EPFL
ENAC TRANSP-OR
Prof. M. Bierlaire

Mathematical Modeling of Behavior
Fall 2020

Netherlands Mode Choice

This lab continues the tasks of the previous lab 7. 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.

For these tasks, you should follow the instructions provided in the Biogeme documentation (sections `Market shares and revenues` and `Elasticities`), 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

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

   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,'car_time')/Derive(V_CAR,'car_cost_euro')
```

   Then, we just need to add this instruction in the `simulate` dictionary (example for car):

```
simulate = {'VOT car': VOT_CAR,
'Weighted VOT car': sampleNormalizedWeight * 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, database)` 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:

- 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 simulated values of (unweighted) values of time, 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. We can use the binary variables `age1` and `age2` defined in the previous lab and multiply them by the value of time of each mode, and add it to the `simulate` dictionary. We can then compute the resulting weighted averages and confidence intervals.

Remark 1: As these columns will contain zeros for individuals not belonging to the age group, we need to make sure to compute the weighted **non-zero** average. This can be done by directly filtering the `simulatedValues` dataframe. You can filter a pandas dataframe in the following way:

```
filter = (df['column'] != 0)  #filter = True if the value in the column is not 0
df = df[filter] #update the dataframe keeping only True values
```

Remark 2: In order to obtain the 90%, we have to define the following command in the specification of the simulation (see instructions 8-9 in documentation):

```
betas = biogeme.freeBetaNames
...
b = results.getBetasForSensitivityAnalysis(betas, size = 100)
left, right = biogeme.confidenceIntervals(b, 0.9)
```

The resulting `left` and `right` dataframes have the same structure as `simulatedValues`, so we can compute the weighted non-zero averages in the same way. The values of the confidence intervals might be slightly different than those reported here, but they should have the same order of magnitude.

	Age_1	Age_2
Car	15.42 [8.87, 28.51]	23.03 [14.48, 40.98]
Rail	3.47 [-1.10, 10.00]	5.18 [-1.53, 14.30]

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

2. *Aggregate elasticities.* To compute the aggregate point elasticities, we need to first compute the disaggregate elasticities. This can be done by defining a new variable containing the formula, then adding it to the `simulate` dictionary. Then, we need to compute the normalization factors. We can do so by using the simulated weighted probabilities (example for car):

```
normalization_car = simulatedValues['Weighted prob. Car'].sum()
```

The aggregate elasticities are thus computed using the simulated disaggregate point elasticities (example for car with respect to time):

```
agg_elast_car_time = (simulatedValues['Weighted prob car']
* simulatedValues['elast_car_time'] / normalization_car).sum()
```

The normalization factors are the following:

- car: 145.811,
- rail: 82.189.

And the aggregate elasticities:

(a) *Elasticity of the share of car with respect to travel time by car:* -0.99
(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):

```
  rail_cost_euro_inc = DefineVariable('rail_cost_euro_inc', rail_cost_euro * 1.1, database)
```

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: 66.84%, 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 * car_cost_euro
```

The total generated revenue is the (weighted) sum of this variable.

---

jp / th / an / no / mpp / mw