Decision-aid methodologies in transportation – Transport modelling module – Lab 4

1 Part 1: Mode choice

The objective of this lab is to understand the implementation of the logit model. There are three main files to go through this lab (in the **00_mode_choice** folder of the lab):

1. 00_mode_choice_MNL.ipynb: Code for the estimation of a logit model
2. 01_mode_choice_MNL_forecasting.ipynb: We use the results of the model that we estimated in 00_mode_choice_MNL.ipynb to generate an OD matrix with the modal split shares by mode.
3. 02_mode_choice_MNL_generalised_cost.ipynb: We can use this script to extract generalised cost between every OD pair.

Acknowledgement: The data that we are using for model estimation are part of the Mobility behaviour in Switzerland» (MOBIS) project and were kindly provided by the Institute for Transport Planning and Systems, ETHZ to be used in our course. For further details, you can have a look at: MOBIS – Institute for Transport Planning and Systems | ETH Zurich

1.1 Estimating the logit model

To estimate a logit model, we should use the 00_mode_choice_MNL file. The data that we are using for model estimation are part of the

Be careful! We need to install **Biogeme** before running this script. Biogeme is a python package designed for the estimation of discrete choice models. In theory, the installation is straightforward and the same as any python package:

1. Run in Jupyter notebook the environment you are planning to use Biogeme. It is always better if you create a new environment to install Biogeme (to avoid package incompatibilities), but it can also work on an existing environment
2. On Jupyter simply type: pip install biogeme, and wait for the installation to complete. It may take a few minutes.

Alright, now that Biogeme is successfully (hopefully) installed, let's quickly scan the code to see its main functionalities (check the 00_mode_choice_MNL file to see a more detailed description of the script).

- In the very first step, we load a few biogeme related packages that we need for the model estimation
- Then we **read the data**. This is basically information about the socio-demographic characteristics, trip purpose, the attributes of the alternatives (e.g., travel time, travel cost etc.), and the actual choice of an individual (e.g. the person chose car for a specific trip)
- We convert the variables that we are planning to use into a variable object that Biogeme can use. This process works as:

-
    - variable_name = Variable('variable_name ')

  where
    - variable_name is the name of the variable object. We can use any name but typically we use the same variable name as in our data
    - Variable(): is a Biogeme function to convert a column of in our data to a variable object
    - 'variable_name ': This is the name of the variable as it appears in our csv file (without the quotes). We need to use the quotes when defining the Biogeme variable object.

- Define some new variables: This step is optional but we can further process our variable objects to generate some new variables; very useful for categorical variables.
- Parameters to be estimated: These are our "betas", the values that show our sensitivity to the attributes of the alternatives and how much they influence our choices. The Beta expression has several inputs to define. Let's see their interpretation e.g. for B_TIME = Beta('B_TIME', 0, None, None, 0):
    - 'B_TIME': this is the name of the parameter that will be printed in our output file
    - The first zero (0) value denotes the starting value that we give to the parameter. Typically, we initiate from 0 but sometimes we have to use a non-zero value e.g., imagine that we estimate the standard deviation of a normal distribution. By definition, standard deviation cannot be zero or even negative.
    - None, None: these denote the lower and upper boundaries of the parameter in case we do constrained optimisation. E.g., sometimes a parameter cannot be negative so we do ***0, None*** or it must be between 0 and 1, se we replace the two None values with ***0, 1***.
    - The final 0 value indicates whether we want to estimate the parameter. If we change the value to *1*, then the parameter will keep the starting value during the optimisation process. This is a useful feature when we have very complex models with a large set of parameters and we want to estimate them gradually.
- Definition of the utility functions: This is basically our model. In this case it is a linear combination of the parameters and attributes. We need one utility function per alternative. To keep things simple, we only consider car and public transport (pt) as alternatives in this example.
- Associate utility functions with the numbering of alternatives: Our choices are indicated as numbers in the data (Biogeme **ONLY** accepts data sets with numerical values only. If there are columns with text values in your data, they should be either converted to numbers or removed). In this example, choice=1 suggests car as the chosen alternative for the trip, and choice=2 public transport. We use a dictionary "V" to indicate this association.
- We should repeat the process for the availability conditions (e.g., some individuals may not have access to car)
- Definition of the model: We use the Biogeme function 'loglogit' to indicate that this is the logarithm of a logit probability (remember that we maximise the log-likelihood and not the likelihood function). If we do not wish to use availabilities

(e.g. when forecasting) we replace the availability object with ***None***, in the *loglogit* function.
- The remainder of the code is for model estimation and display of the results. Please keep this part as is.
- Once we display the results, we see a table with familiar terms such as standard errors, t-tests, p-values etc. If you are unsure how to interpret these terms, have a look at the linear regression in the trip generation step (Step 1 of the 4–step model).

Now that we have a model, we should use it to forecast the modal split per OD pair!

1.2 Forecasting using a logit model

The purpose of a logit model (and any model) is to use it for forecasting. This means that we first need to estimate a model (we did so in Section 1.1). The idea is then the following:

- For each OD pair, we have the travel time, travel cost and any other relevant variable by mode. E.g., travel time by car, travel time by pt, travel cost by car, travel cost by pt etc.
- We implement our model using the attributes by OD (forecasting)
- The result should be the proportion of each mode for each OD pair

For the forecasting part, we will use the data in the zone_attributes_mode_choice.csv file. The data includes information with respect to the travel time (tt) and travel cost (tc) between the zones of the study area – including the external zones. The data is a processed version of the impedance matrix data of the Swiss National Passenger Transport Model 2017. The raw data is available (Impedance matrices of demand 2017 | opendata.swiss) but for now let's stick to the processed version that is available on moodle.

- Note: In theory the values of travel time should change after we implement network assignment. We then need to use the new values to get new travel times and forecast new modal split shares. For simplicity, we will ignore this step. In real world applications, we repeat this process several times until our results do not change from one repetition to the next.

The forecasting part takes place in the 01_mode_choice_MNL_forecasting.ipynb. Follow along the code to generate the shares.

1.3 Tasks: Mode choice

1. The example of the model that you estimated in 00_mode_choice_MNL.ipynb includes generic parameters i.e., the betas of travel time and travel cost are the same for each mode. Try changing to alternative–specific parameters i.e., different betas by mode.
2. Add trip purpose as a categorical variable in your model (and avoid the dummy variable trap!)
3. Obtain the car shares by trip purpose (work, education, and home).
4. Obtain the generalised cost OD matrices by trip purpose.

2 Network assignment

2.1 Preparing the data

If we have followed all labs so far, we should have completed the following steps:

1. Generate trip productions per zone per trip purpose with a linear regression model
2. Use the trip productions to get the attractions per zone (and per trip purpose)
3. Convert the productions and attractions to origin and destinations (per trip purpose)
4. Extract the proportion of daily trips for each trip purpose that takes place during the morning peak time
5. Estimate a mode choice model by trip purpose
6. Use the model to get the generalised cost by OD pair
7. Combine the data from Step 3 and Step 6 and get OD matrices by trip purpose

Now it is time to feed these OD matrices into the network that we generated in the first lab. However, before doing so we need to implement the following steps (in the **01_assignment** folder of the lab):

1. We have not modelled the non–home–based trips explicitly so far. We know from data (see linear regression code) that they represent approximately the 25% of the observations. (Follow notebook **00_Add_non_home_based_trips.ipynb**)
2. Within each matrix (by trip purpose) implement get the share of trips by car using the result of the logit model. (Follow notebook **01_Shares_of_trips_by_car.ipynb**). We must run the notebook once per trip purpose.
3. Address the issue of external zones. If you remember, we assumed four external zones when we created our model network. Our current OD matrix only assumes one external zone. We will just split the trips of this one zone into four segments. (Follow notebook **02_Expand_external_zones.ipynb**). We must run the notebook once per trip purpose.
4. Generate an OD matrix in a format that can be imported to QAequilibraE in QGIS. (Follow notebook **03_OD_for_network.ipynb**)

Once we are done with these steps, we can move to QAequilibraE and run the assignment.

2.2 Running the assignment model in QAequilibraE

It is now time to load the OD matrix to QAequilibraE and run the assignment model. First, let's load and visualise our project. As a reminder:

- We open QGIS
- In the QAequilibraE panel → Project → Open Project
- Click on the folder of your project (usually named new_project by default) and click on the Select Folder button
- Nothing happens! (this is normal)
- In the QAequilibraE panel → Routing → Travelling Salesman Problem (please do this for real, this is the best way to visualise the road network…)
- Close the window that just opened
- The nodes and links should now appear as new layers

Now we should import the OD matrix as a layer. As a reminder this can be done as:

- **Layer → Add Layer → Add Delimited Text Layer…**
- We browse and find the links csv file
- In Geometry Definition, we choose No geometry (attribute only table)
- Then we click on the 'Add' button

Now let's import the OD in our project:

- In the QAequilibraE panel → Data → Import matrices
- Select the settings as in ***Figure 1***
- Press LOAD
- When the data is loaded (you should be able to see the dimensions of the matrix) press Save
- Save the data in the *matrices* folder of your project

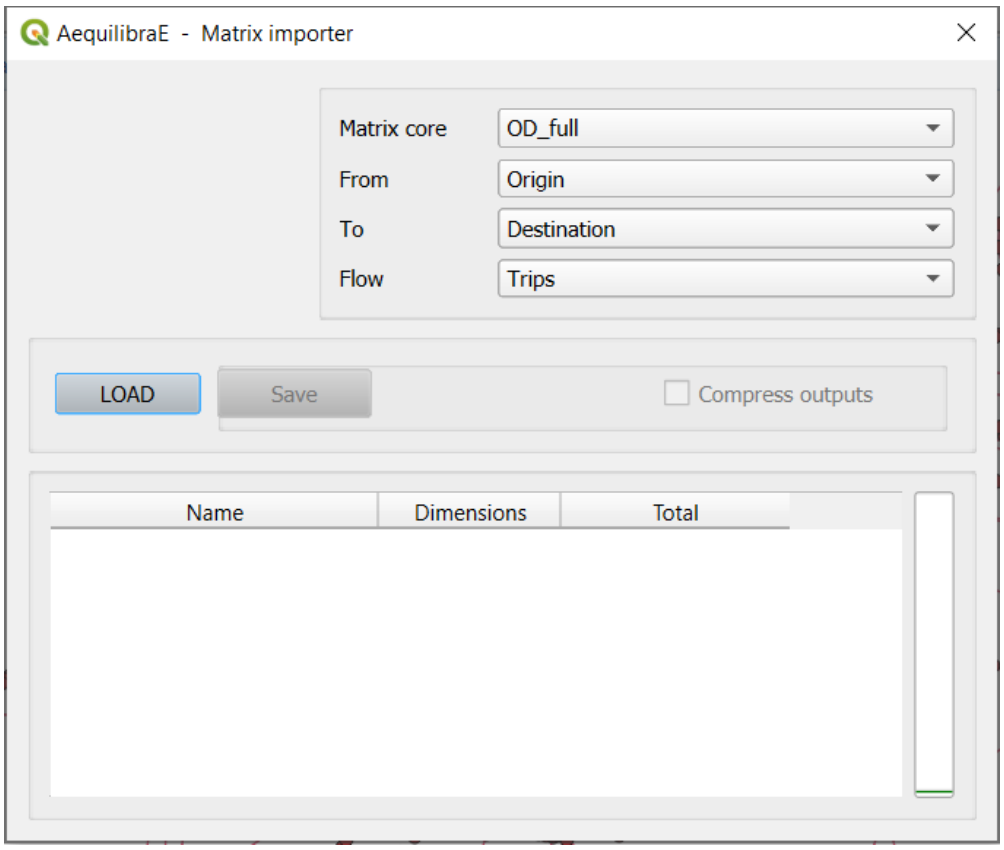

Figure 1 *OD matrix settings*

Now let's visualise the data:

- In the QAequilibraE panel → Data → Visualize data
- Pres the *Update matrix table* button
- Now the Matrices should be updated
- Select the matrix and then press the *Load matrix* button
- Note that by loading the data will also load the zones layer

If everything seems correct, we should do some extra processing in the links data before running the assignment model. We need to do three different tasks:

1. Replace missing speed values (this happens in the centroid connectors)
2. Compute travel time per link in free flow conditions
3. Compute capacity (we assume that some of the total capacity is occupied by other traffic than cars)

First, let's fix the missing speed values:

- Right click on the *links* layer in the layers panel and select *Open Attribute Table*
- Click on the *Open field calculator*
- Click on *Update existing field*
- Under *Update existing field* select speed_ab
- Expression: if(speed_ab IS NULL, 50, speed_ab)
- Click the *Apply* button (leave the field calculator open)

Second, let's compute the free flow travel time:

- Create a new field:
    - Output field name: travel_time_free
    - Output field type: Decimal (double)
    - Expression: distance/speed_ab
- Click the *Apply* button (leave the field calculator open)

Third, let's compute the new capacity:

- Create a new field:
    - Output field name: capacity_scenario_base
    - Output field type: Decimal (double)
    - Expression: capacity_ab*0.90 (we assume that 10% of the capacity is occupied by other modes of transport)
- Click the *Apply* button
- Click the *OK* button

Now we save the links table (disk icon), deactivate the editing mode (pencil icon) and close the attribute table.

Let's proceed with the assignment model:

- In the QAequilibraE panel → Paths and assignment → Traffic assignment
- In the *Traffic Classes* tab, in the Matrix name option, we select our data (if the file is missing, we should restart QGIS) and then click the *Add Traffic class to assignment* button (***Figure 2***)
- In the *Assignment* tab, follow ***Figure 3*** and press the ASSIGN button

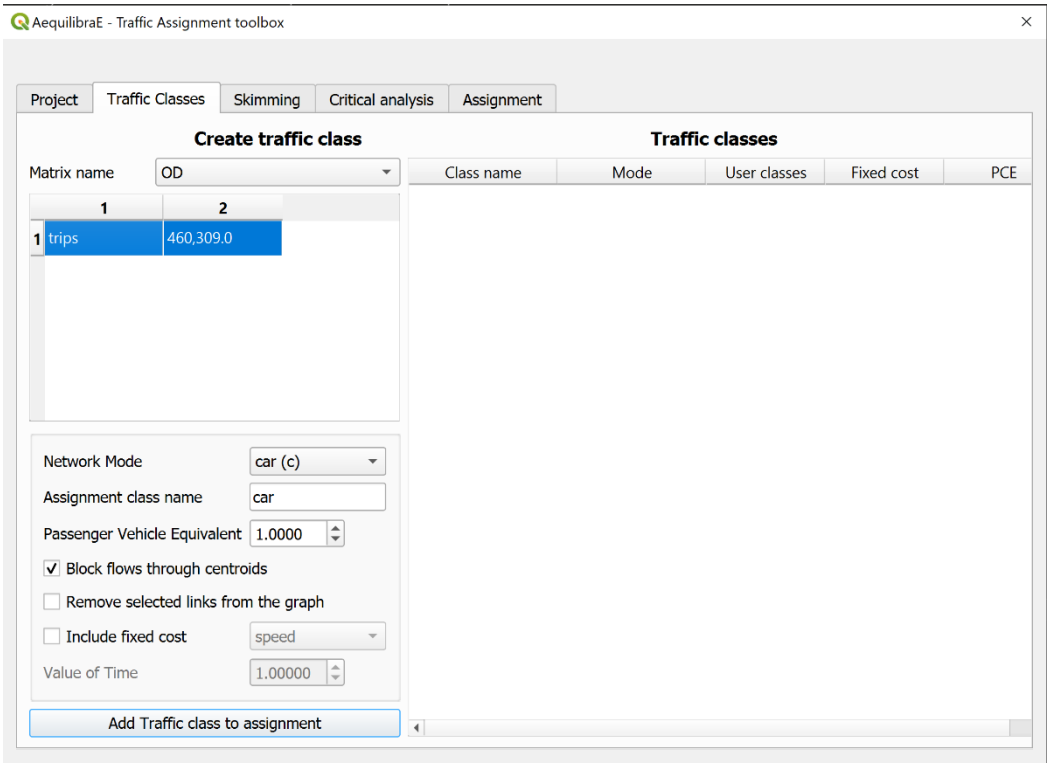

Figure 2 *Traffic classes input*

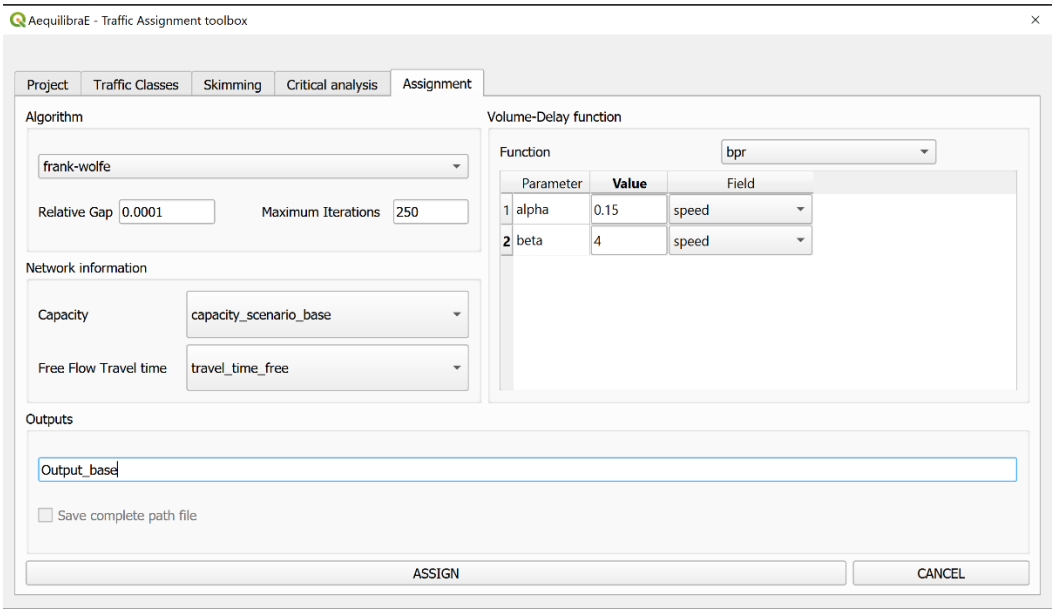

Figure 3 *Assignment input*

Hopefully, the bar reaches to 100% without errors. Now it is time to visualise the results:

- In the QAequilibraE panel → Data → Visualize data
- We go to the results tab
- We select the Output_base and press the *Load Result table as data layer* button (it may take a while to generate the layer)

Now there should be a new layer loaded in the layers panel. We can access the attribute table like any other layer. In the attribute table it is worth focusing on a few specific columns:

- trips_ab: shows the number of vehicles used the link
- Congested_Time_AB: shows the travel time of a link under congestion (congestion likely if the "Delay_factor" column > 1)
- VOC_AB: shows the Volume Over Capacity:
    - VOC = 1 → the link is operating at capacity
    - VOC < 1 → under capacity (free-flow or light congestion)
    - VOC > 1 → over capacity (congestion likely)

Comment: Given that our trips refer to traffic of 1 hour, if we observe Congested_Time_AB > 1, then our network is severely congested.