Traffic Analysis Nice

June 1, 2021

1 Introduction

With the data at disposition of different areas of the city, the goal of the project is to analyze the traffic of Nice area in the year 2019 and 2020. The aim is to identify different seasonal and weekly paths of the traffic over the periods considered, and try to study the impact of the restrictions due to Covid2019 in the traffic dynamic. The data are used with respect their temporal order, daily time series are created with respect to the variables at disposition. In the first section the techniques and algorithms used are presented with a theoretical approach. In the second section Promenade Des Anglais data are considered, after having explain the pre-process implemented result are showed for both 2019 and 2020.

2 Methodology

In order to identify seasonal and weekly traffic paths, using daily time series, clustering procedures are proposed. Clustering is the task of dividing the time series data into a number of groups such that data in the same groups are more similar, present analogous behaviours during the day, than those in other groups. To better identify the path represented by a group (cluster) a barycenter, also call centroid, is computed. Since I am dealing with temporal data I have to define a strategy in order to compare different series both for assign each series in a cluster and update the centroid. I would need a dissimilarity measure that is able to "match" a point of time series x even with "surrounding" points of time series y. The Euclidean distance, assumes the $i-th$ point of the x series is aligned with the $i-th$ point of the y series, it will produce a pessimistic dissimilarity measure. Two clustering procedures are discussed below: Soft-Dynamic Time Warping with K-means algorithm and Global Alignment kernel (GAK) inside of a kernel K-means algorithm. The two procedure are connected each other and can be used together as showed later.

In addition to futher highlight long-term trends in the traffic behaviour a simple moving average calculation is taken in account. This techniques allows to analyze temporal data by creating a series of averages of different subsets of the full data set that delete short-term fluctuations.

2.1 Soft-Dynamic Time Warping and K-means algorithm.

Dynamic Time Warping is a technique to measure similarity between two temporal sequences considering not only the temporal alignment but every binary alignment of the two series. For example a similar traffic condition based on different variables could be recognized in different hours of the day in two different series. The calculation of the DTW similarity involves a dynamic programming algorithm that tries to find the optimum warping path between two series under certain constraints.

Given two multivariate series that corresponds to two different days:

$x \in R^{d \times n}$ and $y \in R^{d \times n}$ valued in R^d (d dimension of the multivariate time series).

Consider a function in order to compare different points of the two series ($x_i \in R^d$ and $y_j \in R^d$) $d : R^d \times R^d \Rightarrow R$, such as $d(x_i, y_j) = (\sum_{i,j=1}^{n} (|x_i - y_j|^p)$, where usually $p = 2$ and $d(x_i, y_j)$ is the quadratic Euclidean distance between two vectors.

A matrix of similarity is computed:

$\Delta(x, y) := [d(x_i, y_j)]_{i,j} \in R^{n \times n}$

$\Delta(x, y)$ can also be defined as local cost matrix, such a matrix must be created for every pair of series compared.

The DTW algorithm finds the path that minimizes the alignment between x and y by iteratively stepping through $\Delta(x, y)$, starting at $[d(x_i, y_j)]_{1,1}$ (bottom left) and finishing at $[d(x_i, y_j)]_{n,n}$ (upper right). and aggregating the cost [1]. At each step, the algorithm finds the direction in which the cost increases the least allowing 3 elementary moves $\rightarrow$, $\uparrow$, $\nearrow$. To improve the discrimination between different warped paths a chosen constraint can be specyfied. This constraint typically consists in forcing paths to lie close to the diagonal of the local cost matrix [2].

By considering $A_{n,n} \subset \{0, 1\}^{n,n}$ the set of all binary alignment between two time series x and y of the same lenght n, the DTW similarity measure reads as follow:

$$DTW(x, y) = \min_{A \in A_{n,n}} \langle A, \Delta(x, y) \rangle \tag{1}$$

This creates a warped "path" between x and y that aligns each point in x to the nearest point in y.

However I can not define dynamic time warping as a distance because does not satisfy the triangular inequality, moreover it is not differentiable everywhere due to the min operator.

Soft-Dynamic Time Warping is a variant of DTW that is differentiable. It uses the log-sum-exp formulation [3]:

$$DTW^{\gamma}(x, y) = -\gamma \log \sum_{A \in A_{n,n}} exp(-\frac{\langle A, \Delta(x, y) \rangle}{\gamma}) \quad where \quad \gamma \geq 0 \tag{2}$$

Despite considering all alignments and not just the optimal one, soft-DTW can be computed in quadratic time $O(d \times n^2)$ as DTW, however as DTW soft-DTW does not satisfy the triangular inequality. Soft-DTW is a symmetric similarity measure, it supports multivariate series as DTW, and it can provide differently smoothed results by means of a user-defined parameter γ.

The "path" created between x and y is smoother than the one created with DTW. Soft-DTW depends on a hyper-parameter γ that controls the smoothing. As showed in equation (3) DTW corresponds to the limit case when γ =0.

[1] Sardà-Espinosa:"Comparing Time-Series Clustering Algorithms in R Using the dtwclust Package" chapter 2

[2] Sakoe,Chiba: "Dynamic programming algorithm optimization for spoken word recognition," IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 26(1), pp. 43–49

[3] Cuturi,Blondel:"Soft-DTW: a Differentiable Loss Function for Time-Series"

$$DTW^{\gamma}(x,y) = \begin{cases} \min_{A \in A_{n,n}} \langle A, \Delta(x,y) \rangle, & \gamma = 0 \\ -\gamma \log \sum_{A \in A_{n,n}} exp(-\frac{\langle A, \Delta(x,y) \rangle}{\gamma}), & \gamma \geq 0 \end{cases} \tag{3}$$

SoftDTW is then using with Centroid-based clustering, such as K-means and K-medoids. These algorithms use an iterative way to create the clusters by moving data points from one cluster to another, based on a distance measure, starting from an initial partitioning. [4].

The soft-DTW is used in K-means algorithm to assign the series to the clusters and to upload the centroid of the cluster. Centroid in a cluster correspond to the multivariate time series $\in R^{d \times n}$ that minimizes the sum of the similarity measures between that time series and all time series inside the cluster. The centroid in a cluster is computed artificially, it does not correspond to a time series inside the cluster. Given the ts multivariate time series each of them composed by n daily observations the algorithm work as follow:

Algorithm $k - means clustering \ (T, K)$
$Input : T = (t_1, t_2, ..., t_{ts})$ *set of daily time series where the generic series* $t_i \in R^{d \times n}$
$Input : k$ *the number of clusters*
$Output :$ *(partion of time series in cluster :* $\{C_j\}_{j=1}^{k}$, *centroid of each cluster* $c_j \in R^{d \times n}$)
$p = 0$
Randomly choose k *series* $\in R^{d \times n}$ *and make them as initial centroids* $(c_1^{(0)}, c_2^{(0)}, ..., c_k^{(0)})$
repeat
Assign each time series to the nearest cluster using $\arg\min_j DTW^{\gamma}(c_j^{(p)}, t_i)$
$p = p + 1$
// **Centroid update**
for $j = 1$ *to* k **do**
Update the centroid $c_j^{(p)}$ *by considering the subset of series* $(t_1, t_2, ..., t_t$ *where* $t \leq ts)$ *that*
end for
until
$c_j^{(p)} \approx c_j^{(p-1)} \quad j = 1, 2, ..., k$
Return $c_1, c_2, ..., c_k$

Sometimes different initializations of the centroids lead to very different final clustering results. To overcome this problem the K-means algorithm is run 5 times with different centroids randomly placed at different initial positions. The final results will be the best output of the 5 times consecutive runs in terms of inertia [5].

[4] K-medoids clustering is very similar to the K-means clustering algorithm. The major difference between them is that while a cluster is represented with its center in the K-means algorithm, it is represented with the most centrally located data point in a cluster in the K-medoids clustering

[5] 5 different sets of randomly chosen centroids are used in the algorithm use. For each different centroid, a comparision is made about how much distance did the clusters move with respect to other centroids. For example if the clusters for a given centroid travelled small distances than it is highly likely that it is closest to the best solution and it is returned

2.2 Global Alignment Kernel and Kernel K-Means

Kernel methods use a kernel function in order to separate high dimensional data. Given two time series x,$y \in R^{d \times n}$ A kernel $k(,)$ is defined as

$$k(x, y) = \langle \rho(x), \rho(y) \rangle_\nu \tag{4}$$

where an appropriate non linear mapping $\rho : R^{d \times n} \rightarrow \nu$ is used to compute the dissimilarity measure $k(x, y)$ in some (unknown) embedding space ν (also know as latent space). This approach is called the "kernel trick". It is used to map time series in a higher-dimensional feature space ν by computing the inner products between the images of pairs of data using a nonlinear function ρ. The "kernel trick" perform computations in the embedding space ν by the inner product of the transformed vectors $\rho(x), \rho(y)$ in the higher dimensional space. For example, one can compute distance between x and y in ν as

$$\begin{aligned} \|\rho(\mathbf{x}) - \rho(\mathbf{y})\|_\nu^2 &= \langle \rho(\mathbf{x}) - \rho(\mathbf{y}), \rho(\mathbf{x}) - \rho(\mathbf{y}) \rangle_\nu \\ &= \langle \rho(\mathbf{x}), \rho(\mathbf{x}) \rangle_\nu + \langle \rho(\mathbf{y}), \rho(\mathbf{y}) \rangle_\nu - 2 \langle \rho(\mathbf{x}), \rho(\mathbf{y}) \rangle_\nu \\ &= k(\mathbf{x}, \mathbf{x}) + k(\mathbf{y}, \mathbf{y}) - 2k(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{5}$$

such computations are used in the Kernel K-means algorithm as illustrated later [6]. In Kernel K means before clustering, time series are mapped to a higher-dimensional feature space ν using a nonlinear functionρ, and then kernel K-means partitions the series by linear separators in the new space. Kernel methods require only a user-specified kernel $k(,)$, such as similarity function over pairs of data in raw representation.

The Global Alignment Kernel (GAK) is a kernel that operates on time series. Considering two time series x,$y \in R^{d \times n}$ I indicate with π a generic warping function, composed by a pair of integral vectors: π_1 for x and π_2 for y. π maps the generic alignment between two series from $\Delta(x, y)_{1,1}$ (bottom left) to $\Delta(x, y)_{n,n}$ (upper right) by allowing 3 elementary moves $\rightarrow$, $\uparrow$, $\nearrow$ as previoulsy showed. The (GAK) for a given bandwith σ is defined as:

$$k_{GAK}^{\gamma}(x, y) = \sum_{\pi \in \mathcal{A}(n,n)} \prod_{i=1}^{|\pi|} \exp\left(-\frac{\left\| x_{\pi_1(i)} - y_{\pi_2 i} \right\|^2}{2\sigma^2} \right) \tag{6}$$

where $|\pi|$ is the length of the generic alignment and the discrepancy between any two points x_i and y_i observed in x and y is the squared euclidean distance.

As reported in [7] the bandwidth σ can be set as a multiple of a simple estimate of the median distance of different points observed in different time-series of the training set, scaled by the square root of the length of time-series in the training set (In my case all daily multivariate time series have the same lenght previously indicated with n). The suggested estimation is

$$\sigma = median||x - y||\sqrt{n} \tag{7}$$

and it is available in **tslearn** [8].

[6] Dhillon,Guan,Kulis :"Kernel k-means, Spectral Clustering and Normalized Cuts"
[7] Cuturi:"Fast Global Alignment Kernels"
[8] Tavernard et.Al: "Tslearn, A Machine Learning Toolkit for Time Series Data"

The(GAK) is related to softDTW, it can be defined as the exponentiated soft-minimum of all aligment distances:

$$k^{\gamma}_{GAK}(x,y) = \sum_{A \in A_{n,n}} exp(-\frac{\langle A, \Delta(x,y) \rangle}{\gamma}) \tag{8}$$

where the γ hyperparameter that controls the smoothness in softDTW is linked with σ by

$$\gamma = 2\sigma^2 \tag{9}$$

Through this relation GAK can be used to estimate γ hyperparameter of softDTW.

Both GAK and softDTW, that scales in $O(d \times n^2)$, incorporates the set of all possible alignments $A \in A_{n,n}$ in the cost matrix $\Delta(x,y)$. For this reason they provide richer statistics than the minimum of that set, which is instead the unique quantities considered by DTW. However only Global Alignment Kernel (GAK) is positive definite [9].

Given a kernel $k^{\gamma}_{GAK}(,)$ and as input a set of daily time series $T = (t_1, t_2, ..., t_{ts})$ where the generic daily series $t_i \in R^{d \times n}$, the $ts \times ts$ matrix $K = (k^{\gamma}_{GAK}(t_i, t_j))_{ij}$ is called the Kernel matrix. Furthermore, the positive definiteness of kernel function translates $k^{\gamma}_{GAK}(,)$ in practice into the positive definiteness of the so called Kernel matrix K [10]. .

Positive definite Kernel Matrix can be used in kernel K-means algorithm.

The aim of kernel Kmeans algorithm is to partition the set of daily time series $T = (t_1, t_2, ..., t_{ts})$ in k clusters denoted as $\{C_j\}_{j=1}^{k}$. The algorithm introduce also weight for each time series denoted by w. A first significant difference (when compared to k-means) is that clusters centers are never computed explicitly, hence time series assignments to cluster are the only kind of information available once the clustering is performed. The algorithm only finds the "best" cluster representative in the embedding space. For the generic cluster j the "best" cluster representative is computed as:

$$m_j = \frac{\sum_{t_i \in C_j} w(t_i)\rho(t_i)}{\sum_{t_i \in C_j} w(t_i)} \tag{10}$$

The squared euclidean distance from $\rho(t_i)$ to m_j used to update the partitioning of the time series at every step read as:

$$\begin{aligned} \|\rho(t_i) - m_j\|^2_{\nu} &= \langle \rho(t_i) - m_j, \rho(t_i) - m_j \rangle_{\nu} \\ &= \langle \rho(t_i), \rho(t_i) \rangle_{\nu} + \langle m_j, m_j \rangle_{\nu} - 2 \langle \rho(t_i), m_j \rangle_{\nu} \end{aligned} \tag{11}$$

where the inner product of time series are computed using kernel function $k^{\gamma}_{GAK}(,)$ and contained in the kernel matrix K

The kernel K-means algorithm finally reads as follow:

[9] Blondel,Mensch,Vert:"Differentiable Divergences Between Time Series"
[10] Cuturi:"Positive Definite Kernels in Machine Learning" pag 9

Algorithm $Kernelk-meansclustering\ \ (T,K)$

$Input: K\ \ Kernel\ \ Matrix$

$Input: k\ \ the\ \ number\ \ of\ \ clusters$

$Output: C_1, C_2, ..., C_k\ \ (partion\ \ of\ \ time\ \ series\ \ in\ \ clusters)$

$p = 0$

$Randomly\ \ assign\ \ each\ \ series\ \ to\ \ a\ \ cluster\ \ (C_1^{(0)}, C_2^{(0)}, ..., C_k^{(0)})$

repeat

$For\ \ every\ \ cluster\ \ \{C_j\}_{j=1}^{k}\ \ find\ \ the\ \ "best"\ \ cluster\ \ representative\ \ m_j = \arg\min_m \sum_{t_i \in C_j} w(t_i)||\rho($

$Update\ \ the\ \ partitioning\ \ of\ \ time\ \ series\ \ by\ \ considering\ \ the\ \ "best"\ \ cluster\ \ representative\ \ pre$

$p = p + 1$

until

$C_j^{(p)} \approx C_j^{(p-1)} \quad j = 1, 2, ..., k$

$Return\ \ C_1, C_2, ..., C_k$

A Second difference with respect to k-means is that clusters generated by kernel k-means are phase dependent rather than in shape. For instance two days that present same level of traffic congestion (same shape) reached in different hours (different phase) can be assigned to different clusters. This is because Global Alignment Kernel is not invariant to time shifts, as demonstrated in [11] for the closely related soft-DTW, however the difference in time shift becomes less emphatic when considering series of the same lenght.

As mentioned before GAK and softDTW are closely related and can be used together. GAK can be used to tune the hyperparameter γ in softDTW by applying equation() and equation() while the latter can be used to computed the cluster center once kernel k-means is computed. The barycenter corresponds to the time series that minimizes the sum of the distances between that time series (computed artificially) and all the time series in the cluster. Given the set of series $(t_1, t_2, ..., t_t$ that belong to cluster j the centroid $c_j \in R^{d \times n}$ reads as:

$$c_j = \arg\min_j \sum_{i=1}^{t} DTW^{\gamma}(c_j, t_i) \tag{12}$$

2.3 Simple moving average

The moving average is commonly used with time series to smooth random short-term variations and to highlight other components such as trend present in the data. This technique is used to highlight the traffic trend of long periods (see most trafficated months), deleting the impact of the traffic peak in the morning and in afternoon. The simple moving average is the unweighted mean of the previous M data points. The selection of M (sliding window) depends on the amount of smoothing desired. By increasing the value of M improves the smoothing at the expense of accuracy. For a sequence of values, the simple moving average at time period t reads as follows:

$$SMA_t = \frac{x_t + x_{t-1} + x_{t-2} + ... + x_{M-(t-1)}}{M} \tag{13}$$

[11] Janati, Cuturi, Gramfort: "Spatio-Temporal Alignments: Optimal transport through space and time,"

3 Promenade Des Anglais

The data at disposition are collected from 9 detectors positioned on the Promenade in both directions. 3 detectors collect the traffic in the north direction and 6 detectors collect the traffic in the south direction (facing the sea). Two period of time are registered from 01/06/2019 to 31/12/2019 and from 06/02/2020 to 30/12/2020. Two variables are reported for each lane of the road: the flow (n° veh/h) indicated as $q(x,t)$ and the occupancy rate of the detector in percentage indicated as $o(x,t)$. The data are recorded every minute. Rough data needs to be treated and aggregated. Here a list of operation computed in order to study the dynamics in both intervals time 2019 and 2020:

1. Duplicate removal.
2. Mark as missing observations of the occupancy rates that are greater than 50%.
3. Mark as missing observations of the flow that are greater than 60 veh/h and have an occupancy rate equal to 100%.
4. For every detector all lanes are aggregated together (any lane containing NaN, cause the resulting output column flow or occupancy to be NaN as well).
5. By looking at the fundamental diagram outlier observations are removed.
6. Fill jumps in the series due to the omission of data with Nans values for both flow and occupancy rate. 1430 omissions in 2019 period and 2401 omissions in 2020 period.
7. 6-minute averages aggregation to mitigate the impact of traffic lights.
8. Compute the percentage of missing values in the series in order to decide which detectors can be used in the analysis.
9. Fill Nans values of usable detector with an imputation procedure.

To apply the clustering procedure the NaNs values must be filled. The imputation procedure takes in account the temporal order of observations, thus if there are too many consecutive NaNs to fill the procedure is not applied. The procedure reads as follow:

1. If an isolated Nan value is recognized (both previous and next observations are present), the observation before the Nan value is propagate forward to fill the missing value.
2. If the number of consecutive Nan values is lower than 20 (2 hours). The consecutive Nans are filled with a linear method based on the temporal alignment of the previous observations available. Otherwise if the number of consecutive Nans exceed 2 hours the imputation is not performed.

Time series, flow and occupancy rate, are preprocessed using normalization over all periods (51360 observations for 2019 and 66480 observations for 2020 6-minute averages). This scaler is such that each output time series is in the range [0,1] allowing to have identical scales for time series with originally different scales (veh/h and \$ %\$):

$$q_{norm}(t,x) = \frac{q(t,x) - MIN}{MAX - MIN}$$

.

$$o_{norm}(t,x) = \frac{o(t,x) - MIN}{MAX - MIN}$$

.

The MinMaxScaler function from the Python machine learning library **Scikit-learn** [12] trasforms each values of the time series proportionally within the range [0,1] preserving the shape. Density and flow scaled series are not amplitude invariant, they do not have the same standard deviation (reached instead by using standardization).

The inverse_transform method of MinMaxScaler, that undo the scaling of a data point according to feature_range,

$$q(t,x) = q(t,x)_{norm} * (MAX - MIN) + MIN$$

$$o(t,x) = o(t,x)_{norm} * (MAX - MIN) + MIN$$

is used in the centroids representations to have a better comprehension of paths, no more in the [0,1] range.

Despite the MIN MAX for $q(t,x)$ and $o(t,x)$ are defined over the years 2019 and 2020, the clustering procedure is applied to the ts daily multivariate time series with n daily observations (where n is equal to 240 observations with 6-minute averages). The choice of scaling the two varaibles with respect to the entire periods not with respect to the single daily time series is done to preserve variability with respect to different days of the week. For example I simple assume that the MAX flow reached on Sunday or Saturday is really low compared to MAX flow reached on another working day.

To create the ts daily normalized multivariate time series a simple **reshape** procedure is applied:

$[ts * n, 2] \implies [ts, n, d]$ where $d = 2$ is the dimensionality of the daily multivariate time series (flow and occupancy rate).

3.1 2019

For 2019 period only 3 detectors contain all data from 1/6 to 31/12. A list of the main problem reported for the other detectors is available in the Appendix. The three detector used in the analysis are C601,C009, C094. The locations are illustrated in Figure 2.1 and Figure 2.2.

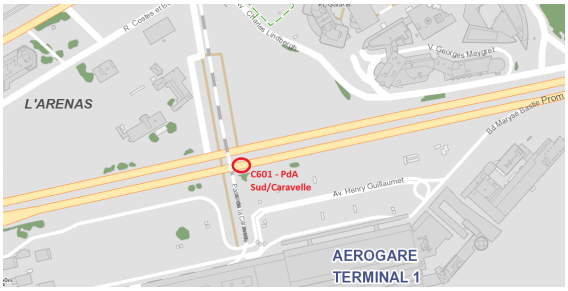

Figure 2.1 Location of C601 detector

[12] Pedregosa et.Al "Scikit-learn: Machine Learning in Python"

Figure 2.2 Location of C009 and C094 detector

3.1.1 C601

To have an idea of the dynamic of the traffic over all period available the simple moving average with a window size of one week is computed for C601 detectors located near the airport, terminal 1. As showed in Figure 3.1 the traffic in term of $q(t, x)$ (number of veh/h) presents seasonal behaviours. I can distinguish three different level of traffic also detected by the kernel K-means algorithm in Figure 3.2. The most trafficated periods are from the start of June until the second week of July and from the start of September to the third week of October. The less trafficated period is the summer that coincide roughly with the school break: from 5/7 to 2/9. The winter period from the third week of October to the third week of December (before Christmas Holidays) has a low level of traffic compared with the September period.

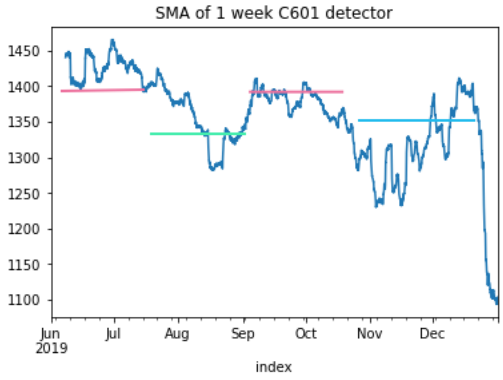

Figure 3.1 Simple moving average 2019 period

In Figure 3.2 the Global Alignment Kernel is seeded in the kernel K-Means algorithm with four clusters. 214 bi-variate (flow and occupancy rate) daily time series are taken as input. The Algorithm in addition to the three seasonal behaviours listed above recognize also the no-working days (Saturday, Sunday and Public Holidays).

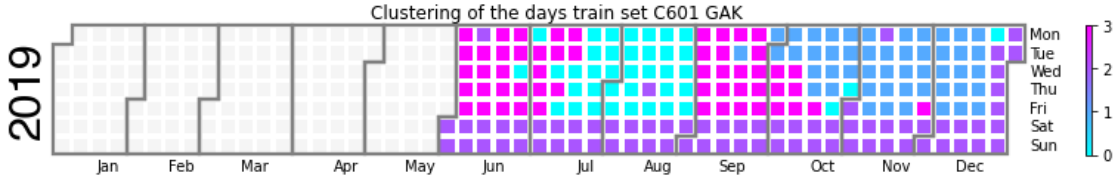

Figure 3.2 Kernel K-means, four clusters 2019 period

In Figure 3.3 a random sample of bi-variate time series for each cluster is represented. To better render each cluster, after all time series are assigned with Kernel K-Means, the centroids of the clusters are computed using softDTW equation().The γ parameter is set to 41.5936. The first cluster $k = 0$ identifies the summer in which the peak of the morning is flatter than the ones present in the second $k = 1$ and fourth $k = 3$ cluster for both flow and occupancy rate. The second cluster $k = 1$ identifies the winter behaviour, where the occupancy rate in the range 10:00 - 16:00 is lower than the ones present in cluster $k = 3$. The third cluster $k = 2$ portrays the dynamic of the traffic during no-working days, where during the morning hours 7:00 - 10:00 the traffic level remains low. The fourth cluster $k = 3$ describes the most trafficated period in which even outside traffic peaks in the morning and in the afternoon the occupancy rate register higher values with respect to the other clusters.

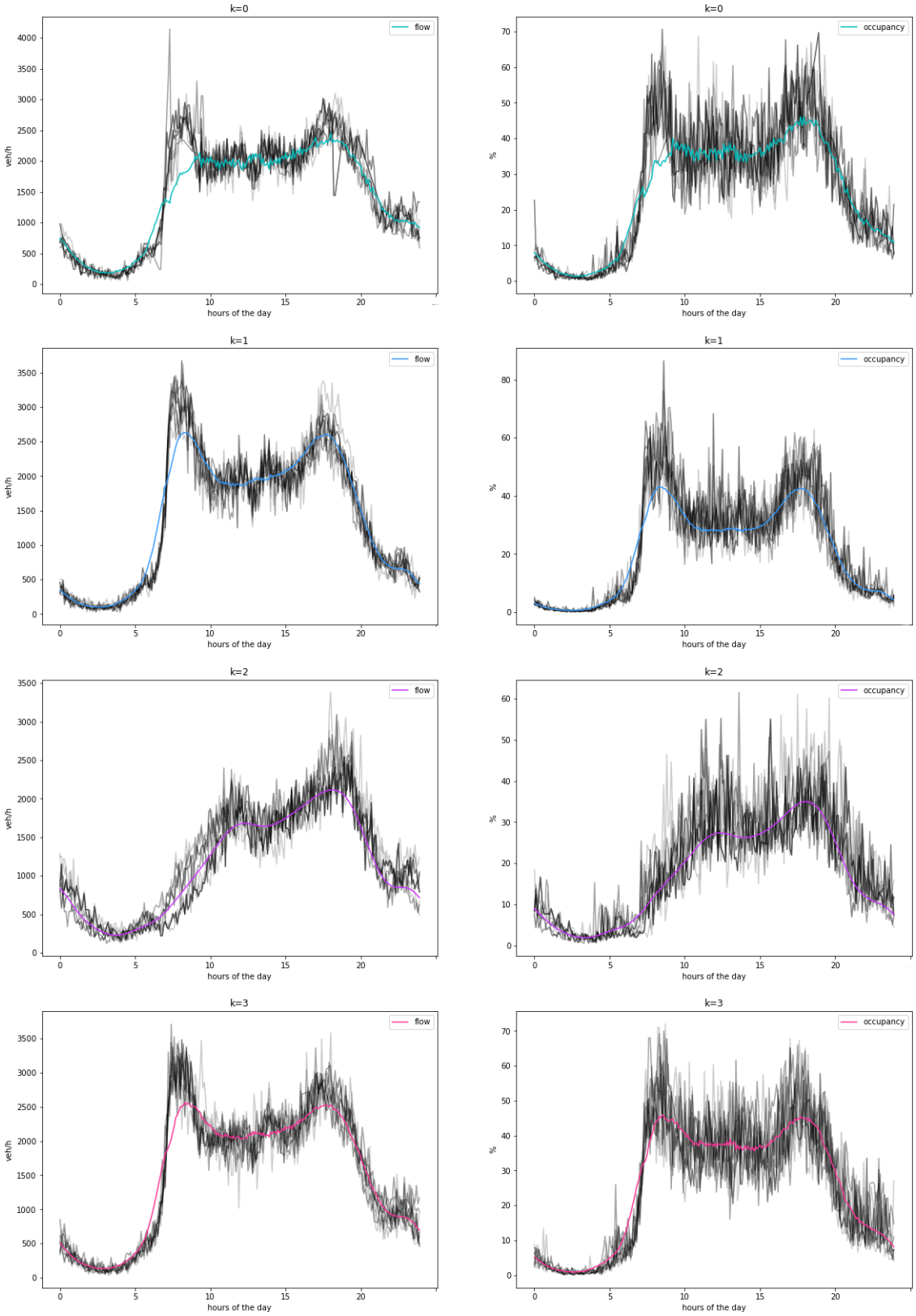

Figure 3.3 Centroids of Kernel K-means, four clusters 2019 period, computed with softDTW

In Figure 3.4 to better distinguish the behaviour of the first $k = 0$ and fourth $k = 3$ cluster weekly series of flow (veh/h) of August and September are compared. The September series presents a higher level of traffic especially in the first part of the day.

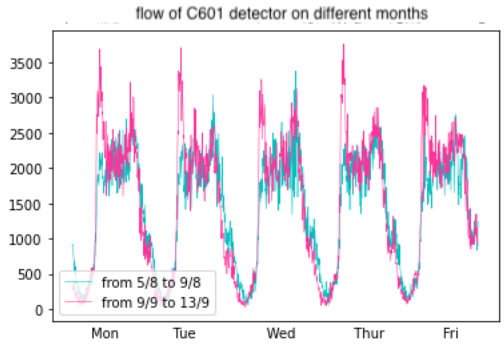

Figure 3.4 Comparison of working days in a week of August and September 2019

In Figure 3.5 only the subperiod from 2/9 to 21/12 is considered. The Global Alignment Kernel is seeded in the kernel K-Means algorithm with four clusters, and 111 bi-variate (flow and occupancy rate) daily time series are taken as input. Despite some weekend days form a unique cluster, the seasonal behaviour is once again confirmed with a separation between September and the winter period starting from the third week of October. To separate better these two behaviour in Figure 3.5.2 weekly series of occupancy rate of September and November are compared. As already mention the occupancy rate in the interval 10:00 - 16:00 is the main difference between the two period.

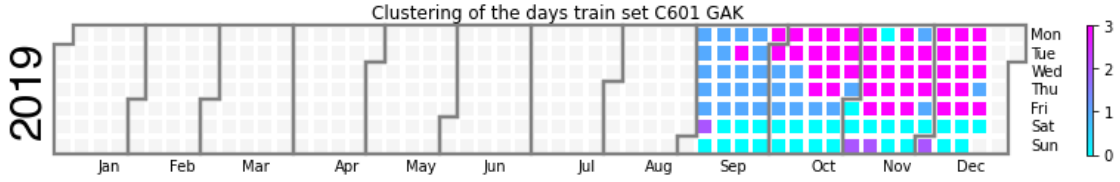

Figure 3.5 Kernel K-means, four cluster from 2/9/2019 to 21/12/2019

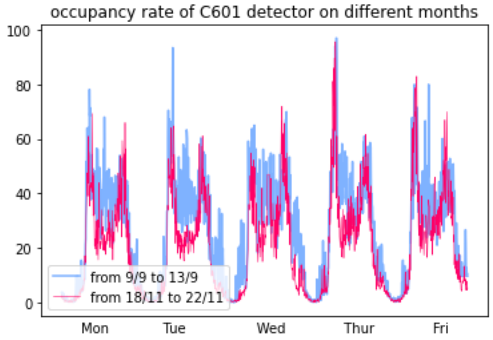

Figure 3.5.2 Comparison of working days in a week of September and November 2019

3.1.2 C009/C094

In this section results for C009/C094 are showed. C009 and C094 are used in this section without any distinction, due to their proximity. The algorithm and the moving average indicate a different behaviour of the traffic especially in the summer months compared with the C601. This mainly due to the location of the detectors close to the city centre of Nice and the beach as illustrated in Figure 2.2. In particular comparing Figure 3.5.5 with 3.1, in which the moving average of one week for C009 and C601 detector are computed, the month of August in Figure 3.5.5 does not present an inflection in terms of flow in the two centrals weeks.

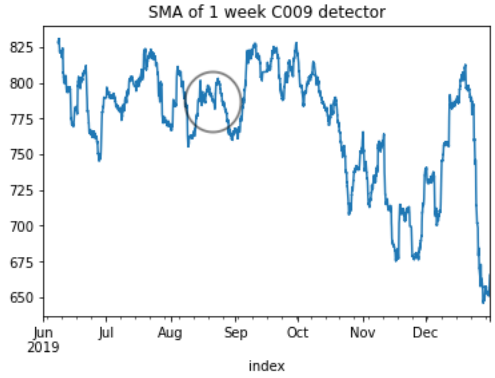

Figure 3.5.5 Simple moving average 2019 period

In Figure 3.6 the softDTW is seeded in the K-Means algorithm with four clusters. The γ parameter estimated using equation () and equation () is set to 18.5761. Again 214 bi-variate (flow and occupancy rate) daily time series are taken as input. The Algorithm put together the month of August and Saturdays, while Sundays form an unique cluster. The remaining days of the week are represent by the other two clusters which mostly differ for the traffic dynamic in the range 10:00 - 16:00 and in the range 20:00 - 24:00.

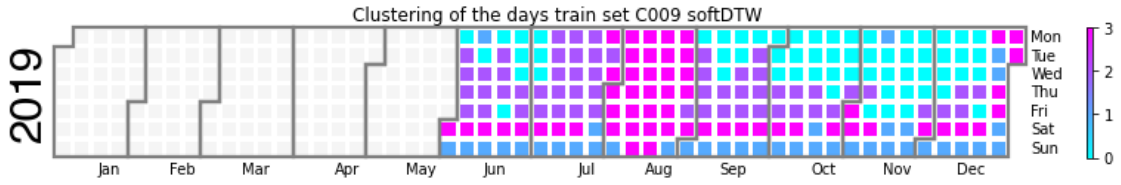

Figure 3.6 K-means, four clusters 2019 period

In Figure 3.7 a random sample of bi-variate time series for each cluster is represented with the corresponding centroid. The first $k = 0$ and the third $k = 2$ cluster identifies working days. I can recognize the traffic peaks in the morning hours 7:30-9:30 and in the afternoon hours 17-19. While the second $k = 1$ and the fourth $k = 4$ cluster represent the central part of the summer and no working days in which the majority of the traffic is distributed in the central hours of the day, without the morning peak. The level of traffic is more consistent in the fourth cluster $k = 3$ than the second one $k = 2$, with an higher level of flow and occupancy rate. In addition the level of traffic remains significant until 20:00 in the fourth cluster $k = 3$ while in the second one $k = 1$ starts to decrease before 20:00. The first $k = 0$ and third $k = 2$ cluster mostly differ for the level of traffic

in the intervals 10-16 and 20-24 in which the third $k = 2$ cluster has higher values than the first one $k = 0$. The third cluster $k = 2$ mostly maps the summer period June,July, September (considered as months of work) and the last parts of working days (Thursdays and Fridays) in October and December as showed in Figure 3.6.

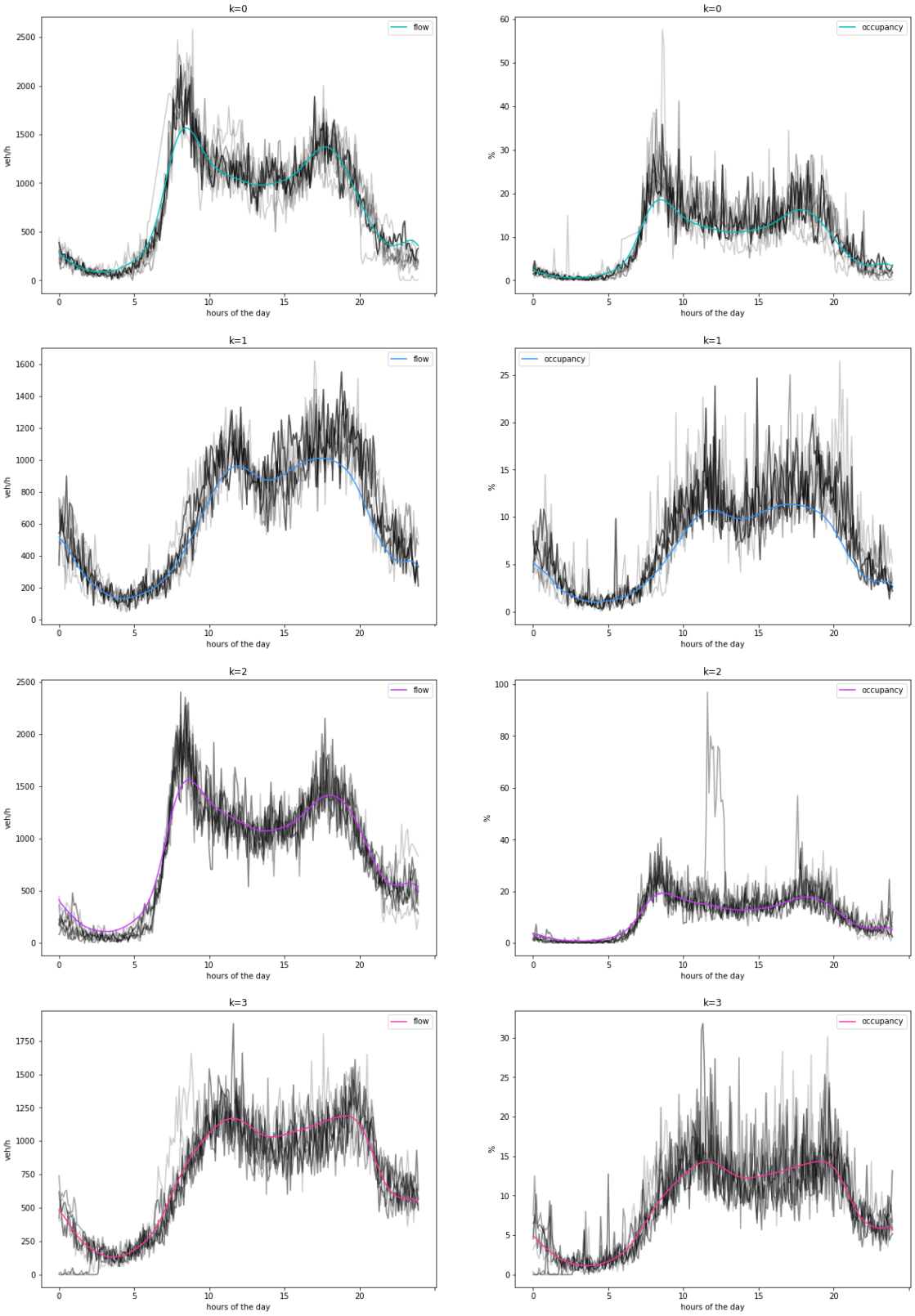

Figure 3.7 Centroids of K-means, four clusters 2019 period

To separate better the behaviours captured by the first $k = 0$ and third $k = 2$ cluster in Figure 3.5.2 weekly series of the flow of July and November are compared. I can see an higher level of traffic in the evening in July series than in November series.

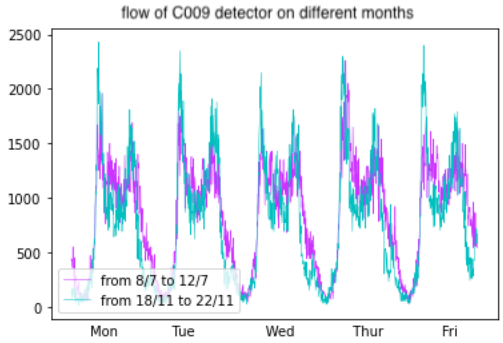

Figure 3.8 Comparison of working days in a week of July and November 2019

In figure 3.9 only the subperiod from 2/9 to 21/12 is taken in account. The softDTW is seeded in the K-Means algorithm with four clusters. The γ parameter estimated is set to. 111 bi-variate (flow and occupancy rate) daily time series are taken as input. The Algorithm recognizes a day with aberrant data 5/12 in which the detector was out of order for 6 hours. No working days form a unique cluster, fourth one $k = 3$, and working days are separated between the first $k = 0$ and second $k = 1$ cluster. The second cluster $k = 1$ maps mostly Thursdays and Fridays in the months of September, October and December, while the first cluster $k = 0$ identifies the others working days.

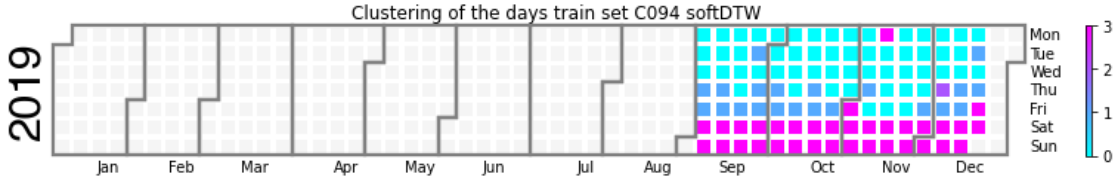

Figure 3.9 K-means,four cluster from 2/9/2019 to 21/12/2019

As showed in Figure 3.10, where a random sample of bi-variate time series for each cluster is represented with the corresponding centroid, the second cluster $k = 1$ presents higher level of traffic than the first one $k = 0$ in the morning 7:30-9:30, in the afternoon 17-19 and in the evening 21-22. This tendency can be seen in the flow and occupancy rate.

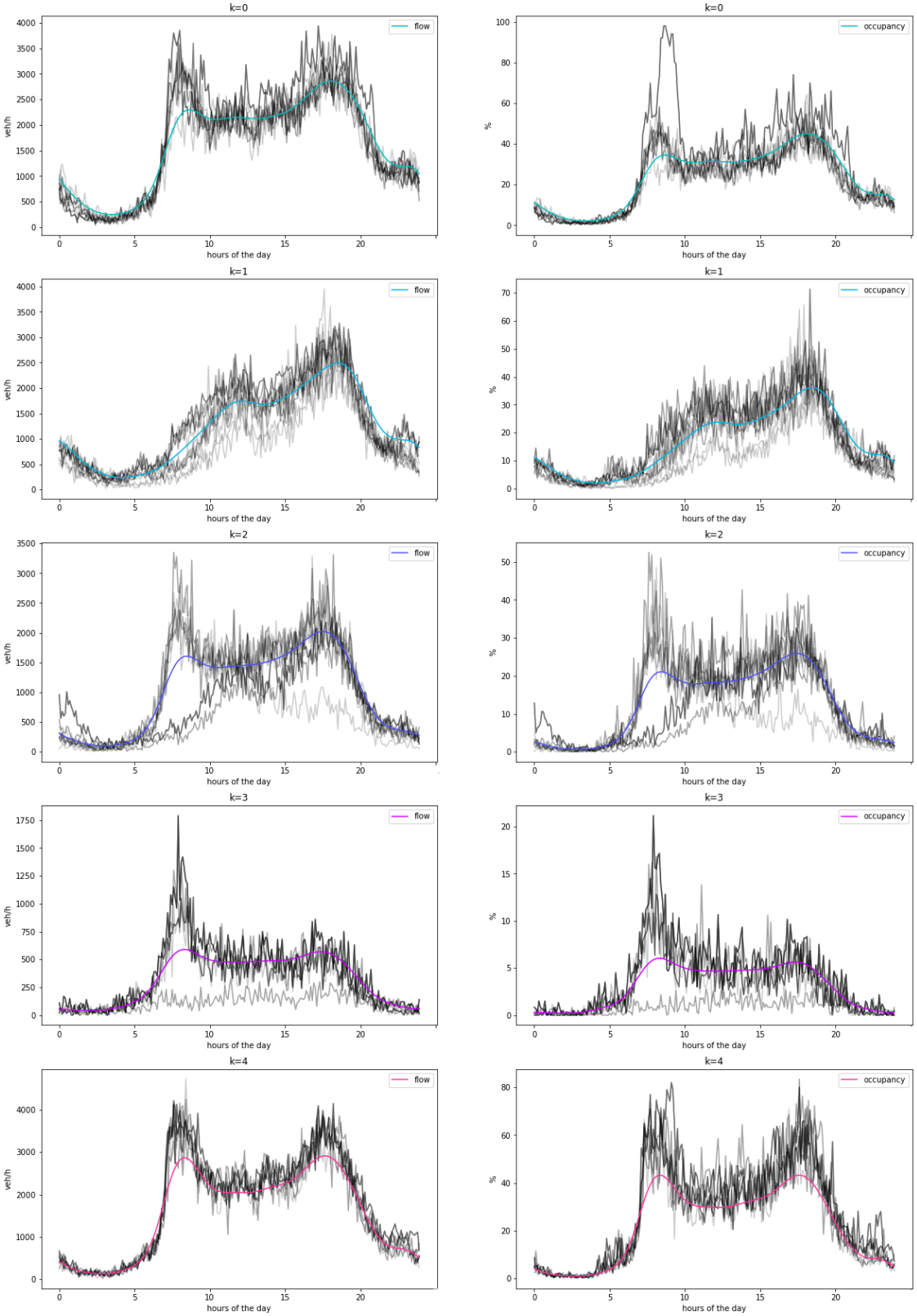

Figure 3.10 Centroids of K-means, four clusters from 2/9/2019 to 21/12/2019

3.2 2020

For 2020 period all detectors were out of order from 11/11 to 24/11. For this reason the effect of the second lockdown started the 30/11 is not competely studied. In addition only 6 detectors are considered due to others problems encountered in C077, C615 and C598 detectors see Appendix(). The locations of all detectors is illustrated in Figure 3.10.5.

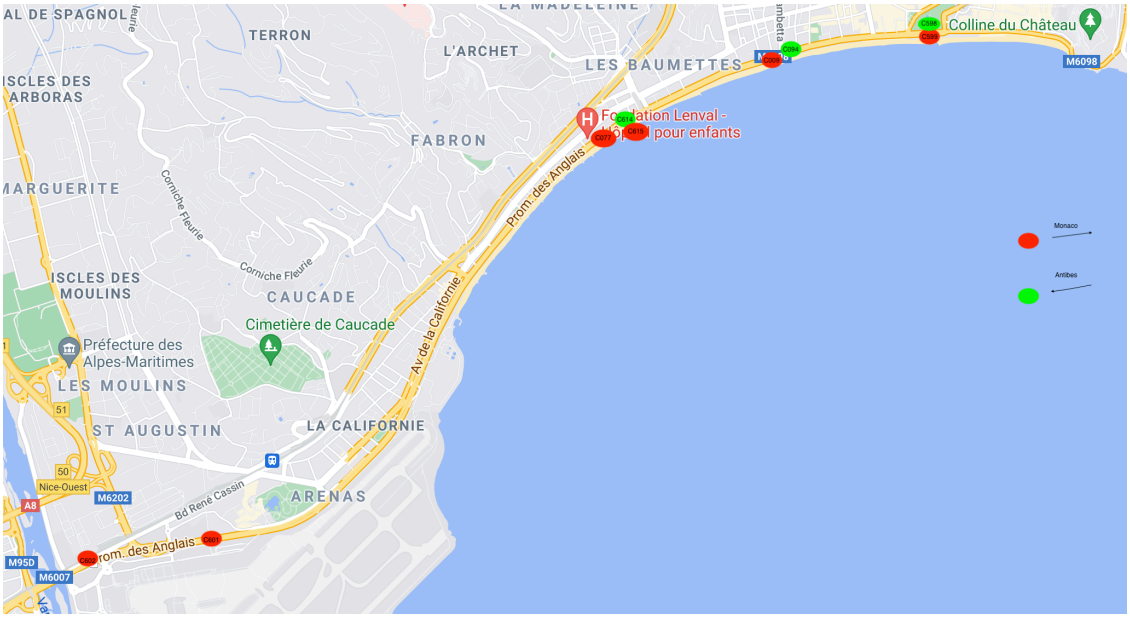

Figure 3.10.5 Location of detectors

The period considered for the analysis is from 6/2/2020 to 8/11/2020. The simple moving average with a window size of one week is computed for C601 detectors located near the airport, Terminal 1. As showed in Figure 3.11 the traffic in term of $q(t, x)$ (number of veh/h) presents seasonal behaviours. I can clearly see the impact of the first lockdown started the 17/03 and part of the second one started the 30/10. The results showed below are of C601 data, but repeating the experiment with the data of the other detectors available I obtained the same general tendency.

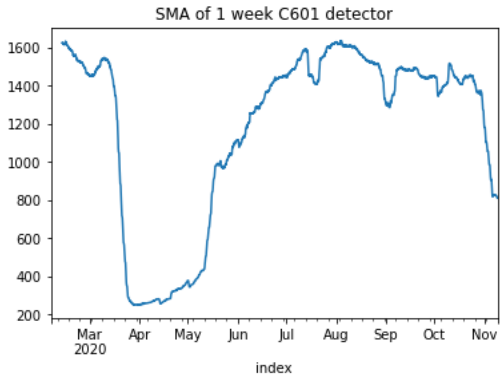

Figure 3.11 Simple moving average 2020 period

In Figure 3.12 the softDTW is seeded in K-Means algorithm with five clusters. The γ parameter

estimated is . set to bi-variate (flow and occupancy rate) daily time series are taken as input. The algorithm distinguish well the two phases of the first lockdown detected by the third $k = 2$ and fourth $k = 3$ cluster: from 17/03 to 11/05 "stay home phase"(fourth cluster $k = 3$) and from 11/05 to 02/06 where primary schools and some middle schools were allowed to reopen (third cluster $k = 2$). These two clusters ($k = 2$ and $k = 3$) identifies also the traffic behaviours in the first days analyzed of the second lockdown from 30/10, where the fourth $k = 3$ cluster maps no-working days and the third one $k = 2$ identifies working days. The first cluster $k = 0$ represents working days during the summer (from 6/7 to 29/8), the second cluster $k = 1$ identifies no-working days in no-restriction period and the fifth cluster maps working days out of restriction periods and summer.

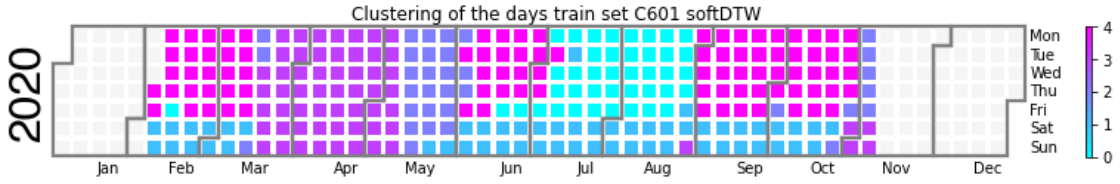

Figure 3.12 K-means, five clusters 2020 period

In Figure 3.13 a random sample of bi-variate time series for each cluster is showed with the corresponding centroid. The fourth cluster $k = 3$ ("stay home" period) has the lowest level of traffic, I can barely distinguish the traffic peak in the morning and in the afternoon. The third cluster $k = 2$(second part of first lockdown) presents a higher level of traffic than the fourth one $k = 3$, with an afternoon traffic peak greater than the one in the morning. The second cluster $k = 1$ represents no-working days (Saturdays, Sundays and public holidays) out of restriction periods, in which the highest level of traffic is reached between 17-20 and the morning peak is absent. The first cluster $k = 0$ and the fifth one $k = 4$ identifies the most trafficated days. The fifth cluster $k = 4$ differ from the first one $k = 0$ (July, August) for an higher peak of the traffic in the morning hours (7:30-9:30).

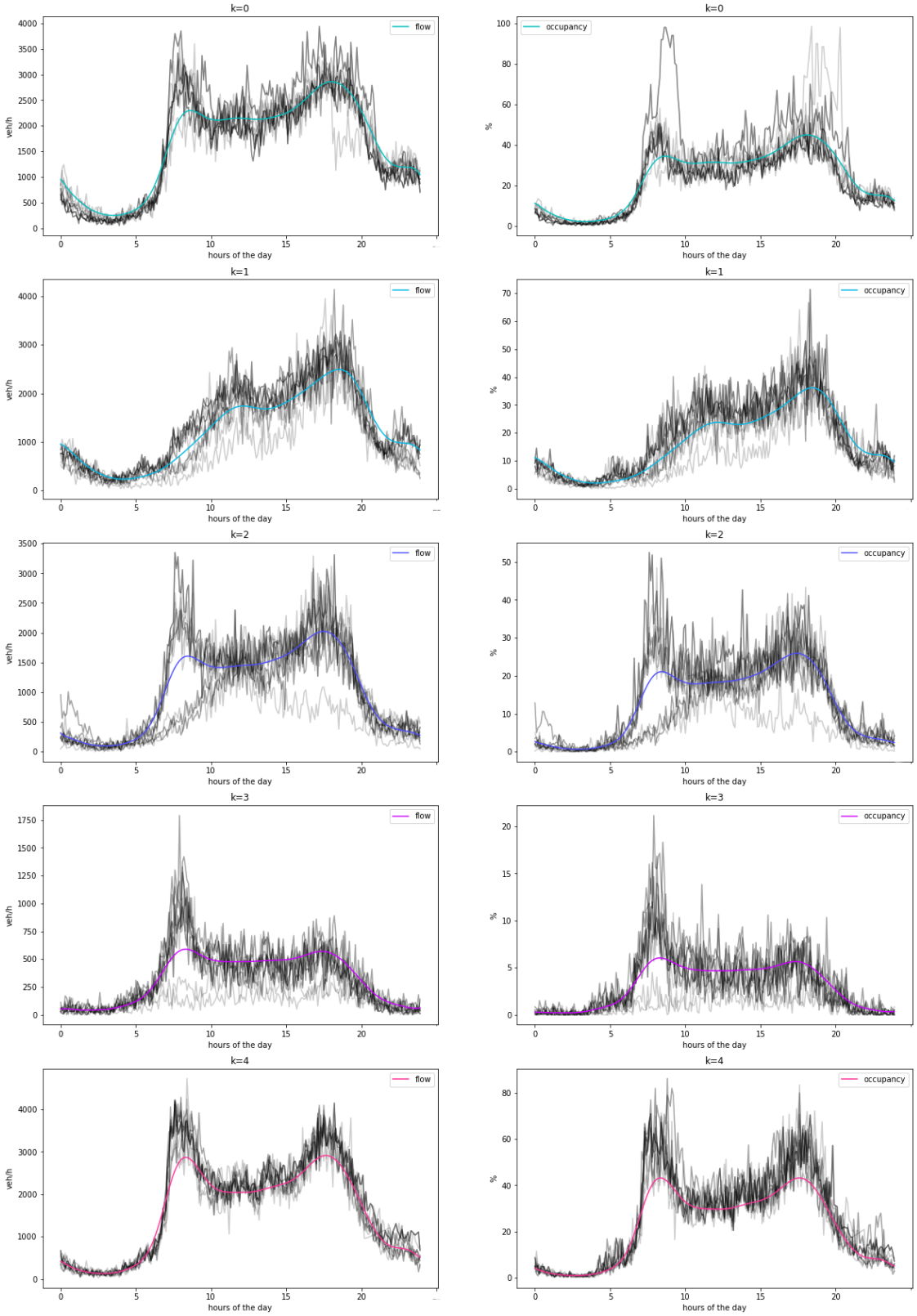

Figure 3.13 Centroids of K-means, five clusters 2020 period

In Figure 3.14 a comparison between a weekly flow series of April and May is proposed to better visualize the discrepancy between third $k = 2$ and fourth $k = 3$ cluster.The flow in the April series (third cluster $k = 2$) is really far from the level of flow in the May series (fourth cluster $k = 3$). The main impact of the restriction is in the "stay home" period, while from 11/05 the traffic level starts to grow.

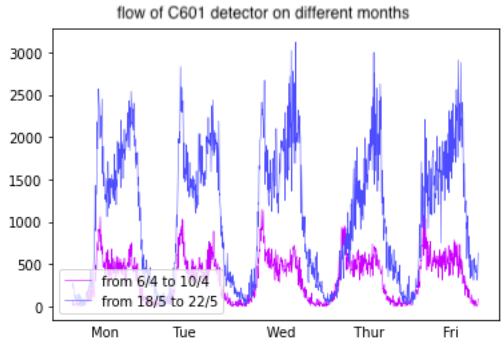

Figure 3.14 Comparison of working days in a week of April and May 2020

In Figure 3.15 a comparison between a weekly flow series of August and September is showed to better visualize the difference between the first $k = 0$ and fifth $k = 4$ cluster. The flow in September series (fifth cluster $k = 5$) presents peaks in the morning hours higher than the ones in August series (first cluster $k = 0$).

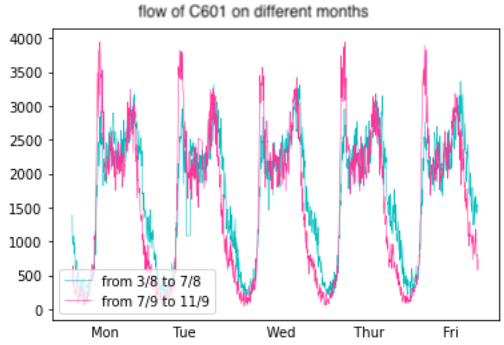

Figure 3.15 Comparison of working days in a week of August and September 2020

3.3 2019 vs 2020

Plotting the flow against the occupancy rate of C094 detector for 2019 and for 2020, I can see different layouts in the two years in Figure 3.18 and 3.19.
In Figure 3.19, representing 2019, data are more concentrated around an occupancy rate of 20 % and a flow of 1500 veh/h, while in Figure 3.20, representing instead the 2020, data are more scattered. Probably in 2019 the traffic data are more concentrated in an area of the plot than the one registered during 2020 due to the Covid restriction.

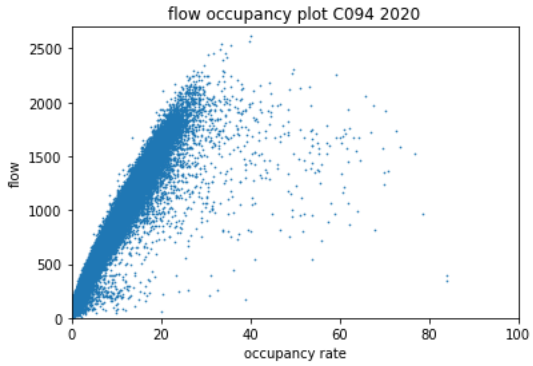

Figure 3.18 Fundamental diagram 2020

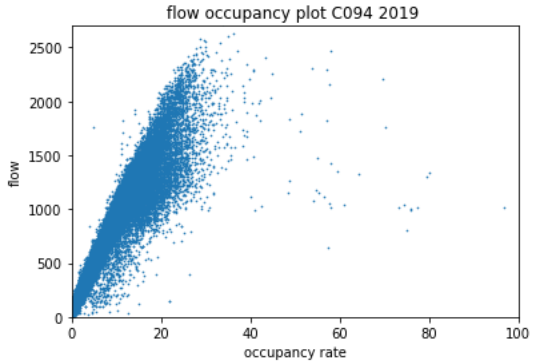

Figure 3.19 Fundamental diagram 2019

Despite different behaviour of traffic captured by cross sectional plot (flow against occupancy plot), when considering time series in specific periods of the years traffic dynamic seems to be equal. Restrictions applied during 2020 do not have a prolonged impact of traffic dynamic. In fact by comparing the weekly flow series of the second week of September for 2019 and 2020, in Figure 3.20, the two series overlap.

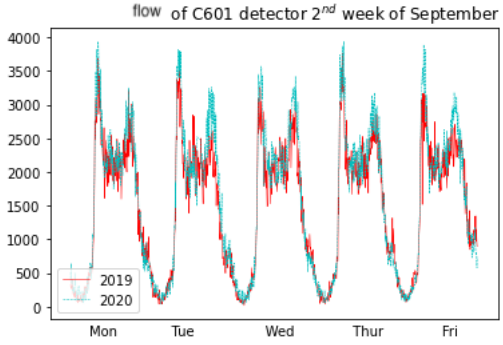

Figure 3.20 Comparison September 2020 and September 2019

4 Voie Mathis

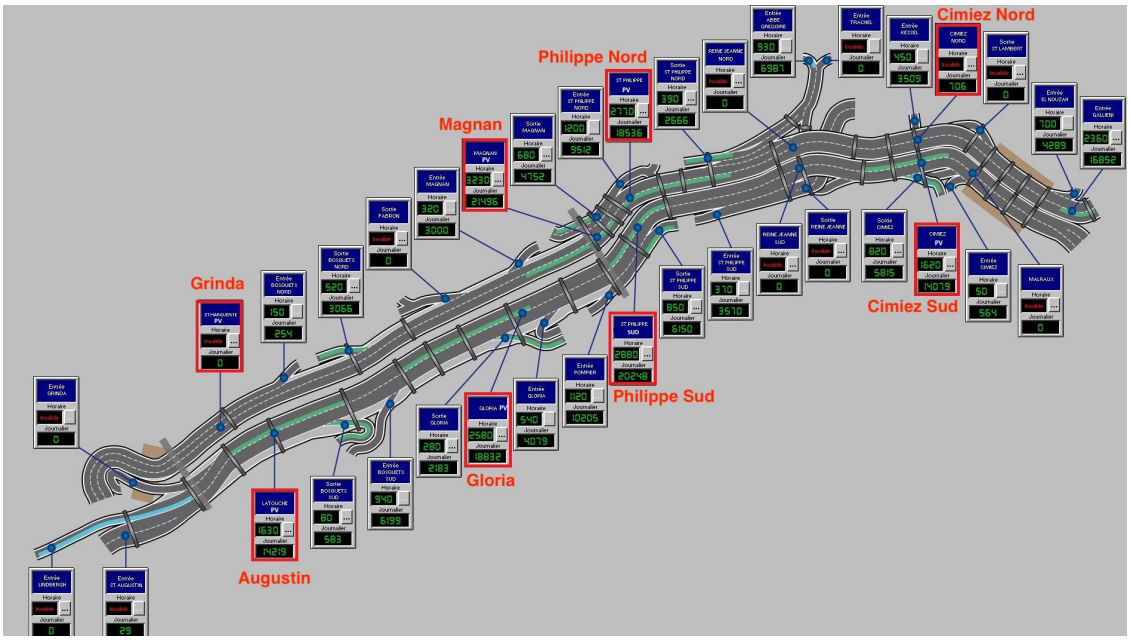

Figure 4.0 Location of detectors "Voie Mathis"

4.1 2019

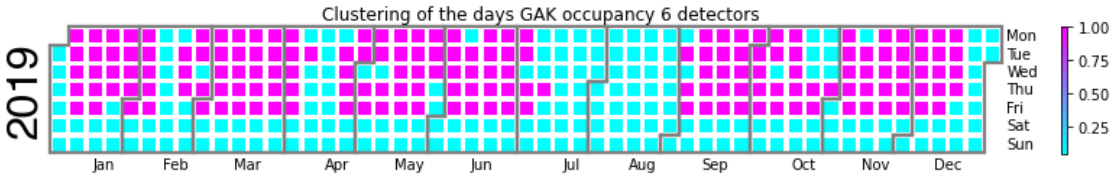

Figure 4.1 Kernel K-means, two clusters 2019

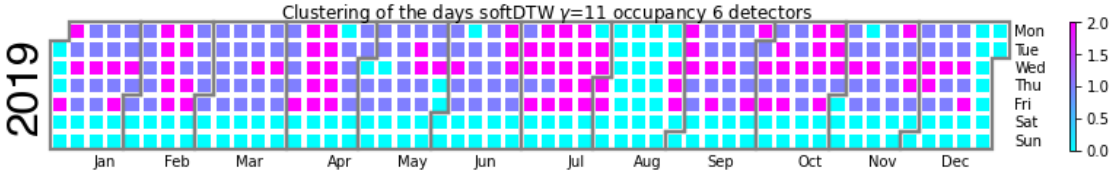

Figure 4.2 K-means, three clusters 2019

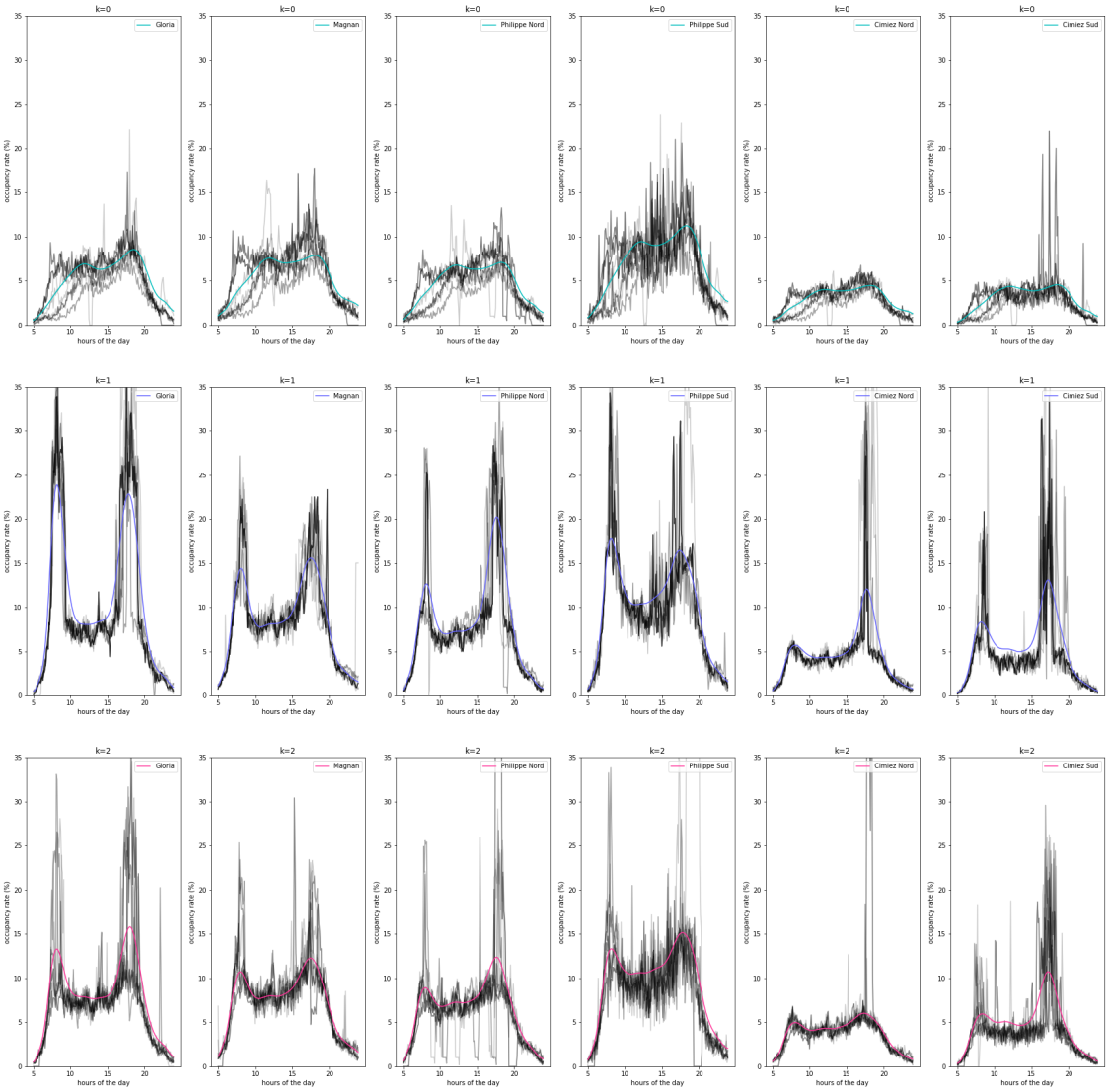

Figure 4.3 centroids of K-means, three clusters 2019

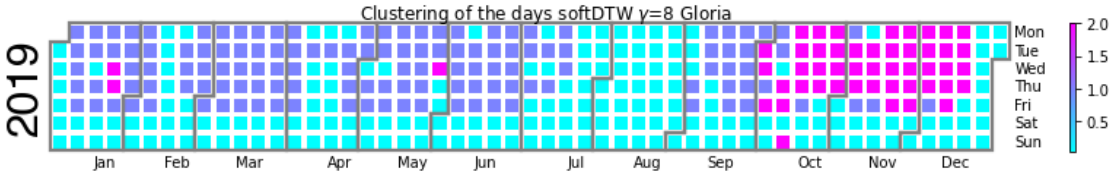

Figure 4.4 K-means, three clusters 2019 Gloria detector

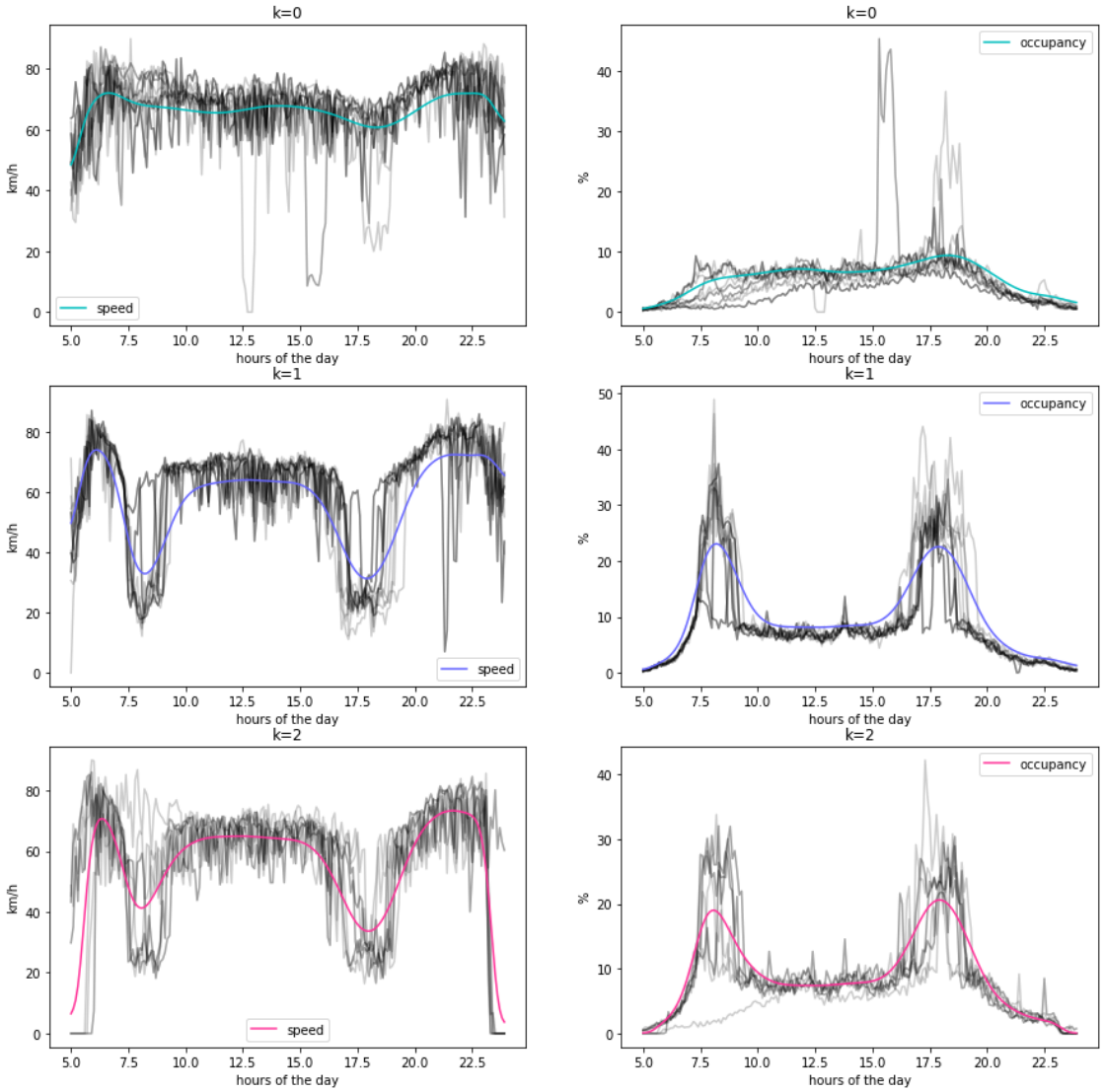

Figure 4.5 centroids of K-means, three clusters 2019 Gloria detector

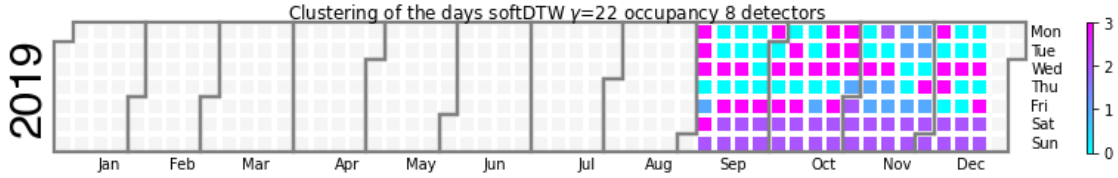

Figure 4.6 K-means, four clusters from 2/9/2019 to 22/12/2019: train set

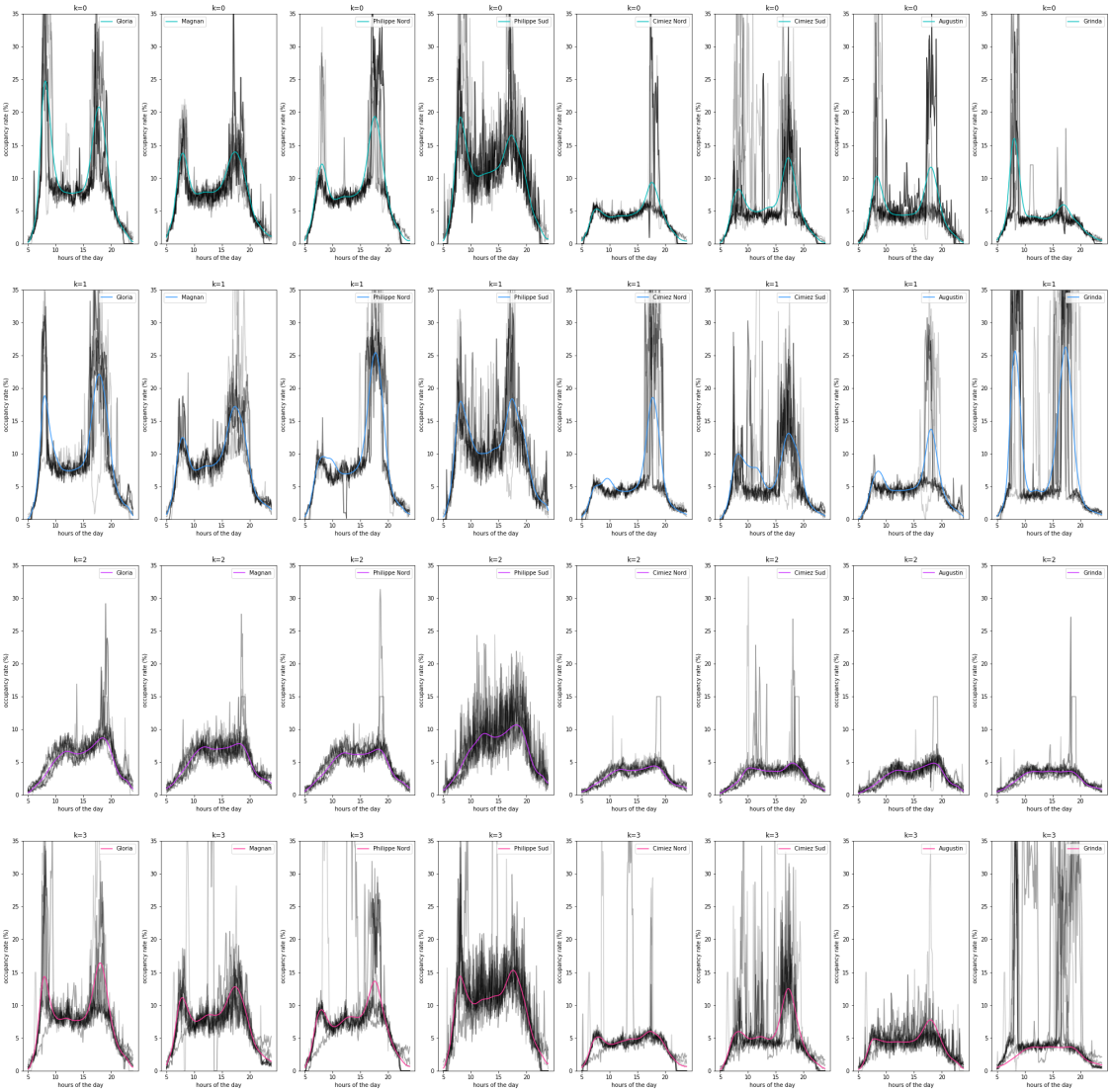

Figure 4.7 centroids of K-means, four clusters from 2/9/2019 to 22/12/2019

5 2020

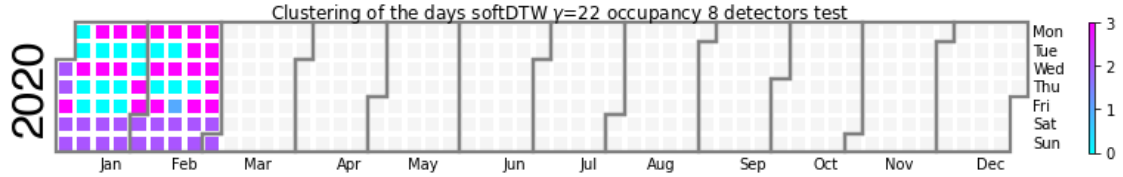

Figure 4.8 K-means, four clusters from 1/1/2020 to 1/3/2020: test set

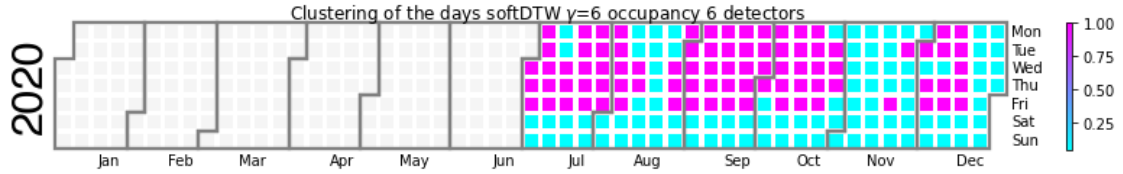

Figure 4.9 K-means, two clusters from 31/8/2020 to 31/12/2020

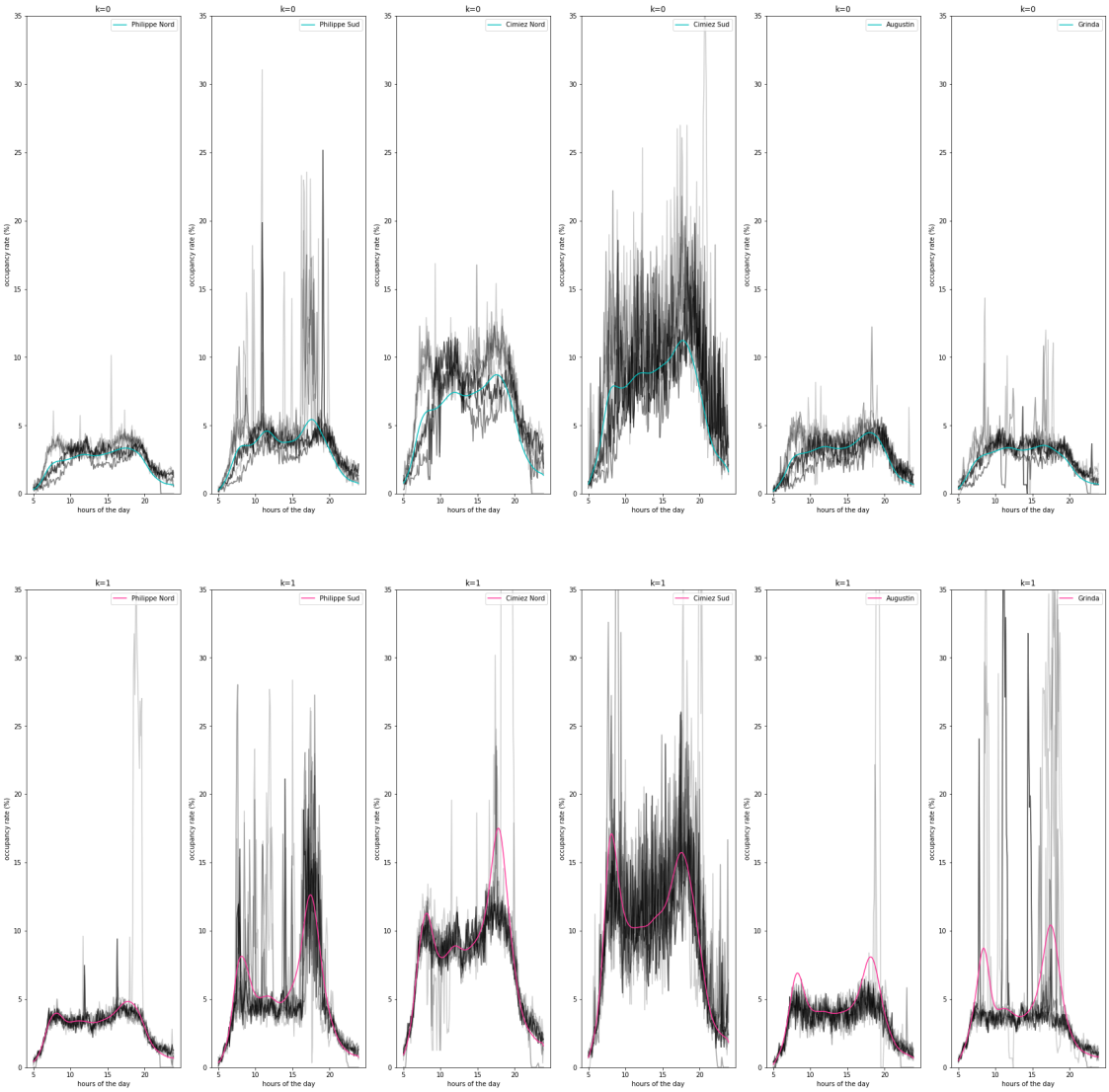

Figure 4.10 centroids of K-means, two clusters from 31/8/2020 to 31/12/2020

6 Appendix

6.1 Promenade Des Anglais out of order detectors

6.1.1 2019

- C602 detector: 4.32 $ % $ of missing values for both flow and occupancy rate. No data from 19/8 at 6 to 8:54. No data from 29/8 at 21:12 to 30/8 14:48. No data from 3/11 at 9 to 11:36. No data from 23/12 at 17:42 to 31/12.
- C077 detector: 56.72 $ % $ of missing values. No data registered from 1/6 to 30/9 at 9:06.
- C614: 7.05 $ % $ of missing values. No data available from 4/6 to 6/6 and from 21/8 to 3/9.
- C615: 46.81 $ % $ of missing values for the occupancy rate. No data registered from 1/6 to 30/9 at 9:06. 7.04 $ % $ of missing values for the flow. No data registered from 4/6 at 17:00 to 6/6 at 15:00 and from 21/8 at 13:12 to 3/9 at 16:18 for the flow.
- C599: 12.68 $ % $ of missing values. No data available from 12/7 to 15/7 and from 11/10 to 30/10.
- C598: 64.39 $ % $ of missing values for both flow and occupancy.

6.1.2 2020

- C615: 30.15 % of missing values for the occupancy rate. No data registered for the occupancy from 6/4 to 21/6.
- C077 10.735 % of missing values for both flow and occupancy. No data from 24/2 at 22:24 to 28/2 at 8:30. No data from 8/9 at 15:24 to 10/9 at 9:48. No data from 12/9 at 13:06 to 16/9 at 14:42.
- C598: 8.24 % of missing values for both flow and occupancy. No data from 9/9 to 14/9 . No data from 3/10 to 6/10.

7 References

- Sardá-Espinosa, Alexis:“Comparing Time-Series Clustering Algorithms in R Using the dtwclust Package”.The R Journal (2019) https://journal.r-project.org/archive/2019/RJ-2019-023/RJ-2019-023.pdf.
- Cuturi, Marco and Blondel, Mathieu:“Soft-DTW: a Differentiable Loss Function for Time-Series”. Journal of Machine Learning Research (2018) https://arxiv.org/abs/1703.01541v2.
- H. Sakoe and S. Chiba:“Dynamic programming algorithm optimization for spoken word recognition,” in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 26, no. 1, pp. 43-49 (1978) https://www.irit.fr/~Julien.Pinquier/Docs/TP_MABS/res/dtw-sakoe-chiba78.pdf.
- Dhillon, Inderjit and Guan, Yuqiang and Kulis, Brian: “Kernel k-means, Spectral Clustering and Normalized Cuts”. Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 551-556 (2004) https://www.cs.utexas.edu/users/inderjit/public_papers/kdd_spectral_kernelkmeans.pdf .

- Cuturi, Marco:" Fast Global Alignment Kernels". Proceedings of the 28th International Conference on Machine Learning, ICML 929-936 (2011) https://icml.cc/2011/papers/489_icmlpaper.pdf .
- Tavenard, Romain and Faouzi, Johann and Vandewiele, Gilles and Divo, Felix and Androz, Guillaume and Holtz, Chester and Payne, Marie and Yurchak, Roman and Rußwurm, Marc and Kolar, Kushal and Woods, Eli:" Tslearn, A Machine Learning Toolkit for Time Series Data". Journal of Machine Learning Research (2020) http://jmlr.org/papers/v21/20-091.html .
- Blondel, Mathieu and Mensch, Arthur and Vert, Jean-Philippe:" Differentiable Divergences Between Time Series". Journal of Machine Learning Research (2020) https://arxiv.org/pdf/2010.08354.pdf .
- Cuturi, Marco:"Positive Definite Kernels in Machine Learning". Journal of Machine Learning Research (2009) https://arxiv.org/pdf/0911.5367.pdf .
- Janati, Hicham and Cuturi, Marco and Gramfort, Alexandre:"Spatio-temporal alignments: Optimal transport through space and time". Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (2020) https://arxiv.org/pdf/1910.03860.pdf .
- Treiber, Martin: "Traffic Flow Dynamics: Data, Models and Simulation". ISBN 978-3-642-32459-8. Springer-Verlag Berlin Heidelberg (2013).
- Pedregosa, Fabian and Varoquaux, Gael and Gramfort, Alexandre and Michel, Vincent and Thirion, Bertrand and Grisel, Olivier and Blondel, Mathieu and Prettenhofer, Peter and Weiss, Ron and Dubourg, Vincent and Vanderplas, Jake and Passos, Alexandre and Cournapeau, David and Brucher, Matthieu and Perrot, Matthieu and Duchesnay, Edouard and Louppe, Gilles:" Scikit-learn: Machine Learning in Python". Journal of Machine Learning Research (2012) https://arxiv.org/pdf/1201.0490.pdf .