

Working with the “10 minute city” model

Spatial Application Division Leuven

KU Leuven

Department of Earth and Environmental Sciences

Title	Working with the “10 minutes city” model
Subject	Data integration and Automation using FME software
Creator	Kelly Wittemans, Fien Vanongeval
Contact	kelly.wittemans@kuleuven.be
Date Issued	12-11-2021
Publisher	KU Leuven: R&D Division SADL
Description	Tutorial on how to work with the model « 10 minutes city »
Format	WORD
Audience	City of Brussels, department of urban planning
Language	English

Version history:

Version	Date	Modified by	Comments
2021a	12-11-2021	Kelly Wittemans	Document created for FME Desktop 2021 1.0.1 and R-4.0.3
2021b	19-11-2021	Fien Vanongeval	Writing of paragraph 3. Heatmap
2021c	13-12-2021	Fien Vanongeval	Appendix, Writing of paragraph 3. Heatmap, isochrones public transport
2021d	21-12-2021	Kelly Wittemans	Final modifications

Introduction

This tutorial is written in the framework of the project: *“Scientific support for the refinement of the concept of the “10-minute city” that in the context of the municipal plan for sustainable development of the City of Brussels and the operationalization through a cartographic tool must be integrated.”*

In Phase 3 and 4 of the project the theoretical proposal of the model of the ‘City of Proximity’ as developed in phases 1 and 2 is concretized in the technological reality of the city of Brussels in order to obtain spatial results on the territory of the city and to analyze the obtained results. This tutorial guides the user through the implementation and the execution of the model to construct maps indicating the levels of accessibility within the Administration of the City of Brussels. The level of accessibility is determined from the presence or absence of certain facilities (defined in phases 1 and 2) within the territory of the city. For each facility, hereafter referred to as supply variables, a map showing isolines of equal travel time (isochrones) is constructed. Thereafter, a superposition is carried out of the isochrone maps belonging to the same (sub)theme (defined in phases 1 and 2). In this way, a heatmap for each respective (sub)theme is obtained. Each supply variable receives a weight (defined in phase 2), which is used to calculate an overall score in the heatmaps. This score finally translates into a level of accessibility.

The workflow is split in three parts: data preparation, isochrones and heatmaps. A background knowledge of FME is necessary.

1. Data Preparation

Before calculating the isochrones or heatmaps, the input data needs be preprocessed. There are two types of input data: the supply variables (Modified_input.fmw) and the roadmap (Cleaning Urbis.fmw).

1.1 Supply variables

Based on the summary diagram (cf. phases 1 and 2), the supply variables were selected. The data provided in shapefile format were downloaded after connecting to the Brucity / I-city network and filtered, i.e., selecting the necessary attributes, using FME software (FME desktop 2021 1.0.1).

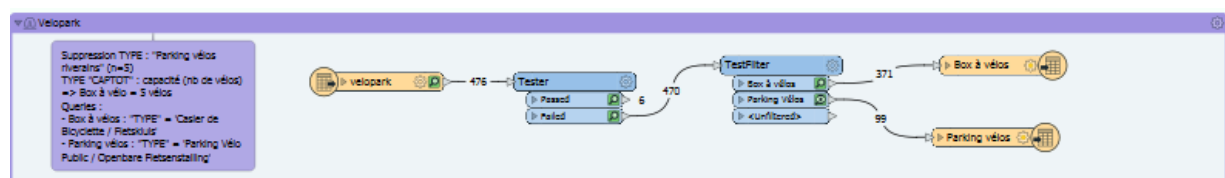


Figure 1: The following steps are executed in FME: 1. Load shapefile data of veloparks, 2. Filter based on the type “Parking Vélo Riverains / Buurtstalling”, 3. Filter the two variables, i.e. Box à vélos and Parking vélos and 4. Write respectively separate shapefiles for the two variables.

For each supply variable, a specific query is implemented in order to create separate shapefiles. In Figure 1 an example is given for the variables “box à vélos” and “Parking vélos” filtered from the shapefile “vélopark”. The new shapefiles created for each variable are used for further processing in later steps of the FME flow.

When running the whole FME workspace or part of it, the user is prompted to fill in the destination folder to save the newly created shapefile and the source files(s) to specify the file path(s) of the original shapefile(s) (cf. Figure. 2)

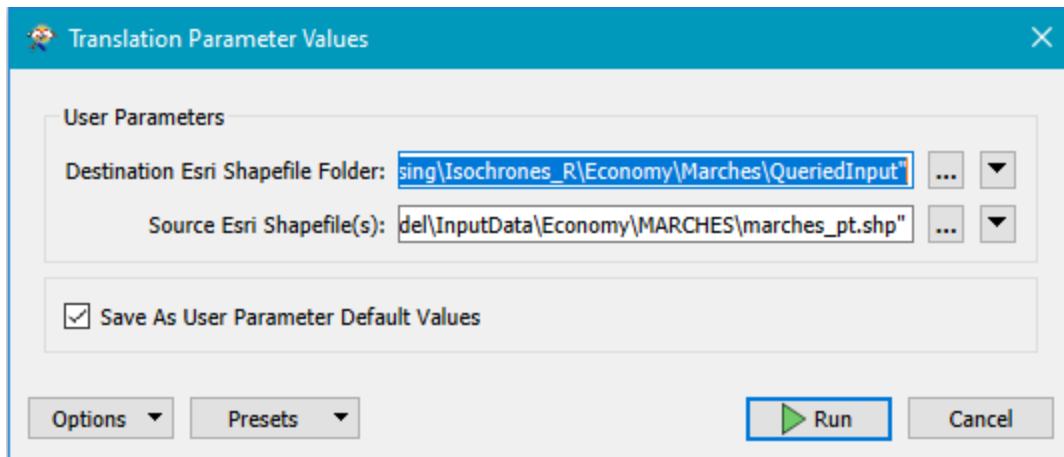


Figure 2: User parameters (destination folder and source file) for the variable 'marchés'.

The data¹ containing future scenario's (2024 and 2040) also needs some preprocessing. Firstly, the dataset is uploaded in FME and separated based on the time horizon and on the type of variable (cf. Figure 3). By doing so a separate shapefile is created for each variable within each time horizon.

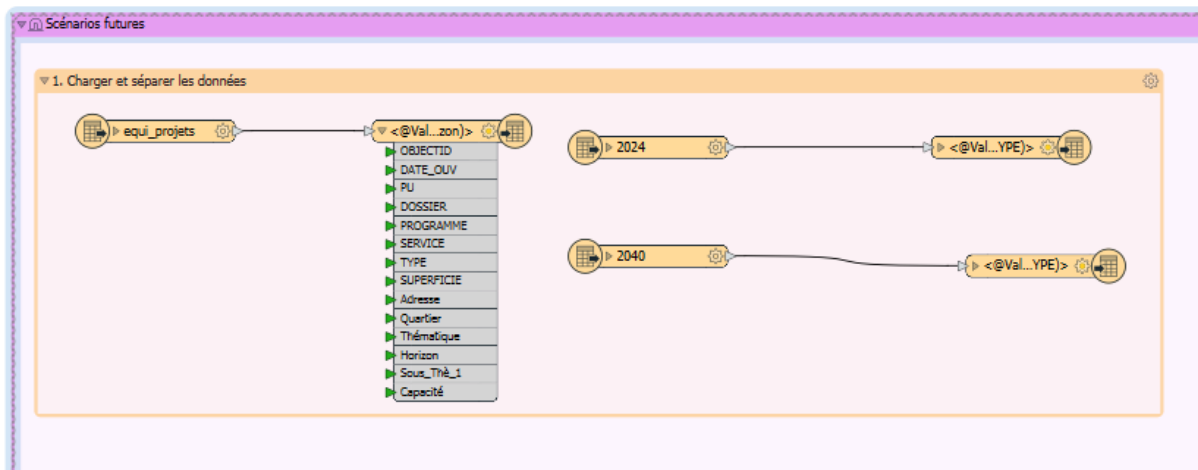


Figure 3: The complete dataset of future scenarios is ('equi-projets') is uploaded in FME and separated based on the time horizon ('Horizon'). Subsequently the datasets of the two time horizons are uploaded and separated based on the variable ('TYPE').

Within the dataset of the future scenarios, the variables which normally consist of polygon data are provided as point data instead. Therefore, some additional steps are needed. Around each point buffers are created with a surface area equal to the surface area specified as attribute (cf. Figure 5). Minor adjustments were first made in the attribute table of for example micro, meso and macro green spaces. The values within the attribute of surface area were converted to the same units (m2), the unit strings were removed and if the cell was empty, it was filled in (cf. Figure 4).

¹ Dataset first preprocessed by ULB: spatial dataset containing all necessary attributes (type of variable according to the technical scheme of the variables used in the 10-minute city model of Brussels (Appendix 1), horizon, geometry).

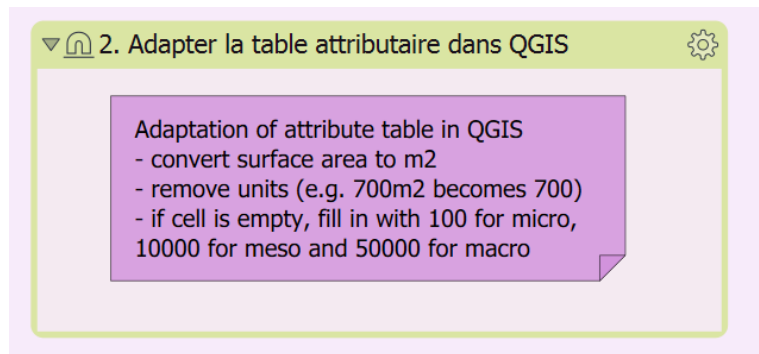


Figure 4: Adaptations made in the attribute table of micro, meso and macro green spaces using QGIS.

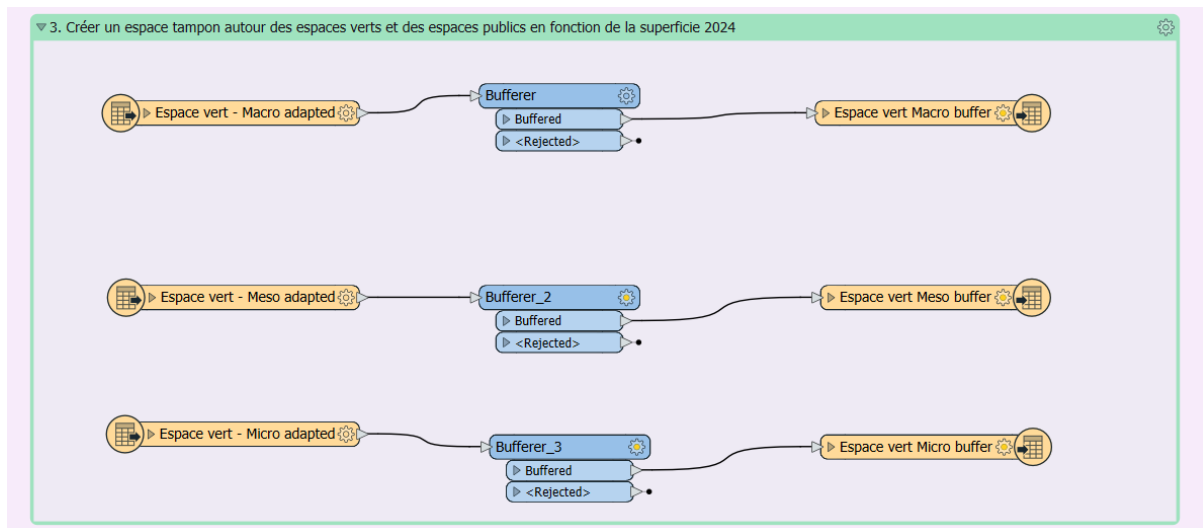


Figure 5: Create buffer around each point representing green spaces based on the surface area attribute ('SUPERFICIE').

1.2 Roadmap

A roadmap is required to calculate isochrones around the collection of datapoints of a supply variable. The map was obtained from UrbIS, the reference database of Brussels' region. Highways, tunnels and tramlines were filtered out from the original map since these roads are not intended for pedestrians and cyclists (cf. Figure 6). The final roadmap still has some shortcomings (cf. final report).

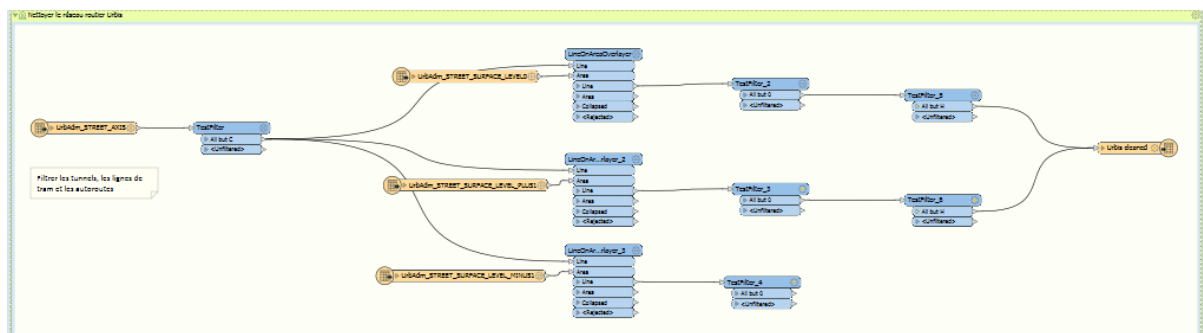


Figure 6: The following steps are executed in FME: 1. Load shapefile data of UrbAdm_STREET_AXIS, 2. Filter the tramlines (type C), 3. Load the shapefile data of UrbAdm_STREET_SURFACE_LEVEL0 and UrbAdm_STREET_SURFACE_LEVEL_PLUS1, 4. filter the roads that overlay with the previous two shapefiles, so including everything except tunnels (including level 0 and level 1 roads and not including minus1 roads), 5. Filter the autoroutes (type H), 6. Write a new shapefile containing the filtered road UrbIS network.

The filtered roadmap is saved in a separate file and used in later steps.

In order to run the script, the user has to specify the file paths of the different UrbIS road networks used in the FME workspace (UrbAdm_STREET_AXIS, UrbAdm_STREET_SURFACE_LEVEL0, UrbAdm_STREET_SURFACE_LEVEL_PLUS1) and the file path of the location where the shapefile of the filtered road network that is created should be saved. This is presented in Figure 7.

Translation Parameter Values

User Parameters

Source Esri Shapefile(s):

Source Esri Shapefile(s):

Source Esri Shapefile(s):

Destination Esri Shapefile Folder:

☒ Save As User Parameter Default Values

Options Presets **Run** Cancel

Figure 7: User parameters (source files and destination folder) for the filtering of UrbIS road network.

Optional: An additional FME Workspace ('Cleaning OSM.fmw') is created to filter road features from OpenStreetMap data (Figure 8).

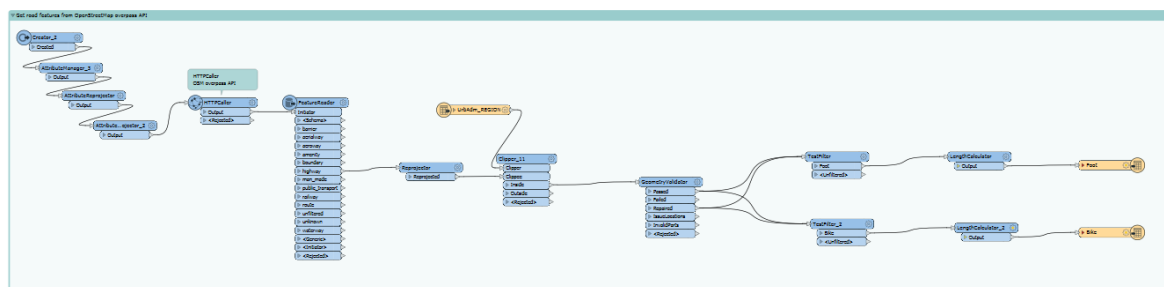


Figure 8: The following steps are executed in FME: 1. Load data directly from Openstreetmap using an overpass API, 2. Select highway features, 3. Clip data within the within the Administration of the City of Brussels, 4. Select road features with labels corresponding to roads intended for pedestrians and roads intended for cyclists, 5. Write a new shapefile containing the filtered Openstreetmap road network (separate shapefile for pedestrians and cyclists).

2. Isochrones

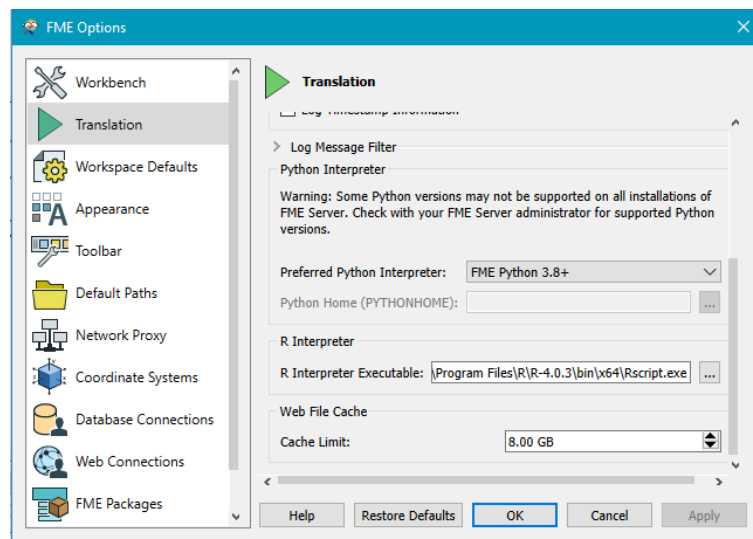
For each of the supply variables, isochrone maps depicting the areas accessible from all datapoints within a certain time threshold are generated. In order to optimize and automate this process, an R script is integrated in FME (isochrones – final.fmw). The FME workflow is different depending on the spatial element, points or polygons.

! IMPORTANT !:

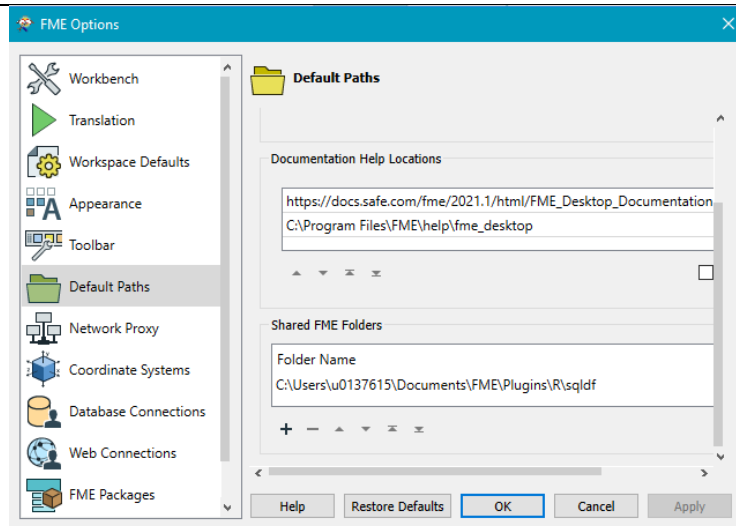
To execute R scripts from FME Server, such as through the “RCaller” transformer in an FME Desktop workspace, you must perform the following on all machines that run FME Engines:

1. Install R (Download R Installer from <https://www.r-project.org/>).
2. Install Rstudio (Download Rstudio from <https://www.rstudio.com/products/rstudio/download/>).
3. Install the sqldf package for R
 - i) Open R studio
 - ii) Run the following command at the R command prompt of Rstudio:
`install.packages("sqldf")`
 - iii) This will launch the installation procedure. The sqldf package will be installed to the system wide R library
4. Install all other packages needed to run your Rscript (for the script used for calculating isochrones run command line 15 till 68 of the Rscript provided in Appendix 2)

FME will try its best to find R as installed on your system; however, if R is installed in a non-default location, or you have multiple R interpreters installed, it may be necessary to specify the *R Interpreter Path* under **Tools > FME Options > Translation > R Interpreter** (see image below)



It's also best to place your R libraries in a shared resources folder. The R libraries are in Windows by default located in **Documents > R > win-library > 4.0**. The location of the shared resources folder is set in Workbench under **Tools > FME Options > Default Paths > Shared FME Folders** (see image below).



In Windows, the R folder is located by default in **Documents > FME > Plugins**.

Additional modules dropped into the R folder will be picked up by FME.

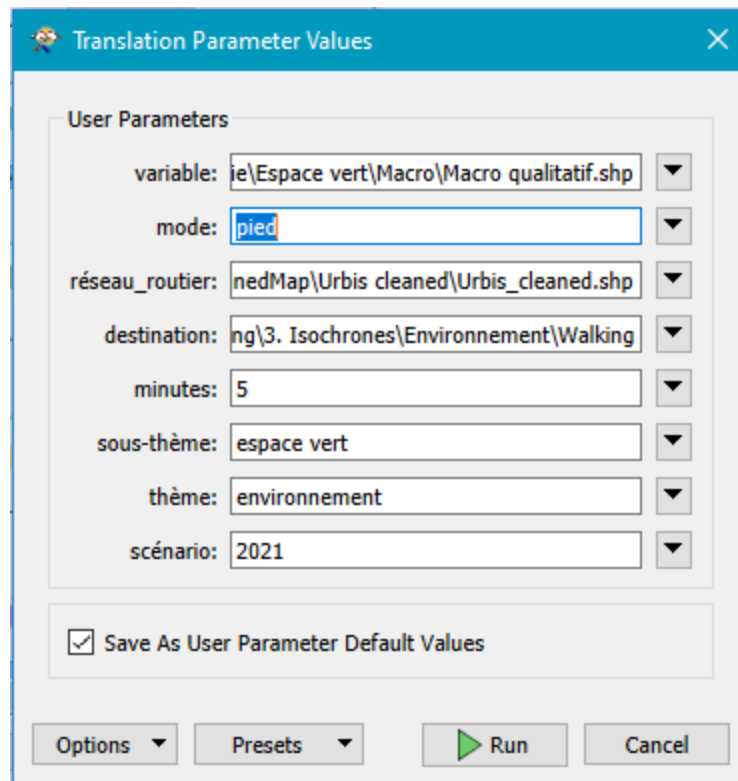
More information:

- https://docs.safe.com/fme/html/FME_Server_Documentation/AdminGuide/Using-R-With-FME-Server.htm
- https://docs.safe.com/fme/2016.1/html/FME_Desktop_Documentation/FME_Transformers/Transformers/rcaller.htm

2.1 Point data

Most of the variables used in the 10-minute city model consist of point data. Before running the script, the user is prompted to fill in list of user parameters as shown in Figure 9. The user can choose the time threshold (in minutes), the mode of transport (by bike or by foot), the folder location of the road network and the folder locations of the different variables of choice. Subsequently, the FME workflow calculates automatically all isochrone maps. The resulting maps are stored as shapefiles in a destination folder specified by the user.

Below an explanation is given for each user parameter:



The dialog box titled "Translation Parameter Values" contains the following fields and options:

- variable:** ie\Espace vert\Macro\Macro qualitatif.shp
- mode:** pied
- réseau_routier:** nedMap\Urbis cleaned\Urbis_cleaned.shp
- destination:** ng\3. Isochrones\Environnement\Walking
- minutes:** 5
- sous-thème:** espace vert
- thème:** environnement
- scénario:** 2021
- ☒ Save As User Parameter Default Values
- Buttons: Options, Presets, Run, Cancel

Figure 9: User parameters (file path(s) of variable(s), mode of transport, file path of the road network, destination folder for the created isochrones, number of minutes, sub-theme, theme and scenario) for calculation the isochrones for point data.

- **Variable:** the complete path to the shapefile(s) of the interested variable(s) that were preprocessed in a previous step (cf. § Data preparation). Multiple variables of the same subtheme can be processed at the same time. The file paths should be separated by commas as shown in Figure 10.

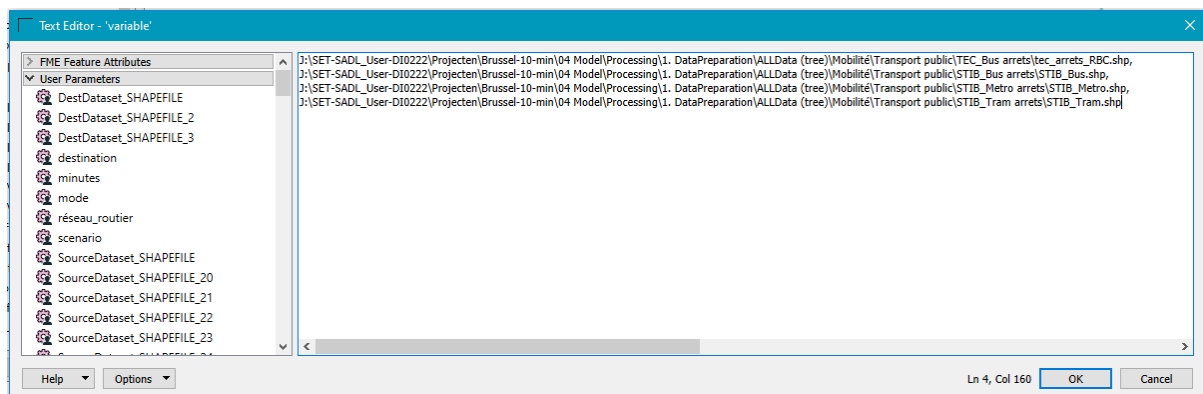


Figure 10: Multiple variables are processed by filling in all file paths of all variables of interest, separated by a comma.

- **Mode:** This refers to the mode of transport. The user can chose between by foot ("**pied**") or by bike ("**velo**")
- **Reseau routier:** the path to the shapefile of the filtered road network ((cf. § Data preparation)
- **Destination:** the path to the folder where the isochrone maps will be saved
- **Minutes:** the time threshold for the isochrones expressed in minutes
- **Sous-thème:** specification of the subtheme to which the variable(s) belong(s): "**transport**", "**partage**", "**stationnement velo**", "**commerces alimentaires**", "**commerces non-**

alimentaires”, “espace public”, “recyclage”, “espace vert”, “petite enfance”, “jeunesse”, “senior”, “sport”, “culture”, “cohesion sociale”, “sante”.

- Thème: specification of the subtheme to which the variable(s) belong(s): “mobilité”, “économie”, “environnement”, “vivre ensemble”.
- Scénario: specification of the scenario for which the user is calculating isochrones (“2021”, “2024” or “2040”)

In the FME workspace a short legend is provided to help the user to fill in the user parameters (cf. Figure 11). To get more insight in the variables, the themes and the subthemes, the user is advised to take a look at the technical scheme made during the project (Appendix 1).

User parameters:
 "mode": pied / velo
 "theme": mobilité / économie / environnement / vivre ensemble
 "sousthème": transport public/ partage / stationnement velo
 commerces alimentaires / commerces non-alimentaires
 espace public / recyclage / espace vert
 petite enfance / jeunesse / senior / sport / culture / cohesion sociale / sante
 "scenario": 2021 / 2024 / 2040

Figure 11: Legend for the user parameters 'mode', 'theme', 'sousthème' and 'scenario'.

In the FME workspace, the different user parameters are added as attributes in tables (cf. Figure 12).

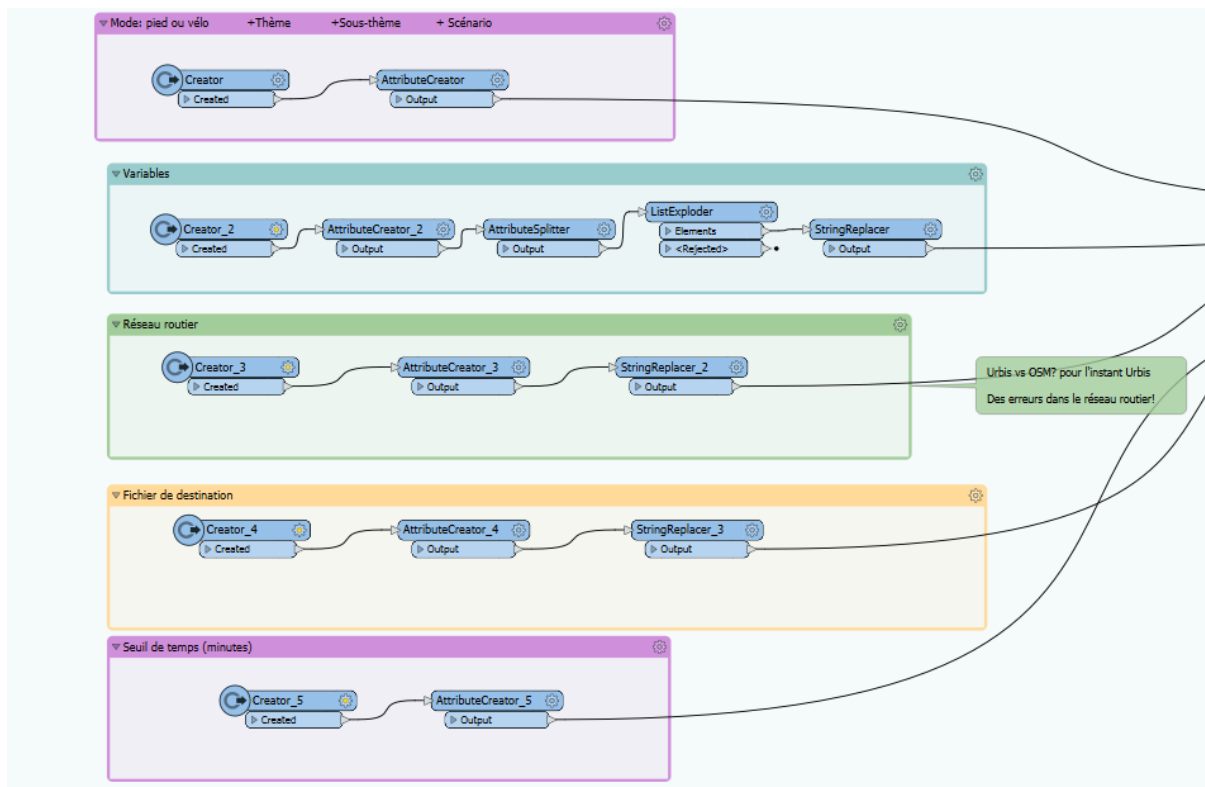


Figure 12: New tables are created (“Creator”) and filled with the user parameters (“AttributeCreator”). The string “\” is replaced by “/” in the file paths since this is the notation accepted by R (“StringReplacer”). The different variables are splitted into different attributes (“AttributeSplitter”).

After the user parameters are converted into tables, these tables are then used as input to the RCaller which runs the Rscript to calculate isochrones (cf. Figure 13).

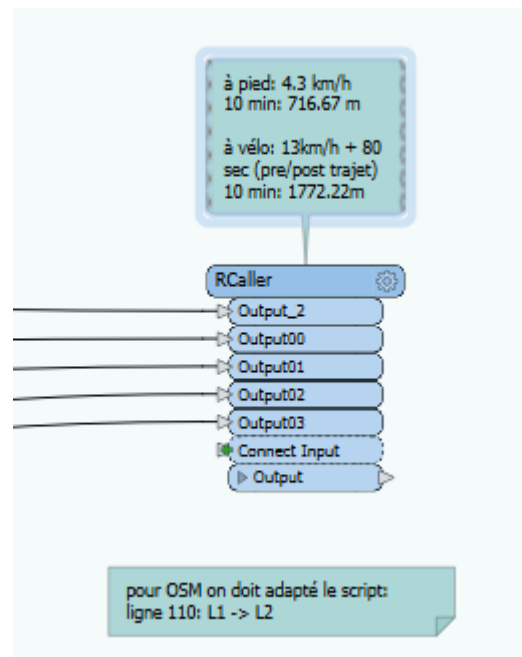


Figure 13: R script integrated in FME using the "RCaller" with the user parameters as input.

Figure 14 shows the RCaller transformer parameters. These parameters are already complete and don't need extra input from the user. The *inputs* consist of the user parameters that were converted into tables. Under *Rscript* the entire Rscript is included (see Appendix 2). This Rscript calculates automatically all the isochrone maps based on the user parameters. The variables of interest and the road network are loaded based on the file path given (user parameter: 'variables' and 'reseau_routier'). The points representing the variables are snapped to the closest road feature using the `snapPointsToLines` function (`maptools` package). Subsequently the mode of transport (user parameter: 'mode') and the amount of minutes (user parameter: 'minutes') determine the distance limit for the calculation of the isochrones. The main R package used for the calculation of the isochrones is `cppRouting` (more information: <https://cran.r-project.org/web/packages/cppRouting/cppRouting.pdf>). Lastly, the generated isochrone maps are saved as shapefiles in the destination folder specified by the user (user parameter: "destination"). The user parameters 'theme', 'soustheme' and 'scenario' are added as attributes to the shapefiles. In the following step, i.e. creating heatmaps, these additional attributes will be further used.

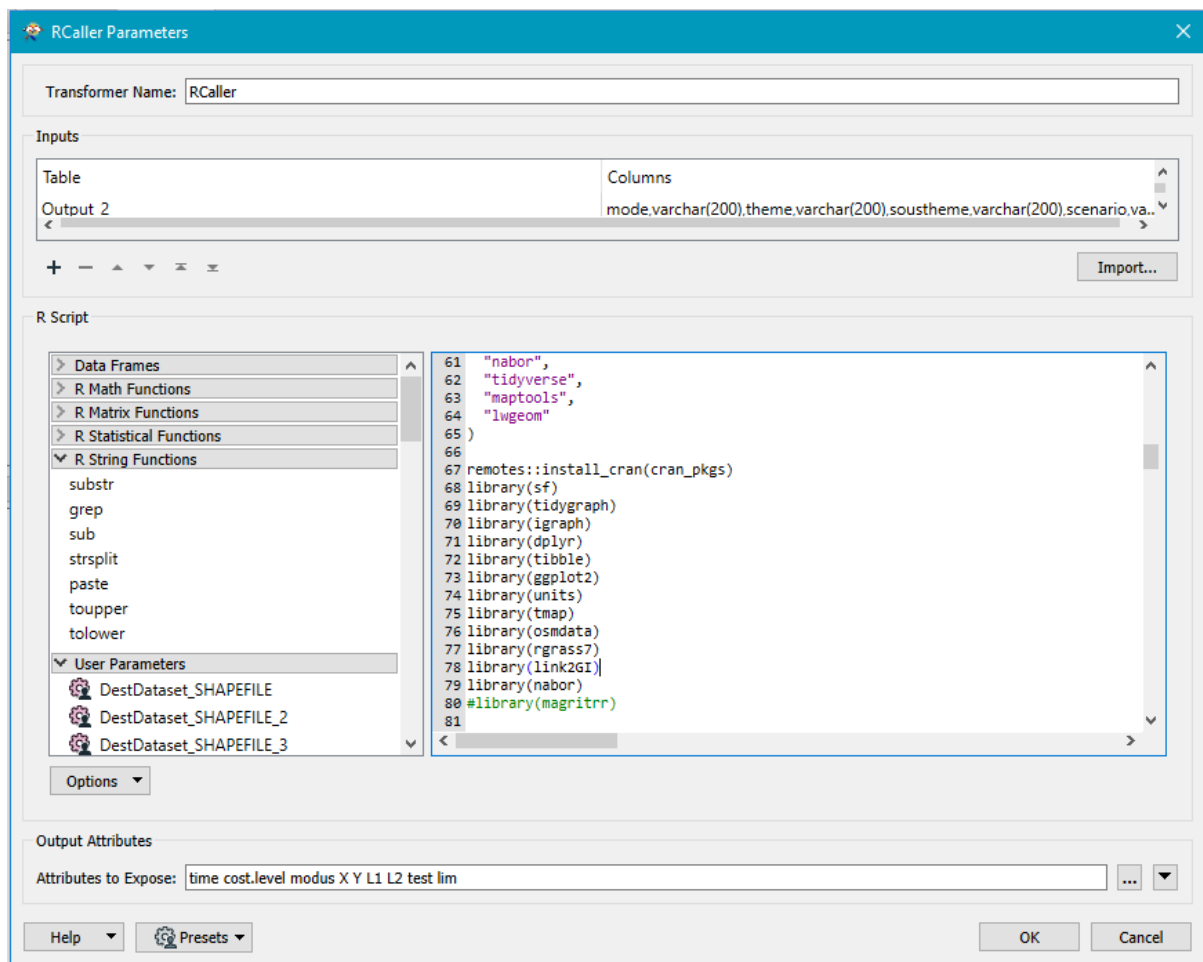


Figure 14: RCaller parameters which includes the entire Rscript.

Optional: It is possible to split large shapefiles of input variables in smaller segments and process them separately. Later the isochrone maps of the smaller segments can be rejoined. In Figure 15 an example is given for the variable 'STIB Bus data'.



Figure 15: Splitting the STIB Bus data in order to speed up processing.

Optional: An Rscript has been created that is able to calculate isochrones for travel by public transport.

The R script consists basically of following steps:

1. Closest bus stops near the facility

- a. Calculate the distance between the STIB bus stops and the facility (e.g. swimming pool)
 - b. Convert distance into minutes based on walking speed (4.3 km/h by foot)
2. Reachable bus stops
 - a. Select all bus stops accessible according to public transport routes
 - b. Convert to minutes according to the timetable
3. Isochrones
 - a. Calculate the isochrones around each accessible bus stop with the remaining time

Currently, the script is based on the travel times between STIB bus stops (afternoon trips between 14h and 19h). The input data for travel times and stops can be extended to metro and tram data.

An example:

```
stops <- read.table("stop_times.txt", header = T, sep = ",")
```

```
stib_bus <- st_read("J:/SET-SADL_User-DI0222/Projecten/Brussel-10-min/04 Model/Processing/1. DataPreparation/ALLData (tree)/Mobilité/Transport public/STIB_Bus arrêts/STIB_Bus.shp")
```

To use the Rscript for different variables, two elements need to be changed inside the Rscript. First of all, the path file of the input data needs to be adapted.

An example:

```
piscines <- st_read("J:/SET-SADL_User-DI0222/Projecten/Brussel-10-min/04 Model/Processing/1. DataPreparation/ALLData (tree)/Vivre ensemble & Bien-être/Cohésion & inclusion sociale/Formations & Services sociaux/Bureau d'accueil/BureauxAccueilPA_RBC.shp")
```

Secondly, the name of the column that contains the name of the facility (e.g. the name of the swimming pool) needs to be corrected:

An example:

```
*For 'Bureau d'accueil': Name <- row_interest$NOM.OFFICI
```

```
*For 'Piscine': Name <- row_interest$Nom
```

The main issue when running the script is the very long processing time due to the high complexity. One way to circumvent the issue, is to run the script for each facility within the variable separately. This means that the extend of the first loop need to be adjusted.

An example: When you want to run the script for the first swimming pool in the row, you will change the loop extend:

```
for(m in 1:nrow(piscines)) → for(m in 1:1)
```

or for the second swimming pool:

```
for(m in 1:nrow(piscines)) → for(m in 2:2)
```

The R script is also integrated in a FME Workspace (isochrones – transport public.fmw) (cf. Figure 16). It is constructed in the same way as the FME Workspace isochrones – final.fmw. A small step is added in order for the user to calculate the isochrones for each facility separately.

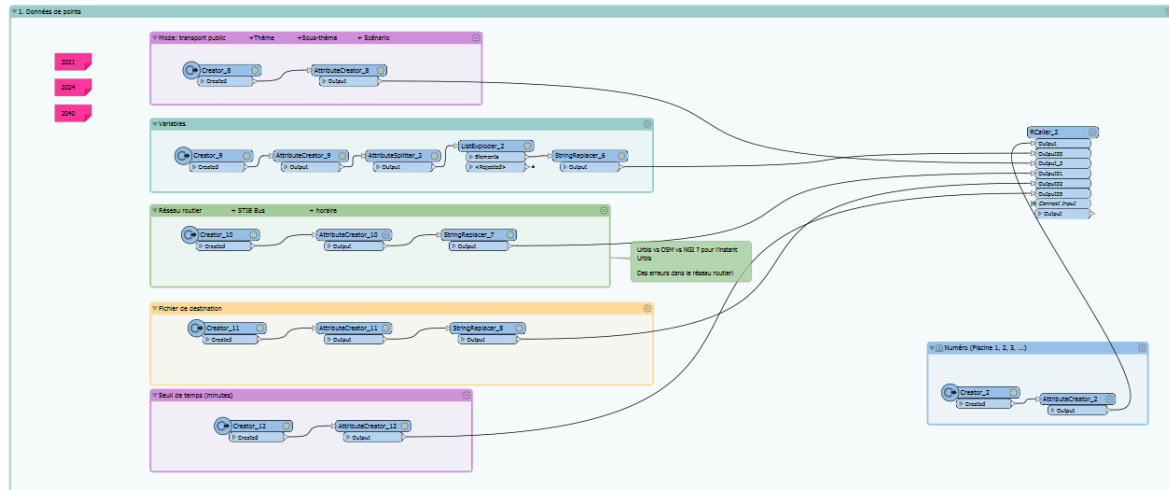



Figure 16: FME workspace with integrated R script to calculate the isochrones using public transport (STIB Bus).

Figure 17 shows the user parameters. Additional user parameters compared to the FME Workspace 'isochrones – final.fmw' need to be defined by the user. In particular, the user has to define the file path of the shapefile of the STIB Bus stops and the file path of the table containing the travel times between bus stops and the number of the facility. So, for each facility separately an isochrone map using public transport is created.

 Translation Parameter Values

User Parameters

variable: reau d'accueil/BureauxAccueilPA_RBC.shp

mode: TP

réseau_routier: inedMap\Urbis cleaned\Urbis_cleaned.shp

destination: Processing\3. Isochrones\Mobilité\Walking

minutes: 20

sous-thème: cohésion sociale

thème: vivre ensemble

scénario: 2021

STIB_Bus: public/STIB_Bus arrêts/STIB_Bus.shp


STOP_times: transport travel time/STIB/stop_times.txt

Numéro: 1

☒ Save As User Parameter Default Values

Options

Presets

 Run

Cancel

Figure 17: User parameters (file path(s) of variable(s), mode of transport (TP=Transport public), file path of the road network, destination folder for the created isochrones, number of minutes, sub-theme, theme, scenario, file path of the STIB bus stops, file path of the table with travel times and the number of the facility) for calculation the isochrones for point data using public transport.

After calculating all isochrones for each facility separately, the different isochrone maps can be rejoined into one shapefile. In Figure 18 an example is given for the variable 'Piscines'.

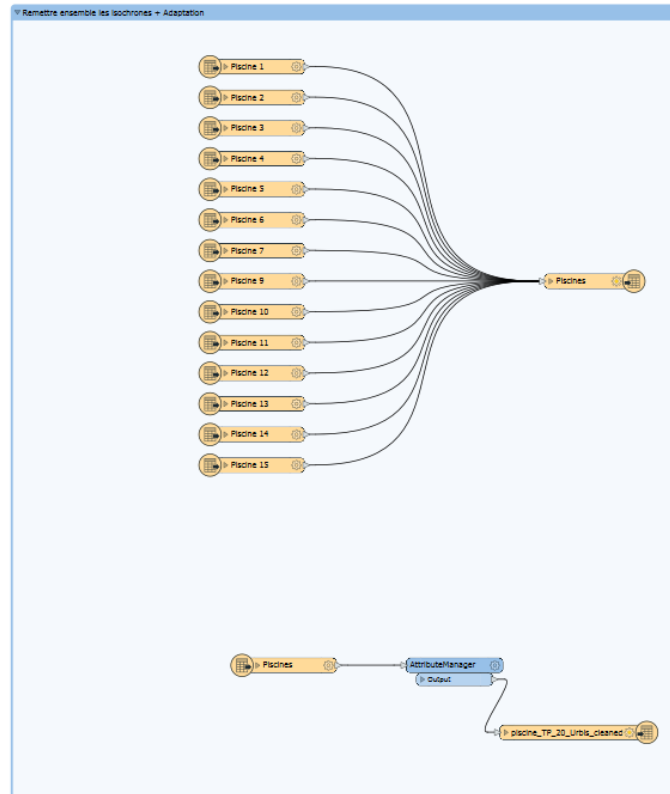


Figure 18: Rejoining the isochrones calculated for each facility (in this case swimming pool) separately. If needed, attributes such as mode, theme, subtheme and scenario can be added ('AttributeManager').

2.2 Polygon data

Some variables consist of polygon data. For these variables a buffer of equal distance is created around each polygon. The distance is calculated based on the velocity (by foot: 4.3 km/h and by bike: 13 km/h + 80 seconds needed for pre/post journey) and the amount of minutes provided by the user. So for example 10 minutes by foot means 716.67 m and 10 minutes by bike 1772.22 m. The workflow is presented in Figure 19.

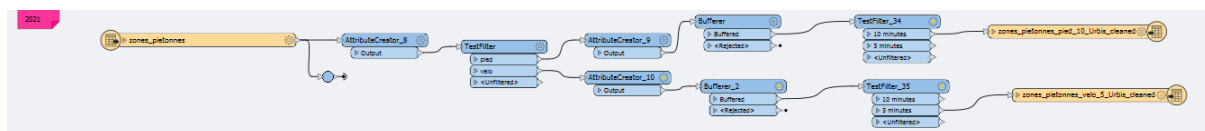


Figure 19: Creating a buffer around the variable 'zones pietonnes', which consist of polygon data.

The user parameters for this part of the FME workspace are the same as for the point data; except the road network is missing (cf. Figure 20). The source shapefile is the shapefile containing polygons. Make sure that all other parameters are also filled in correctly.

Figure 20: User parameters for creating buffers around polygon data (file path(s) of variable(s), mode of transport, destination folder for the created isochrones, number of minutes, sub-theme, theme and scenario).

For future scenario's, i.e. 2024 and 2040, the process needs to be repeated. The user can chose to either change the source file in the user parameters or to upload a new shapefile in the FME Workspace and connect it with the other transformers. (! Do also not forget to change the user parameter 'scenario' to the correct one). The template to add new shapefiles is already created in the FME workspace as presented in Figure 21.

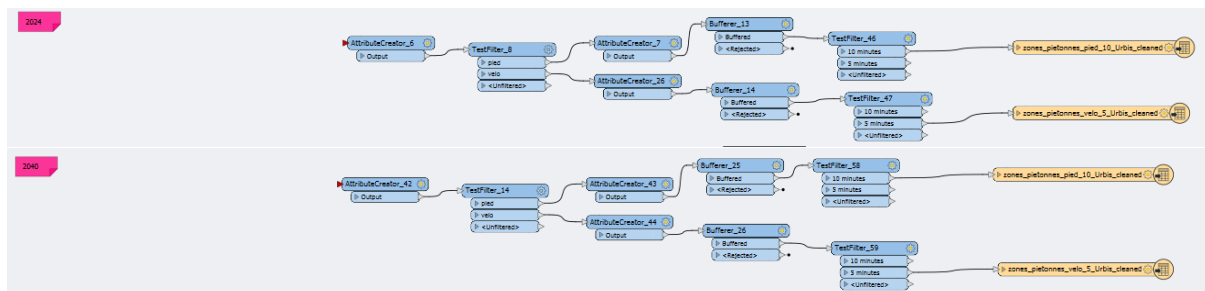


Figure 21: Create buffers around the input variables of future scenarios.

Optional: As mentioned in the final report, creating buffers is a quick and easy solution, but the results represent distances as the crow flies around the perimeter and are therefore less precise because the method does not take into account the road network. There exists the possibility of transforming the perimeters of the polygons into a series of points (cf. Figure 22). The isochrones can then be calculated for the set of points using the built-in R script (cf. § Point data).

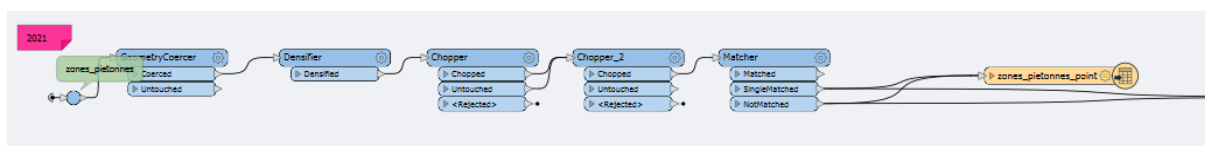


Figure 22: Split the perimeter of the polygons into a series of points. The following steps are executed in FME: 1. Convert polygon into line segments ('GeometryCoercer'), 2. Add vertices at fixed interval (30m) ('Densifier'), 3. Chop data into points by length and by vertex ('Chopper'), 4. Delete duplicate points (SingleMatched and NotMatched output from 'Matcher')

Splitting the perimeters of the polygons into a series of points results in a very large set of points. Consequently, it is better to divide the large set of points into smaller segments and process them separately. Isochrones are calculated using the workflow presented in § Point data. Afterwards, the separate isochrone maps of the smaller segments can be rejoined. In Figure 23 an example is given for the variable ‘zones piétonnes’.

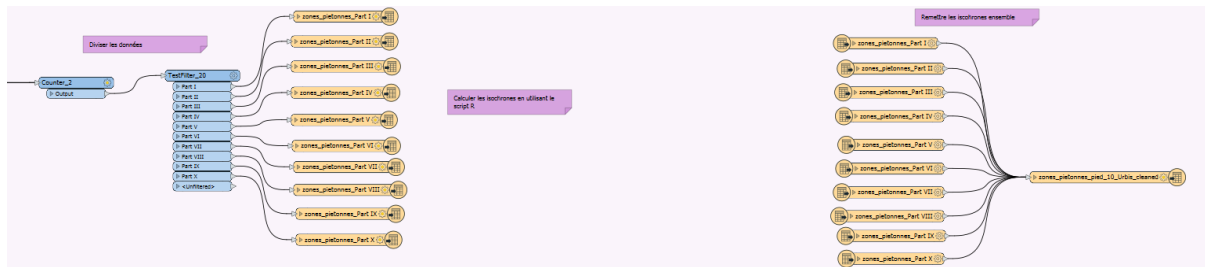


Figure 23: Dividing the set of points around the perimeters of the variable ‘zones piétonnes’ into smaller segments in order to speed up processing. Afterwards the isochrones maps are rejoined.

If in the future data with access points to public spaces or green spaces becomes available, more precise results can be obtained.

3. Heatmap

In order to measure the spatial-temporal accessibility for different (sub)themes, heatmaps were created. A heatmap is constructed by overlaying the isochrone maps of each variable within one (sub)theme. The resulting intersecting areas receive a score, calculated as the sum of each variable’s score multiplied by the variable’s weight. The score of a variable equals its weight if the variable is present in the area and equals zero if the variable is not present in the area. The weights are assigned by the user. The level of accessibility of an area is expressed in % and equals its score times 100. The calculations are automated using the software FME (HEATMAPS.fmw). The user can specify the mode of transport (by bike or by foot) and the (sub)theme(s) for which the model should run heatmaps. Also the file location of the supply variables and the destination folder of the resulting heatmaps can be set by the user.

3.1 Current situation

The FME Workspace consists of different steps. The isochrone maps created in a previous step (cf. § Isochrones) are used as input.

Figure 24 shows the first 5 steps in the procedure. First, each of the variables passes a filter (‘Tester’) such that only isochrones are considered that comply to the specified user parameters. These user parameters are mode of transport, theme, subtheme and scenario. Secondly, the individual isochrones of each variable are dissolved in order to create one connected area. Then the weights of each variables is uploaded from the excel table (Figure 25). The complete table with all the weights can be found in Appendix 3. The workflow of dissolving isochrones (‘Dissolver’) and uploading weights (‘Feature Joiner’) is comprised in the ‘Custom Transformer’ (Figure 26). Subsequently the variables within one subtheme are superimposed (‘Area on Area Overlayer’). Each new area that is created in this way, has a different presence of variables. For example, in area 1 variables A and B are present while in area 2 variables A and C are present. In the fifth step, the score of the variables is defined as 1 when the variable is present and as 0 if the variable is not present (‘Attribute Manager’). Based on the score and the weight of the individual variables, the score of the subtheme is calculated as following:

$$\text{score}(\text{subtheme}) = \text{score}(\text{var1}) * \text{weight}(\text{var1}) + \text{score}(\text{var2}) * \text{weight}(\text{var2}) + \dots \quad (\text{eq.1})$$

In some cases, a condition is set by the user (Figure 27). The score of the subtheme will then be calculated according to eq.1 if the condition holds. If the condition does not hold, the score of the subtheme is set equal to zero ('Attribute Manager').

The accessibility of the subtheme is calculated as following ('Attribute Manager'):

$$\text{accessibility}(\text{subtheme}) = \text{score}(\text{subtheme}) * 100 [\%] \quad (\text{eq.2})$$

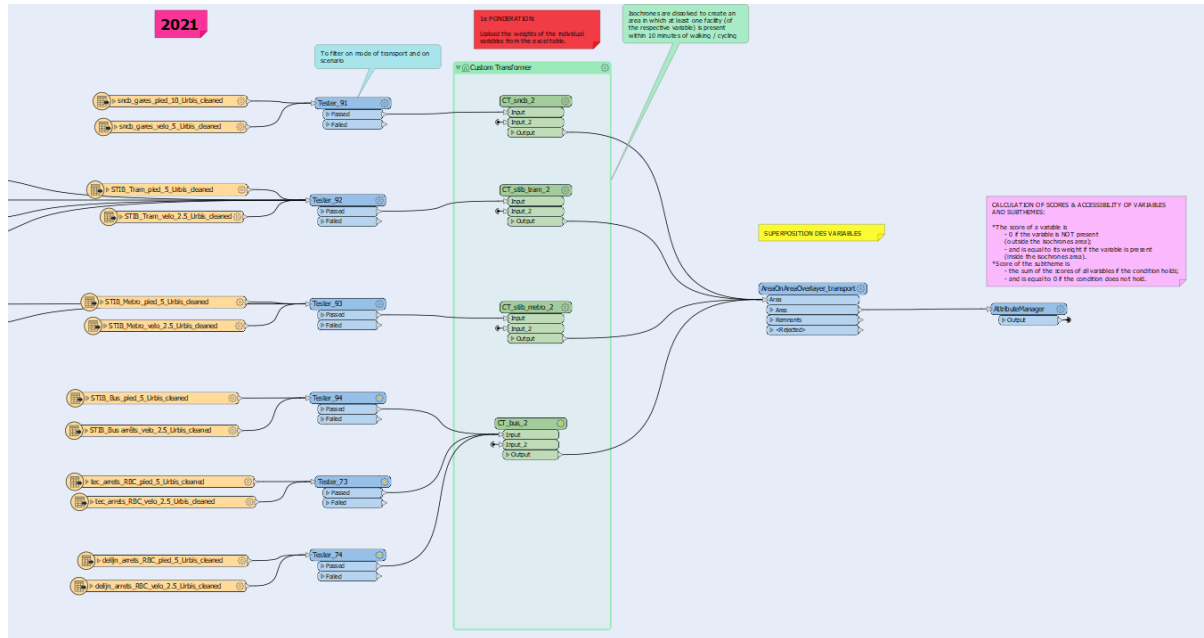


Figure 24: PART I of the FME workflow. Following steps are executed: 1. Load shapefile data of isochrones. 2. Filter based on the mode of transport and scenario, 3. Dissolve isochrones and upload the weights of the individual variables from the excel table, 4. The variables of one subtheme are superimposed, 5. Calculation of scores and accessibility of the subtheme.

Any changes in the weights of the variables, subthemes and themes need to be adjusted in the excel table. The new table then needs to be uploaded into the FME workbench (Figure 25). Appendix 3 explains how the weights need to be calculated for use in this model.

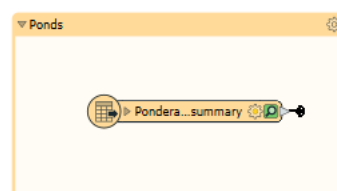


Figure 25: Loading the excel table with weights for each variable, subtheme and theme.

The custom transformer (Figure 26) first of all dissolves the incoming isochrones. Then a new attribute is created, named "DONNEE". The value of this attribute depends on the variable to which the custom transformer is attached. An example: for the variable 'Plaines de jeux', the value of "DONNEE" is 'PDJ', which corresponds to the abbreviation of the variable given in the excel table (Table 2 in Appendix 3). In this way, the variable's isochrones are matched with the variable's weight. The 'Bulk Attribute Renamer' and 'Attribute Remover' are added to the flow to change the name or to remove some attributes, such that the workflow remains clear and organized.

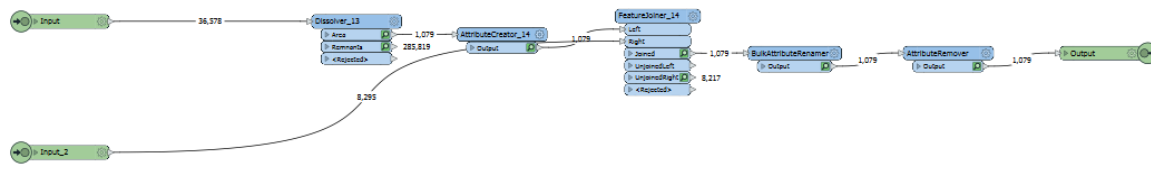


Figure 26: Custom transformer to dissolve isochrones and upload weights of the individual variables.

The condition is specified by the user and needs to be adapted inside the FME workflow if any changes are made to the condition. Figure 27 shows an example of a condition and how it is integrated into the FME workflow. Inside the 'Attribute Manager', the score of the subtheme is set to a conditional value: the score is calculated according to eq.1 if the condition holds and is zero if the condition does not hold.

Subtheme	Condition
Espace vert accessible au public (E.V.)	un E.V. macro à moins de 10 min OR un E.V. macro à moins de 20 min AND un E.V. micro à moins de 5 min OR un E.V. meso à moins de 10 min

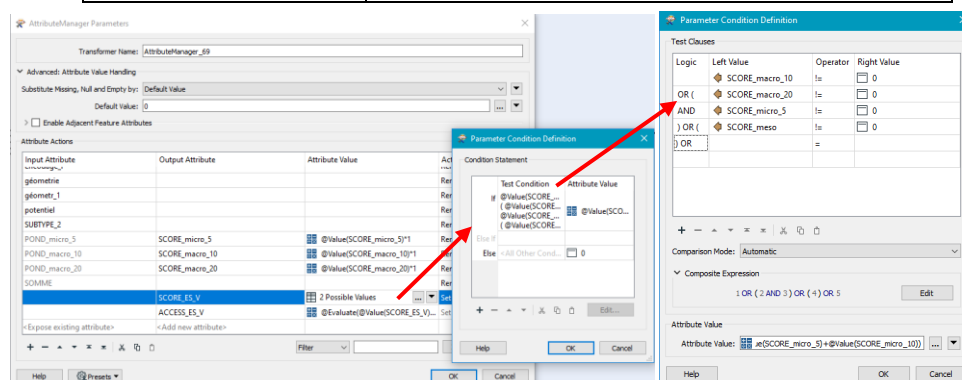


Figure 27: Setting a conditional value in FME 'Attribute Manager': example for the conditional value of the subtheme "Espace vert".

Figure 28 shows the second part of the FME workflow. First, the weights of the subthemes are uploaded from the excel sheet ('Feature Joiner'). Once again, a new attribute is created ("DONNEE") for which the value equals the abbreviation of the respective subtheme (corresponding to Table 2 in Appendix 3). When changing the abbreviations in the table, the FME workbench needs to be adapted at this point. The 'Attribute Manager' that follows is used to remove unnecessary columns, in order to keep the workflow clear and organized. Next, the subthemes within one theme are superimposed ('Area on Area Overlayer'). Finally, the score and accessibility of the theme is calculated ('Attribute Manager'). The theme score is calculated as following:

$$score(theme) = score(subtheme1) * weight(subtheme1) + score(subtheme2) * weight(subtheme2) + ... \quad (eq.3)$$

The accessibility of the theme is then calculated as following:

$$accessibility(theme) = score(theme) * 100 [\%] \quad (eq.4)$$

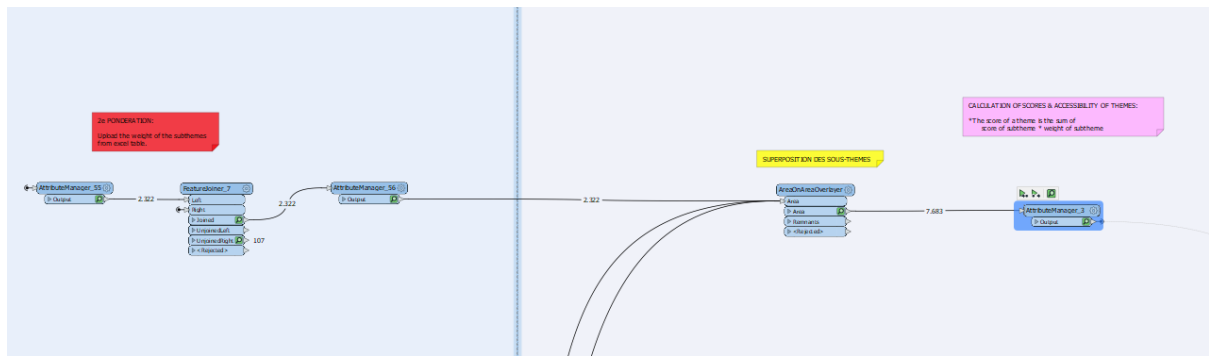


Figure 28: PART II of the FME workflow. Following steps are executed: 1. Upload weights to the subthemes from the excel table. 2. The subthemes of one theme are superimposed. 3. Calculation of the theme score and accessibility.

In the third part of the FME workflow, the maps are clipped to the area comprised by the Administration of the City of Brussels ('Clipper') (Figure 29). The 'Tester' filters out the Administration of the City of Brussels from the total municipality of Brussels.

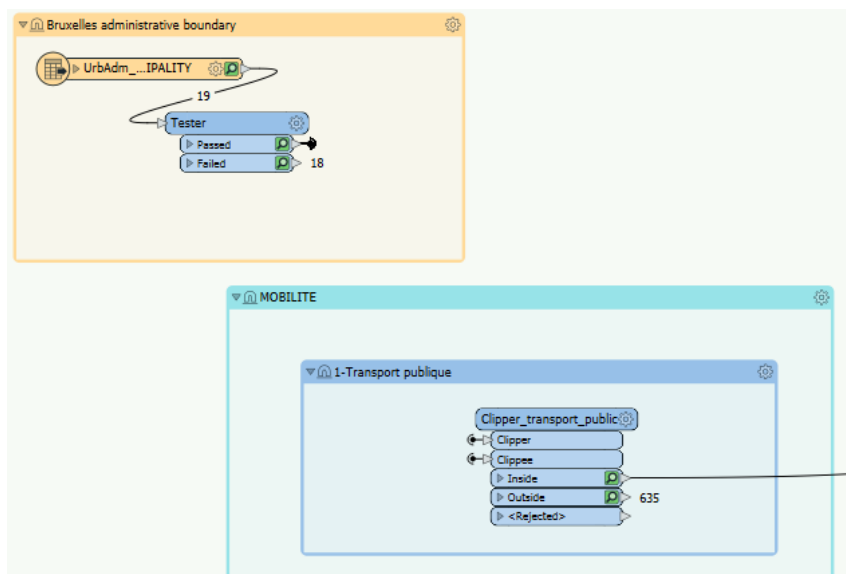


Figure 29: PART III of the FME workflow. Clip the created maps with Brussels administrative boundary.

To keep overview, heatmaps are created for each subtheme and each theme separate. At the end, the maps are filtered again to make the distinction between the modes of transport and the scenarios ('Test Filter') (Figure 30). The 'Attribute Manager' is used to rename the columns of the attribute table that belongs to the shapefile. In this way, the attribute table that is connected to the final heatmap will be easy to understand.

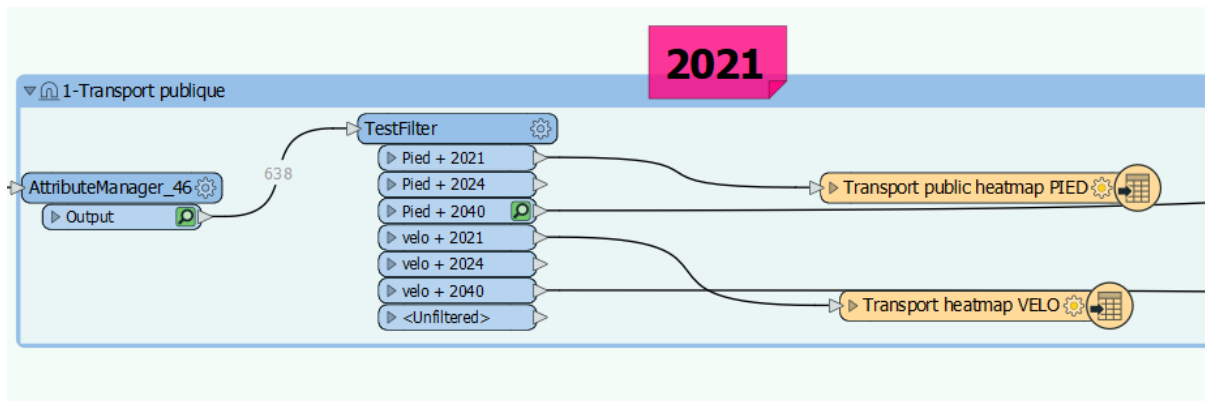


Figure 30: PART IV in the FME workflow. Creating heatmaps for each subtheme.

In the fifth part of the FME workflow, the accessibility for all themes together is calculated. First, the weights of the themes are uploaded from the excel sheet ('Feature Joiner') (Figure 31). Once again, a new attribute is created ("DONNEE") for which the value equals the abbreviation of the respective theme (corresponding to Table 2 in Appendix 3). The 'Attribute Manager' that follows is used to remove unnecessary columns, in order to keep the workflow clear and organized. Next, the themes are superimposed ('Area on Area Overlayer') and the total score and accessibility are calculated ('Attribute Manager') (Figure 31).

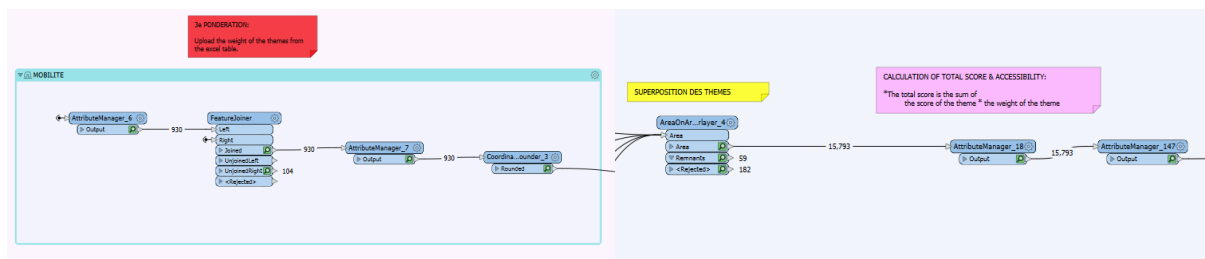


Figure 31: Part V of the FME workflow. The following steps are executed in FME: 1. Upload weights of the themes from the excel table. 2. Superposition of themes. 3. Calculation of total score and accessibility.

Finally, heatmaps are created for each mode of transport and for each scenario. A filter is placed to distinguish between mode of transport and scenario ('Test Filter'). From the attribute table in the final heatmap, it can be easily derived which variables are present and which are not present in a specific area. A "yes" or "no" value is assigned by means of the 'Null Attribute Mapper' and the 'String Replacer' in FME (Figure 32).

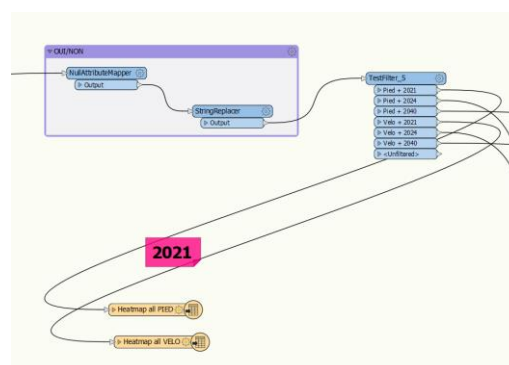


Figure 32: PART VI of the FME workflow. The following steps are executed in FME: 1. "Yes" (present) or "no" (not present) value is given to the individual variables. 2. A total heatmap is created for each mode of transport and for each scenario.

When running the model for the first time, the user has to make sure that to select the correct file path for each of the variables. Also, he/she has to select the destination folder to which the heatmaps should be written. Next, the user has to specify the mode of transport, the theme(s) or subtheme(s) and the scenario. To run for multiple modes of transport, for multiple themes or subthemes, the user parameters should be separated by a comma and a space (Figure 33).

Figure 33: Panel to specify the user parameters: examples on how to fill in.

3.2 Future situation

Some variables are expected to change in future, due to the construction of more facilities. For two future scenarios (2024 and 2040), isochrones were constructed for the future variable layers (cf. § Isochrones). These isochrones are loaded separately into the 'Heatmap.fmw' workbench. Figure 34 shows how the workbench is structured.

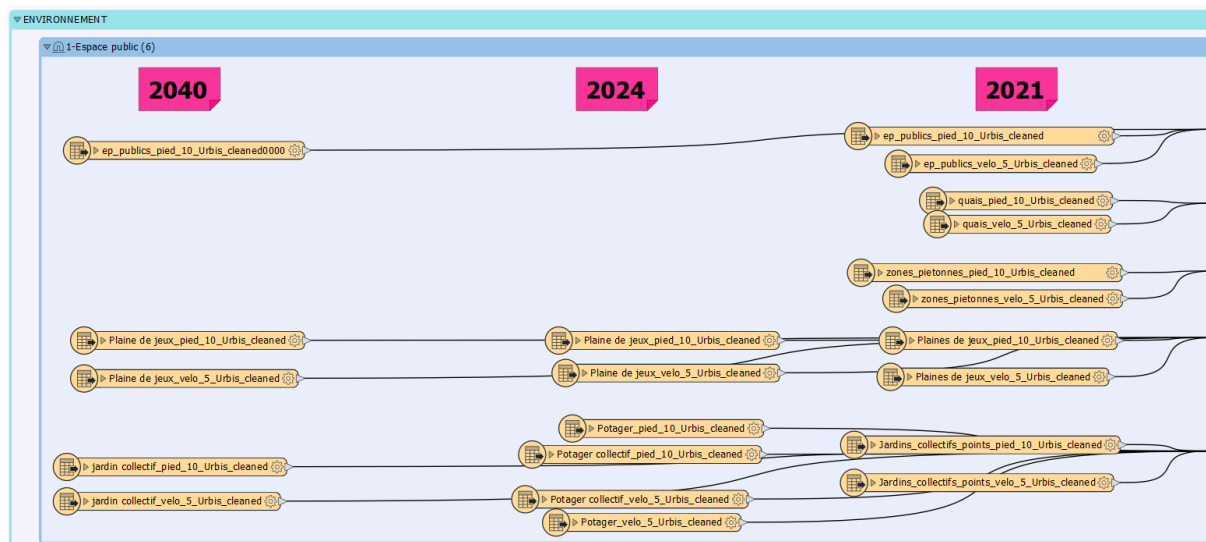


Figure 34: Isochrones of the variables for three scenarios are loaded separately into the FME workflow.

In the user parameters, the scenario can be specified (either 2021, 2024 or 2040). At the end of the workflow, the heatmap will be written under the corresponding name in the destination folder specified by the user.

Appendix

Appendix 1: Technical scheme of the variables used in the 10-minute city model of Brussels

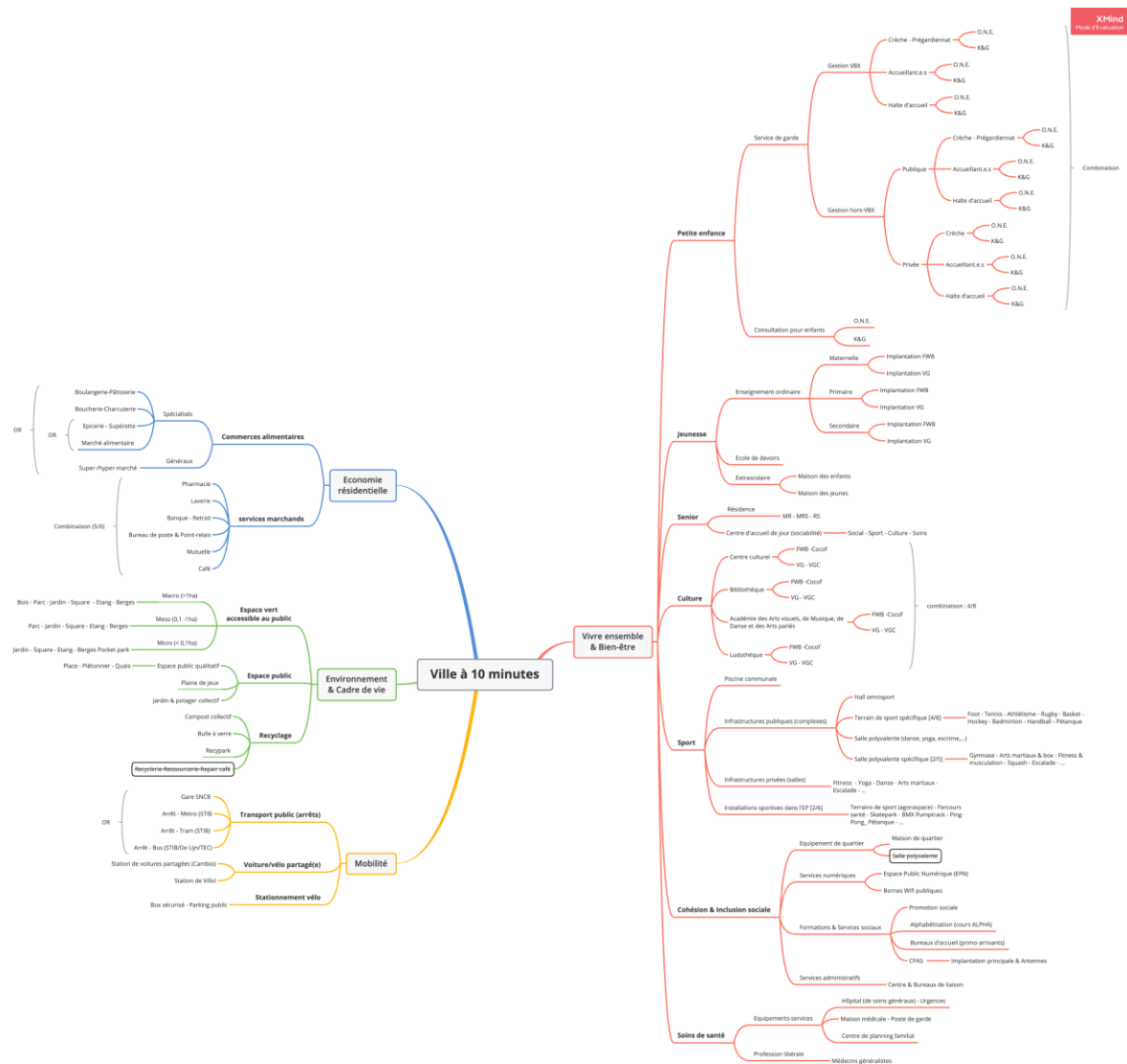


Figure 279: Scheme of the variables, and their respective subthemes and themes.

Appendix 2: R script

1	#####
2	##### Calculation of isochrones#####
3	#####
4	
5	#cpprouting method to calculate isochrones
6	
7	#More information:
8	# https://cran.r-project.org/web/packages/cppRouting/cppRouting.pdf
9	# https://github.com/vlarmet/cppRouting/blob/master/README.md#compute-isochrones
10	# https://r-spatial.org/r/2019/09/26/spatial-networks.html
11	
12	#set the number of digits
13	options(digits = 15)
14	
15	#installation of packages
16	if(!"remotes" %in% installed.packages()) {
17	install.packages("remotes")
18	}
19	
20	cran_pkgs = c(
21	"cppRouting",
22	"dplyr",
23	"sf",
24	"ggplot2",
25	"concaveman",
26	"ggmap",
27	"microbenchmark",
28	"reshape2",
29	"sp",
30	"rgdal",
31	"tidygraph",
32	"igraph",
33	"osmdata",
34	"tibble",
35	"units",
36	"tmap",
37	"rgrass7",
38	"link2GI",
39	"nabor",
40	"tidyverse",
41	"maptools",
42	"lwgeom",
43	"data.table",
44	"magrittr"

45)
46	
47	library(cppRouting)
48	library(dplyr)
49	library(sf)
50	library(ggplot2)
51	library(concaveman)
52	library(ggmap)
53	library(tmap)
54	library(microbenchmark)
55	library(reshape2)
56	library(sp)
57	library(rgdal)
58	library(tidygraph)
59	library(igraph)
60	library(tibble)
61	library(units)
62	library(tmap)
63	library(osmdata)
64	library(rgrass7)
65	library(link2GI)
66	library(nabor)
67	library(data.table)
68	library(magrittr)
69	
70	#preparation of the data based on user parameters
71	
72	selection1 <- Output_2\$mode
73	selection2 <- Output00\$X_list
74	selection3 <- Output01\$réseau_routier
75	selection4 <- Output02\$destination
76	selection5 <- Output03\$minutes
77	selection6 <- Output_2\$theme
78	selection7 <- Output_2\$soustheme
79	selection8 <- Output_2\$scenario
80	
81	#read shapefile of urbis road network
82	#based on file path provided as user parameter for 'reseau_routier'
83	urbis<-read_sf(file.path(selection3[1]))
84	
85	
86	#Loop for all input variables. For each input variable an isochrone map is created.
87	for (i in 1:length(selection2)) {
88	
89	#read shapefile of variable based on file path provided as user parameter 'variable'
90	a <- sub(".*?/(.*?).shp.*", "\\1", selection2[i])

91	assign(a, read_sf(file.path(selection2[i])))
92	location <- get(sub(".*?/(.*?).shp.*", "\\1", selection2[i]))
93	#possibility to set CRS but sometimes this can give errors
94	#st_crs(urbis) = CRS("+init=epsg:31370")
95	#st_crs(location) = CRS("+init=epsg:31370")
96	
97	#project points onto the road network using function snappointsToLines
98	library("maptools") #load package
99	location_sp <- as(location, 'Spatial') #convert to spatial objects
100	location_sp <- SpatialPoints(location_sp) #convert to spatial points
101	urbis_sp <- as(urbis, 'Spatial') #convert to spatial objects
102	test <- snapPointsToLines(location_sp,urbis_sp) #project points (=variables) to lines (=road network)
103	test2 <- as.data.frame(test) #convert back to dataframe
104	
105	#split road network based on the projected points.
106	#By doing this extra nodes will be created in the road network.
107	#this is a very time consuming step
108	library("lwgeom")
109	test3 <- st_as_sf(test2,coords = c("X", "Y"))
110	on_line_all = st_cast(test3, "POINT")
111	buf_all <- st_combine(st_buffer(on_line_all,0.000000001))
112	parts_all = st_collection_extract(lwgeom::st_split(urbis\$geometry, buf_all),"LINESTRING")
113	
114	parts_all = st_as_sf(
115	data.frame(
116	id = 1:length(parts_all),
117	geometry = parts_all
118)
119)
120	
121	#calculate length
122	parts_all\$length <- st_length(parts_all)
123	
124	#change the urbis dataset with the newly created road network that contains additional edges
125	urbis <- parts_all
126	
127	#give each edge of the road network a unique index
128	edges <- urbis %>%
129	mutate(edgeID = c(1:n()))
130	
131	#create nodes at the start and end point of each edge
132	nodes <- edges %>%
133	st_coordinates() %>%
134	as_tibble() %>%
135	rename(edgeID = L1) %>%

136	group_by(edgeID) %>%
137	slice(c(1, n())) %>%
138	ungroup() %>%
139	mutate(start_end = rep(c('start', 'end'), times = n()/2))
140	
141	#give each node a unique index
142	library(tidyverse)
143	nodes <- nodes %>%
144	mutate(xy = paste(.\$X, .\$Y)) %>%
145	group_by(X, Y) %>%
146	mutate(nodeID =cur_group_id()) %>%
147	#mutate(nodeID = group_indices(., factor(xy, levels = unique(xy)))) %>%
148	select(-xy)
149	
150	#combine the node indices with the edges
151	source_nodes <- nodes %>%
152	filter(start_end == 'start') %>%
153	pull(nodeID)
154	
155	target_nodes <- nodes %>%
156	filter(start_end == 'end') %>%
157	pull(nodeID)
158	
159	edges = edges %>%
160	mutate(from = source_nodes, to = target_nodes)
161	
162	#remove duplicate nodes
163	nodes <- nodes %>%
164	distinct(nodeID, .keep_all = TRUE) %>%
165	select(-c(edgeID, start_end)) %>%
166	st_as_sf(coords = c('X', 'Y')) %>%
167	st_set_crs(st_crs(edges))
168	
169	#change column names
170	roads <- edges[c('from','to','length')]
171	roads <- data.frame(roads)
172	roads <- roads[c('from','to','length')]
173	colnames(roads) <- c('from','to','weight')
174	
175	coordinates <- nodes %>%
176	st_coordinates() %>%
177	as_tibble()
178	nodeID <- nodes[c('nodeID')]
179	coord <- cbind(coordinates,nodeID)
180	coord <- data.frame(coord)
181	coord <- coord[c('nodeID','X','Y')]

182	colnames(coord)<- c('ID','X','Y')
183	
184	graph<-makegraph(roads,directed = F,coords = coord)
185	
186	# set the maximal distance based on the time threshold and the mode of transport
187	# provided by the user parameters ('minute' and 'mode')
188	temps <- as.data.frame(selection5[1])
189	temps[1] <- as.numeric(temps[1])
190	#distance is different depending on the mode of transport (walking or cycling)
191	#velocity for walking: 4.3 km/h
192	#velocity for cycling: 13 km/h + 80 seconds pre/post journey
193	if(selection1[1]=="pied") {
194	lim <- (4.3*1000/60*temps[1]) #4.3 km/h
195	} else if (selection1[1]=="velo") {
196	lim <- (13*1000/3600*((temps[1]*60)-80)) #13 km/h + 80 seconds pre/post journey
197	}
198	
199	lim <- as.numeric(lim)
200	lim <- as.data.frame(lim)
201	
202	#select the node ID's that match with the location of the variable
203	#(node from a location of a variable = point projected to the road network)
204	y <- coord[order(coord\$ID),]
205	row.names(y) <- y\$ID
206	node <- data.frame()
207	node_tot <- data.frame()
208	for (j in 1:dim(test2)[1]) {
209	node_loop <- node_tot
210	node <- subset(y\$ID, y\$Y %like% test2[j,\$Y])
211	node_tot <- rbind(node_loop, node)
212	}
213	node_isochrones <- node_tot[,1]
214	node_isochrones <- data.frame(node_isochrones)
215	colnames(node_isochrones) <- "nodeID"
216	
217	#Compute isochrones for X minutes walking at 4,3 km/h or cycling at 13 km/h
218	#distance/time limit was previously calculated (line 185-196)
219	#time consuming step
220	iso<-get_isochrone(graph,from = node_isochrones\$nodeID,lim = c(lim))
221	#Convert nodes to concave polygons with concaveman package
222	coord\$ID <- as.character(coord\$ID)
223	poly<-lapply(iso,function(t){
224	t<-data.frame(noeuds=t,stringsAsFactors = F)
225	t<-left_join(t,coord,by=c("noeuds"="ID"))
226	return(concaveman(summarise(st_as_sf(t,coords=c("X","Y"),crs=31370))))
227	})

228	
229	#add attributes to the isochrone polygons
230	poly <- st_zm(poly)
231	poly<-do.call(rbind,poly)
232	poly\$time<-as.factor(names(iso))
233	poly\$minutes <- selection5[1]
234	poly\$mode <- selection1[1]
235	poly\$theme <- selection6[1]
236	poly\$soustheme <- selection7[1]
237	poly\$scenario <- selection8[1]
238	
239	#delete geometries which are not correct due to errors in the road network
240	invalid <- st_coordinates(poly\$polygons)
241	invalid <- as.data.frame(invalid)
242	invalid\$L2 <- as.character(invalid\$L2)
243	invalid <- count(invalid, invalid\$L2)
244	invalid2 <- invalid[invalid\$n<4,]
245	poly\$ID <- row.names(poly)
246	poly <- poly[!(poly\$ID %in% invalid2\$`invalid\$L2`),]
247	poly <- poly %>% select(-ID)
248	
249	#convert to shapefile
250	shapefile<- as(st_geometry(poly), "Spatial")
251	# Extract polygon ID's
252	pid <- sapply(slot(shapefile, "polygons"), function(x) slot(x, "ID"))
253	# Create dataframe with correct rownames
254	p.df <- data.frame(ID=1:length(shapefile), row.names = pid)
255	# Try coercion again and check class
256	variable <- SpatialPolygonsDataFrame(shapefile, p.df)
257	# add user parameters as attributes - will later be used when creating heatmaps
258	variable\$mode <- selection1[1]
259	variable\$theme <- selection6[1]
260	variable\$soustheme <- selection7[1]
261	variable\$minutes <- selection5[1]
262	variable\$scenario <- selection8[1]
263	a <- sub(".*?/(.*?).shp.*", "\\1", selection2[i])
264	b <- sub(".*?/(.*?).shp.*", "\\1", selection3[1])
265	#write to shapefile
266	#destination folder specified as user parameter
267	writeOGR(variable, layer = paste0(a,"_",selection1[1],"_",selection5[1],"_", b), file.path(selection4[1]), driver="ESRI Shapefile", overwrite_layer = T)
268	}
269	
270	time <- selection5[1]
271	modus <- selection1[1]
272	fmeOutput<-data.frame(time, modus, lim)

273	#####
274	##### Calculation of isochrones #####
275	#####
276	
277	#cpprouting method to calculate isochrones
278	
279	#More information:
280	# https://cran.r-project.org/web/packages/cppRouting/cppRouting.pdf
281	# https://github.com/vlarmet/cppRouting/blob/master/README.md#compute-isochrones
282	# https://r-spatial.org/r/2019/09/26/spatial-networks.html
283	
284	#set the number of digits
285	options(digits = 15)
286	
287	#installation of packages
288	if(!"remotes" %in% installed.packages()) {
289	install.packages("remotes")
290	}
291	
292	cran_pkgs = c(
293	"cppRouting",
294	"dplyr",

Appendix 3: Calculation of weights

The initial weights of the variables, subthemes and themes are defined by means of their frequency of usage and essentiality (see final report chapter X) (Table 1).

Table 1: Abbreviation (“donnee”) and weight (“pond”) of the variables, subthemes and themes.

Sous-Thème	Pondération (fréquence * essentiel)
Commerces alimentaires	$(7*7) * 1 = 1$
Commerces non-alimentaires	$(3/7) * 1 = 0,5$
Espace vert accessible au public	$(7/7) * 1 = 1$
Espace public	$(7/7) * 0,5 = 0,5$
Recyclage	$(2/7) * 1 = 0,3$
Arrêts TP	$(7/7) * 1 = 1$
Voiture/vélo partagé(e)	<ul style="list-style-type: none"> - Cambio : $(1/7) * 0,5 = 0,1$ - Villo : $(5/7) * 0,5 = 0,4$ <p>TOTAL = 0,25</p>
Stationnement vélo	$(7/7) * 0,5 = 0,5$
Services de garde	$(5/7) * 1 = 0,8$
Consultations Enfance	$(0,5/7) * 1 = 0,1$
Enseignement	$(5/7) * 1 = 0,8$
Ecoles des devoirs	$(5/7) * 0,5 = 0,4$
Extrascolaires	$(5/7) * 0,5 = 0,4$
Résidence Seniors	$(7/7) * 1 = 0,5$
Centre de jours Senior	$(5/7) * 0,5 = 0,4$
Equipements culturels	$(3/7) * 1 = 0,5$
Equipements sportifs	$(3/7) * 1 = 0,5$
Services numériques	$(5/7) * 0,5 = 0,4$
Formations & Services sociaux	$(5/7) * 0,5 = 0,4$
Services administratifs	$(0,25/7) * 1 = 0,1$
Soins de santé	$(0,5/7) * 10 = 0,7$

In order to use the FME model, these weights are normalized ($weight_{norm}$). The weights of the individual variables are normalized to the subtheme they belong to. In the case no weight based on frequency/essentiality is given, the variables within the subtheme are considered equally important and the normalized weight will be calculated as following:

$$weight_{norm}var(1) = \frac{1}{\text{number of variables within the subtheme}}. \quad (eq.5)$$

In case an initial weight is assigned based on frequency/essentiality, the normalized weight will be calculated as following:

$$weight_{norm}var(1) = \frac{weight(var1) + \dots + weight(varN)}{\text{sum of } weight(var1 - varN)}, \quad (eq.6)$$

with var(1) to var(N) belonging to the same subtheme.

An example: The subtheme “Transport public” consists of 4 variables (‘SNCB gares’, ‘Arrêts Métro’, ‘Arrêts Tram’ et ‘Arrêts Bus’). These variables are equally important within the subtheme and so their normalized weight is equal to 0.25 (Table 2) such that the sum equals 1. The subtheme “Voiture/vélo partagé(e)” consists of 2 variables that are not equally important (‘Cambio’ and ‘Villo’). ‘Cambio’ has initial weight of 0.1 and ‘Villo’ has initial weight of 0.4. The sum of these weights is 0.5, and thus the normalized weight for ‘Cambio’ is 0.2 and for ‘Villo’ is 0.8 such that the sum equals 1.

The weights of the subthemes are normalized to the theme they belong to. The same reasoning holds:

$$weight_{norm}subtheme(1) = \frac{weight(subtheme1) + \dots + weight(subthemeN)}{\text{sum of } weight(subtheme1 - subthemeN)}, \quad (eq.7)$$

An example: The theme “mobility” consists of 3 subthemes (“Transport public”, “Voiture/vélo partagé(e)” and “Stationnement vélo”) which received a weight of 1, 0.5 and 0.25 respectively (table X). The sum of these weights is 1.75, and so the normalized weight for the subthemes is 0.571, 0.286 and 0.143 respectively such that the sum equals 1 (table X).

Each of the 4 themes (“Mobilité”, “Economie”, “Environnement” and “Vivre ensemble & bien-être”) are equally important and thus receive a normalized weight of 0.25 such that the sum is 1.

Table 2: Abbreviation (“donnee”) and normalized weight (“pond”) of the variables, subthemes and themes.

Variable/subtheme/theme	DONNEE	POND
Environnement	ENV	0.25
Espace public	ES_P	0.277777778
Jardin & potager collectif	POTAGERS	0.333333333
Espace public qualitatif	PIET	0.333333333
Plaine de jeux	PDJ	0.333333333
Recyclage	RECY	0.166666667
Compost collectif	COMPOSTS	0.25
Recypark	RECYPARK	0.25
Recyclerie-Ressourcerie-Repair café	RRR	0.25
Bulle à verre	BULLE	0.25
Espace vert accessible au public	ES_V	0.555555556
Macro (>1ha) à 10 minutes	MACRO_10	0.2
Macro (>1ha) à 20 minutes	MACRO_20	0.2
Meso (0,1–1ha)	MESO	0.2
Micro (<0,1ha) à 10 minutes	MICRO_10	0.2
Micro (<0,1ha) à 5 minutes	MICRO_5	0.2
Mobilité	MOB	0.25
Transport public (arrêts)	TRANS	0.571428571
Gare SNCB	SNCB_GARE	0.25
Arrêts metro (STIB)	STIB_M	0.25
Arrêts tram (STIB)	STIB_T	0.25
Arrêts bus (STIB/De Lijn/TEC)	BUS	0.25
Voiture/vélo partagé(e)	PART	0.142857143

Station de voitures partagées (Cambio)	CAMBIO	0.2
Station de Villo!	VILLOS	0.8
Stationnement vélo	VELO	0.285714286
Parking public	VELOS_P	0.5
Box sécurisé	BOX	0.5
Vivre ensemble et bien-être	V_E	0.25
Petite enfance	PE	0.147540984
Service de garde	SERVICE	0.888888889
Crèche – préguardiennat publique – O.N.E	CRECHE_PU_FR	0.083333333
Crèche – préguardiennat publique – K&G	CRECHE_PU_NL	0.083333333
Crèche privée - ONE	CRECHE_PR_FR	0.083333333
Crèche privée – K&G	CRECHE_PR_NL	0.083333333
Accueillant.e.s publique – O.N.E.	ACC_PU_FR	0.083333333
Accueillant.e.s publique – K&G	ACC_PU_NL	0.083333333
Accueillant.e.s privée – O.N.E.	ACC_PR_FR	0.083333333
Accueillant.e.s privée – K&G	ACC_PR_NL	0.083333333
Halte d'accueil publique – O.N.E.	HACC_PU_FR	0.083333333
Halte d'accueil privée – O.N.E.	HACC_PR_FR	0.083333333
Halte d'accueil publique – K&G	HACC_PU_NL	0.083333333
Halte d'accueil privée – K&G	HACC_PR_NL	0.083333333
Consultation pour enfants	CONSUL	0.111111111
O.N.E	ONE	0.5
K&G	K&G	0.5
Jeunesse	JEU	0.262295082
Enseignement ordinaire	ECOLE	0.5
Ecole maternelle francophone	MAT_FR	0.166666667
Ecole maternelle néerlandophone	MAT_NL	0.166666667
Ecole primaire francophone	PRIM_FR	0.166666667
Ecole primaire néerlandophone	PRIM_NL	0.166666667
Ecole secondaire francophone	SEC_FR	0.166666667
Ecole secondaire néerlandophone	SEC_NL	0.166666667
Extrascolaire	EXTRA	0.25
Maison des enfants	M_JEUNES	0.5
Maison des jeunes	M_ENFANTS	0.5
Ecole de devoirs	DEVOIRS	0.25
Senior	SENIOR	0.147540984
Centre d'accueil de jour (sociabilité)	ES_SENIOR	0.555555556
Résidence	RES	0.444444444
Culture	CUL	0.081967213
Académie des Arts, de Musique et de Danse francophone	ACA_FR	0.125
Académie des Arts, de Musique et de Danse néerlandophone	ACA_NL	0.125
Bibliothèque francophone	BIBLIO_FR	0.125
Bibliothèque néerlandophone	BIBLIO_NL	0.125
Centre culturel francophone	CC_FR	0.125
Centre culturel néerlandophone	CC_NL	0.125
Ludothèque francophone	LUDO_FR	0.125

Ludothèque néerlandophone	LUDO_NL	0.125
Sport	SPORT	0.081967213
Piscine communale	PISCINE	0.25
Infrastructures publiques (complexes)	SPORT_PU	0.25
Hall omnisport	HALL	0.25
Terrain de sport spécifique	TERRAIN	0.25
Salle polyvalente	POLY	0.25
Salle polyvalente spécifique	POLY_SPEC	0.25
Infrastructures privées (salles)	SPORT_PR	0.25
Installations sportives dans l'espace public	ES_SPORT	0.25
Cohésion & inclusion sociale	CIS	0.163934426
Espace public numérique	EPN	0,2
Bornes wifi publiques	WIFI	0,2
Promotion sociale	PROMO	0,1
Maison de quartier	M_QUARTIER	0,05
Salle polyvalente	SAL_POL	0,05
CPAS	CPAS	0,1
Bureaux d'accueil	BA	0,1
Alphabétisation	ALPHA	0,1
Centre & Bureaux de liaison	SA	0,1
Soins de santé	SANTE	0.114754098
Maison médicale – poste de garde	MS	0.25
Hôpital - urgences	URGENCE	0.25
Centre de planning familial	PLANFAM	0.25
Médecins généralistes	MED	0.25
Economie résidentielle	ECO	0.25
Commerces alimentaires	COM_ALI	0.666666667
Boulangerie-pâtisserie	BOUL	0.2
Boucherie-Charcuterie	BOUCH	0.2
Epicerie-Supérette	EPI	0.2
Marché alimentaire	MARCHE	0.2
Super-/hyper-marché	SUPER	0.2
Commerces non alimentaires	COM_NON_ALI	0.333333333
Bureau de poste & point relais	P_RELAIS	0.166666667
Pharmacie	PHARMA	0.166666667
Laverie	LAVERIE	0.166666667
Banque - retrait	BANQUE	0.166666667
Mutuelle	MUTUELLE	0.166666667
Café	CAFE	0.166666667