EPFL
ENAC TRANSP-OR
**Prof. M. Bierlaire**

Mathematical Modeling of Behavior
Fall 2018

Netherlands Mode Choice

This lab continues the tasks of the previous lab 10. In the previous lab, you have learned how to predict the aggregated market shares and how to run the simulation in Biogeme. Today, you will compute relevant indicators for policy analysis and forecast the prices that maximize the expected revenue.

This lab requires the dataset that contains the individual weights. You can continue using the dataset that you created in the previous lab, but the dataset `netherlandsRP_weight.dat` is also provided together with this description.

For these tasks, you should follow the instructions provided in *Description of the simulation file*, which is available during the MOOC (`Course > 7. Forecasting > 7.1 Aggregation > Description of the simulation file`), as in the previous lab.

With respect to the data and the base model specification, you can refer to the description of the previous lab.

# 1 Indicators

In this section, we ask you to compute the different indicators you have seen in the lecture.

1. *Value of time.* The value of time is calculated as

$$\mathrm{VOT}_{in} = \frac{(\partial V_{in}/\partial t_{in})(c_{in}, t_{in})}{(\partial V_{in}/\partial c_{in})(c_{in}, t_{in})}, \tag{1}$$

where $c_{in}$ and $t_{in}$ are the cost and travel time of alternative $i$ and individual $n$, respectively.
In Biogeme, the calculation of derivates is written as follows (example for car):
`VOT_CAR = Derive(V_CAR,'TravelTimeCar')/Derive(V_CAR,'CostCarEuro')`
Then, we just need to add this instruction in the `simulation` variable to be reported by Biogeme (example for car):
`simulate = {'VOT CAR': VOT_CAR}`

**Remark:** Notice that the `DefineVariable` operator is designed to preprocess the data file, and can be seen as a way to add another column in the data file, defining a new variable. However, the relationship between the new variable and the original one is lost.

Therefore, Biogeme is not able to properly calculate the derivatives. For instance, if you have scaled one variable, your variable of interest is the original variable and not the scaled one. Thus, if your variable of interest is `Time`, and not `Time_Scaled`, its definition must be explicitly known to correctly calculate the derivatives. Consequently, all statements such as `Time_Scaled = DefineVariable('Time_Scaled', Time/100)` should be replaced by statements such as
`Time_Scaled = Time/100`
in order to maintain the analytical structure of the formula to be derived. In our case this step is not necessary because our variable of interest is the one we are defining.

(a) *Provide an estimate of the average value of time in the population in eur/h for each alternative.*
The estimate of the average value of time in the population is obtained from the weighted average of the sample. The aggregate values are found in the "Weighted average" row of the `.html` file. The estimates of the value of time are equal to:
- car: 17.89 eur/h, and
- rail: 4.02 eur/h.

(b) *Analyze the distribution of the value of time for each alternative in the sample by identifying the socioeconomic characteristic(s) that play(s) a role in the calculation of the value of time and report its value together with the 90% confidence interval.*
By looking at the deterministic utilities and the `.html` file, one can easily identify two groups of the population with different values of time for both rail and car: individuals within Age1 and individuals within Age2. To easily identify which individuals belong to each group, we can include in the `simulate` variable one of the two variables (`Age1` or `Age2`) to be reported. The average values, together with their confidence intervals, which will be found in "Weighted non zeros average", are provided in Table 1. Note that the lower values of the intervals for rail are negative. This is due to the method that Biogeme uses to calculate these values.
**Remark:** In order to obtain the 90% confidence interval we need to uncomment/include the following statement:
`BIOGEME_OBJECT.VARCOVAR = vc`

| | $\text{Age}_1$ | $\text{Age}_2$ |
|---|---|---|
| Car | 15.42 [9.82, 28.82] | 23.03 [13.51,44.03] |
| Rail | 3.47 [-1.22, 9.56] | 5.18 [-1.54, 14.38] |

Table 1: Average value of time and 90% confidence interval for car and rail

2. *Aggregate elasticities.* Since the aggregate point elasticities are obtained by aggregating the disaggregate elasticities, we need to calculate the normalization factors. To do so, we include the following statements in the simulation file and run the resulting file:
```
BIOGEME_OBJECT.STATISTICS['Normalization for elasticities CAR'] = Sum(theWeight
* prob_CAR ,'obsIter')
```

```
BIOGEME_OBJECT.STATISTICS['Normalization for elasticities RAIL'] = Sum(theWeight
* prob_RAIL ,'obsIter')
```
Then, a simulation file with the statements for the aggregate elasticities of interest has to be created. For instance, if we want to calculate the aggregate elasticity of the choice of car with respect to its travel time, the following instructions need to be included:
```
normalization_car = ...
elas_car_time = Derive(prob_CAR,'TravelTimeCar') * TravelTimeCar / prob_CAR
```
and,
```
'Agg. Elast. PT - Time': elas_car_time * prob_CAR / normalization_car
```
where the first statement corresponds to the normalization factor (calculated when running the simulation file with the additional statements for the normalization factors), the second statement calculates the disaggregate elasticity of the choice of car with respect to its travel time and the third statement is the entry to the simulation dictionary (`simulate` variable) that is designed to calculate the aggregate elasticities.

The normalization factors are the following:

- car: 145.811, and
- rail: 82.189.

Now we can compute all the aggregate elasticities by including all the corresponding instructions in the simulation file (i.e., the two normalization factors, all the elasticities expressions and we have to include them in the `simulate` variable). The aggregated values can be found in the columns "Weighted total":

(a) *Elasticity of the share of car with respect to travel time by car:* -0.98
(b) *Elasticity of the share of car with respect to the cost of car:* -0.23
(c) *Elasticity of the share of rail with respect to travel time by rail:* -0.47
(d) *Elasticity of the share of rail with respect to the cost of rail:* -0.82
(e) *Elasticity of the share of car with respect to travel time by rail:* 0.27
(f) *Elasticity of the share of car with respect to the cost of rail:* 0.46
(g) *Elasticity of the share of rail with respect to travel time by car:* 1.76
(h) *Elasticity of the share of rail with respect to the cost of car:* 0.41

## 2 Forecasting

Now we ask you to forecast the market shares for car and rail under a certain increase of the latter, as well as to decide on the price that maximizes the generated revenue from a given set of price levels.
*Compute the predicted market shares for an increase of the cost of rail of 10%.*
We need to define a new variable capturing the increase in the cost of rail (before the utility statements):

`CostRailEuro_inc = DefineVariable('CostRailEuro_inc', CostRailEuro*1.10)`
Then, we replace the original cost variable by the new variable wherever it appears in the utility statements, and run the file as usual. The obtained market shares are the following:

- car: 66.83%, and
- rail: 33.16%.

Note that this technique can also be used to test different price levels and to decide on the one maximizing the generated revenue. Indeed, we can increase/decrease the price of an alternative in the same way as described before and then include in the simulation the following statement to report the revenue generated by the alternative. For instance, for the car alternative with its original cost we will write:
`'Revenue CAR': prob_CAR * CostCarEuro`
Then, the total generated revenue is included in the row "Weighted total".

---

mpp / yo / tr