Data Mining

INFO-H423

Hack my Ride

BAKKALI Yahya (000445166)
FALLAHI Amirmohammad (000460073)
HAUWAERT Maxime (000461714)
NIZERY Damien (000544100)

December 2021

Université Libre de Bruxelles (ULB)

Contents

1 Introduction

Public transport is a vital part of nowadays societies. In addition to allowing people to travel from a location to another location, it can really be useful to find solutions to some of today's environmental problems. If a public transport company provides good services, people will give it some credit, so that in addition to increasing its profit, urban traffic, energy consumption and air pollution can be decreased.

A profound study and good analysis of public transport companies' data and statistics can help the managers and the experts of these companies to provide a more accurate schedule, and it can also be useful for the users, to organize their daily program. In this project, data from Brussels' intercommunal transport company, **STIB/MIVB**[1], has been analyzed. The following analyses have been performed on data ranging from the *3rd September 2021* to the *23rd September 2021*:

- Vehicle's speed variation over time.
- Vehicle's delays at the different stops.
- Prediction on the vehicles' arrival time at a specific stop.
- Guessing of the transport mode of a given trajectory over STIB network.
- Reachable places within a time interval.

In this report, data exploration, data processing, implementation evaluation and also results of each task are explained in detail.

2 Data preparation

2.1 Steps

2.1.1 JSON to CSV

The dataset is composed of several JSON files which contain the location of all vehicles approximately every 30 seconds over a period of 3 weeks.

Analyzing the JSON files is far from easy with the tools at our disposal, and so the first step was to convert the JSON files to CSV files. The way the conversion was performed can be found below:

[1] "Société des Transports Intercommunaux de Bruxelles" in French or "Maatschappij voor het Intercommunaal Vervoer te Brussel" in Dutch

Before


```
{"data":[{
  "time":"1632409236387",
  "responses":[
    {"lines": [
      {"lineId": "1",
      "vehiclePositions": [
          {"directionId":
              "8161",
          "distanceFromPoint":
              1,
          "pointId": "8122"},
              ...]}, "lineId":
              ...]
    },{"lines":...}
 ],}, {...}, ... ]}
```


After

time	lineId	directionId	distanceFromPoint	pointId
1632409236387	1	8161	1	8122
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

2.1.2 Lines information

The data, generated in CSV format, was not sufficient to begin the analyses. The next step was to link the data to other information stored in other dataset files.

The information about the lines in the GTFS files were not easy to handle and were lacking some useful data, such as the distance between the different stops of the lines.

To this purpose, the Esri Shape file describing the map stops of the **STIB/MIVB** network was used. First, it was loaded using QGIS tools and then saved as a CSV file containing its base attribute values and a new distance field. This new field was computed for each stop and represents the distance to the start point of the line. The following formula was used to compute this distance in QGIS.

```
line_locate_point(geometry:=geometry(get_feature('Lines','LIGNE',"Code_Ligne")),point:=$geometry)
```

2.2 Main challenges

2.2.1 Incoherent stop IDs

The stop IDs given in the positions files do not seem to correspond directly to the stop IDs in the GTFS files. Some stops present in the different lines have an ID which include a letter. By performing a quick search on the positions files, none of these stops showed up, which led this issue to be investigated further.

First, two sets, **realStops** and **posStops** are created and filled the extracted stop IDs from the GTFS

files, and from the positions file, respectively.

Direct matching Matching directly the stop IDs of **posStops** with the stops of **realStops**

By removing these stop IDs, there are 21% of the stops that have not been successfully found.

Leading zeroes Adding leading zeroes to the stop IDs of **posStops** to get a four characters ID.

By removing these stop IDs, there are still 19% of the stops that are still not found.

There are not any stops in **realStops** which have the same ID without leading zeroes.

Letter Matching the stops of **posStops** with the stops of **realStops**, which are taken without the possible letter in their IDs.

By deleting these stop IDs, there are finally 4% of the stops that cannot be found.

There are stops in **realStops** which have the same ID with different letters, but they belong to different lines, which is not a problem as the vehicle positions have a specific line.

The stops that are left in **posStops** are stops that do not correspond to any of the stops of **realStops**. Following the STIB documentation, these stops are *technical* stops and must be ignored.

Result The stop IDs contained in the positions files underwent a certain trim function. To retrieve their real IDs, the verifications specified above are used. When an ID is still not found even after these verifications, it is treated as a *technical* stops, thus, ignored.

2.2.2 Variance

Another useful information to add to the positions files is the variance. Indeed, the variance shows in which direction the vehicles, in a line, go, which is important to get the order of the stops the vehicles should follow.

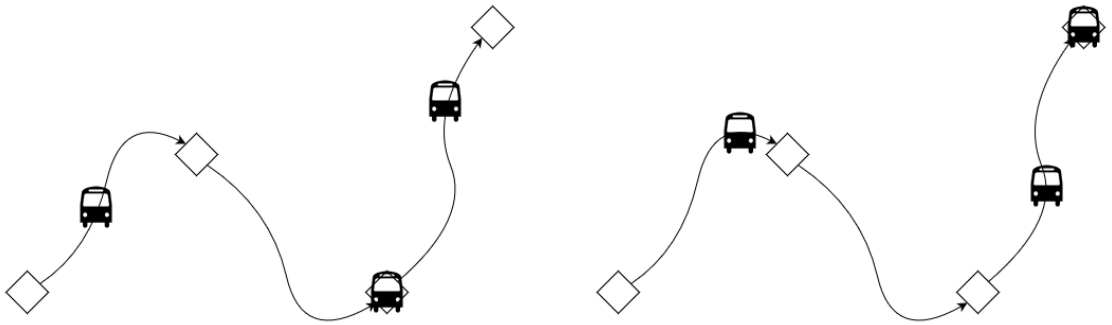
The way this information is computed by taking the two orders of the stops ID of a line, the first and second variance. Then, iterating backwards in the two lists at the same time and returning the first variance which matched the direction ID.

This works because normally there are no stops IDs in one variance that are also present in the other. There is only one exception, as certain lines have their last stop ID of a variance equal to the first stop ID of the other variance. The computation explained above takes into account this exception.

2.2.3 Vehicle ID

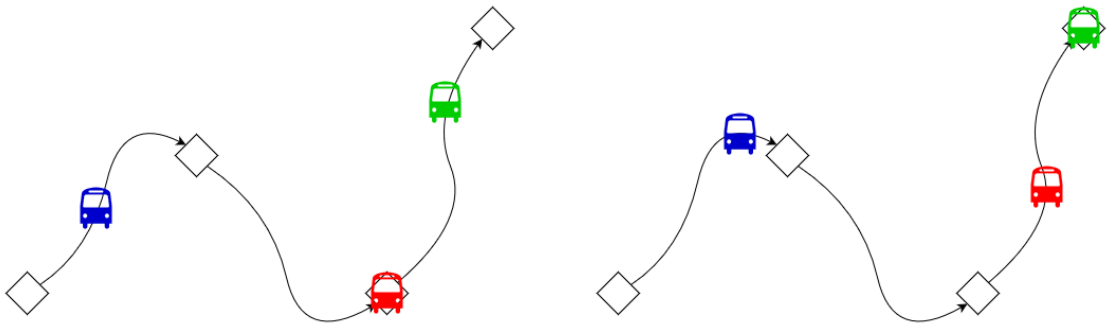
One of the most important yet difficult to get information is the vehicle ID. It is the ability of tracking a vehicle through different positions at different times.

It has been chosen to link the different positions to the nearest vehicles available. In other words, all the positions of a specific line and variance at a certain time T are taken and ordered by their distance from the start of the line. Then, all of these positions are tried to be associated with the available vehicles which have their last position the nearest from it but located before it on the line. It is assumed that a vehicle cannot go backwards. The notion of availability of a vehicle is defined such that a position with a time T cannot be linked to a vehicle which its last position has the same T. The positions which could not have been associated with another vehicle have been linked to new vehicles.



(a) T (b) T+1

Figure 1: Raw vehicles positions



(a) T (b) T+1

Figure 2: Labelized vehicles positions

For this information to be able to be computed as accurately as possible, some additional assumptions had to be made:

- The maximum speed a vehicle can have is 100 km/h.
- The vehicle cannot have the time of two consecutive positions separated by more than two minutes.

2.3 Cleansing

With all the new information computed and extracted, some erroneous or incoherent pieces of data can be detected and thus removed.

- Some positions have a direction ID which is not present in the specified line.
- Some positions have a **distance from point** bigger than the distance between the last stop passed and the next one.

In the first position file, 175624 positions were not successfully attributed a variance and 2119 positions had an invalid distance. It is negligible as 1292680 positions were valid.

3 Analysis of vehicle speed

It is pretty obvious that for calculating the speed of a vehicle, the displacement of that vehicle and the duration of that displacement are required. In other words, the following kinematic formula should be used for speed calculation:

$$\overline{V} = \frac{\Delta x}{\Delta t}$$

Where:

$$\begin{cases} \overline{V} & \text{indicates average speed (velocity)} \\ \Delta x & \text{indicates distance (displacement)} \\ \Delta t & \text{indicates the duration that the vehicle spent for the displacement } \Delta x \end{cases}$$

As just the scalar value of velocity and displacement are required for purposes of this project, the term speed and distance will be used instead of velocity and displacement from now.

3.1 Data processing

In order to analyze the vehicle speed over the different network segments, two main elements, the speeds and the segments.

To calculate the speed, the file `vehiclePositionID.csv` is used as well as the file `LinesInformation.csv` which gives the distance at which the stops are located from the start of the line. First, the distance of each position is calculated, it is simply the sum of the distance of the last stop passed and the `DistanceFromPoint`.

From

Line	Variance	Stop	DistanceFromPoint	...
71	1	ULB	243	...
71	1	ULB	402	...
71	1	Cimetière d'Ixelles	0	...
71	1	Cimetière d'Ixelles	34	...
...	...	...	...	...

`LinesInformation.csv`

Line	Variance	Stop	Distance	...
71	1	ULB	5420.399	...
71	1	Cimetière d'Ixelles	5988.051	...
...	...	...	...	...

To

Line	Variance	Stop	Distance	...
71	1	ULB	5663.399	...
71	1	ULB	5822.399	...
71	1	Cimetière d'Ixelles	5988.051	...
71	1	Cimetière d'Ixelles	6022.051	...
...	...	...	...	...

Then, the calculations of the speeds are quite straightforward, for each vehicle, the difference of time and distance is calculated between each consecutive pair of locations. To obtain more rich data out of these speeds, it has been chosen to keep the day as well as the hour these speeds have been registered.

From

Line	Variance	Stop	Distance	Time	...
71	1	ULB	5663.399	08/09/2021 8:33:49	...
71	1	ULB	5822.399	08/09/2021 8:34:21	...
71	1	Cimetière d'Ixelles	5988.051	08/09/2021 8:34:53	...
71	1	Cimetière d'Ixelles	6022.051	08/09/2021 8:35:25	...
...	...	...	...	...	...

To

Line	Variance	Stop	Speed	Day	Hour	...
71	1	ULB	?	Wed	8	...
71	1	ULB	17.974	Wed	8	...
71	1	Cimetière d'Ixelles	18.665	Wed	8	...
71	1	Cimetière d'Ixelles	3.856	Wed	8	...
...	...	...	...	...	...	...

In order to create the segments, the file `LinesInformation.csv` is used. To achieve that, this file is joined with itself and only are kept the combinations in which the difference between the second stop and the first

one is equal to one. This gives for each line, each variance, all of its segments which are composed of the first stop as the `From` column and the second one as the `To` column.

From

Line	Variance	Stop	Succession	...
2	2	Elisabeth	1	...
2	2	Ribaucourt	2	...
2	2	Yser	3	...
2	2	Rogier	4	...
...	...	...	...	...

To

Line	Variance	From	To	...
2	2	Elisabeth	Ribaucourt	...
2	2	Ribaucourt	Yser	...
2	2	Yser	Rogier	...
...	...	...	...	...

3.2 Results

In this section, the analysis is divided into two parts, the first relating to a specific line sample, since it is not possible to make the same charts for all lines, therefore it was necessary to choose a single line. On the other hand, the second part is more general, and the charts focus more on the speed of the transport mode than on the lines speed.

3.2.1 Speed analysis on a specific line (line 2)

The line chosen for the analysis is line 2, a metro line containing 19 stops on each of its two variants. The figure 3 shows the vehicles average speed in km/h over the different line segments and figure 4 shows the vehicles average speed in km/h over day hours for the same line.

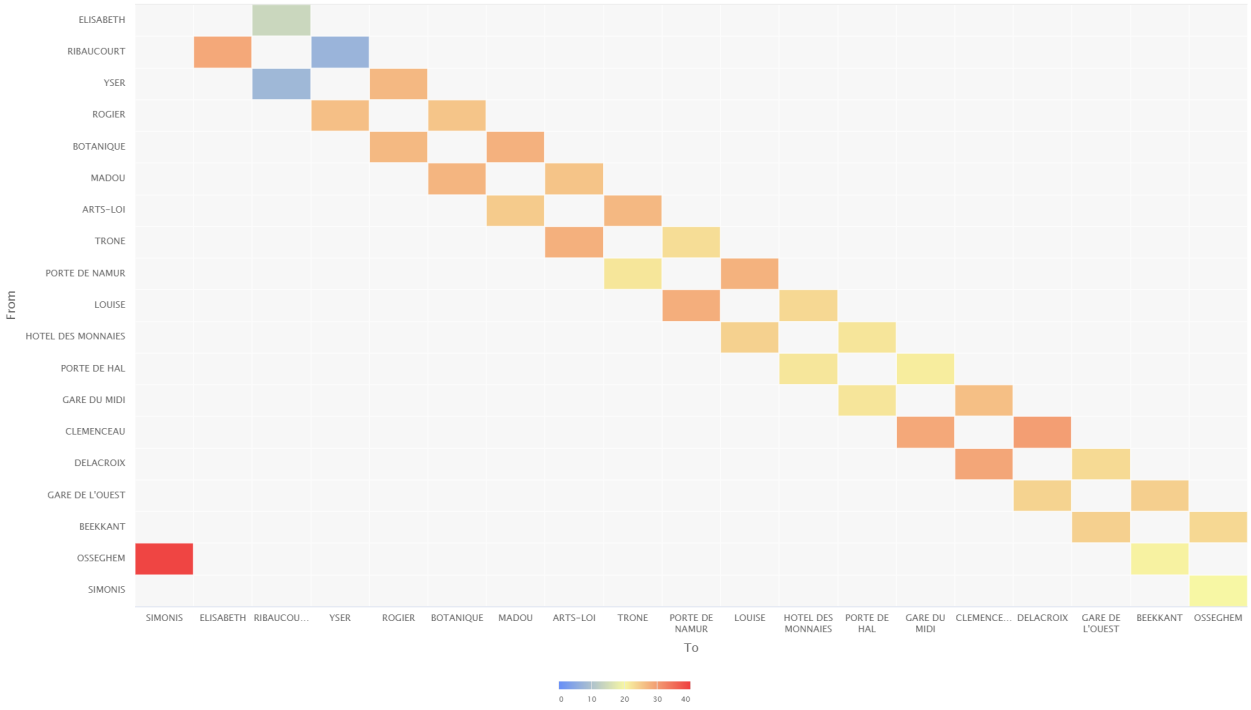


Figure 3: Vehicles average speed in km/h over the different segments of line 2

The obtained result indicates that the speed values vary in the range between 0 km/h and 40 km/h with most of values in the interval [20, 30]. The result also shows that traveling by two variants of the same segment does not affect the speed of vehicles since there is not a high difference between them.

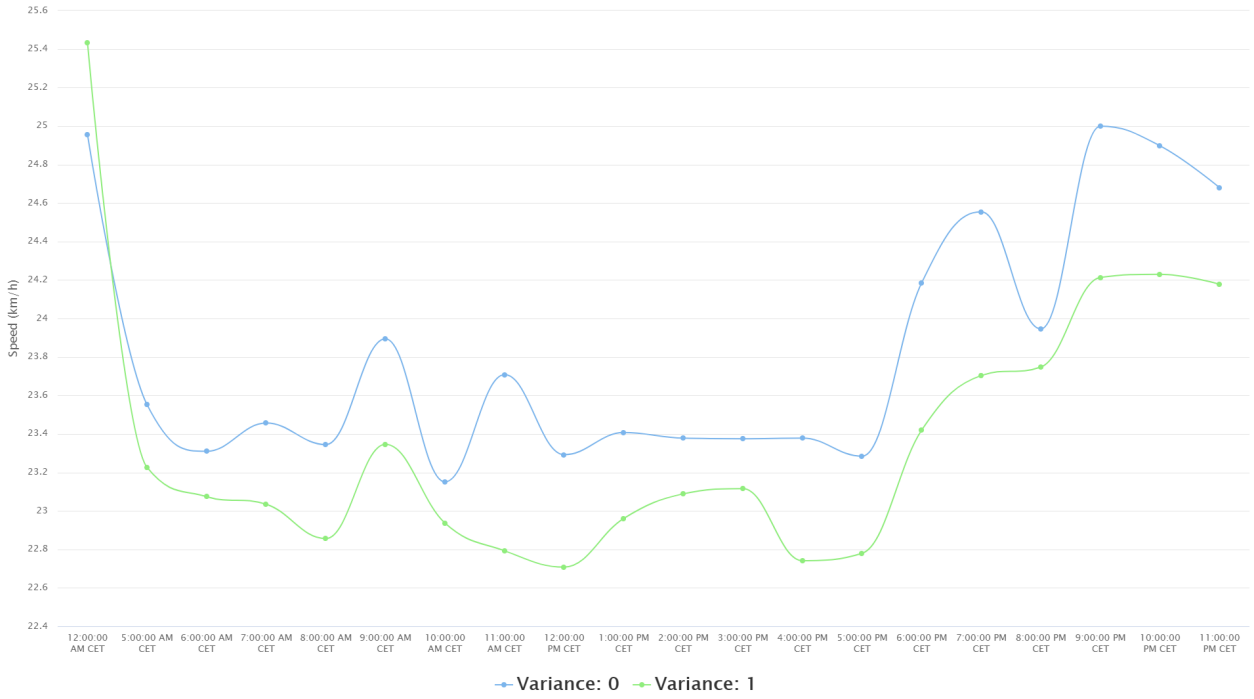


Figure 4: Vehicles average speed in km/h over day hours for line 2

In the first sight, a huge peak in the diagram can be observed at 6:00 PM. In fact, from 6:00 PM to 12:00 AM, the average speed of line 2 is higher in both variances compared to other hours. As diagram shows in the range out of 6:00 PM to 12:00 AM, the speed variation is almost between 0 $\frac{km}{h}$ to 1 $\frac{km}{h}$.

3.2.2 Speed of each transport mode according to weekdays

The first verification that has been performed, was on speed of each transport mode according to weekdays. To do so, for each day of week, the average of speeds of each transport mode calculated. The result is as follows:

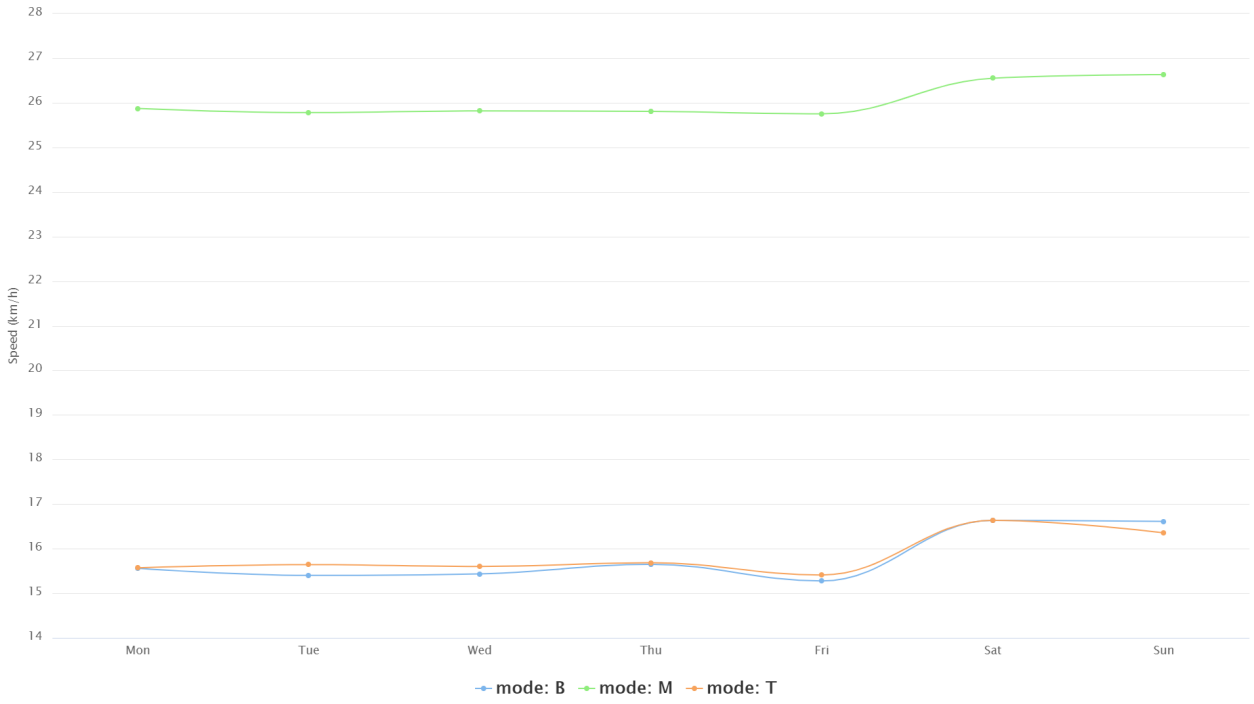


Figure 5: Transport modes speed according to each day of week

An overview of the diagram displays that the metro average speed is at least $10\frac{km}{h}$ greater than average speed of bus and tram. This can be explained by considering the route that each transport mode, traverse. Metros normally traverse a route underground, so it is away from urban traffic while bus and tram traverse on congested areas of the city.

As the diagram also shows, for all the transport modes, the average speed at weekends (Saturdays and Sundays) is greater than the average speed of other days of week. The reason for that is pretty obvious. In the weekends traffic congestion in the city is lower compared to other days as offices, schools and etc are closed.

Another noteworthy thing about this diagram is that the speed of tram is a little bit more than speed of bus except in the weekends. That can be described by taking into account that the quantity of special routes for trams over the city is greater than the quantity of special routes of buses over the city. But as the weekend traffic congestion decreases, buses can go more rapid compared to trams.

A comparison which can be interesting is, to verify the result that have been calculated by our calculation with the official statistic of STIB [2]. The average speeds of each transport modes, according to our calculation and STIB statistics are as follows:

calculator / transport mode	Metro	Tram	Bus
STIB statistics	27.9 km/h	16.2 km/h	15.5 km/h
Our calculatuion	26.03 km/h	15.84 km/h	15.79 km/h

[2] https://www.stib-mivb.be/irj/go/km/docs/WEBSITE_RES/Attachments/Corporate/Statistiques/2020/STIB_RA2020_Statistiques_EN_web.pdf

The table shows that the calculation that have been made in this project almost corresponds to official statistics of STIB.

3.2.3 Speed of each transport mode according to day hours

The second verification performed is on the speed of each transport mode according to hours. To do so, for each hour of a day, the average of speed of each transport mode at that hour is calculated. The result is as follows:

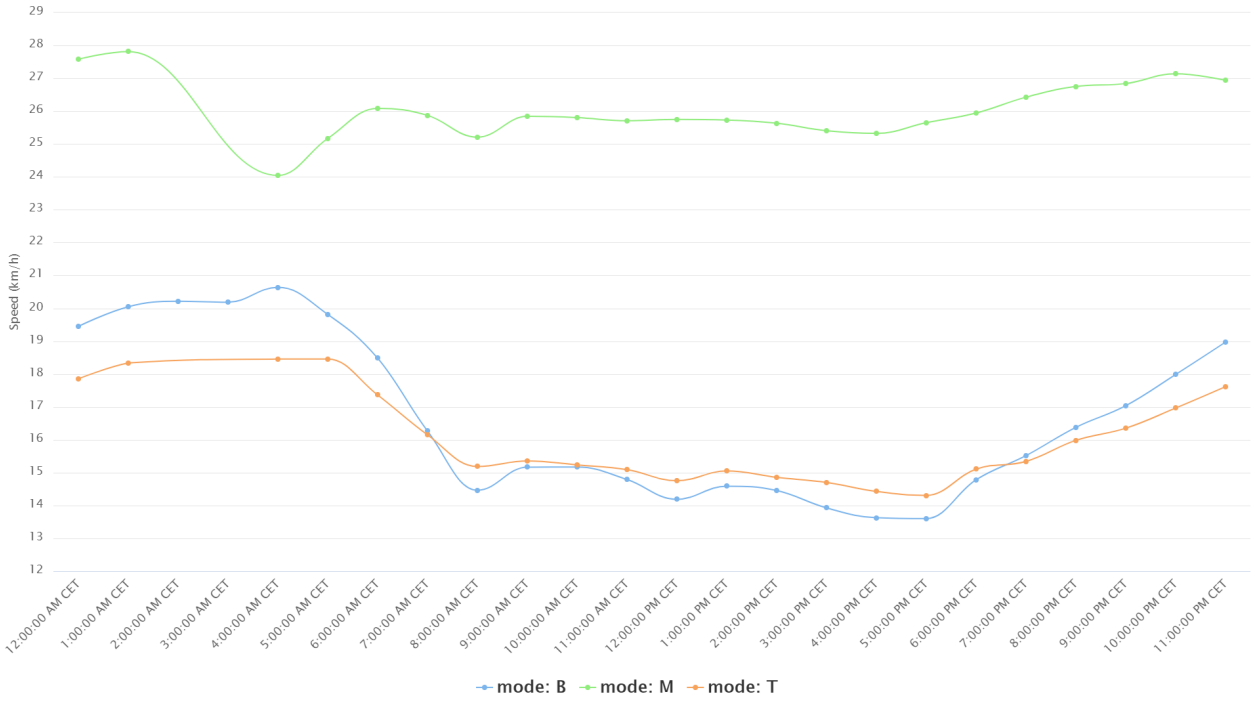


Figure 6: Transport modes speed according to day hours

As the overview shows, metros have less speed variation during the day compared to buses and tramways. In general, the metro is faster than the bus and the tramway. Buses and tramways have almost the same range of variation in the diagrams, but in rush hours, the speed of tramways is higher than the speed of buses. In other hours of the day the bus speed is larger than the tramway speed. The reason of that is the greater quantity of dedicated routes for tramways compared to buses as explained in the previous diagram.

It can be also seen that there exists a speed reduction at 8:00 AM and 5:00 PM in the diagram. These hours are rush hours of a day. As at 8:00 AM people are going to work and students are going to school and as at 5:00 PM offices and schools close, it is normal to have such kind of diagram. Between 8:00 AM and 5:00 PM the speed rises compared to these hours, but it is lower than the speed out of range 8:00 AM to 5:00 PM.

3.2.4 Metros speed vs. hours

The below graph (figure 7) shows the average speed distribution of metros for each hour of the day. This average speed distribution is calculated for each day of the week.

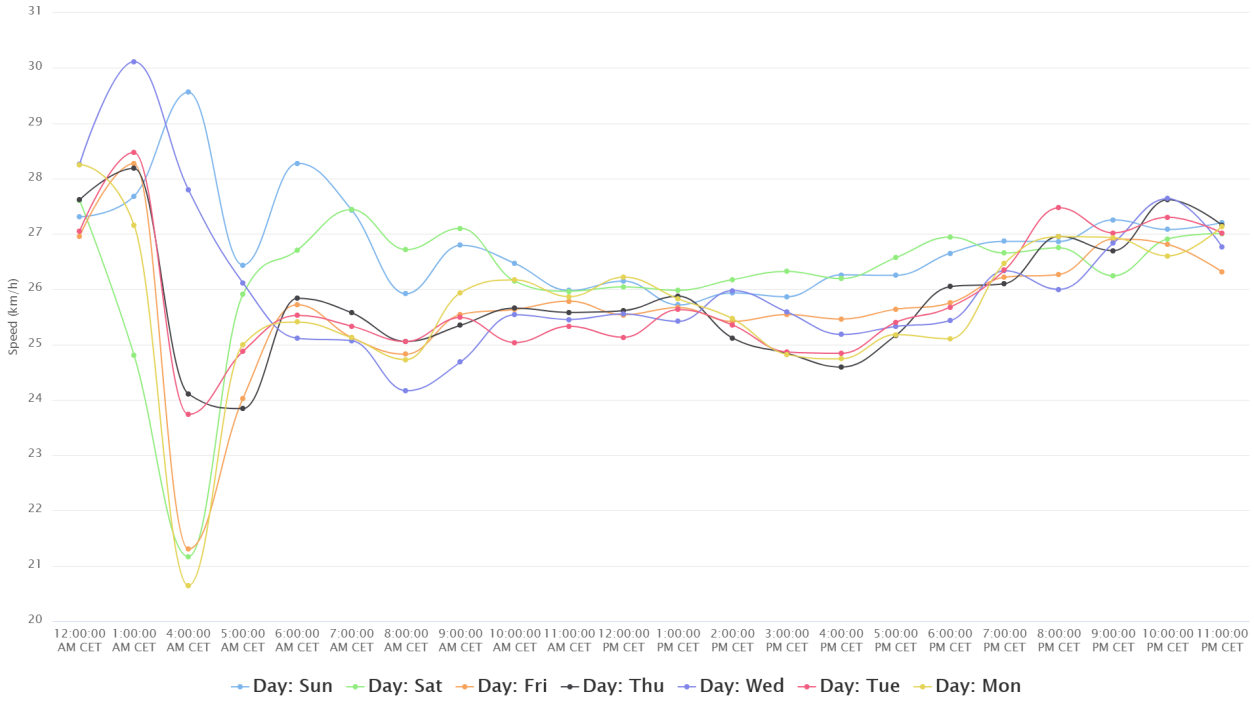


Figure 7: metro

The curves corresponding to Saturdays and Sundays (weekend) show globally higher values of speed than the other days (working days). In addition, each working day show a strong speed reduction at 4:00 AM. This value is to be considered with care because most metro lines do not start before 5:00 AM. This low value comes from very few metros which are starting their daily service and are therefore not representative of the normal operations.

The rest of the day is more interesting with a small reduction of speed around 8:00 AM and around 4:00 PM. The variations are, however, not huge with most average speed values being between 24 km/h and 29 km/h.

3.2.5 Buses speed vs. hours

The below graph (figure 8) shows the average speed distribution of buses for each hour of the day. This average speed distribution is calculated for each day of the week.

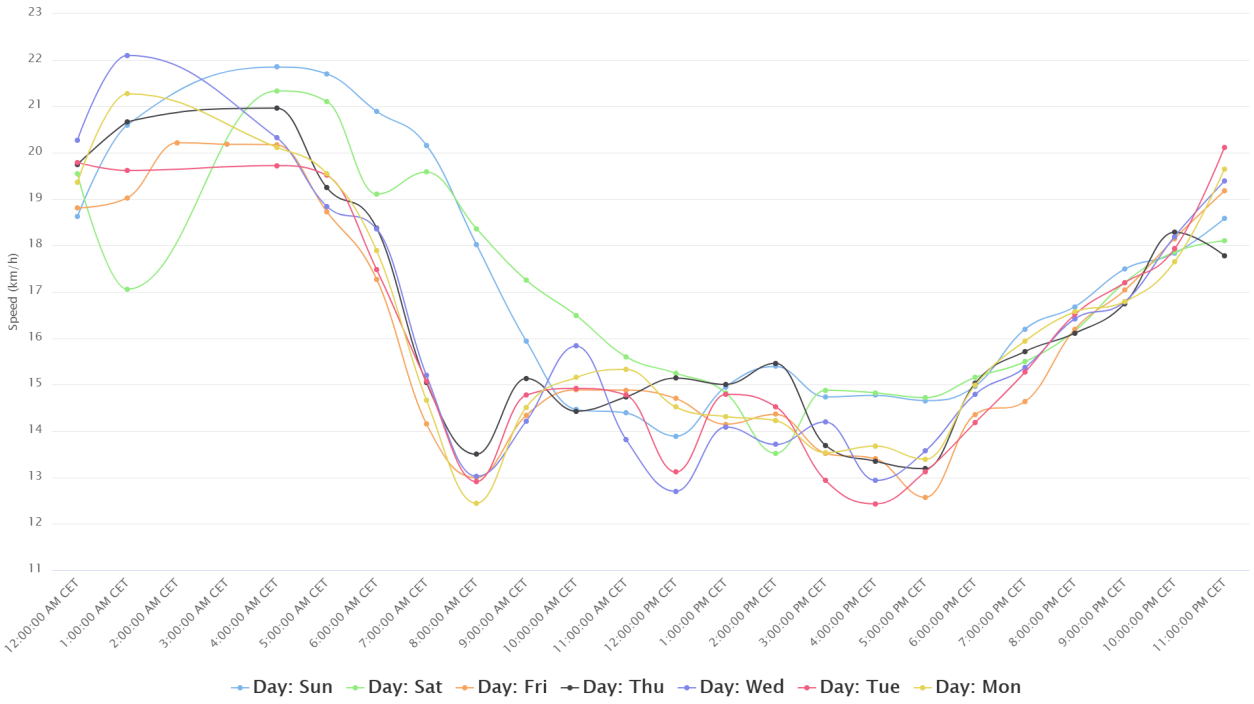


Figure 8: bus

This graph shows higher variations than for metros with values ranging between 22 km/h and 12 km/h. The difference between the weekend and working days is also clearer, especially in the morning. The 8:00 AM reduction of speed during working days is easily identifiable, while it is not present during the weekend. The increase of speed in the evening is very consistent through all days.

3.2.6 Tramways speed vs. hours

The below graph (figure 9) shows the average speed distribution of tramways for each hour of the day. This average speed distribution is calculated for each day of the week.

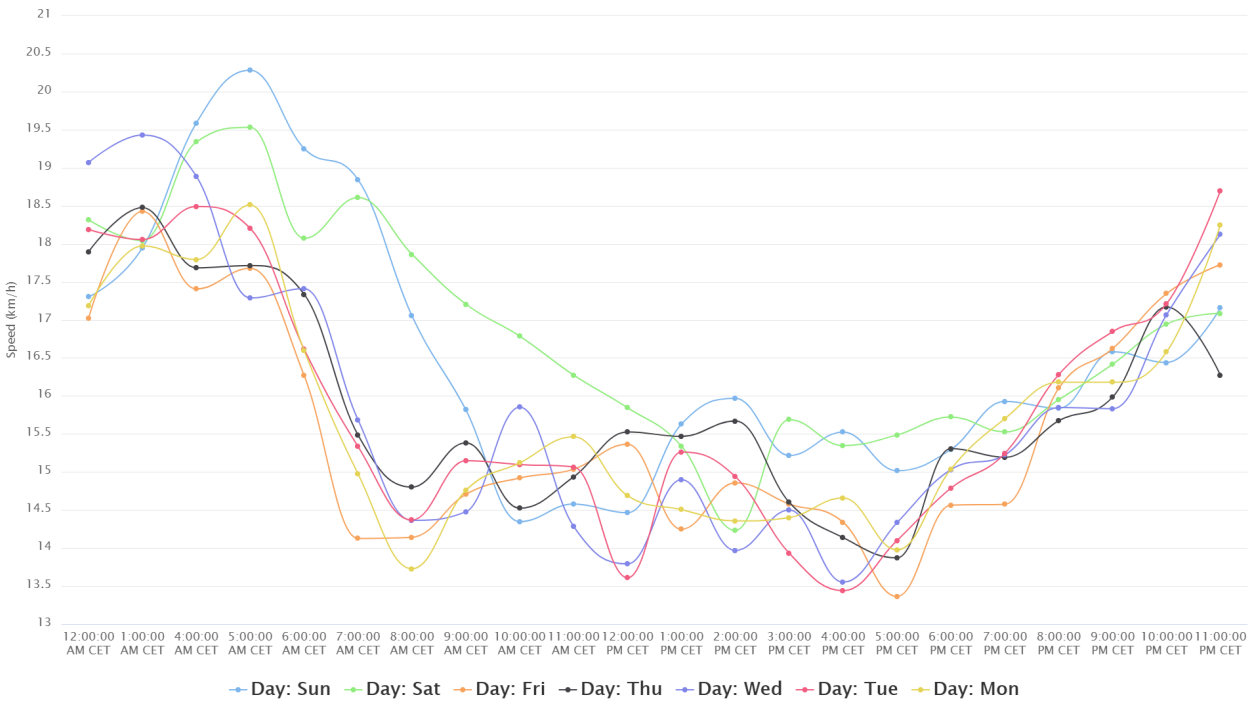


Figure 9: tram

This graph is quite similar to the buses graph with a bit more variation between each working day. The variations during the day are slightly smaller than for buses (between 20 km/h and 13,5 km/h).

4 Analysis of vehicle delays

This task consists of analyzing the vehicle delays at the different stops, how it varies across stops, and over time. The following files are required to perform the current task. The duration of the vehicle trips was not fully covered by only one of the GTFS snapshot files, thus both snapshots were used. The explanation of the GTFS files that have been used, comes from the official Google API documentation [3].

[3] https://developers.google.com/transit/gtfs/reference#dataset_files

Filename	Source	Defines
vehiclePositionID.csv	Generated from JSON	Vehicles' trips following a variance of the network's lines
routes.txt	gtfs3Sept, gtfs23Sept	Transit routes. A route is a group of trips that are displayed to riders as a single service.
trips.txt	gtfs3Sept, gtfs23Sept	Trips for each route. A trip is a sequence of two or more stops that occur during a specific time period.
stop_times.txt	gtfs3Sept, gtfs23Sept	Times that a vehicle arrives at and departs from stops for each trip.
calendar.txt	gtfs3Sept, gtfs23Sept	Service dates specified using a weekly schedule with start and end dates.
calendar_dates.txt	gtfs3Sept, gtfs23Sept	Exceptions for the services defined in the calendar.txt.

4.1 Data processing

The source data does not contain trip information, therefore, to determine the delay of each vehicle, it is necessary to match each vehicle trip to one of the possible trips which exist at the same time period.

First of all, it is important to generate a new `calendar.txt` file to take into account the exceptions defined in `calendar_dates.txt`. This file contains the services which should be modified, their modification date and the type of modification. Value 1 means that the service has been added for the specified date while value 2 means that the service has been deleted for the specified date.

For instance, by using the `calendar_dates.txt` and `calendar.txt` files, the obtained result is represented by the new `calendar.txt`. This exception handling process can be seen below:

```
service_id,date,exception_type
238162502,20210920,1
238074051,20210927,2
238074051,20210928,2
238074051,20210929,2
238074051,20210930,2
238074051,20211004,2
238074051,20211005,2
238074051,20211006,2
238074051,20211007,2
238074051,20211011,2
238074051,20211012,2
238074051,20211013,2
238074051,20211014,2
```

Listing 1: calendar_dates.txt

Before

```
service_id,monday,tuesday,wednesday,thursday
    ,friday,saturday,sunday,start_date,end_date
238162502,0,0,0,0,0,1,0,20210918,20210919
238074051,1,1,1,1,1,0,0,20210924,20211015
```

Listing 2: calendar.txt

After

```
service_id,monday,tuesday,wednesday,thursday
    ,friday,saturday,sunday,start_date,end_date
238162502,0,0,0,0,0,1,0,20210918,20210919
238162502,0,0,0,0,0,1,0,20210920,20210920
238074051,1,1,1,1,1,0,0,20210924,20210926
238074051,1,1,1,1,1,0,0,20211001,20211003
238074051,1,1,1,1,1,0,0,20211015,20211015
```

Listing 3: new calendar.txt

Once the first step is completed, the next step is to select only the useful information fields in each file. The result is as follows:

- `vehiclePositionID.csv` : All fields (i.e., Time, LineID, Variance, DirectionID, pointID and distanceFromPoint).
- `calendar.txt` : All fields (i.e., service_id, "Days of week", start_date and end_date).
- `routes.txt` : route_id and route_short_name (i.e., LineID).
- `trips.txt` : route_id, trip_id, service_id and direction_id (i.e., Variance).
- `stop_times.txt` : trip_id, stop_id and arrival_time.

As the selected fields show, the theoretical travel time and the real travel time cannot be linked directly. It is therefore important to join the files together to create the missing link. Thus, these files have been used to generate these links and to simplify understanding. The fields are renamed as follows:

- Vehicles = {`vehiclePositionID.csv`} : Time, VehicleID, LineID, Variance, DirectionID, pointID and distanceFromPoint.
- Calendar = {`calendar.txt, routes.txt, trips.txt`} : service_id, "Days of week", start_date, end_date, LineID, and Variance.
- Stop Times = {`stop_times.txt, routes.txt, trips.txt`} : trip_id, stop_id and arrival_time, service_id, LineID, and Variance.

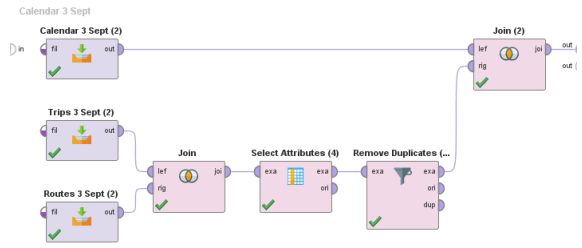


Figure 10: Calendar

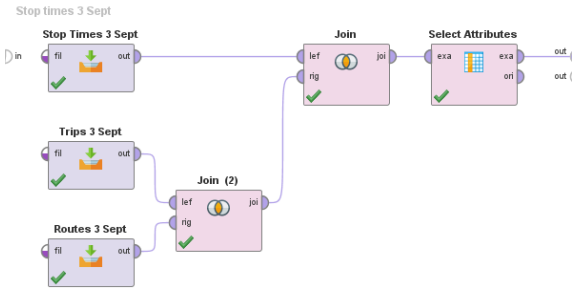


Figure 11: Stop Times

Now that these links are created, things get more interesting. As the result shows, each vehicle trip can be matched to one of the possible trips. For this purpose, a filtering is performed on the vehicles as shown below:

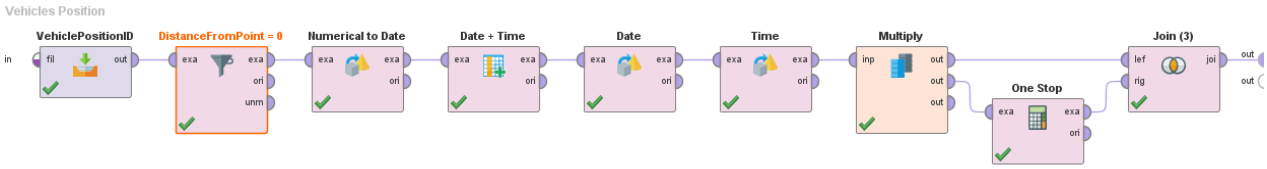


Figure 12: Vehicles filtering

1. Filter vehicle trips to obtain only the entries for which the vehicle position is exactly the same as the position of a stop (i.e., distanceFromPoint = 0).
2. Keep only one entry for each different stop (i.e., the one with the earliest time) because it is this entry which represents the arrival time.

After filtering the vehicles, the following phase consists in extracting from the calendar each possible service in the same period as the one in which the trips were made, then assigning it to each vehicle belonging to the same line variance as that of the service. Among all the possibilities that have been obtained, filtering was applied to retain only the valid services (i.e., the service which is available on the same day as the given trip).

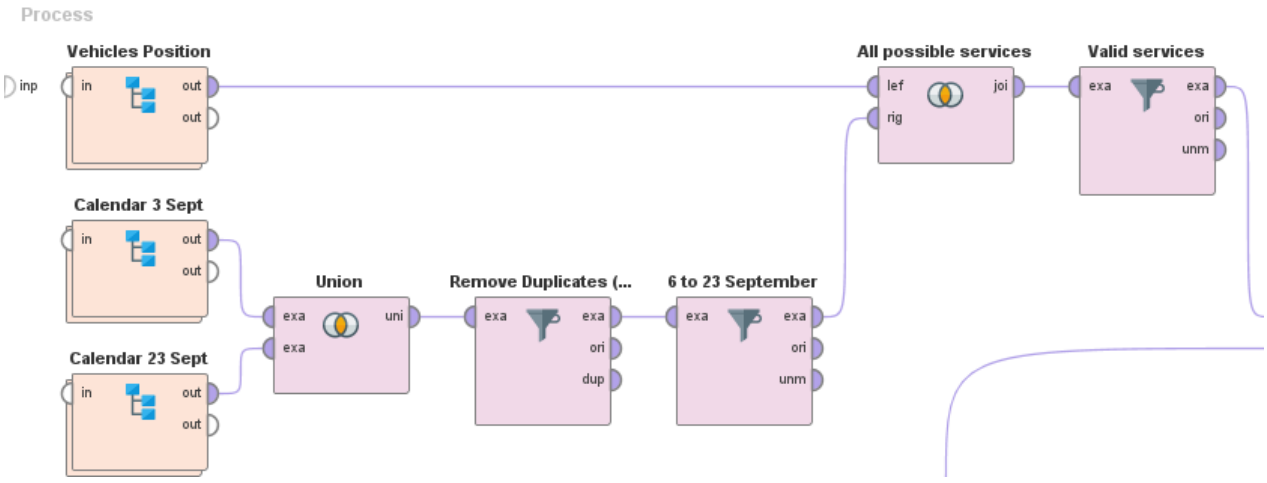


Figure 13: Linking vehicles to services

At this stage of the process, the information obtained is all the possible services that a trip can have. This service information is used to connect a real trip to all the theoretical trips which belong to the stops schedule.

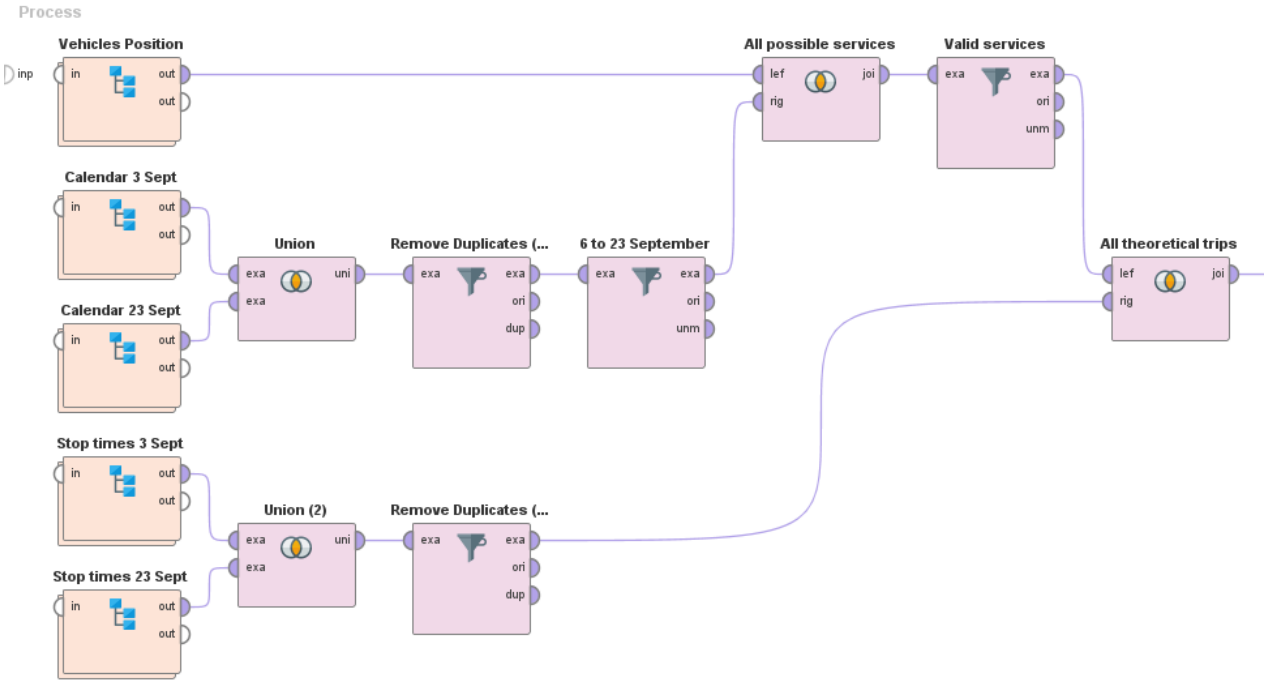


Figure 14: Linking vehicles to theoretical trips

Now that each real route is linked to one or more theoretical trips, the most accurate one must be picked. The chosen approach is to pick only those that cover all the stops of the trip. Note that some vehicle trips are temporary or exceptional (i.e., they do not appear on the schedule). Therefore, they are not taken into

account. Afterwards, the real trip is compared to all the remaining ones by computing the overall delay with respect to each trip and choosing the one with the lowest delay value.

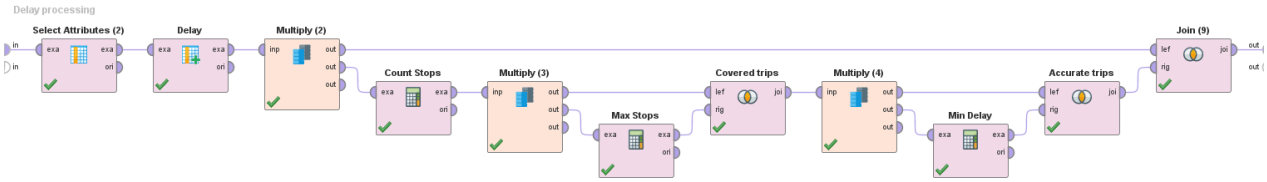


Figure 15: Delay processing

Note that the method of computing the delay is as follows: the difference between the theoretical time and the real time is calculated in absolute value (i.e., arriving before the scheduled time is also considered as a delay).

All the steps explained are grouped in this complete process below:

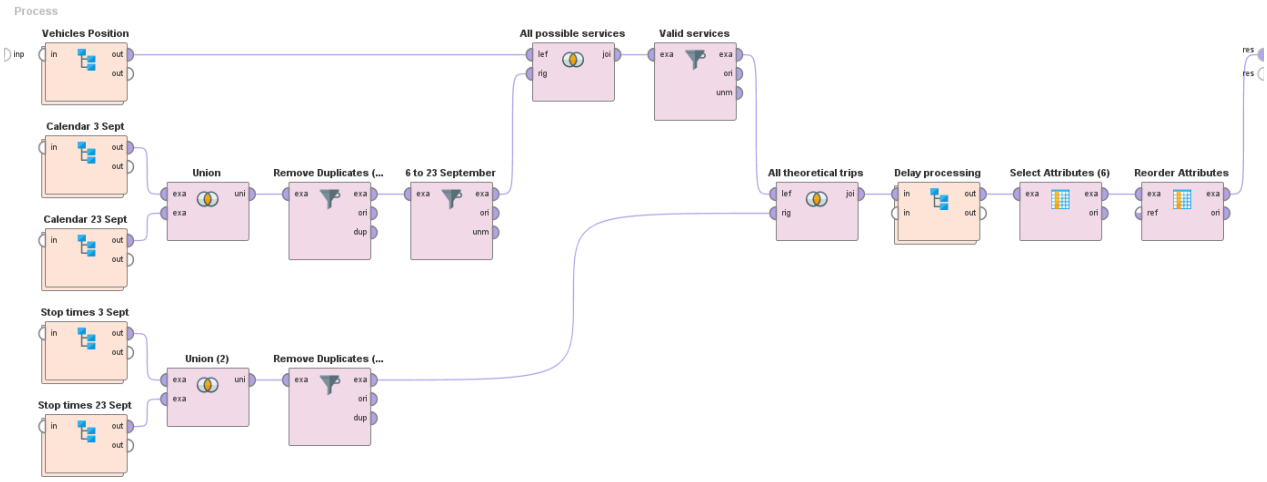


Figure 16: Complete process

4.2 Results

For this section, the analysis has been performed on a specific line as the analysis of all lines together was not possible due to limited amount of available memory. The diagrams below, show the result that has been obtained for line 2 which is a metro line with 19 stops in both of its variances.

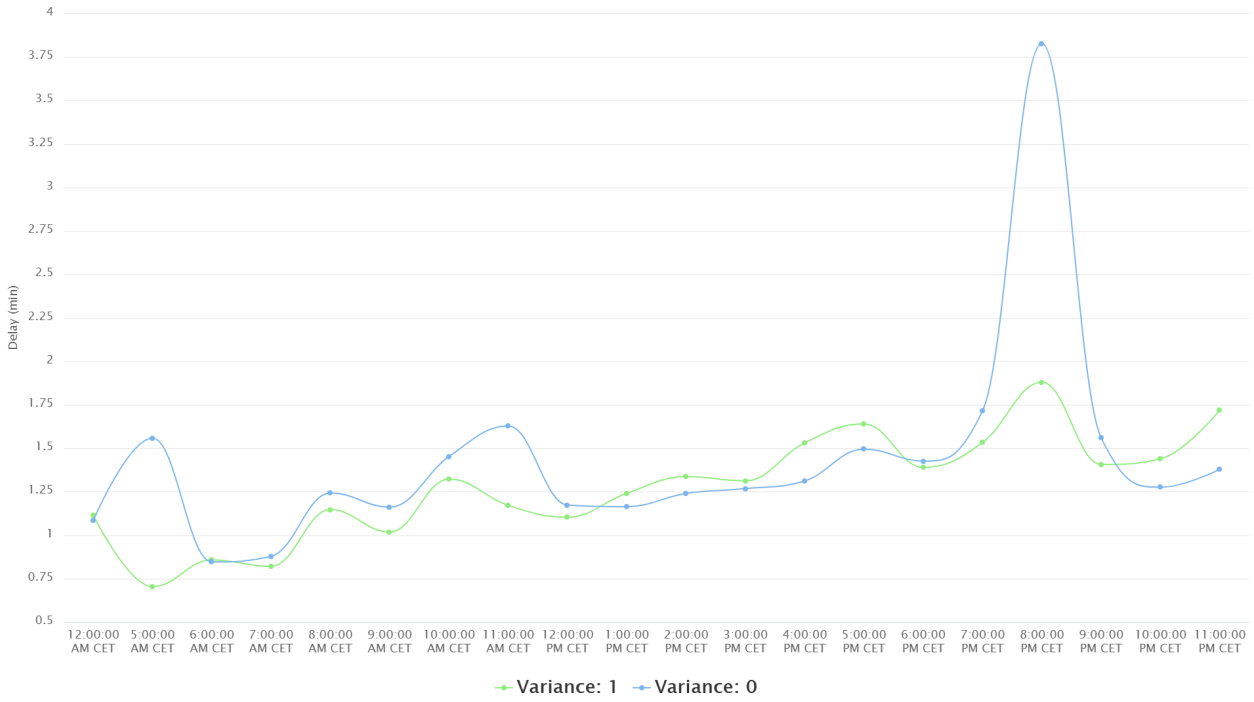


Figure 17: Total delay of line 2 per hour

As figure 17 shows, there exists a peak at 8:00 PM in diagrams in both variances. This shows that at 8:00 PM this metro line is more crowded compared to other hours. Something which is interesting is that at this hour, delay of the first variance is almost 2 minutes more than delay of second variance, in average. So, it can be concluded that first variance of line 2 is more crowded at the evening, compared to its second variance.

General overview on the diagram shows that in average there exists a delay between 1 to 2 minutes, in each hour in average. This delay is tolerable considering that other lines and transport modes also have almost the same range of delay as line 2.

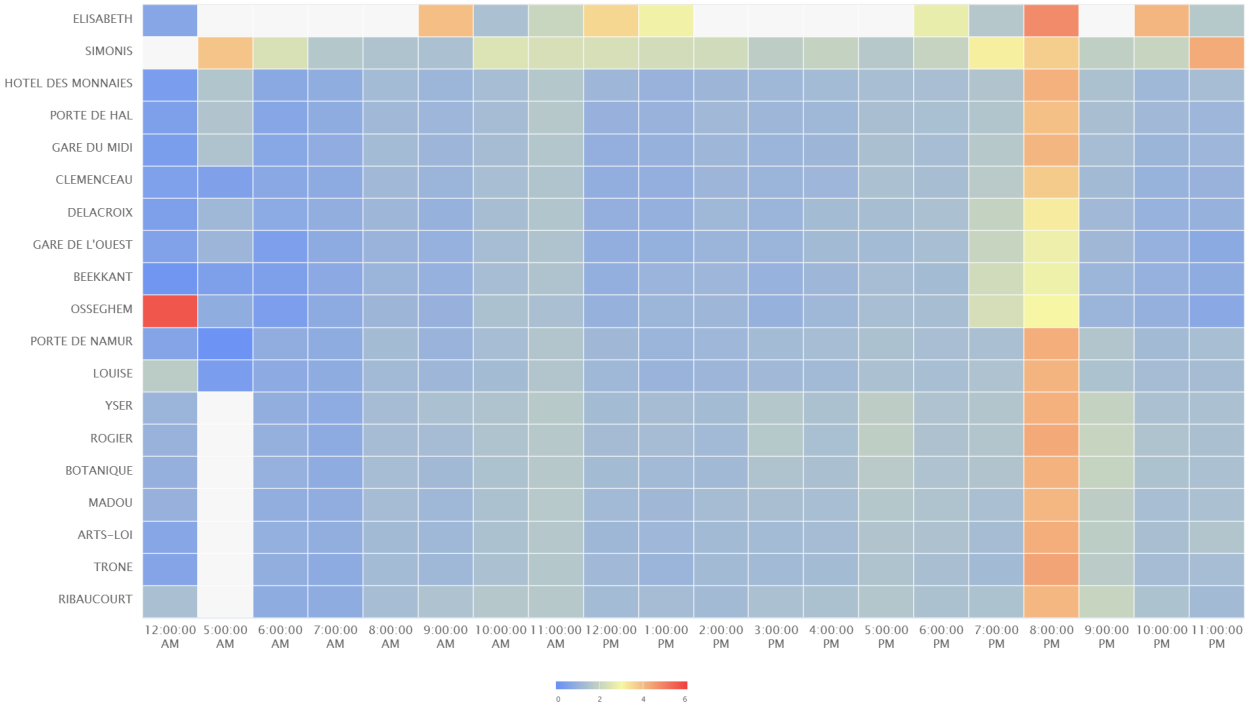


Figure 18: Line 2 variance 0 stops delay per hour

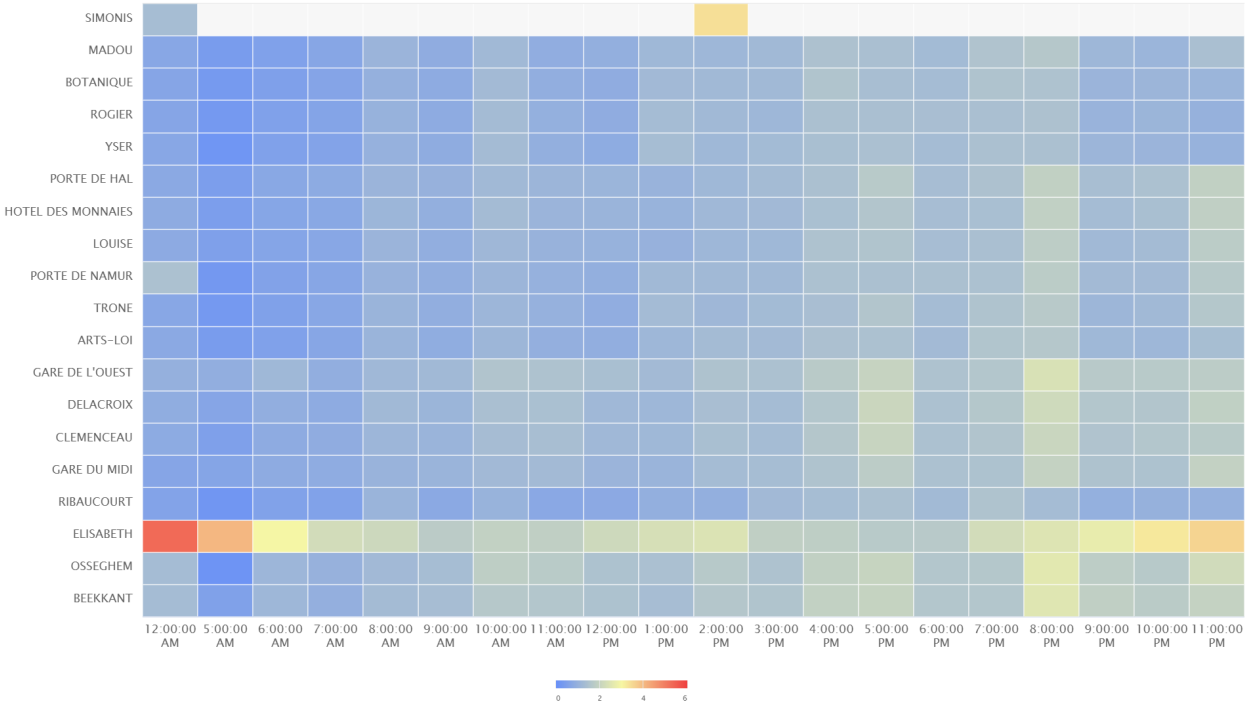


Figure 19: Line 2 variance 1 stops delay per hour

The two diagrams are mostly blue which indicates that, the range of delay is almost between 0 and 2

minutes for each stop. It can be mentioned that in both diagram the average delay in the first stop (Simonis for variance 0 and Elisabeth for variance 1) is higher compared to other stops of the line. Maybe by decreasing this initial delay at first stops, STIB reaches a more exact time schedule with lower minutes of delay compared to the actual schedule.

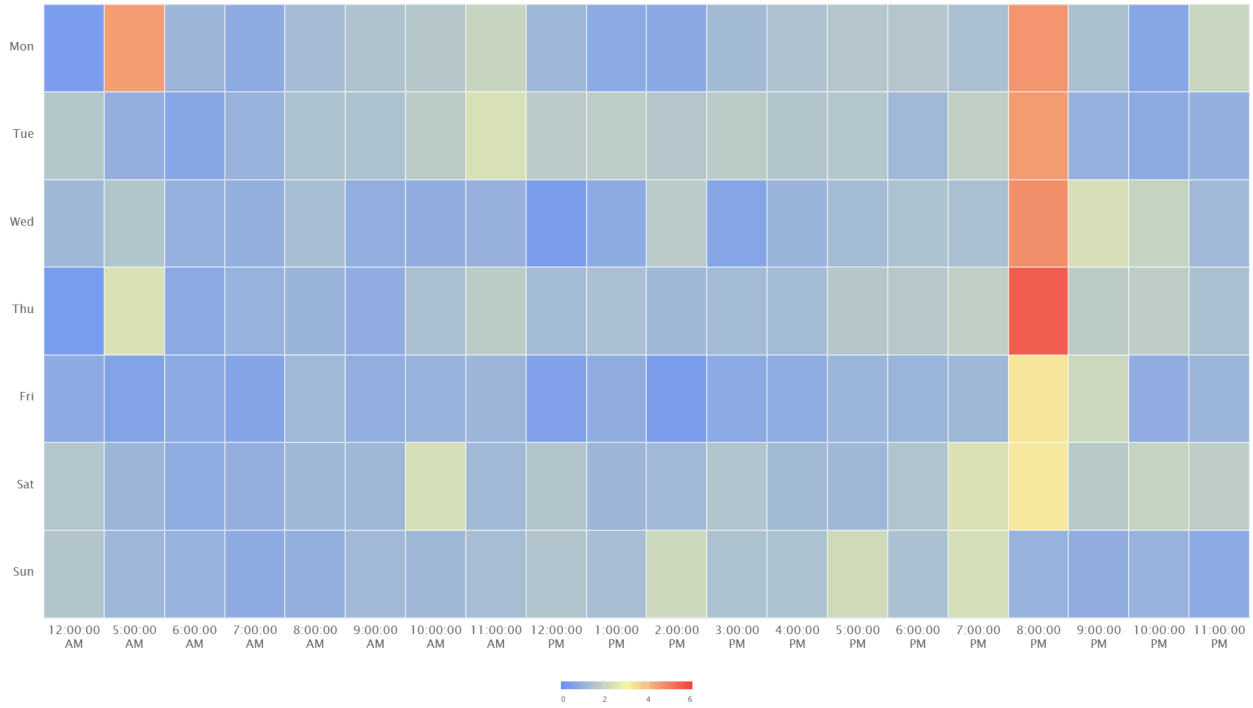


Figure 20: Line 2 variance 0 according to weekdays and hours

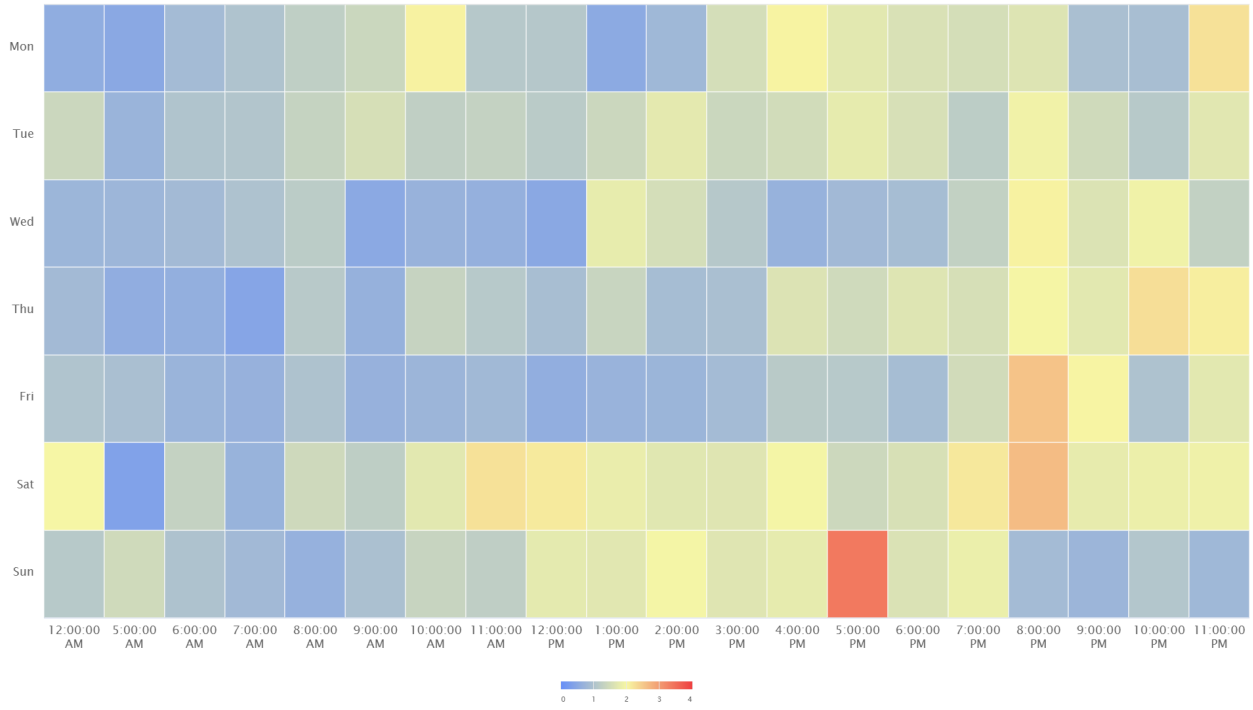


Figure 21: Line 2 variance 1 according to weekdays and hours

An overview of the diagrams shows that in the afternoon, there exist more delay compared to the morning. It can also be observed that the variance one has greater delay in average compared to the variance 0 as it has more yellow cases.

As diagrams shows, in average, the total delay on Saturdays and Sundays in both variances is greater than any other days. The reason of that can be explained by considering that during the weekend, the metro line is crowded a greater portion of the day compared to the other days.

5 Arrival time forecasting

In this part of the project, it is requested to do arrival time forecasting at a given stop in the route of a vehicle, given this vehicle start time. It is also requested to test the accuracy of the forecasting by randomly splitting the dataset in disjoint training and testing subsets.

5.1 Data processing

This question is solved using RapidMiner, based on the `vehiclePositionID.csv` file. The elaborated RapidMiner process follows the below mentioned steps:

1. Request the considered line number, the variant (direction), the start stop and the end stop as input.

2. Remove "duplicates", i.e. the vehicle IDs which appear several times in a row at distance 0 from the considered stop. We only keep the first time at the stop (the arrival time).
3. Join the Start stop data to the end stop data.
4. Calculate the time difference between start stop and end stop for each vehicle.
5. Keep only the arrival time and the time difference.
6. Generate new attributes:
    - `Hour` : Integer containing departure time hour.
    - `Day` : Integer containing departure time day of the month.
    - `Time_in_minute`s : Time difference in minutes.
7. Calculate the averages of the time differences in minutes for every hour in the data to obtain a computable time series.
8. Fill in the missing values with zeroes (for night time or any other time frame without any measurements).
9. Generate an ID in order to be able to make some operations on the time series.

5.2 Results

5.2.1 Data exploration

To learn a bit more about the generated time series, the following operations are performed:

1. Fast Fourier Transformation
2. STL decomposition
3. Autocorrelation
4. Partial autocorrelation

In the following paragraphs, the example of Metro line 2, variant 1, from stop *Hôtel des Monnaies* to stop *Clémenceau*, is chosen for illustration.

Stationarity : it is clear that, just like the speed time series, the time difference time series are clearly stationary (see figure 22, below). There is therefore no need to convert it to a new time series that would be stationary.

As a reminder, the data covers the whole range of days provided in the input `JSON` files and it is ranging from the 3rd September to the 23rd September 2021. Each point (id) is an hour. The zero values correspond to hours without any vehicle measurements between start stop and end stop.

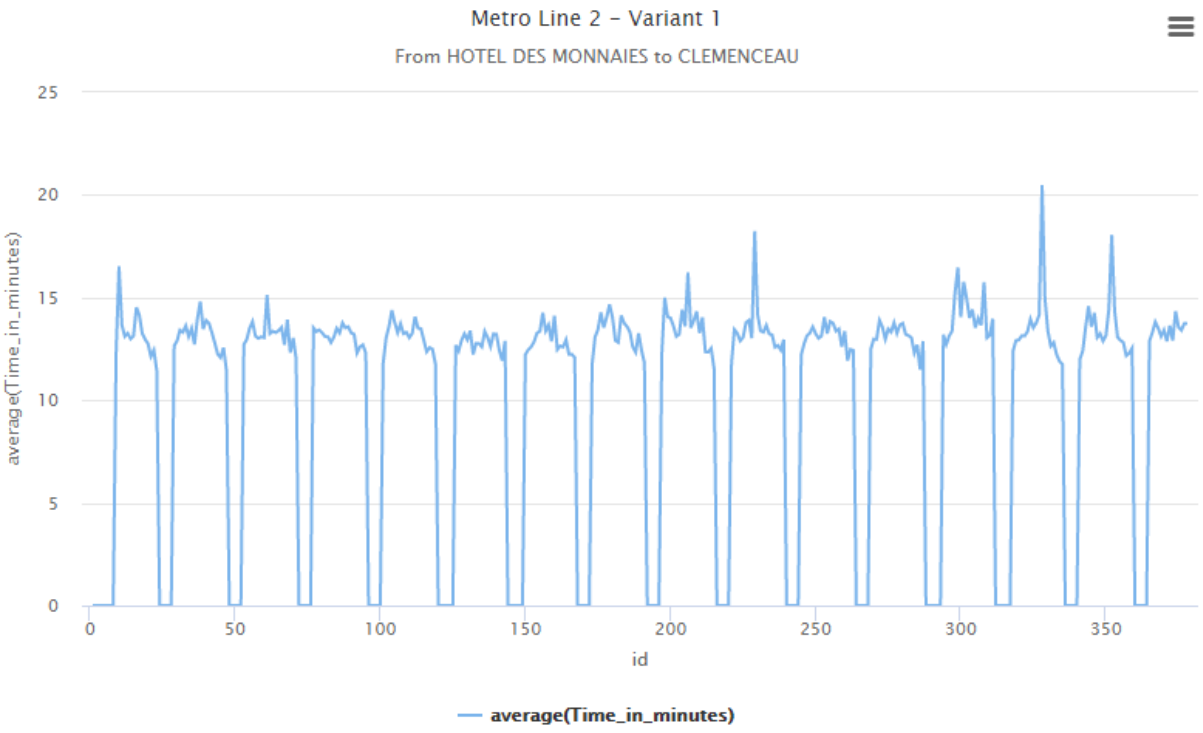


Figure 22: Time series

FAST FOURIER TRANSFORMATION (FFT) : The Fast Fourier transformation changes the data from a time-based data to a frequency-based data in order to identify the underlying patterns. In particular, it can help identify the seasonality in time series. As shown in figure 23, below, the first peak in seasonality has a frequency of 0,04308 $hour^{-1}$, which corresponds to a period of 23,2 hours, i.e. quite close to an intuitively significant seasonality : 24 hours. Interestingly enough, there are 2 other easily identifiable seasonal peaks at 0,08225 $hour^{-1}$ and 0,12533 $hour^{-1}$. These 2 peaks correspond to 12,2 hours (close to half a day) and 7,98 hours (close to a third of a day).

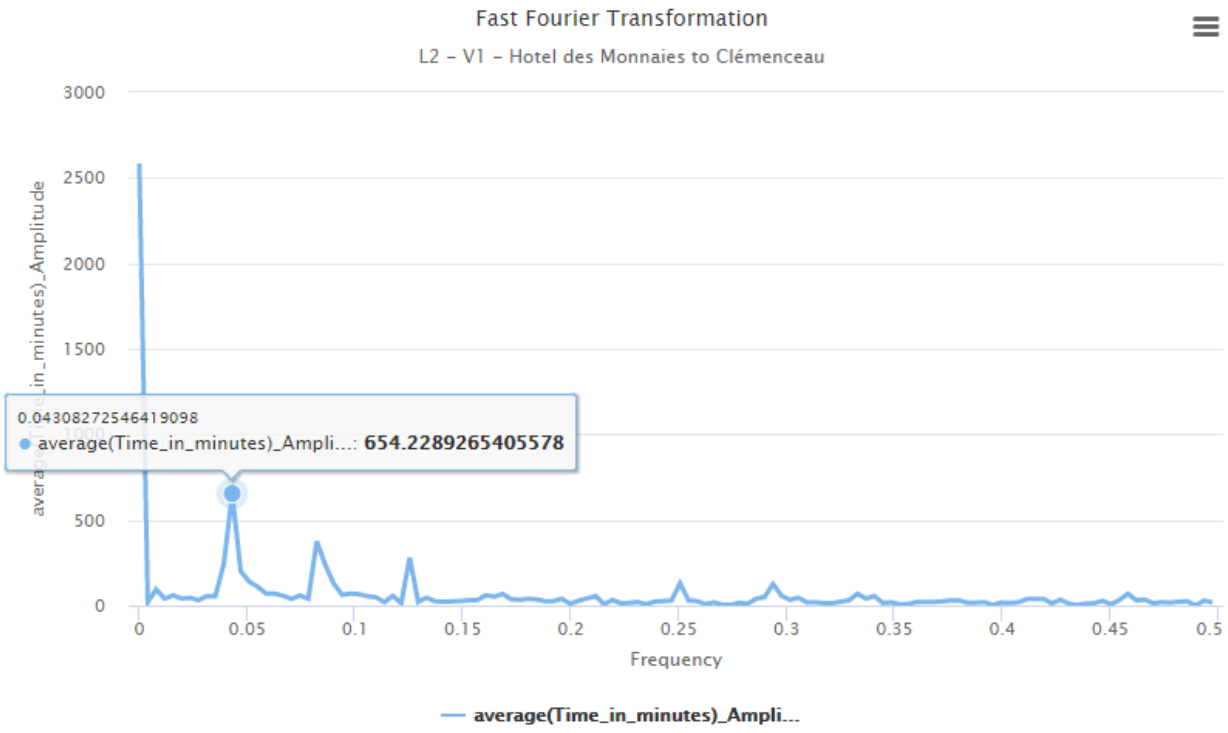


Figure 23: Fast Fourier transformation

STL DECOMPOSITION : STL decomposition (Seasonal and Trend decomposition using Loess) helps identify the seasonal component, the trend, the remainder and potentially the cycles. The selected seasonality is an output of the FFT.

Based on the observation of the FFT, we can therefore use seasonality = 24 in the STL decomposition.

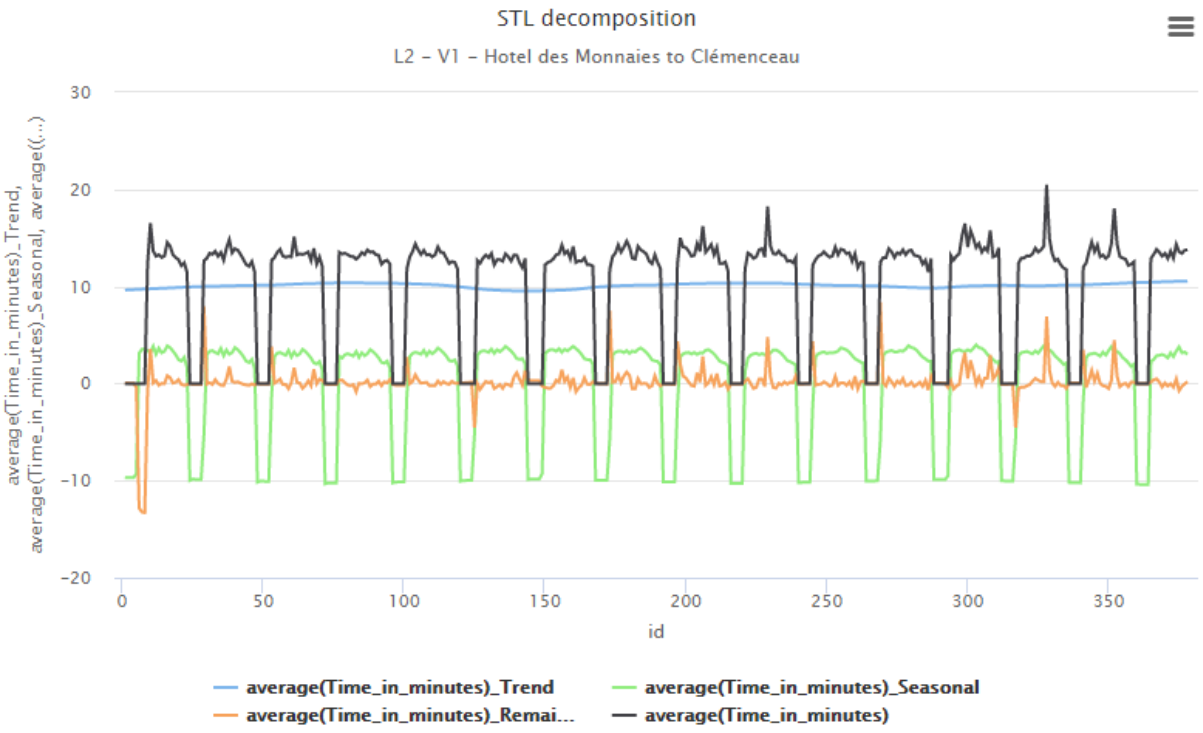


Figure 24: STL Decomposition

Based on figure 24, it can be noted that the trend of the time differences (blue curve) is a rather flat curve with little variation over the days. It is visible that the seasonal component (green curve) explains most of the variance. The remainder (black curve), however, remains quite large especially at the early stages of the measurements.

AUTOCORRELATION : Autocorrelation (as a function of time difference L) is the ratio between covariance of a term and the term at t + L and the variance of this term. It corresponds to the correlation between adjacent timestamps in the time series. Is used to define q in the ARIMA model.

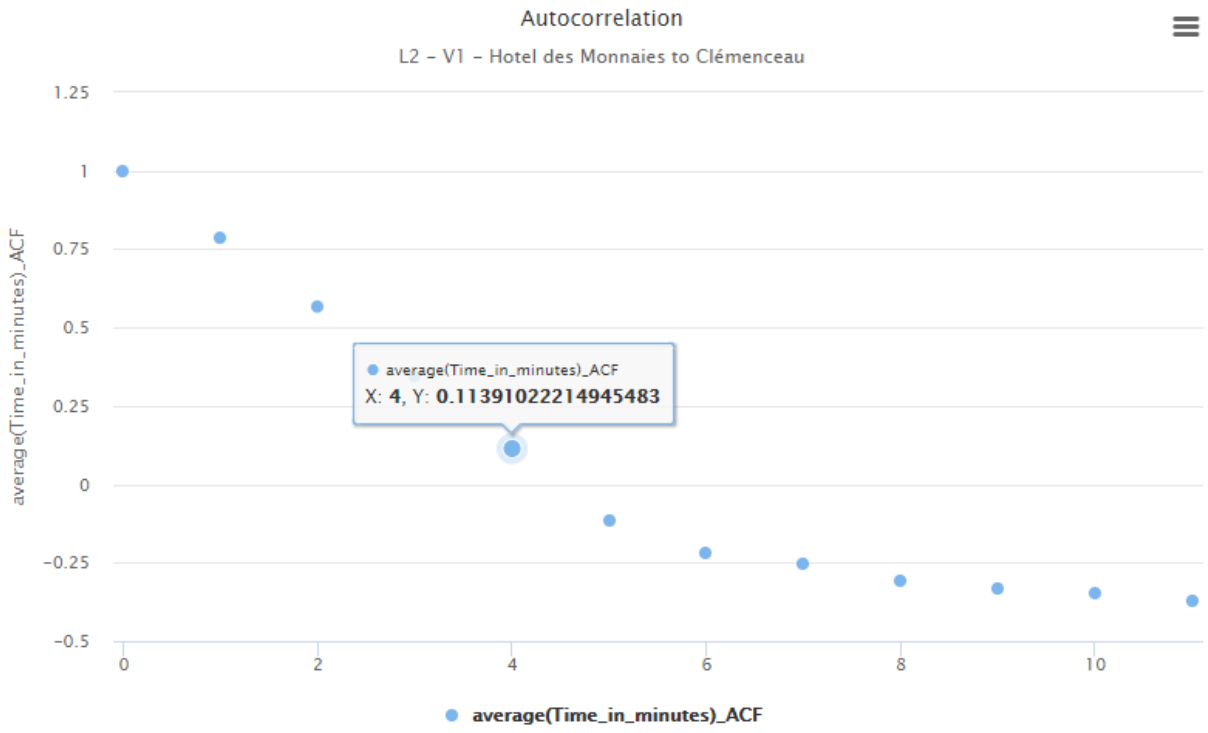


Figure 25: Autocorrelation

As shown in figure 25, above, the 4^{th} neighbor in the time series still explains more than 11% of the variance. We will therefore set q = 4 for this specific time series.

PARTIAL AUTOCORRELATION : Partial autocorrelation is the amount of correlation between the term at time t and the term a time t+L which is not explained by the correlation between the term at time t and the ones located between t and t+L. Is used to define p in the ARIMA model.

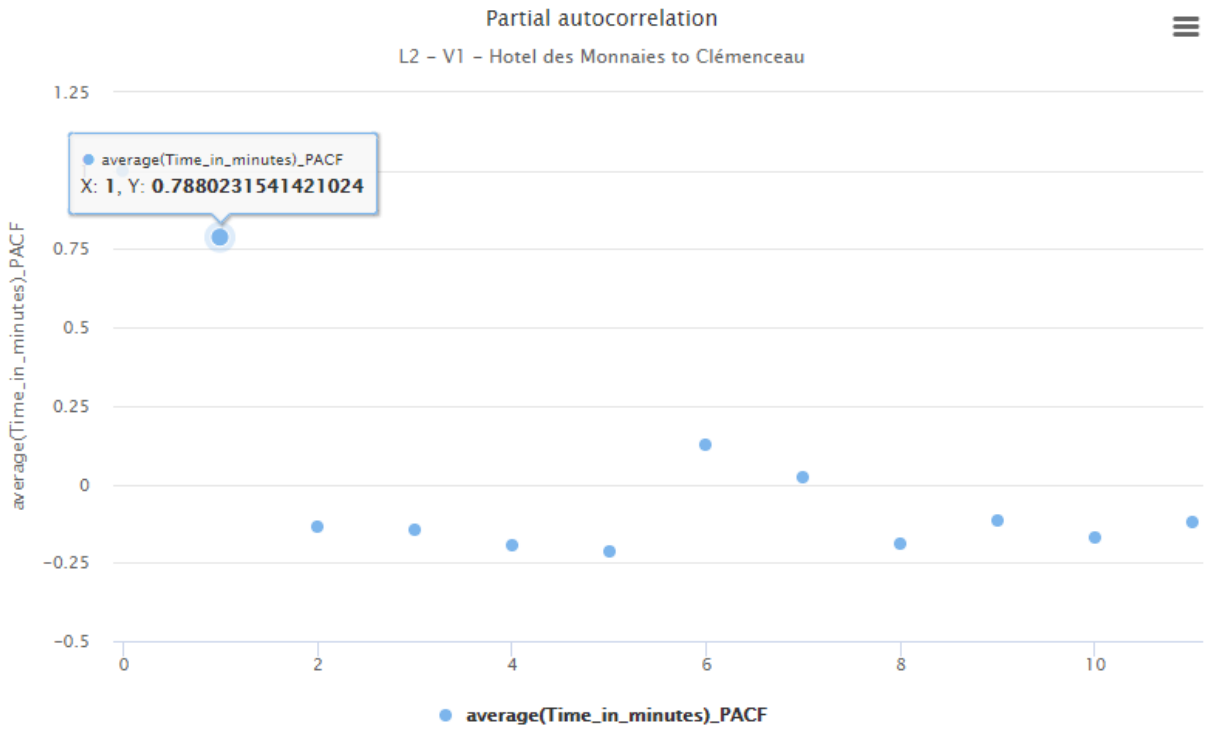


Figure 26: Partial autocorrelation

As shown in figure 26, above, the 1st neighbor in the time series has a direct impact on the value of the considered value, but the 2nd neighbor already has no impact other than the one explained by the correlation between the one considered and the 1st neighbor. We will therefore set p = 1 for this specific time series.

It should be noted that these p and q parameters should be checked for every time difference series computed.

5.2.2 ARIMA forecast

In order to perform ARIMA forecasting, and as requested in question 3 of this project, it was required to "test the accuracy of [our] forecasting by randomly splitting the given dataset in disjoint training and testing subsets". Forecast validation was therefore performed in RapidMiner. A window size of 120 hours (training set) and a step size of 1 hour were selected. The horizon size (test set) was set to 5 hours.

These input parameters, as well as the p = 1, d = 0 (stationary time series) and q = 4, give a Root Mean Squared Error of 3,898 minutes, an Absolute Error of 3,354 minutes and a relative error of 17,72%. Let us be clear, these results are not as accurate as one would have liked.

These accuracy figures are given for a main criterion being AIC (Akaike information criterion). Choosing BIC (Bayesian information criterion) or AICC (AIC with correction for small sample sizes) does not drastically modify the accuracy of the model.

The forecast validation graph shown in figure 27, below, illustrates the differences between the model and the actual data.

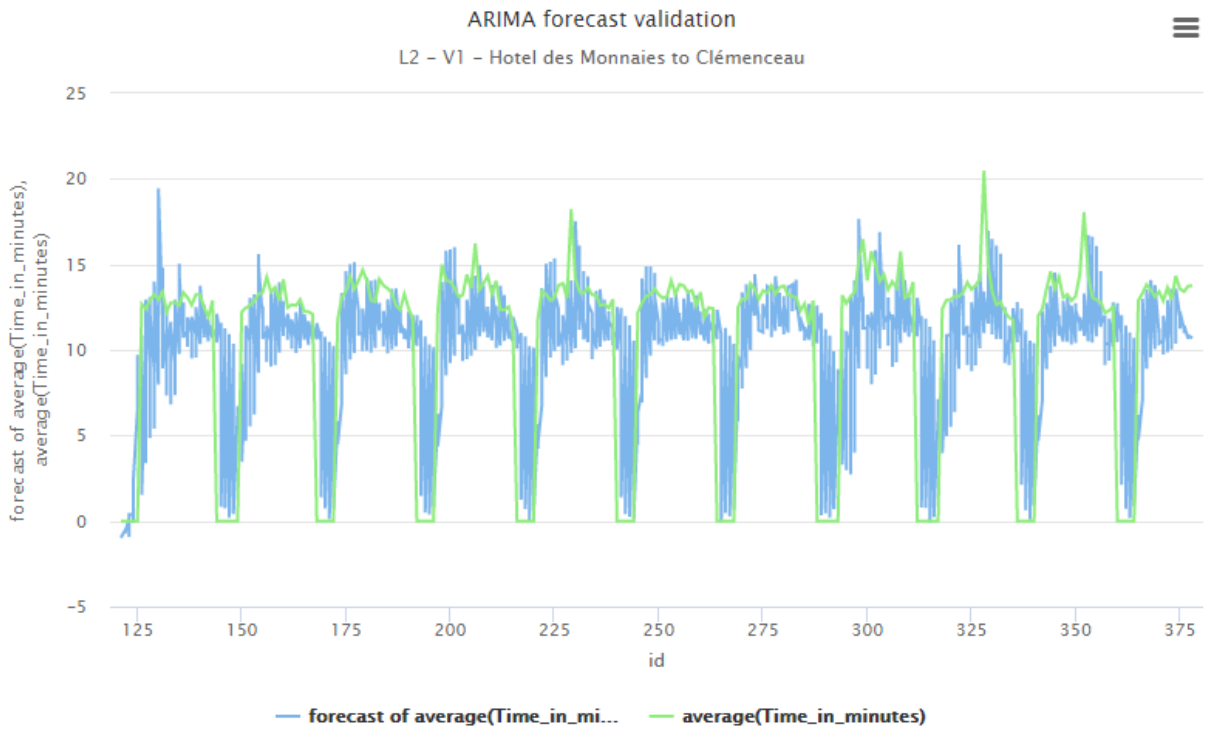


Figure 27: Forecast validation

At last, the proposed model is used to forecast the travel time 3 hours after the end of the time series. Figure 28, below, shows that the travel time should be decreasing. This proposition of the model is in line with intuition since the measurements were stopped at the end of a weekday.

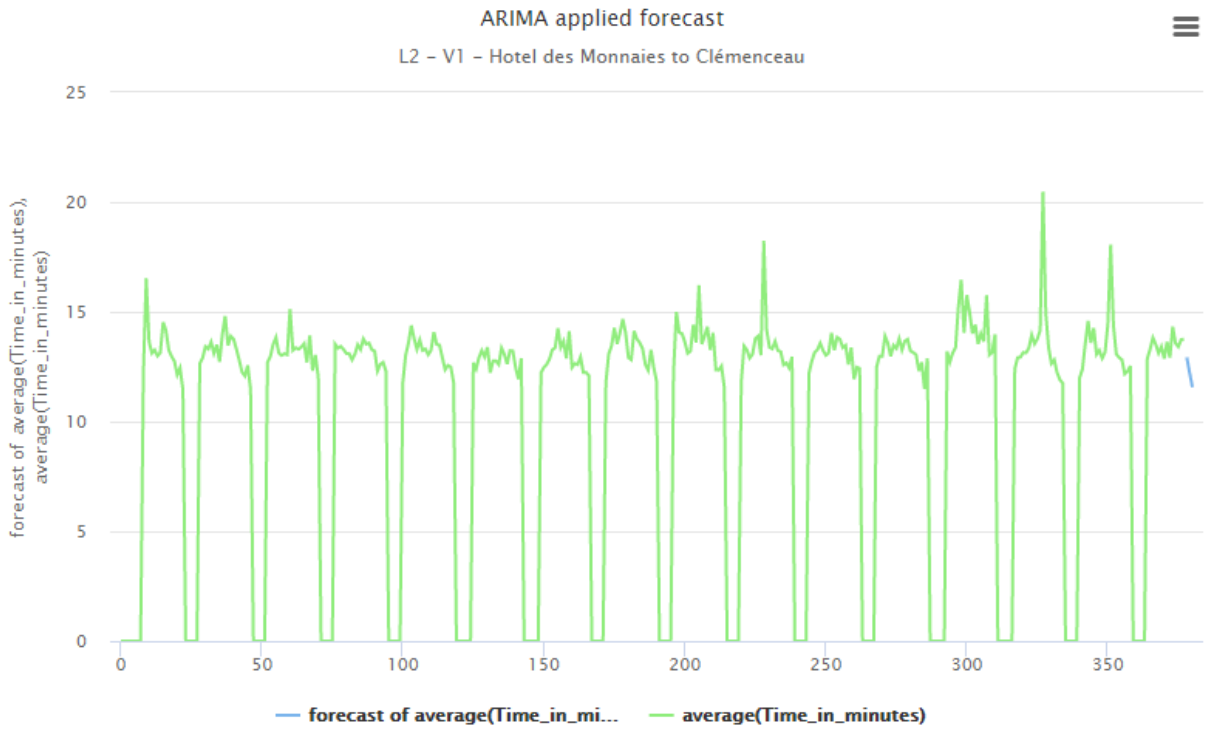


Figure 28: ARIMA forecast

6 GPS tracks inference

For this task of the project, a list of tracks is given, and it is requested the transport mode of these tracks is inferred. The tracks file is a simple `csv` file which contains several points for each track. Each point has the following data:

- `TrackID`: An Integer indicating the track to which the point belongs.
- `lat` : A float which indicates the latitude of the point.
- `lon` : A float which indicates the longitude of the point.
- `time` : A string which indicates at which instant this data has been collected.

6.1 Data processing

The most challenging part of this task was to find an *appropriate intersection* between these data and the data which was available from STIB. Finally, it has been decided to use both **position analysis** and **speed analysis** in order to indicate transport mode of a given track.

6.1.1 Position analysis

In order to find such a positional intersection, STIB `shape files` have been used. Several libraries of Python have been used to extract the points of each line. The library **GeoPandas** maybe was the most useful library that has been used. The STIB `shape files` contain feature dictionary or implement the `__geo_interface__` of lines. This data has been extracted by using the `GeoDataFrame.from_features` method. The following figures illustrate the data extraction that has been performed in this step:

Figure 29: STIB lines scheme

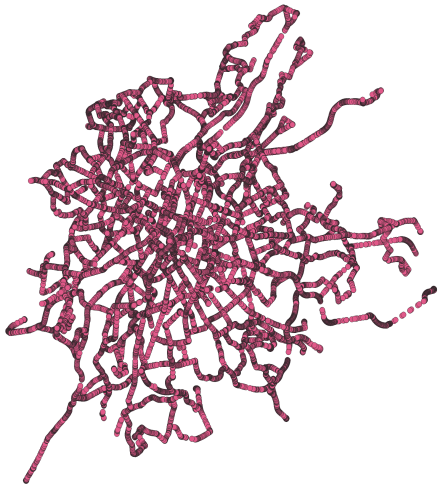


Figure 30: Points of STIB lines

Once lines points have been extracted, the following process have been used:

1. Each point of each track, P_t, has been compared with each point of each line, P_l (line variant also have been taken into consideration). For this comparison, the challenging part was that P_t and P_l had different CRS (Coordinate Reference System). STIB data use the *Belgian Lambert 72 (EPSG: 31370)* CRS while tracks CRS is *WGS 84 (World Geodetic System) (EPSG: 4326)*. So the tracks points CRS have been set to *Belge Lambert 72 (EPSG: 31300)* in order to be comparable with STIB lines points.
2. If the distance between P_t and P_l was less than 10 meters, then the line l is set to be a candidate for the point P_t.
3. If the line l was a candidate for at least $\frac{3}{4}$ of track points, it has been considered as a candidate line for that track.
4. A list provided in order to contain all candidate lines.

6.1.2 Speed analysis

Once candidate lines list is prepared, speed analysis is performed. The speed analysis is very important because in this step it becomes possible to identify if the traveller is in a STIB vehicle or he/she is using his/her own vehicle. The following steps are implemented in order to analyze speed:

1. For each candidate line, the closest stop to the first point of the track, P_s, and the last point of the track, P_e, is found.
2. By using the analysis of the first task, the average speed between P_s and P_e is calculated.
3. The average speed of the track is calculated. As each point of the track is provided almost instantly, a scale of 30 seconds is defined for this calculation. The reason of using this scale, is that the points that have been used in order to calculate average speed for the speed analysis task, has a time difference of almost 30 seconds.
4. If, for a candidate line, the average speed of the track and the average speed that has been calculated by using the first task analysis, have a difference larger than 15 $\frac{km}{h}$, this candidate line is ignored. Otherwise, the candidate line is considered as the track line.

It is noteworthy that, if more than one candidate line satisfies the conditions, the line with maximum number of points (position) correspondence has been considered as the track line.

6.2 Results

The result of execution using the file `GPStracks.csv` is shown below:

6.2.1 Track ID 1

```
Track ID:   1
.......................
Position analysis:

Candidate lines:  []
Votes:  []  /  178
Percentages:  []
.......................
Speed analysis:

.......................
Result:

Track:  1  --> transport mode Other (Probably the line is:  None )
```

6.2.2 Track ID 3

```
Track ID:   3
.......................
Position analysis:
```

```
Candidate lines:  [('032t', 1), ('032t', 2), ('082t', 1), ('082t', 2), ('049b', 2), ('050b', 2)]
Votes:  [88, 91, 88, 91, 91, 91]  /  115
Percentages:  [76.522, 79.13, 76.522, 79.13, 79.13, 79.13]
.........................
Speed analysis:

Line and variance  ('32', '1')
Stops (from --> to):  0631  -->  5123
Average line speed:  0.0
Average track speed:  22.29658714734742
.........................
Line and variance  ('32', '2')
Stops (from --> to):  0636  -->  5159F
Average line speed:  -inf
Average track speed:  22.29658714734742
.........................
Line and variance  ('82', '1')
Stops (from --> to):  0631  -->  5123
Average line speed:  15.062294002987745
Average track speed:  22.29658714734742
.........................
Line and variance  ('82', '2')
Stops (from --> to):  0636  -->  5159F
Average line speed:  -inf
Average track speed:  22.29658714734742
.........................
Line and variance  ('49', '2')
Stops (from --> to):  2500  -->  5159
Average line speed:  -inf
Average track speed:  22.29658714734742
.........................
Line and variance  ('50', '2')
Stops (from --> to):  2500  -->  5159
Average line speed:  -inf
Average track speed:  22.29658714734742
.........................
Result:

Track:  3  --> transport mode t (Probably the line is:  82 )
```

6.2.3 Track ID 4

```
Track ID:   4
.........................
Position analysis:
```

```
Candidate lines:  [('050b', 2), ('050b', 1)]
Votes:  [233, 211]  /  241
Percentages:  [96.68, 87.552]
.......................
Speed analysis:

Line and variance  ('50', '2')
Stops (from --> to):  5159  -->  2663
Average line speed:  -inf
Average track speed:  14.16390422669891
.......................
Line and variance  ('50', '1')
Stops (from --> to):  2546  -->  5721
Average line speed:  15.765670238368289
Average track speed:  14.16390422669891
.......................
Result:

Track:  4  --> transport mode b (Probably the line is:  50 )
```

6.2.4 Track ID 5

```
Track ID:   5
.......................
Position analysis:

Candidate lines:  [('032t', 1), ('032t', 2), ('082t', 1), ('082t', 2), ('097t', 1), ('097t', 2)]
Votes:  [200, 198, 200, 198, 198, 193]  /  218
Percentages:  [91.743, 90.826, 91.743, 90.826, 90.826, 88.532]
.......................
Speed analysis:

Line and variance  ('32', '1')
Stops (from --> to):  2608F  -->  5123
Average line speed:  -inf
Average track speed:  15.9978424378629
.......................
Line and variance  ('32', '2')
Stops (from --> to):  2663G  -->  5159F
Average line speed:  0.0
Average track speed:  15.9978424378629
.......................
Line and variance  ('82', '1')
Stops (from --> to):  2608F  -->  5123
```

```
Average line speed:  -inf
Average track speed:  15.9978424378629
.......................
Line and variance  ('82', '2')
Stops (from --> to):  2663G  -->  5159F
Average line speed:  14.998223517057735
Average track speed:  15.9978424378629
.......................
Line and variance  ('97', '1')
Stops (from --> to):  2608F  -->  5123
Average line speed:  -inf
Average track speed:  15.9978424378629
.......................
Line and variance  ('97', '2')
Stops (from --> to):  2663G  -->  5175
Average line speed:  14.435552062173322
Average track speed:  15.9978424378629
.......................
Result:

Track:  5  --> transport mode t (Probably the line is:  82 )
```

6.2.5 Track ID 6

```
Track ID:   6
.......................
Position analysis:

Candidate lines:  []
Votes:  []  /  277
Percentages:  []
.......................
Speed analysis:

.......................
Result:

Track:  6  --> transport mode Other (Probably the line is:  None )
```

6.2.6 Track ID 7

```
Track ID:   7
.......................
Position analysis:
```

```
Candidate lines:  [('007t', 1), ('007t', 2), ('025t', 1), ('025t', 2)]
Votes:  [166, 189, 152, 175]  /  195
Percentages:  [85.128, 96.923, 77.949, 89.744]
.......................
Speed analysis:

Line and variance  ('7', '1')
Stops (from --> to):  0906  -->  5258
Average line speed:  -inf
Average track speed:  19.948385691111582
.......................
Line and variance  ('7', '2')
Stops (from --> to):  0901  -->  3481
Average line speed:  13.851716761672703
Average track speed:  19.948385691111582
.......................
Line and variance  ('25', '1')
Stops (from --> to):  0906  -->  3480
Average line speed:  -inf
Average track speed:  19.948385691111582
.......................
Line and variance  ('25', '2')
Stops (from --> to):  0901  -->  2397F
Average line speed:  12.965666453178818
Average track speed:  19.948385691111582
.......................
Result:

Track:  7  --> transport mode t (Probably the line is:  7 )
```

6.2.7 Track ID 8

```
Track ID:   8
.......................
Position analysis:

Candidate lines:  []
Votes:  []  /  314
Percentages:  []
.......................
Speed analysis:

.......................
Result:
```

```
Track:  8  --> transport mode Other (Probably the line is:  None )
```

6.2.8 Track ID 10

```
Track ID:   10
.......................
Position analysis:

Candidate lines:  []
Votes:  []  /  62
Percentages:  []
.......................
Speed analysis:

.......................
Result:

Track:  10  --> transport mode Other (Probably the line is:  None )
```

6.2.9 Track ID 11

```
Track ID:   11
.......................
Position analysis:

Candidate lines:  [('025t', 1)]
Votes:  [179]  /  209
Percentages:  [85.646]
.......................
Speed analysis:

Line and variance  ('25', '1')
Stops (from --> to):  5462F  -->  0906
Average line speed:  14.898977287950377
Average track speed:  26.457712832170717
.......................
Result:

Track:  11  --> transport mode t (Probably the line is:  25 )
```

6.2.10 GPStracks analysis in one sight

The following table shows the result of execution for file `GPStracks.csv`:

Tracks ID/ Information	Transport mode	Possible line
1	Other (o)	—
3	Tram (t)	82
4	Bus (b)	50
5	Tram (t)	82
6	Other (o)	—
7	Tram (t)	7
8	Other (o)	—
10	Other (o)	—
11	Tram (t)	25

7 Reachability analysis

Another useful analysis to perform on this data is the reachability. Given a point in the map and a time interval, the algorithm gives as set of places where this point is reachable within the time limit. It can also be seen the other way around, it gives the set of places that are reachable from the point within the time limit. This information can be used by the police (or `STIB` security) to catch a fugitive, it would allow us to set perimeters around the different places the person could have reached.

7.1 Data processing

Due to the limitations of the data availability, It was decided to take into account only the following means of locomotion:

- The different public transports of STIB
- Walking, assumed to be $5\frac{km}{h}$

7.1.1 Algorithm

The algorithm works in an iterative mode.

The first step is to list all the stop ID that are reachable by running within the time interval from the point, as well as the theoretical arrival time.

Next, the iteration phase begins.

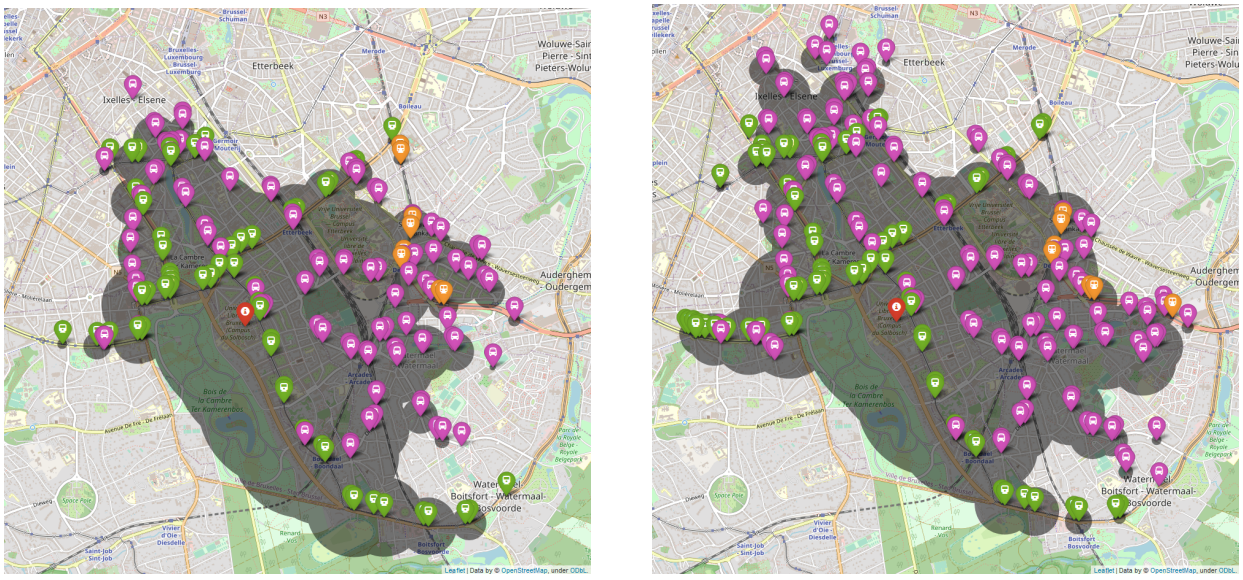
For each stop that just got added or modified at the last iteration, all the stops which are reachable within the time left are listed. These reachable stops are either obtained by walking or by taking transports. Next,

these stops are merged with the main list of stops such as if a stop is already present in the main list, the arrival time is checked, if this arrival time is earlier than the one of the main list, it is modified with the current one, else it is dismissed.

The iteration stops when there are no longer any new or modified stops in the main list.

7.2 Result

The results below were obtained by giving the start position located in the middle of the ULB's campus Solbosch at the 23^{rd} October 2021, at 08:10 and 23:10, respectively.



(a) 08:10

(b) 23:10

Figure 31: Reachable places with 15 minutes from ULB at 23/10/2021

In these results, it can be seen that in the morning the reachable area is more expanded than the one at night. This can easily be explained, as the average speed at 23h is higher than at 8h. It can also be seen that the area in the morning is not totally included in the one at night. It can be due to certain lines having their speed evolves differently in time. Another reason could be simply a problem in the data, such as missing or too few data.

8 Conclusion

Data preparation was certainly the biggest challenge for this project. In particular, the algorithm that creates the vehicle IDs based on the vehicle positions available in the `JSON` file has a great influence on the results. Any mistake in this algorithm may have severe repercussions on the results of all following tasks.

Considering the consistent results (speeds, delays, etc. for the 3 transport modes) exposed in this report, the way this data preparation was done by our team is considered sufficient to obtain interesting results as shown in this report, but could be improved if a vehicle ID was provided in the input data.

In addition to this major drawback, some corrections could be brought to the data, such as:

- Adding the trip id's to all vehicle positions. This would be really helpful to compute the time delays more precisely.
- Adding the metros positions with regard to last stop would be helpful to have a more detailed analysis of metros speed, just like for buses and tramways.
- Harmonising the Stop IDs between the `JSON` files and the `GTFS` files would make the analysis more straightforward (currently some are different between the two databases).
- Etc.

Despite these minor issues, it is very interesting to observe how a public company like STIB / MIVB shares some of its information publicly. The possibilities that this decision has opened are infinite and we believe that this can only contribute to the development of deeper knowledge of Brussels' public transports and to the development of tools to improve users' experience and, eventually, to make their trips faster and safer.

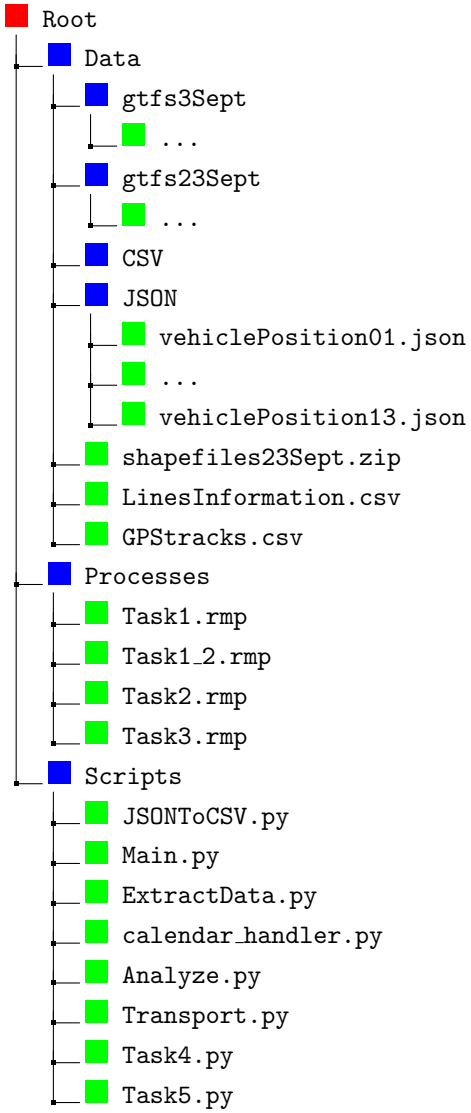
A User manual

In order to reproduce the results presented in this report, the following steps should be taken.

A.1 Prerequisite

A.1.1 Directory disposition

For all the scripts and processes to work as expected, the necessary files should be placed as follows:



The directories are represented in blue and the files in green.

A.1.2 Software

Python 3 (3.9.7) should be installed as well as several libraries listed below:

- Geopandas (0.9.0)
- Folium (0.12.1.post1)
- Haversine (2.5.1)

The libraries themselves have dependencies, which is not a problem, as `pip` should automatically install them.

A.1.3 Data generation

Several files need to be generated in order to be able to launch the different tasks.

These python scripts should be executed in the `Scripts` directory:

```
$ python3 JSONToCSV.py
$ python3 Main.py
```

Then, another files have to be generate by running the `Task1` process on RapidMiner.

A.2 Task 1

The task is entirely processed in RapidMiner.

As the task is a prerequisite of several tasks, all the needed files have already been generated. To visualize the results, it is simply needed to execute the task1_2 process.

A.3 Task 2

First this line should be run:

```
$ python3 calendar_handler.py
```

Then the task is entirely processed in RapidMiner. The file `task2.rmp` should be imported and executed.

A.4 Task 3

This task is entirely processed in RapidMiner. The input file is `vehiclePositionID.csv`. In order to run the process, the file `task3.rmp` should be first imported in RapidMiner.

Then, boxes `START STOP` and `END STOP` should be filled in with the `LineID`, `Variance` and `PointIDs` for the start and end stop.

Based on this input, the data exploration subprocess will highlight the appropriate p and q parameters to use for the ARIMA forecasting model.

The other variable to choose in the applied ARIMA forecasting, is the forecast horizon (number of values to forecast).

The ARIMA forecast validation and the applied ARIMA model will run in a second step in order to calculate and show the final result of forecasting.

A.5 Task 4

In order to run the algorithm of the GPS tracks inference, it is simply needed to execute the `Task4.py` file with Python 3.

By default, only the provided tracks are loaded and analyzed. To analyze other tracks, it is simply needed to change the path of the file passed in the `readTracks` function.

A.6 Task 5

In order to run the algorithm of the reachability analysis, it is simply needed to execute the `Task5.py` file with Python 3.

To change different parameters, it is just needed to modify the values in the main function. By default, these values are set to the one used in the report at the result section of the reachability analysis.