT (FFT) algorithm but the FFT algorithm is very sensitive to missing data. Missing data are relatively frequent in the collected data as we can see in figure 1.3b. As stated above, missing data occur either in a given period or in isolated events. For isolated missing

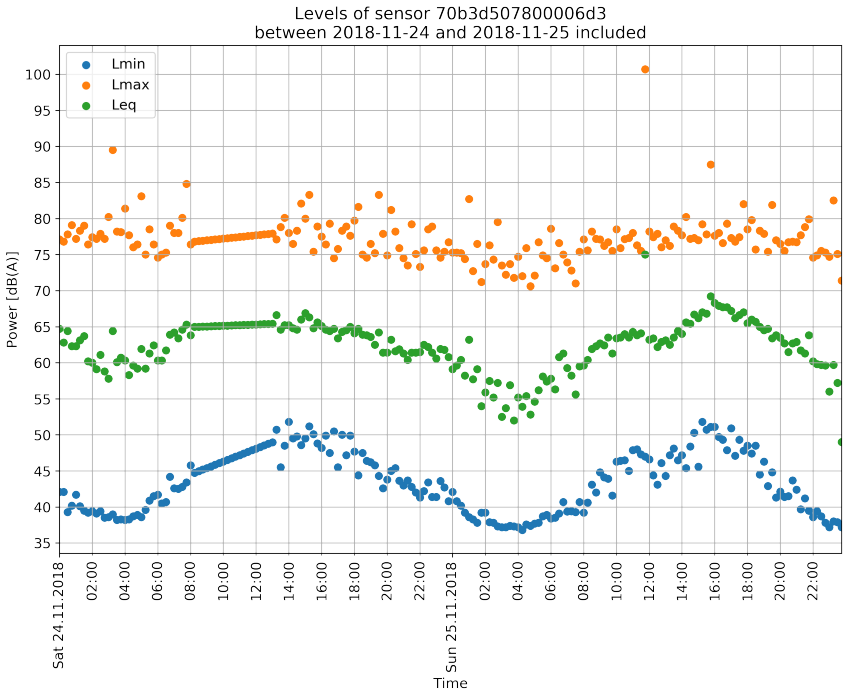


Figure 1.6. Missing data interpolated with a linear interpolation method

data, it is not a problem because we can interpolate these samples without too much information loss using linear interpolation. For missing data occurring in a period, we can not interpolate as it would result in a loss of information. As shown in figure 1.6, interpolating too much missing data all at once leads to unrealistic results. The goal here is not to Therefore instead of interpolating too much missing data at once, we will place a first filter that will pre-filter the RTS. If more than 10 samples are missing in a day, the day is tagged as bad by the filter and it will be removed from the RTS. The aim is not to search for the best interpolation method. Such a search would be costly in time and energy. This is why the above-mentioned policy must be applied: when in doubt, filter out. Now that the RTS has been sanitized from any missing data, we can apply the FFT. We will call the FFT applied on the RTS the Reference FT (RFT). As shows figure 1.7, the highest peak in the RFT represents the 24h period.

Now that the RFT is set we can compare each day with it. We will use the same process as the reference. We are going to apply a FFT on a day that we want to compare with the RFT. In order to be able to compare two Fourier transforms, the number of samples composing them must be equal. Our example in figure 1.5 is composed of 18 days. Each day is composed of 96 samples, it is composed of $18 * 96 = 1728$ samples. Comparing a single day with the RFT without pre-processing is impossible because $96 \neq 1728$. We thus need a solution to compare one day with the reference. The solution is to duplicate the TS of one day according to the number of days composing the reference. In the case of our example in figure

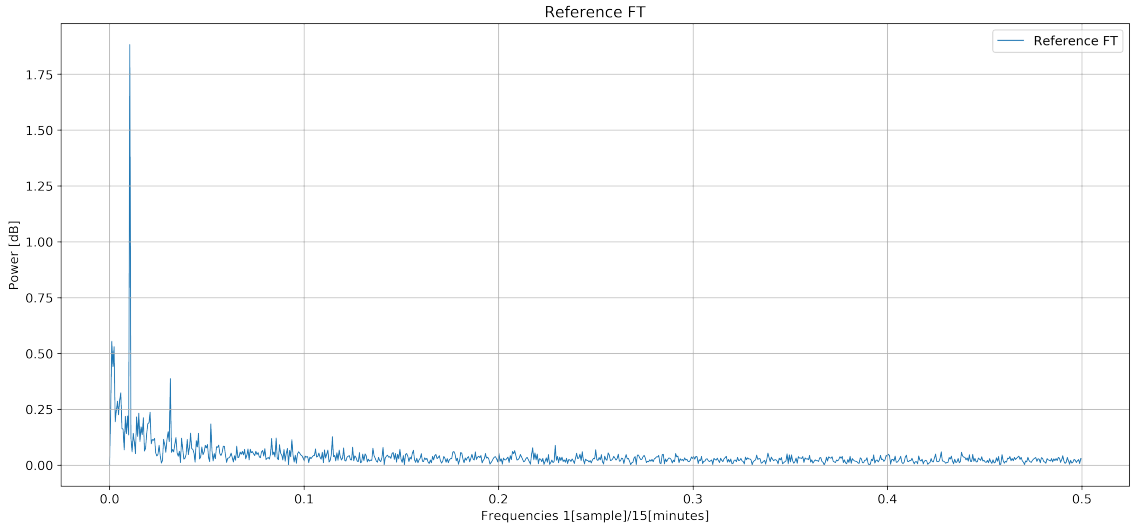
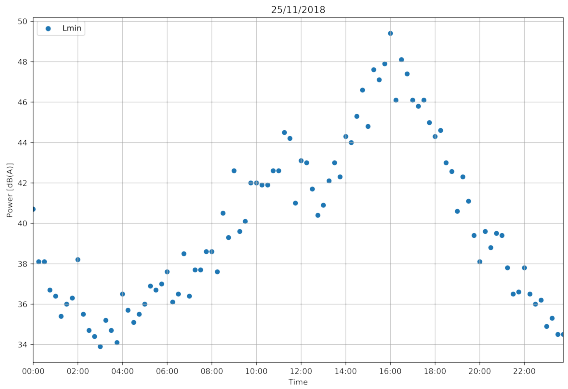


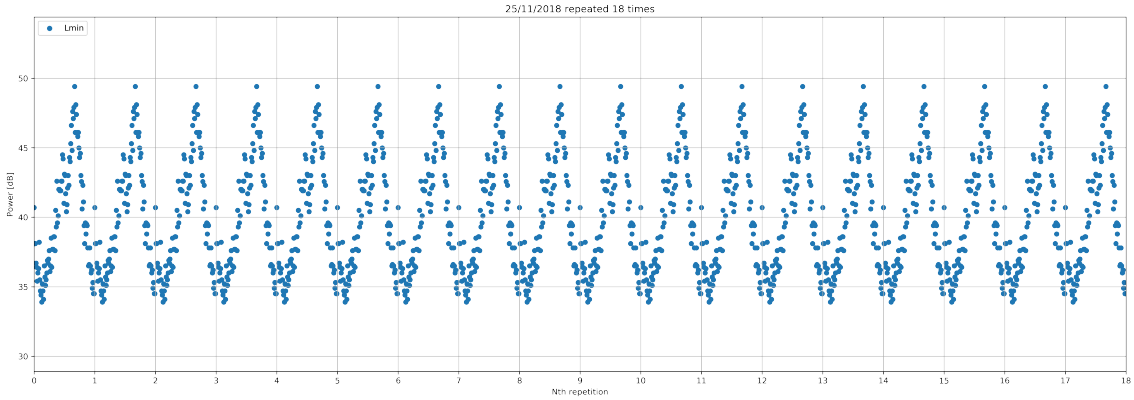
Figure 1.7. Example of RFT, resulting from the FFT algorithm applied on the RTS in figure 1.5

Note: For sake of representation, frequency zero (F_0) has been artificially set to zero since it represents the average power of the signal

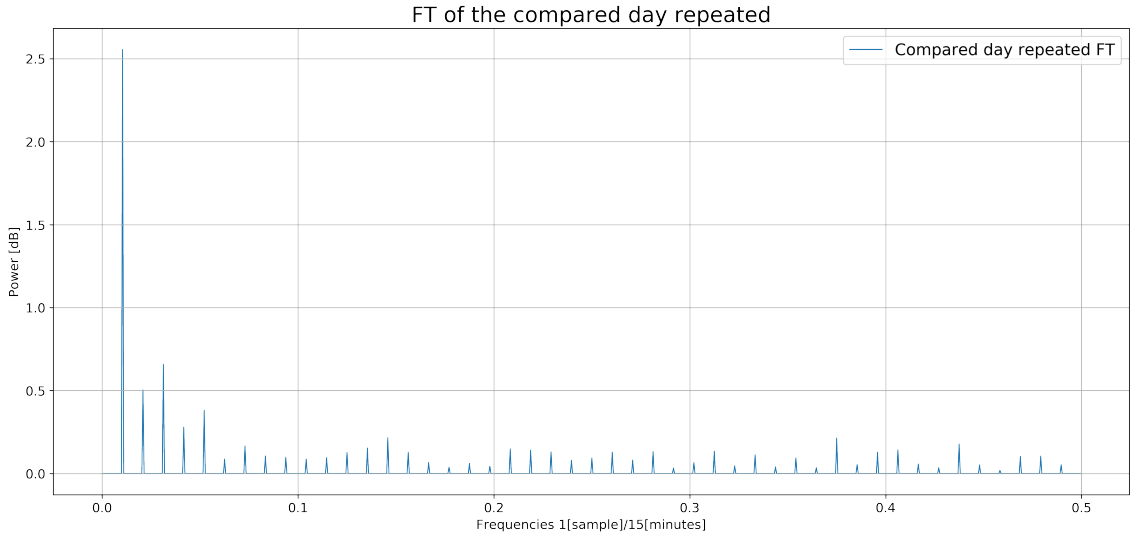
1.5, there are 18 days, so the compared day will be replicated 18 times. The resulting TS is in figure 1.8b. With this TS that matches the size of the RTS, we can apply the FFT algorithm on it and get a FT that matches the size of the RFT.



(a) TS of the compared day



(b) Compared TS built with the compared day replicated to fit the RTS size



(c) FT of the compared day replicated TS

Note: For sake of representation, frequency zero (F_0) has been artific0ially set to zero since it represents the average power of the signal

Figure 1.8. Example of compared TS, with a given day replicated to fit the size of the RTS and then its FT.

As we can see in figure 1.8c, the first peak is also present and is the highest. We can also see that the consequence of compared day TS replication is that the noise is no longer randomly distributed and has been amplified. Now that these two are calculated, we can compare them.

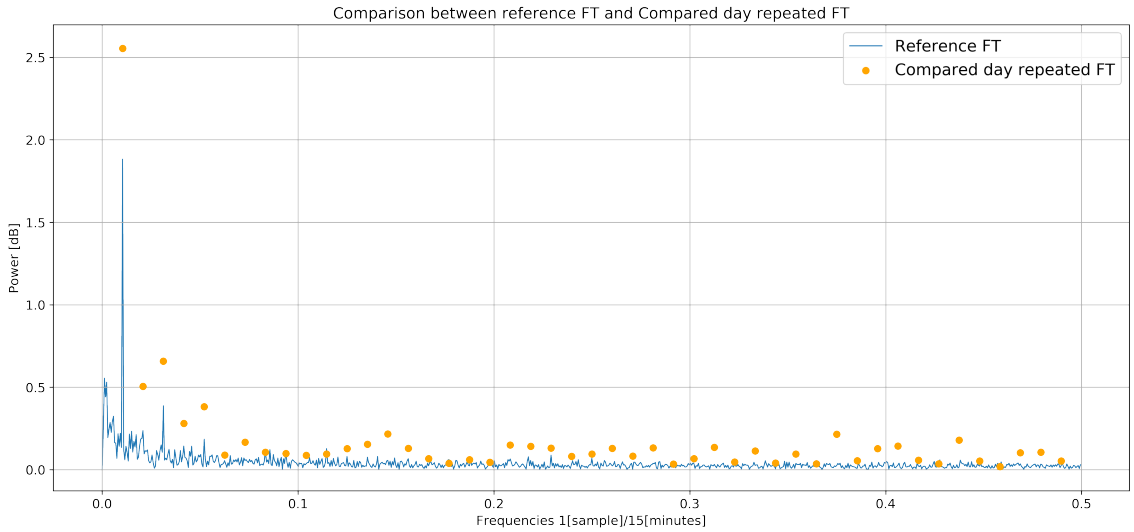
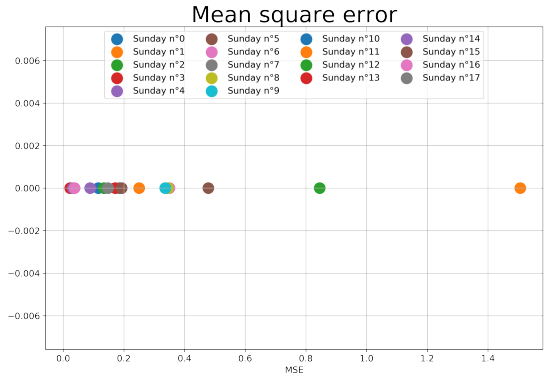


Figure 1.9. Comparison between RFT (blue line) and compared FT (orange dots) Note: For sake of representation, frequency zero (F_0) has been artificially set to zero since it represents the average power of the signal

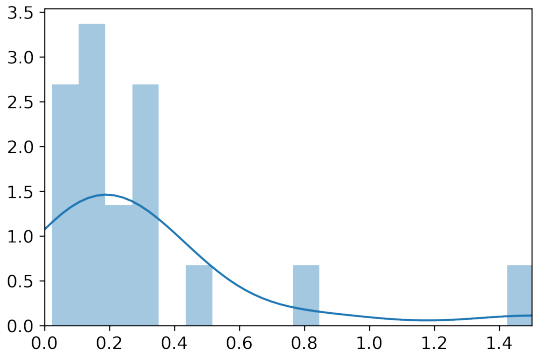
In figure 1.9, the dots represent the power level for each frequency. This graph aims to highlight the differences between the peaks of the two FT. Although this visual representation shows the differences between the two FT, we need a method to quantify the error between the RFT and the compared FT. There are different options available to quantify this difference. Our choice turned to the Mean Squared Error (MSE). Its formula is the following:

$$\text{MSE}(Y) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \tag{1.2}$$

The advantage of the MSE is that it will penalize large differences as it squares them. Our goal is to filter out days that do not have a daily pattern. So we will only consider the first peaks (with F_0) that correspond to low frequencies. The number of peaks we will consider is a parameter of the filter. It can, therefore, be modified to consider more peaks or fewer peaks. Currently, the filter considers the first 10 peaks. Now that we have a measure quantifying the difference between RFT and a compared FT, we can apply this procedure to each day composing the reference in order to get their MSE.



(a) MSE of each day of the reference represented along an axis



(b) figure 1.10a seen as a statistical distribution

Figure 1.10. MSEs viewed along an axis and as a distribution

Once all the MSEs have been computed, we can compare them and decide what is the tolerated error. As shows figure 1.10a, we have several days that are more or less far from the reference. Therefore we have to decide at what point we consider the day to be bad. We can consider this set of MSEs as a statistical distribution. In statistics, outlier detection is a common task. We will thus apply an outlier detection technique in order to automatically define a tolerance threshold. The chosen outlier detection technique chosen is Tukey's boxplot. This technique can be represented visually by the graph of the same name. It calculates the first (Q_1) and third (Q_3) quartile of the distribution and then calculates the space between these 2 quartiles (IQR). A day is then considered as an outlier if its $MSE > Q_3 + 1.5 * IQR$. Outliers are then considered as bad days.

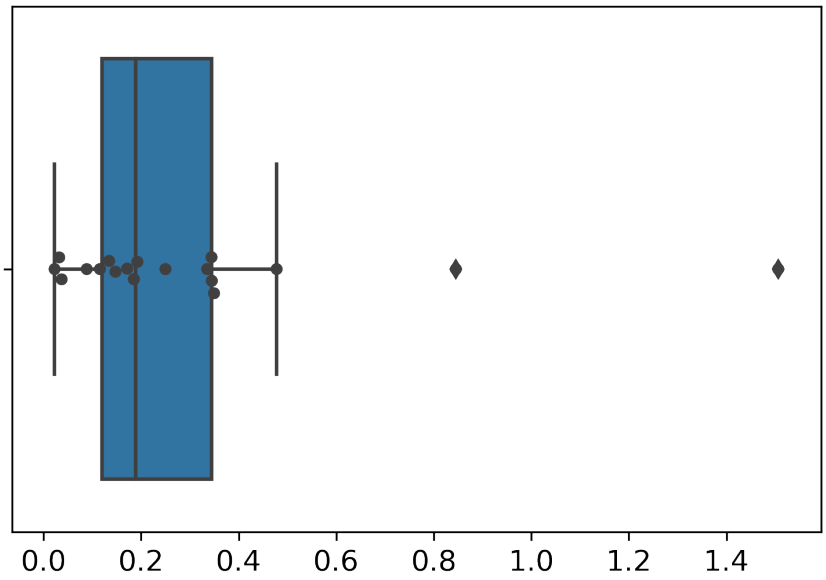


Figure 1.11. Boxplot of the distribution in figure 1.10

Finally, the filtering procedure can be summarized as follows:

Data: Device's time-series, A day to filter $\{0, \ldots, 6\}$
Result: List of good and bad days
extract the RTS from the device's TS
Filter out days without enough samples
compute the RFT
foreach *day in the RTS* **do**
  Extract the day of the RTS
  Replicate the day signal along time so it has the same amount of samples as the RTS
  Compute the compared FT of the replicated signal
  Compute the MSE between the RFT and the compared FT
end
Compute the error threshold
foreach *day in the RTS* **do**
  if $MSE_{day} <= threshold$ **then**
    Add the day to the good day list
  end
  else
    Add the day to the bad day list
  end
end
return good day and bad day lists

Algorithm 1: Filter procedure

1.2 Discussion

As explained in the methodology section, the difficulty in making this filter is that there is no definition of accurate data. According to the information sent to us by SABRA, we had to make several assumptions. These assumptions were as follows: The good data showed a 24-hour pattern and the majority of the data from one sensor was good.

As the definition of "good data" is not clearly defined, it is not possible to quantify the quality of the filter. Only domain experts can validate the quality of the filter by analyzing its results. Therefore, a tool is needed to be able to visualize the data in order to decide if a bad day is not filtered out by the filter. Indeed, with no less than 550 sensors filtered over a period of about 1 year, the amount of data is too large for the filter results to be manually analyzed one by one. In order to facilitate validation work by experts in the field, a Representational State Transfer (REST) Application Programming Interface (API) has been developed. This API makes it easy to request the filtering of the days of a given sensor. With this API, a web interface for data visualization has been set up. The web app calls the REST API This web app allows us to choose a device on the map and display good and bad days on a calendar using green and red colors for respectively good and bad days. The user can then choose a day and get insights information about the filter like its FT and MSE.

Unfortunately, due to the large volume of data, the filter has not yet been validated but the first results are promising and seem to validate the general filtering method. In the case of inaccuracies, several parameters that allow us to act on the filter can be modified according to the type of inaccuracies. In case the unfiltered days do not respect the daily pattern, we can act on the number of peaks to consider and increase or reduce it. In case the filter is too tolerant, i.e. the outlier definition is too lax, we can replace the outlier detection method to apply a more severe one, such as isolation forest for example, in order to filter more strictly. Finally, if the error calculation does not suit our needs, the error calculation method can also be replaced by the Mean Absolute Error (MAE).

Chapter 2

Clustering

2.1 Methodology

Now that we have filtered data, we can analyze it. Thanks to the REST API, we have a data pipeline that allows us to collect data and filter it easily with a simple call to an API route. With these filtered data we will now be able to extract signatures not polluted. With these sensor-specific signatures, we will attempt to group signatures that are close together in order to understand the factors causing their proximity.

The definition of a sound signature is not precise. Any component that helps characterizes the sound can be considered part of the signature. Applying this to our noise TS, several components might make up the signature. Following the work done in [8], in which they performed a similar task and used 24-hour patterns, we thought this would be a good starting point. Since we have a large amount of data, we need to figure out a way to get a 24-hour pattern representing a sensor. One possible technique would be to average each hour and thus have an average level for each hour of the sensor. This would be equivalent to calculating the average level of all samples at 00:00, then all samples at 00:15 and so on until 23:45. The result would be 96 averaged levels which would, therefore, represent a TS of an average day as shown in figure 2.1.

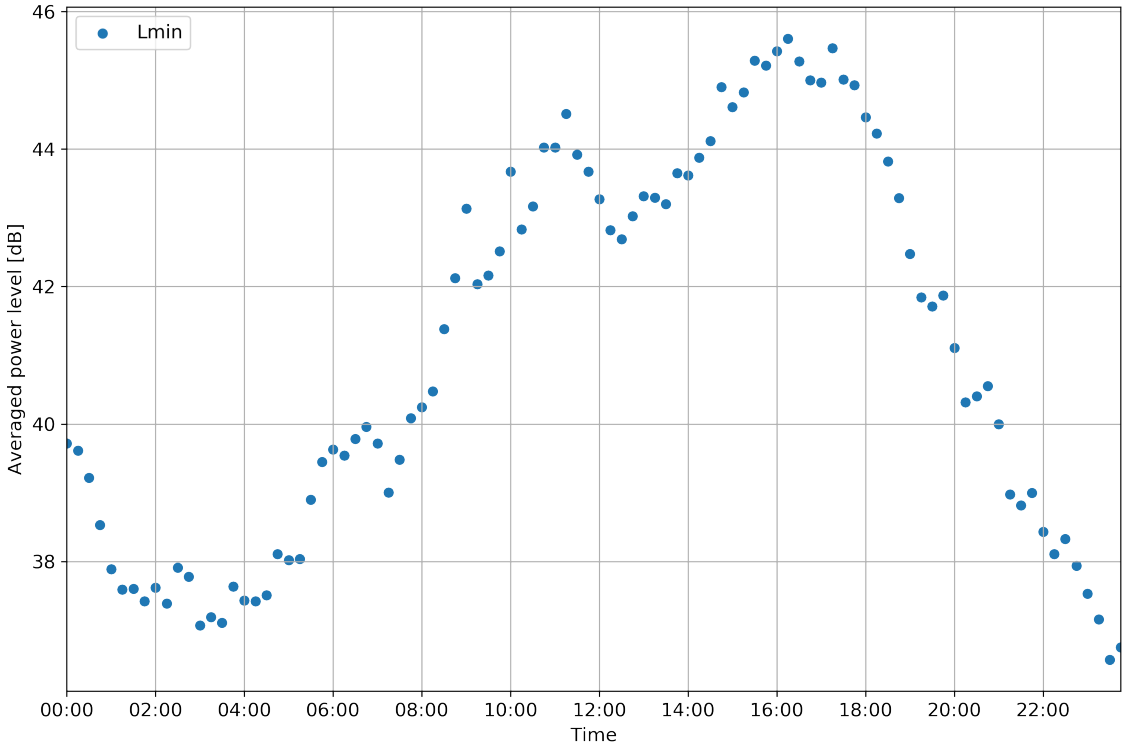


Figure 2.1. Hourly averaged levels representing a 24h pattern

This starting point presents a major problem. As indicated in 1 each day of the week has its pattern. This observation is especially true for the weekend days. We, therefore, decided to take these differences into account by including the day in the averages. Instead of averaging all the samples of a given hour, we calculate the average of all the samples at a given time on a given day. The first average level would then be Monday at 00:00, then Monday at 00:15 and so on until Sunday 23:45. The resulting time-series, presented in figure 2.2, would then be composed of $96 * 7 = 672$ averaged levels.

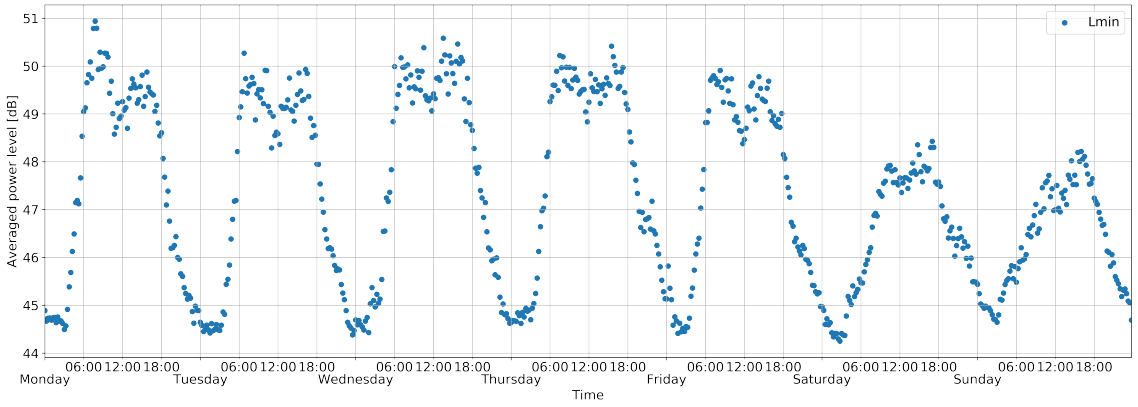


Figure 2.2. TS representing the signature of a device

This TS presented in 2.2 is the signature of a device. This signature will be used in a clustering algorithm to group common signatures. As presented in section , globally the K-means algorithm is particularly adapted to TS. Moreover, it has already been used to solve a problem similar to ours and presented good results. We will, therefore, use the K-means algorithm to group the signatures. This algorithm is known as a machine learning algorithm belonging to the unsupervised learning branch. The result of these algorithms is called a model. The K-means algorithm sees data as a vector space. Thus each device will be defined by a signature which will be defined by a vector of dimension 672. Since we have a total of $N = 551$ devices, our dataset will be composed of 551 vectors and each vector will have a dimension of 672. Each cluster is then defined by a vector defining the center of the cluster. These vectors are called centroids. Each vector belongs to its nearest centroid.

One of the constraints of the algorithm is that the value of K must be defined. This value corresponds to the number of clusters that will be created, i.e. the number of distinct groups at the end of the execution of the algorithm. Fortunately, it is possible to run the algorithm several times while varying the value of K. It is then possible to compare the results of the different K values using a metric. This metric is the final cost function. During training, the k-means algorithm tries to minimize a function called distortion. this function is defined as the sum of the squared distances between each vector and its closest centroid. In other words, the closer the vectors are to their centroids, the smaller the sum of the squared distances, and the better the centroids are positioned. It is therefore logical that by adding clusters the distortion decreases until it reaches $K = N \Rightarrow distortion = 0$. Using the heuristic of the elbow or the knee of a curve, which tells us when adding clusters is no longer worth the reduction in distortion, we can determine what is the best K or at least between which bounds the best K might be located.

In order to find the best K, we ran the K-means clustering training algorithm several times, varying K from 1 to 10. Since the algorithm is not deterministic, it may not find the global minimum of the cost function but a local minimum. To solve this problem, the algorithm was run 10 times for each value of K and the best model was chosen. the distortion measure presented is therefore that of the result of the best model for a given K. As illustrated in figure 2.3, we can see that there is no "knee of the curve" but rather a reduction in the slope of the curve between 4 and 6. For information, SABRA had estimated that this value of K would be between 4 and 8.

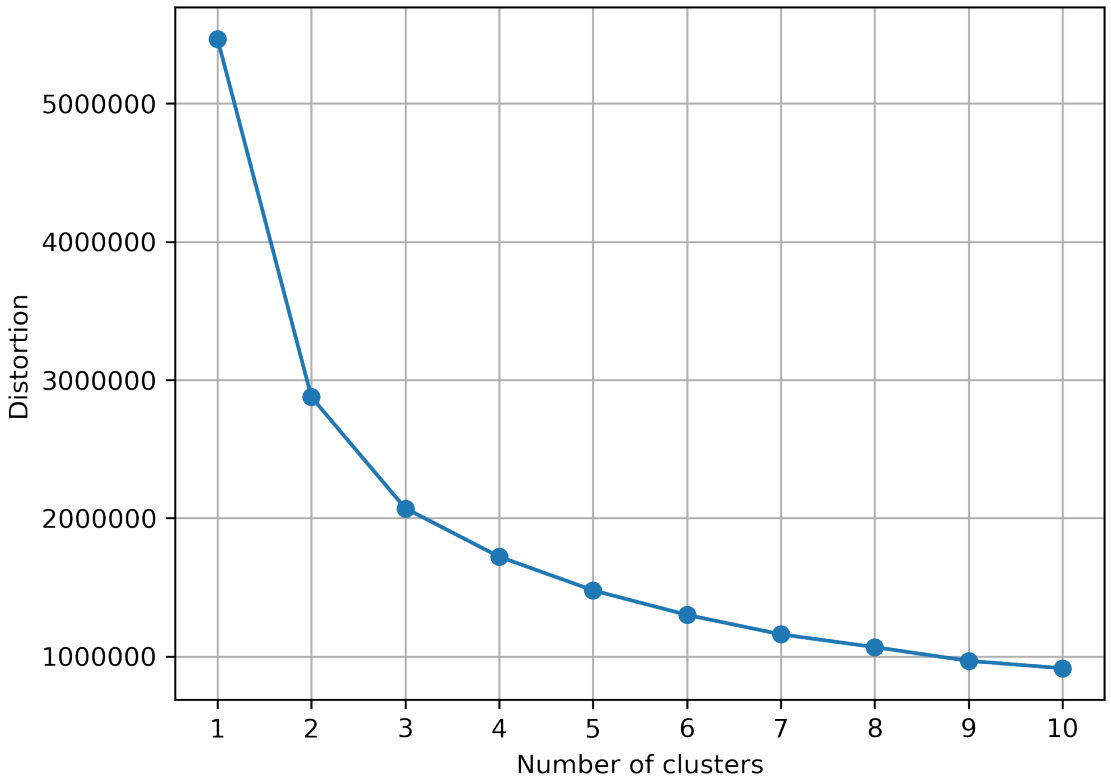


Figure 2.3. Distortion as a function of the number of clusters

2.2 Discussion

Since centroids are vectors, they also are a TS. Therefore we can visualize centroids in the same way as signatures. Figure 2.4 shows centroids for each values of K. In order to facilitate understanding, when we talk about a cluster, we will use the following notation: C_i^K, designating the cluster i belonging to the model composed of K clusters. For example C_0^4 refers to cluster 0 of the first graph where $K = 4$. As shown in the first graph, in which $K = 4$, the levels are distinct during the daytime, and then 2 levels are distinguished at night-time. When adding the fifth cluster, we can find out that something strange is happening. The new C_1^5 cluster is located between clusters C_0^4 and C_3^4. In reality, the C_0^4 and C_3^4 clusters no longer exist in C^5. They split into C_0^5 and C_2^5 to make room for C_1^5. This cluster has relatively average levels during the day but they are particularly high at night. We will see later that these night levels have a very rational explanation. Finally adding the sixth cluster creates the last one that has very low levels at daytime and night-time. Taking these findings into account, we chose to select $K = 6$.

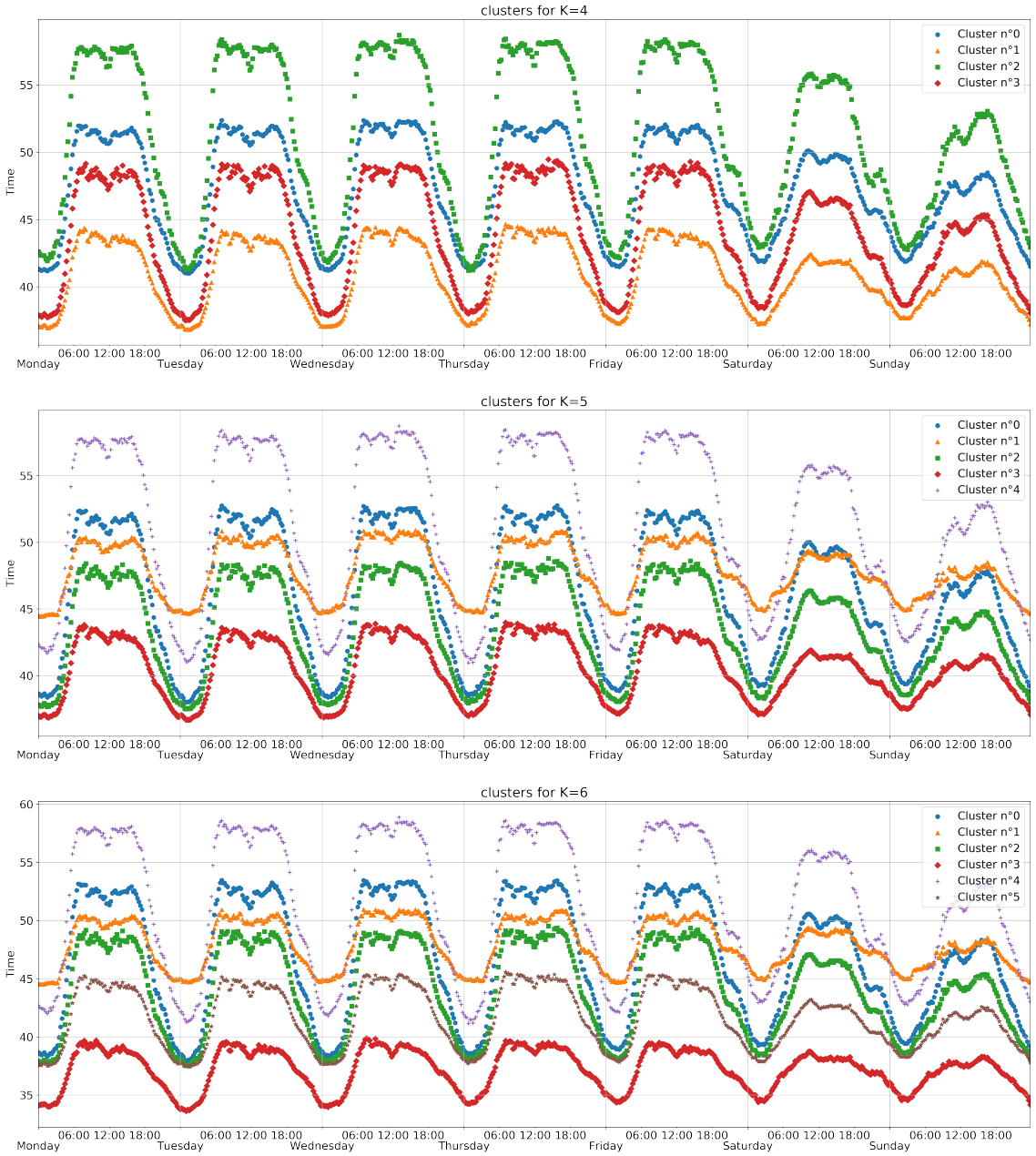


Figure 2.4. Centroids of clusters for $K \in \{4, 5, 6\}$

It is hard to understand how clusters interact with each other. We can see general behaviors of the different clusters but this information is not sufficient to understand how clusters are distributed geographically. Since noise is a propagating wave, adjacent devices should record highly correlated data. These highly correlated data should in return be classified in the same clusters. The intuition would then be that this correlation could be found by displaying the devices on a map.

As the map in figure 2.5 shows us, our initial intuition is quite valid. The first

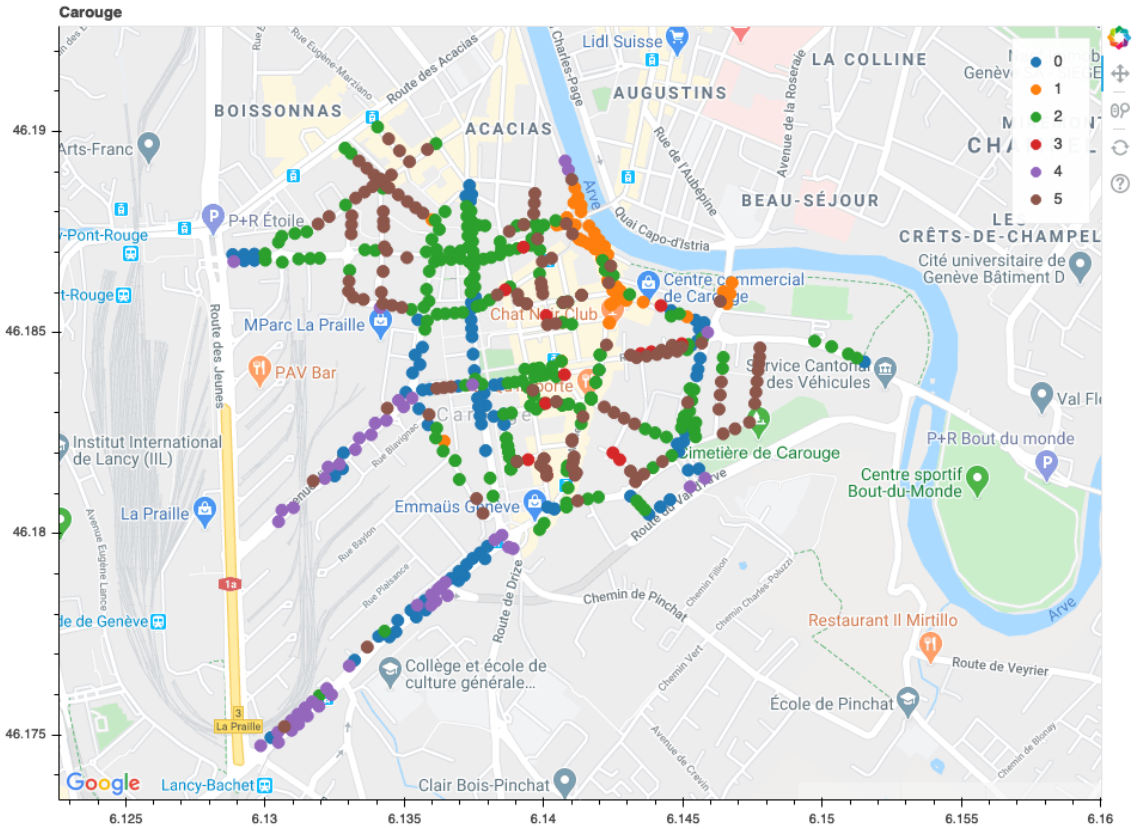


Figure 2.5. Devices according to the cluster to which they are associated placed on the Carouge map

observation we can make is that the cluster C_1^6 is only present around the Arve river. Indeed, the constant flow of water has a non-negligible impact, especially at night. During the day, since water noise is less noisy than road-noise, road noise increases the general sound level. which means the signal keeps a 24-hour pattern. We, therefore, believe that this cluster is a special case for devices close to a river. It also tells us that the type of road plays a big role in the measurements during the day but that at night other parameters can have more impact because road noise is much less impacting. This finding brings us new perspectives for future work. It would be wise to consider a daytime context and a night-time context. An example would be a street that hosts nightlife spots would have a normal daytime context but a much noisier nighttime context than others.

As we can also see, along the main access roads to the city of Carouge, the devices belong either to C_0^6 or C_4^6 which corresponds to the highest levels during the day. Devices on boulevards are also associated with C_4^6. These cluster changes and therefore average levels are probably due to changes in speed limits along these major axes which therefore have an impact on levels along the same axis. It would be interesting to compare these clusters with the speed limits in these streets. The secondary roads are globally part of the C_2^6 cluster. We find these devices in the

heart of the city where the speed is often limited to 50Km/h. Finally, the devices on very quiet residential roads are associated with the C_5^6 cluster. These are often one-way roads, two-way roads but where only one vehicle can pass at a time. Moreover, these roads are often limited to 30Km/h. Finally the last cluster C_3^6 is the strangest. we can't get any information on the geographic location of this cluster. It is also the least represented. Indeed, only 16 devices are associated with this cluster. On the other hand, there is such a difference between the levels of this cluster and the others that we think it does have its context. It could be that this cluster is a cluster of defective devices. The identification of this context and therefore the reason why these devices collect this data is one of the tracks to be explored in future work in order to better understand the interactions of noise in a small town like Carouge.

Conclusion

Road traffic is the main source of environmental noise. As it is the main source of environmental noise, people living in urban areas are much more exposed to these health risks. According to United Nations (UN)'s 2018 revision of world urbanization prospects [1], in 2018 the percentage of the world's population living in urban areas was 56%. By 2050, this percentage will be 68%. In order to reduce noise exposure, it is first necessary to understand how noise propagates in cities.

That's the project the cantonal environment office (OCEV) is undertaking. Through service of air, noise, and non-ionizing radiations (SABRA) they have deployed several hundred devices in the city of Carouge in the canton of Geneva in Switzerland. However, this deployment suffers from several problems related to the heavy constraints to which these sensors are subjected. The environment in which they are deployed makes them vulnerable to weather conditions, tampering, etc. We thus have a data integrity problem with the data collected that we need to resolve. The proposed solution is to use the trust model (TM) proposed by [2]. The purpose of this work is to develop the first building blocks of a signature trust factor (TF). La signature d'un capteur correspond au contexte dans lequel il est déployé (axe principal, route secondaire, etc). Pendant plusieurs mois les appareils déployés ont collectés des données brutes.

In the first phase, it was necessary to filter out the data that lacked integrity. As the meaning of "correct data" is not clearly defined, only an expert in the field can validate whether or not the filter does indeed eliminate incorrect data. However, our filter is based on the assumption that the majority of the data are correct. We decided to filter the data by blocks of 24 hours. We can then speak of a correct or incorrect day. We then use those 24-hour blocks to calculate a reference Fourier Transform (FT) that we can compare to individual days FT.

In a second step, we clustered the sensor signatures using a clustering algorithm to create context-specific signatures. The signatures are composed of the average levels over a week. We have therefore created 6 different clusters, each with its context. As we thought, these contexts are strongly related to the types of roads and other parameters that can have a significant impact in some cases. In future work, we could consider considering night and day contexts in order to better characterize

night behaviors. We could also try to better understand why some devices are classified in clusters that do not at first glance correspond to their contexts. This kind of analysis requires a great knowledge of the context of the city of Carouge and its geography. Unfortunately, this kind of information is not collected by sensors.

Bibliography

[1] United Nations, Department of Economic and Social Affairs, and Population Division, *World urbanization prospects: the 2018 revision*. 2019, OCLC: 1120698127, ISBN: 978-92-1-148319-2.

[2] M. N. Bouchedakh, "Generic trust framework for iot applications," Last accessed: 2020-02-25, Master's thesis, HEPIA - Haute école du paysage, d'ingénierie et d'architecture, 2018. [Online]. Available: `https://lsds.hesge.ch/wp-content/uploads/2018/05/Master-Report-Bouchedakh.pdf`.

[3] F. Theakston and Weltgesundheitsorganisation, Eds., *Burden of disease from environmental noise: quantification of healthy life years lost in Europe*, OCLC: 779684347, Copenhagen: World Health Organization, Regional Office for Europe, 2011, 106 pp., ISBN: 978-92-890-0229-5.

[4] Europæiske Miljøagentur, *Noise in Europe 2014*. European Environment Agency, 2014, OCLC: 918951121, ISBN: 978-92-9213-505-8.

[5] K. Daimi, Ed., *Computer and Network Security Essentials*. Springer, 2018, ISBN: 978-3-319-58423-2. DOI: `10.1007/978-3-319-58424-9`. [Online]. Available: `https://doi.org/10.1007/978-3-319-58424-9`.

[6] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering – a decade review," *Information Systems*, vol. 53, pp. 16–38, Oct. 2015, ISSN: 03064379. DOI: `10.1016/j.is.2015.04.007`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S0306437915000733` (visited on 06/05/2020).

[7] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, p. 100, 1979, ISSN: 00359254. DOI: `10.2307/2346830`. [Online]. Available: `https://www.jstor.org/stable/10.2307/2346830?origin=crossref` (visited on 06/05/2020).

[8] G. Zambon, R. Benocci, and G. Brambilla, "Cluster categorization of urban roads to optimize their noise monitoring," *Environ Monit Assess*, vol. 188, no. 1, p. 26, Dec. 12, 2015, ISSN: 1573-2959. DOI: `10.1007/s10661-015-4994-4`. [Online]. Available: `https://doi.org/10.1007/s10661-015-4994-4` (visited on 06/06/2020).

[9] ——, "Statistical road classification applied to stratified spatial sampling of road traffic noise in urban areas," *International Journal of Environmental Research*, vol. 10, no. 3, Jul. 2016. DOI: 10.22059/ijer.2016.58760. [Online]. Available: http://doi.org/10.22059/ijer.2016.58760 (visited on 05/04/2020).