

Calculating indicators with PythonBiogeme

Michel Bierlaire

May 17, 2017

Report TRANSP-OR 170517

Transport and Mobility Laboratory

School of Architecture, Civil and Environmental Engineering

Ecole Polytechnique Fédérale de Lausanne

`transp-or.epfl.ch`

SERIES ON BIOGEME



1 Introduction

The package Biogeme (`biogeme.epfl.ch`) is designed to estimate the parameters of various models using maximum likelihood estimation. It is particularly designed for discrete choice models. But it can also be used to extract indicators from an estimated model. In this document, we describe how to calculate some indicators particularly relevant in the context of discrete choice models.

2 The model

We consider a case study involving a transportation mode choice model, using revealed preference data collected in Switzerland in 2009 and 2010 (see Atasoy et al., 2013). The model is a nested logit model with 3 alternatives: *public transportation*, *car* and *slow modes*. The utility functions are defined as:

$$\begin{aligned}
 V_{PT} &= \text{BETA_TIME_FULLTIME} * \text{TimePT_scaled} * \text{fulltime} + \\
 &\quad \text{BETA_TIME_OTHER} * \text{TimePT_scaled} * \text{notfulltime} + \\
 &\quad \text{BETA_COST} * \text{MarginalCostPT_scaled} \\
 V_{CAR} &= \text{ASC_CAR} + \\
 &\quad \text{BETA_TIME_FULLTIME} * \text{TimeCar_scaled} * \text{fulltime} + \\
 &\quad \text{BETA_TIME_OTHER} * \text{TimeCar_scaled} * \text{notfulltime} + \\
 &\quad \text{BETA_COST} * \text{CostCarCHF_scaled} \\
 V_{SM} &= \text{ASC_SM} + \\
 &\quad \text{BETA_DIST_MALE} * \text{distance_km_scaled} * \text{male} + \\
 &\quad \text{BETA_DIST_FEMALE} * \text{distance_km_scaled} * \text{female} + \\
 &\quad \text{BETA_DIST_UNREPORTED} * \text{distance_km_scaled} * \text{unreportedGender}
 \end{aligned}$$

where `ASC_CAR`, `ASC_SM`, `BETA_TIME_FULLTIME`, `BETA_TIME_OTHER`, `BETA_DIST_MALE`, `BETA_DIST_FEMALE`, `BETA_DIST_UNREPORTED`, `BETA_COST`, are parameters to be estimated, `TimePT_scale`, `MarginalCostPT_scaled`, `TimeCar_scale`, `CostCarCHF_scale`, `distance_km_scale` are attributes and `fulltime`, `notfulltime`, `male`, `female`, `unreportedGender` are socio-economic characteristics. The two alternatives “public transportation” and “slow modes” are grouped into a nested. The complete specification is available in the file `01nestedEstimation.py`, reported in Section A.1. We refer the reader to Bierlaire (2016) for an introduction to the syntax.

The parameters are estimated using `PythonBiogeme`. Their values are reported in Table 1. A file named `01nestedEstimation_param.py` is also generated. It contains the values of the estimated parameters written in `PythonBiogeme` syntax, as well as the code necessary to perform a sensitivity analysis. This code provides the variance-covariance matrix of the estimates.

Parameter number	Description	Coeff. estimate	Robust Asympt. std. error	t-stat	p-value
1	ASC_CAR	0.261	0.100	2.61	0.01
2	ASC_SM	0.0590	0.217	0.27	0.79
3	BETA_COST	-0.716	0.138	-5.18	0.00
4	BETA_DIST_FEMALE	-0.831	0.193	-4.31	0.00
5	BETA_DIST_MALE	-0.686	0.161	-4.27	0.00
6	BETA_DIST_UNREPORTED	-0.703	0.196	-3.58	0.00
7	BETA_TIME_FULLTIME	-1.60	0.333	-4.80	0.00
8	BETA_TIME_OTHER	-0.577	0.296	-1.95	0.05
9	NEST_NOCAR	1.53	0.306	5.00	0.00

Summary statistics

Number of observations = 1906

Number of excluded observations = 359

Number of estimated parameters = 9

$$\mathcal{L}(\beta_0) = -2093.955$$

$$\mathcal{L}(\hat{\beta}) = -1298.498$$

$$-2[\mathcal{L}(\beta_0) - \mathcal{L}(\hat{\beta})] = 1590.913$$

$$\rho^2 = 0.380$$

$$\bar{\rho}^2 = 0.376$$

Table 1: Nested logit model: estimated parameters

3 Market shares and revenues

Once the model has been estimated, it must be used to derive useful indicators. PythonBiogeme provides a simulation feature for this purpose. We start by describing how to calculate market shares using sample numeration. It is necessary to have a sample of individuals from the population. For each of them, the value of each of the variables involved in the model must be known. Note that it is possible to use the same sample that what used for estimation, **but only if it contains revealed preferences data.** It is the procedure used in this document.

More formally, consider a choice model $P_n(i|x_n, C_n)$ providing the probability that individual n chooses alternative i within the choice set C_n , given the explanatory variables x_n . In order to calculate the market shares in the population of size N , a sample of N_s individuals is drawn. As it is rarely possible to draw from the population with equal sampling probability, it is assumed that stratified sampling has been used, and that each individual n in the sample is associated with a weight w_n correcting for sampling biases. The weights are normalized such that

$$N_s = \sum_{n=1}^{N_s} w_n. \quad (1)$$

An estimator of the market share of alternative i in the population is

$$W_i = \frac{1}{N_s} \sum_{n=1}^{N_s} w_n P_n(i|x_n, C_n). \quad (2)$$

If the alternative i involves a price variable p_{in} , the expected revenues generated by i is

$$R_i = \frac{N}{N_s} \sum_{n=1}^{N_s} w_n p_{in} P_n(i|x_n, p_{in}, C_n). \quad (3)$$

In practice, the size of the population is rarely known, and the above quantity is used only in the context of price optimization. In this case, the factor N/N_s can be omitted.

To calculate (2) and (3) with PythonBiogeme, a specification file must be generated. In our example, the file `02nestedSimulation.py`, reported in Section A.2, has been generated as follows:

1. Start with a copy of the model estimation file `01nestedEstimation.py`.
2. Replace all Beta statements by the equivalent statements in the file `01nestedEstimation.param.py`.

3. Copy and paste the code for the sensitivity analysis, that is

- the names of the parameters: `names=...`
- the values of the variance-covariance matrix: `values=...`
- the definition of the matrix itself:

```
vc = bioMatrix(9,names,values)
BIOGEME OBJECT.VARCOVAR = vc
```

4. Remove the statement related to the estimation:

```
BIOGEME.OBJECT.ESTIMATE = Sum(logprob,'obsIter')
```

5. Replace it by the statement for simulation:

```
BIOGEME.OBJECT.SIMULATE = Enumerate(simulate,'obsIter')
```

The `simulate` variable must be a dictionary describing what has to be calculated during the sample enumeration. In this case, we calculate, for each individual in the sample, the choice probability of each alternative. We also calculate the expected revenue generated by each individual for the public transportation companies, using the following statement:

```
simulate = {'Prob. car': prob_car ,
            'Prob. public transportation': prob_pt ,
            'Prob. slow modes': prob_sm ,
            'Revenue public transportation':
            prob_pt * MarginalCostPT}
```

Each entry of this dictionary corresponds to a quantity that will be calculated. The key of the entry is a string, that will be used for the reporting. The value must be a valid formula describing the calculation. In our example, we have defined

```
prob_pt = nested(V,av,nests,0)
prob_car = nested(V,av,nests,1)
prob_sm = nested(V,av,nests,2)
```

In the output of the estimation (see the file `01nestedEstimation.html`), the sum of all weights have been calculated using the statement

```
BIOGEME.OBJECT.STATISTICS['Sum of weights'] = Sum(Weight,'obsIter')
```

The reported result is 0.814484. Therefore, in order to verify (1), we introduce the following statements:

```
theWeight = Weight * 1906 / 0.814484
BIOGEME.OBJECT.WEIGHT = theWeight
```

as there are 1906 entries in the data file.

In order to prepare for the calculation of elasticities, we have also included the following statements:

```
BIOGEME.OBJECT.STATISTICS['Normalization for elasticities PT'] = \
    Sum(theWeight * prob_pt , 'obsIter')
BIOGEME.OBJECT.STATISTICS['Normalization for elasticities CAR'] = \
    Sum(theWeight * prob_car , 'obsIter')
BIOGEME.OBJECT.STATISTICS['Normalization for elasticities SM'] = \
    Sum(theWeight * prob_sm , 'obsIter')
```

The reasons are discussed in the next section.

The simulation is performed using the statement

```
pythonbiogeme 02nestedSimulation optima.dat
```

It generates the file 02nestedSimulation.html, that contains the following sections:

- The preamble reports information about the version of PythonBiogeme, useful URLs and the names of the files involved in the run.
- Statistics: this section is the same as for the estimation, and reports the requested statistics:

```
Alt. 0 available: 1906
Alt. 0 chosen: 536
Alt. 1 available: 1906
Alt. 1 chosen: 1256
Alt. 2 available: 1906
Alt. 2 chosen: 114
Cte loglikelihood (only for full choice sets): -1524.92
Gender: females: 871
Gender: males: 943
Gender: unreported: 92
Normalization for elasticities CAR: 1244.77
Normalization for elasticities PT: 535.086
Normalization for elasticities SM: 126.147
Null loglikelihood: -2093.96
Number of entries: 1906
Occupation: full time: 798
Sum of weights: 0.814484
```

- The simulation report contains two parts: the aggregate values, and the detailed records. We start by describing the latter. It reports, for each row of the sample file, the weight w_n (last column) and, for each entry in the dictionary defined by BIOGEME.OBJECT.SIMULATE

1. the calculated quantity,

2. the 90% confidence interval for this quantity. It is calculated using simulation. As the estimates have been obtained from maximum likelihood, they are (asymptotically) normally distributed. Therefore, we draw from a multivariate normal distribution $N(\hat{\beta}, \hat{\Sigma})$, where $\hat{\beta}$ is the vector of estimated parameters, and $\hat{\Sigma}$ is the variance-covariance matrix defined by the BIOGEME OBJECT.VARCOVAR statement. The number of draws is controlled by the parameter NbrOfDrawsForSensitivityAnalysis. The requested quantity is calculated for each realization, and the 5% and the 95% quantiles of the obtained simulated values are reported to generate the 90% confidence interval. Note that the confidence interval is reported only if the statement

```
BIOGEME.OBJECT.VARCOVAR = vc
```

is present. If you do not need the confidence intervals, simply remove this statement from the .py file.

- Simulation report: aggregate values. For each calculated quantity, aggregate indicators are calculated. Denote by z_n a calculated quantity (the probability that individual n chooses the car alternative, for instance). Then, the following aggregate values are reported, together with the associated confidence interval (if requested):

- Total:

$$\sum_{n=1}^{N_s} z_n. \quad (4)$$

- Weighted total:

$$\sum_{n=1}^{N_s} w_n z_n. \quad (5)$$

- Average:

$$\frac{1}{N_s} \sum_{n=1}^{N_s} z_n. \quad (6)$$

- Weighted average:

$$\frac{1}{N_s} \sum_{n=1}^{N_s} w_n z_n. \quad (7)$$

- Non zeros:

$$\sum_{n=1}^{N_s} \delta(z_n \neq 0), \quad (8)$$

where

$$\delta(z_n \neq 0) = \begin{cases} 1 & \text{if } z_n \neq 0, \\ 0 & \text{otherwise.} \end{cases} \quad (9)$$

– Non zeros average:

$$\frac{\sum_{n=1}^{N_s} z_n}{\sum_{n=1}^{N_s} \delta(z_n \neq 0)}. \quad (10)$$

– Weighted non zeros average:

$$\frac{\sum_{n=1}^{N_s} w_n z_n}{\sum_{n=1}^{N_s} \delta(z_n \neq 0)}. \quad (11)$$

– Minimum:

$$\min_n z_n. \quad (12)$$

– Maximum:

$$\max_n z_n. \quad (13)$$

Therefore, the result of (2) is available in the row “Weighted average”. In this example, the market shares are:

- car: 65.3078% (confidence interval: [60.5884%,69.0407%]),
- public transportation: 28.0738% (confidence interval: [23.603%,32.391%]),
- slow modes: 6.61844% (confidence interval: 4.54637%,10.417%).

The result of (3) is obtained in the row “Weighted total”. In this case, the expected revenue (generated by the individuals in the sample) is 3018.29 (confidence interval: [2442.87,3826.36]).

4 Elasticities

Consider now one of the variables involved in the model: x_{ink} . The objective is to anticipate the impact of a change of the value of this variable on the choice of individual n , and on the market share of alternative i .

4.1 Point elasticities

We assume that the relative (infinitesimal) change of the variable is the same for every individual in the population, that is

$$\frac{\partial x_{ink}}{x_{ink}} = \frac{\partial x_{ipk}}{x_{ipk}} = \frac{\partial x_{ik}}{x_{ik}}, \quad (14)$$

where

$$x_{ik} = \frac{1}{N_s} \sum_{n=1}^{N_s} x_{ink}. \quad (15)$$

The *disaggregate direct point elasticity* of the model with respect to the variable x_{ink} is defined as

$$E_{x_{ink}}^{P_n(i)} = \frac{\partial P_n(i|x_n, C_n)}{\partial x_{ink}} \frac{x_{ink}}{P_n(i|x_n, C_n)}. \quad (16)$$

It is called

- disaggregate, because it refers to the choice model related to a specific individual,
- direct, because it measures the impact of a change of an attribute of alternative i on the choice probability of the same alternative,
- point, because we consider an infinitesimal change of the variable.

The *aggregate direct point elasticity* of the model with respect to the average value x_{ik} is defined as

$$E_{x_{ik}}^{W_i} = \frac{\partial W_i}{\partial x_{ik}} \frac{x_{ik}}{W_i}. \quad (17)$$

Using (2), we obtain

$$E_{x_{ik}}^{W_i} = \frac{1}{N_s} \sum_{n=1}^{N_s} w_n \frac{\partial P_n(i|x_n, C_n)}{\partial x_{ik}} \frac{x_{ik}}{W_i}. \quad (18)$$

From (14), we obtain

$$E_{x_{ik}}^{W_i} = \frac{1}{N_s} \sum_{n=1}^{N_s} w_n \frac{\partial P_n(i|x_n, C_n)}{\partial x_{ink}} \frac{x_{ink}}{W_i} = \frac{1}{N_s} \sum_{n=1}^{N_s} w_n E_{x_{ink}}^{P_n(i)} \frac{P_n(i|x_n, C_n)}{W_i}, \quad (19)$$

where the second equation is derived from (16). Using (2) again, we obtain

$$E_{x_{ik}}^{W_i} = \sum_{n=1}^{N_s} E_{x_{ink}}^{P_n(i)} \frac{w_n P_n(i|x_n, C_n)}{\sum_{n=1}^{N_s} w_n P_n(i|x_n, C_n)}. \quad (20)$$

This equation shows that the calculation of aggregate elasticities involves a weighted sum of disaggregate elasticities. However, the weight is not w_n as for the market share, but a normalized version of $w_n P_n(i|x_n, C_n)$.

The *disaggregate cross point elasticity* of the model with respect to the variable x_{jnk} is defined as

$$E_{x_{jnk}}^{P_n(i)} = \frac{\partial P_n(i|x_n, C_n)}{\partial x_{jnk}} \frac{x_{jnk}}{P_n(i|x_n, C_n)}. \quad (21)$$

It is called *cross* elasticity because it measures the sensitivity of the model for alternative i with respect to a modification of the attribute of another alternative.

4.2 Arc elasticities

A similar derivation can be done for arc elasticities. In this case, the relative change of the variable is not infinitesimal anymore. The idea is to analyze a before/after scenario. The variable x_{ink} in the before scenario becomes $x_{ink} + \Delta x_{ink}$ in the after scenario. As above, we assume that the relative change of the variable is the same for every individual in the population, that is

$$\frac{\Delta x_{ink}}{x_{ink}} = \frac{\Delta x_{ipk}}{x_{ipk}} = \frac{\Delta x_{ik}}{x_{ik}}, \quad (22)$$

where x_{ik} is defined by (15). The *disaggregate direct arc elasticity* of the model with respect to the variable x_{ink} is defined as

$$E_{x_{ink}}^{P_n(i)} = \frac{\Delta P_n(i|x_n, C_n)}{\Delta x_{ink}} \frac{x_{ink}}{P_n(i|x_n, C_n)}. \quad (23)$$

The *aggregate direct arc elasticity* of the model with respect to the average value x_{ik} is defined as

$$E_{x_{ik}}^{W_i} = \frac{\Delta W_i}{\Delta x_{ik}} \frac{x_{ik}}{W_i}. \quad (24)$$

The two quantities are also related by (20), following the exact same derivation as for the point elasticity.

4.3 Using PythonBiogeme for disaggregate point elasticities

The calculation of (16) involves derivatives. For simple models such as logit, the analytical formula of these derivatives can easily be derived. However, their derivation for advanced models can be tedious. It is common to make

mistakes in the derivation itself, and even more common to make mistakes in the implementation. Therefore, PythonBiogeme provides an operator that calculates the derivative of a formula. This is illustrated in the file 03 nestedElasticities .py, reported in Section A.3. The statements that trigger the calculation of the elasticities are:

```
elas_pt_time = Derive(prob_pt, 'TimePT') * TimePT / prob_pt
elas_pt_cost = Derive(prob_pt, 'MarginalCostPT') * MarginalCostPT / prob_pt
elas_car_time = Derive(prob_car, 'TimeCar') * TimeCar / prob_car
elas_car_cost = Derive(prob_car, 'CostCarCHF') * CostCarCHF / prob_car
elas_sm_dist = Derive(prob_sm, 'distance_km') * distance_km / prob_sm
```

The above syntax should be self-explanatory. But there is an important aspect to take into account. In the context of the estimation of the parameters of the model, the variables have been scaled in order to improve the numerical properties of the likelihood function, using statements like

```
TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )
```

But the variable of interest, for the sake of applications, is TimePT, not TimePT_scaled. The problem is that the DefineVariable operator is designed to preprocess the data file. It can be seen as a way to add another column in the data file, defining a new variable. It means that it loses the analytical relationship between the new variable and the original one. Therefore, PythonBiogeme is not able to properly calculate the derivatives. Consequently, all statements such as

```
TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )
```

should be replaced by statements such as

```
TimePT_scaled = TimePT / 200
```

in order to maintain the analytical structure of the formula to be derived.

4.4 Using PythonBiogeme for aggregate point elasticities

The aggregate point elasticities can be obtained by aggregating the disaggregate elasticities, using (20). This requires the calculation of the normalization factors

$$\sum_{n=1}^{N_s} w_n P_n(i|x_n, C_n). \quad (25)$$

This has been performed during the previous simulation using the statements

```
BIOGEME.OBJECT.STATISTICS['Normalization for elasticities PT'] = \
    Sum(theWeight * prob_pt, 'obsIter')
```

```
BIOGEME.OBJECT.STATISTICS['Normalization for elasticities CAR'] = \
    Sum(theWeight * prob_car , 'obsIter')
BIOGEME.OBJECT.STATISTICS['Normalization for elasticities SM'] = \
    Sum(theWeight * prob_sm , 'obsIter')
```

Therefore, we have now included the following statements:

```
normalization_pt = 535.086
normalization_car = 1244.77
normalization_sm = 126.147
```

The quantities that must be calculated for each individual, in order to derive the aggregate elasticities correspond to the following entries in the dictionary:

```
'Agg. Elast. PT - Time': elas_pt_time * prob_pt / normalization_pt ,
'Agg. Elast. PT - Cost': elas_pt_cost * prob_pt / normalization_pt ,
'Agg. Elast. Car - Time': elas_car_time * prob_car / normalization_car ,
'Agg. Elast. Car - Cost': elas_car_cost * prob_car / normalization_car ,
'Agg. Elast. Slow modes - Distance': elas_sm_dist * prob_sm / normalization_sm
```

Note that the weights have not been included in the above formula, so that the values of the aggregate elasticities can be found in the row “Weighted total”:

- Car — cost: -0.0906321,
- Car — travel time: -0.0440771,
- Public transportation — cost: -0.320246,
- Public transportation — travel time: -0.274315,
- Slow ~~model~~ — distance: -1.09095.

Equivalently, we could have used statements like

```
'Agg. Elast. PT - Time': theWeight * elas_pt_time * prob_pt / normalization_pt ,
```

and the aggregate value would have been found in the row “Total” instead of “Weighted total”. Note also that we have omitted to report the confidence intervals in this example, by commenting out the statement:

```
#BIOGEME.OBJECT.VARCOVAR = vc
```

4.5 Using PythonBiogeme for cross elasticities

The calculation of (21) is performed in a similar way as the direct elasticities, using the following statements:

```

elas_car_cost = Derive(prob_car, 'MarginalCostPT') * MarginalCostPT / prob_car
elas_car_time = Derive(prob_car, 'TimePT') * TimePT / prob_car
elas_pt_cost = Derive(prob_pt, 'CostCarCHF') * CostCarCHF / prob_pt
elas_pt_time = Derive(prob_pt, 'TimeCar') * TimeCar / prob_pt

```

They calculate the following elasticities:

- choice of car with respect to the marginal cost of public transportation,
- choice of car with respect to travel time by public transportation,
- choice of public transportation with respect to cost of the car,
- choice of public transportation with respect to travel time by car.

The corresponding aggregate elasticities are calculated exactly like for the direct case, and their values can be found in the row “Weighted total”.

- Agg. Elast. Car - Cost PT: 0.123008
- Agg. Elast. Car - Time PT: 0.106567
- Agg. Elast. PT - Cost car: 0.199984
- Agg. Elast. PT - Time car: 0.0953097

Note that these values are now positive. Indeed, when the travel time or travel cost of a competing mode increase, the market share increases.

4.6 Using PythonBiogeme for arc elasticities

Arc elasticities require a before and after scenarios. In this case, we calculate the sensitivity of the market share of the slow mode alternative when there is a uniform increase of 1 kilometer.

The “before” scenario is represented by the same model as above. The after scenario is modeled using the following statements:

```

delta_dist = 1
distance_km_scaled_after = (distance_km + delta_dist) / 5
V_SM_after = ASC_SM + \
    BETA_DIST_MALE * distance_km_scaled_after * male + \
    BETA_DIST_FEMALE * distance_km_scaled_after * female + \
    BETA_DIST_UNREPORTED * distance_km_scaled_after * unreportedGender
V_after = {0: V_PT,
            1: V_CAR,
            2: V_SM_after}
prob_sm_after = nested(V_after, av, nests, 2)

```

Then, the arc elasticity is calculated as

$$\text{elas_sm_dist} = \frac{(\text{prob_sm_after} - \text{prob_sm}) * \text{distance_km}}{(\text{prob_sm} * \text{delta_dist})}$$

The aggregate elasticity is calculated as explained above. It is equal here to -1.00708, and the confidence interval is [-1.7212,-0.562574].

5 Willingness to pay

If the model contains a cost or price variable (like in this model), it is possible to analyze the trade-off between any variable and money. It reflects the willingness of the decision maker to pay for a modification of another variable of the model. A typical example in transportation is the *value of time*, that is the amount of money a traveler is willing to pay in order to decrease her travel time.

Let c_{in} be the cost of alternative i for individual n . Let x_{in} be the value of another variable of the model. Let $V_{in}(c_{in}, x_{in})$ be the value of the utility function. Consider a scenario where the variable under interest takes the value $x_{in} + \delta_{in}^x$. We denote by δ_{in}^c the additional cost that would achieve the same utility, that is

$$V_{in}(c_{in} + \delta_{in}^c, x_{in} + \delta_{in}^x) = V_{in}(c_{in}, x_{in}). \quad (26)$$

The willingness to pay is defined as the additional cost per unit of x , that is

$$\delta_{in}^c / \delta_{in}^x, \quad (27)$$

and is obtained by solving Equation (26). If x_{in} and c_{in} appear linearly in the utility function, that is if

$$V_{in}(c_{in}, x_{in}) = \beta_c c_{in} + \beta_x x_{in} + \dots, \quad (28)$$

and

$$V_{in}(c_{in} + \delta_{in}^c, x_{in} + \delta_{in}^x) = \beta_c (c_{in} + \delta_{in}^c) + \beta_x (x_{in} + \delta_{in}^x) + \dots. \quad (29)$$

Therefore, (27) is

$$\delta_{in}^c / \delta_{in}^x = -\beta_x / \beta_c. \quad (30)$$

If x_{in} is a continuous variable, and if V_{in} is differentiable in x_{in} and c_{in} , we can invoke Taylor's theorem in (26):

$$\begin{aligned} V_{in}(c_{in}, x_{in}) &= V_{in}(c_{in} + \delta_{in}^c, x_{in} + \delta_{in}^x) \\ &\approx V_{in}(c_{in}, x_{in}) + \delta_{in}^c \frac{\partial V_{in}}{\partial c_{in}}(c_{in}, x_{in}) + \delta_{in}^x \frac{\partial V_{in}}{\partial x_{in}}(c_{in}, x_{in}) \end{aligned} \quad (31)$$

Therefore, the willingness to pay is equal to

$$\frac{\delta_{in}^c}{\delta_{in}^x} = - \frac{(\partial V_{in}/\partial x_{in})(c_{in}, x_{in})}{(\partial V_{in}/\partial c_{in})(c_{in}, x_{in})}. \quad (32)$$

Note that if x_{in} and c_{in} appear linearly in the utility function, (32) is the same as (30). If we consider now a scenario where the variable under interest takes the value $x_{in} - \delta_{in}^x$, the same derivation leads to

$$\frac{\delta_{in}^c}{\delta_{in}^x} = \frac{(\partial V_{in}/\partial x_{in})(c_{in}, x_{in})}{(\partial V_{in}/\partial c_{in})(c_{in}, x_{in})}. \quad (33)$$

The calculation of the value of time corresponds to such a scenario.

$$\frac{\delta_{in}^c}{\delta_{in}^t} = \frac{(\partial V_{in}/\partial t_{in})(c_{in}, t_{in})}{(\partial V_{in}/\partial c_{in})(c_{in}, t_{in})} = \frac{\beta_t}{\beta_c}, \quad (34)$$

where the last equation assumes that V is linear in these variables. Note that, in this special case of linear utility functions, the value of time is constant across individuals, and is also independent of δ_{in}^t . This is not true in general.

The calculation of (33) involves the calculation of derivatives. It is done in Pythonbiogeme using the following statements:

```
WTP_PT_TIME = Derive(V_PT, 'TimePT') / Derive(V_PT, 'MarginalCostPT')
WTP_CAR_TIME = Derive(V_CAR, 'TimeCar') / Derive(V_CAR, 'CostCarCHF')
```

The full specification file can be found in Section A.6. The aggregate values are found in the “Weighted average” row of the report file: 3.95822 CHF/hour (confidence interval: [1.98696, 7.81565]). Note that this value is abnormally low, which is a sign of a poor specification of the model. Note also that, with this specification, the value of time is the same for car and public transportation, as the coefficients of the time and cost variables are generic.

Finally, it is important to look at the distribution of the willingness to pay in the population/sample. The detailed records of the report file allows you to do so. It is easy to drag and drop the HTML report file into your favorite spreadsheet software in order to perform additional statistics. In this example, the value of time takes two values, depending on the employment status of the individual:

- Full time: 6.68992 (confidence interval: [4.15056, 11.1866])
- Not full time: 2.41847 (confidence interval: [0.829511, 5.91561])

6 Conclusion

PythonBiogeme is a flexible tool that allows to extract useful indicators from complex models. In this document, we have presented how some indicators relevant for discrete choice models can be generated. The HTML format of the report not only allows to display the report in your favorite browser. It also allows to import the generated values in a spreadsheet for more manipulations.

A Complete specification files

A.1 01nestedEstimation.py

```
1  ## File 01nestedEstimation.py
2  ## Simple nested logit model for the Optima case study
3  ## Wed May 10 10:55:12 2017
4
5  from biogeme import *
6  from headers import *
7  from loglikelihood import *
8  from statistics import *
9  from nested import *
10
11  ### Three alternatives:
12  # CAR: automobile
13  # PT: public transportation
14  # SM: slow mode (walking, biking)
15
16  ### List of parameters to be estimated
17  ASC_CAR = Beta('ASC_CAR',0,-10000,10000,0)
18  ASC_SM = Beta('ASC_SM',0,-10000,10000,0)
19  BETA.TIME.FULLTIME = Beta('BETA.TIME.FULLTIME',0,-10000,10000,0)
20  BETA.TIME.OTHER = Beta('BETA.TIME.OTHER',0,-10000,10000,0)
21  BETA.DIST.MALE = Beta('BETA.DIST.MALE',0,-10000,10000,0)
22  BETA.DIST.FEMALE = Beta('BETA.DIST.FEMALE',0,-10000,10000,0)
23  BETA.DIST.UNREPORTED = Beta('BETA.DIST.UNREPORTED',0,-10000,10000,0)
24  BETA.COST = Beta('BETA.COST',0,-10000,10000,0)
25
26
27  ### Definition of variables:
28  # For numerical reasons, it is good practice to scale the data to
29  # that the values of the parameters are around 1.0.
30
31  # The following statements are designed to preprocess the data.
32  # It is like creating a new columns in the data file. This
33  # should be preferred to the statement like
34  # TimePT_scaled = Time_PT / 200.0
35  # which will cause the division to be reevaluated again and again,
36  # throuh the iterations. For models taking a long time to
37  # estimate, it may make a significant difference.
38
39  TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )
40  TimeCar_scaled = DefineVariable('TimeCar_scaled', TimeCar / 200 )
41  MarginalCostPT_scaled = DefineVariable('MarginalCostPT_scaled',
42                                         MarginalCostPT / 10 )
43  CostCarCHF_scaled = DefineVariable('CostCarCHF_scaled',
44                                     CostCarCHF / 10 )
```

```

45 distance_km_scaled = DefineVariable('distance_km_scaled',
46                                     distance_km / 5 )
47
48 male = DefineVariable('male',Gender == 1)
49 female = DefineVariable('female',Gender == 2)
50 unreportedGender = DefineVariable('unreportedGender',Gender == -1)
51
52 fulltime = DefineVariable('fulltime',OccupStat == 1)
53 notfulltime = DefineVariable('notfulltime',OccupStat != 1)
54
55 ### Definition of utility functions:
56 V_PT = BETA.TIME_FULLTIME * TimePT_scaled * fulltime + \
57         BETA.TIME_OTHER * TimePT_scaled * notfulltime + \
58         BETA.COST * MarginalCostPT_scaled
59 V_CAR = ASC_CAR + \
60         BETA.TIME_FULLTIME * TimeCar_scaled * fulltime + \
61         BETA.TIME_OTHER * TimeCar_scaled * notfulltime + \
62         BETA.COST * CostCarCHF_scaled
63 V_SM = ASC_SM + \
64         BETA.DIST_MALE * distance_km_scaled * male + \
65         BETA.DIST_FEMALE * distance_km_scaled * female + \
66         BETA.DIST_UNREPORTED * distance_km_scaled * unreportedGender
67
68 # Associate utility functions with the numbering of alternatives
69 V = {0: V_PT,
70      1: V_CAR,
71      2: V_SM}
72
73 # Associate the availability conditions with the alternatives.
74 # In this example all alternatives are available for each individual.
75 av = {0: 1,
76       1: 1,
77       2: 1}
78
79 ### DEFINITION OF THE NESTS:
80 # 1: nests parameter
81 # 2: list of alternatives
82
83 NEST_NOCAR = Beta('NEST_NOCAR',1,1.0,10,0)
84
85 CAR = 1.0 , [ 1]
86 NO_CAR = NEST_NOCAR , [ 0, 2]
87 nests = CAR, NO_CAR
88
89 # All observations verifying the following expression will not be
90 # considered for estimation
91 BIOGEME.OBJECT.EXCLUDE = Choice == -1
92
93

```

```

94 # The choice model is a nested logit, with availability conditions
95 logprob = lognested(V,av,nests,Choice)
96
97 # Defines an iterator on the data
98 rowIterator('obsIter')
99
100 #Statistics
101 nullLoglikelihood(av,'obsIter')
102 choiceSet = [0,1,2]
103 cteLoglikelihood(choiceSet,Choice,'obsIter')
104 availabilityStatistics(av,'obsIter')
105
106 BIOGEME.OBJECT.STATISTICS['Gender: males'] = \
107     Sum(male,'obsIter')
108 BIOGEME.OBJECT.STATISTICS['Gender: females'] = \
109     Sum(female,'obsIter')
110 BIOGEME.OBJECT.STATISTICS['Gender: unreported'] = \
111     Sum(unreportedGender,'obsIter')
112 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = \
113     Sum(fulltime,'obsIter')
114 BIOGEME.OBJECT.STATISTICS['Sum of weights'] = \
115     Sum(Weight,'obsIter')
116
117 # Define the likelihood function for the estimation
118 BIOGEME.OBJECT.ESTIMATE = Sum(logprob,'obsIter')
119 BIOGEME.OBJECT.PARAMETERS['optimizationAlgorithm'] = "CFSQP"

```

A.2 02nestedSimulation.py

```

1  ## File 02nestedSimulation.py
2  ## Simple nested logit model for the Optima case study
3  ## Wed May 10 11:24:32 2017
4
5  from biogeme import *
6  from headers import *
7  from statistics import *
8  from nested import *
9
10 ##### Three alternatives:
11 # CAR: automobile
12 # PT: public transportation
13 # SM: slow mode (walking, biking)
14
15 ##### List of parameters and their estimated value.
16 ASC_CAR = Beta('ASC_CAR',0.261291,-10000,10000,0,'ASC_CAR' )
17 ASC_SM = Beta('ASC_SM',0.0590204,-10000,10000,0,'ASC_SM' )
18 BETA.TIME.FULLTIME = \
19     Beta('BETA_TIME_FULLTIME',-1.59709,-10000,10000,0,'BETA_TIME_FULLTIME' )
20 BETA.TIME.OTHER = \

```

```

21  Beta('BETA_TIME_OTHER', -0.577362, -10000, 10000, 0, 'BETA_TIME_OTHER' )
22  BETA_DIST_MALE = \
23  Beta('BETA_DIST_MALE', -0.686327, -10000, 10000, 0, 'BETA_DIST_MALE' )
24  BETA_DIST_FEMALE = \
25  Beta('BETA_DIST_FEMALE', -0.83121, -10000, 10000, 0, 'BETA_DIST_FEMALE' )
26  BETA_DIST_UNREPORTED = \
27  Beta('BETA_DIST_UNREPORTED', -0.702974, -10000, 10000, 0, 'BETA_DIST_UNREPORTED' )
28  BETA_COST = \
29  Beta('BETA_COST', -0.716192, -10000, 10000, 0, 'BETA_COST' )
30
31
32  ### Definition of variables:
33  # For numerical reasons, it is good practice to scale the data to
34  # that the values of the parameters are around 1.0.
35
36  # The following statements are designed to preprocess the data. It is
37  # like creating a new columns in the data file. This should be
38  # preferred to the statement like
39  # TimePT_scaled = TimePT / 200.0
40  # which will cause the division to be reevaluated again and again,
41  # throuh the iterations. For models taking a long time to estimate, it
42  # may make a significant difference.
43
44  TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )
45  TimeCar_scaled = DefineVariable('TimeCar_scaled', TimeCar / 200 )
46  MarginalCostPT_scaled = DefineVariable('MarginalCostPT_scaled',
47                                         MarginalCostPT / 10 )
48  CostCarCHF_scaled = DefineVariable('CostCarCHF_scaled',
49                                     CostCarCHF / 10 )
50  distance_km_scaled = DefineVariable('distance_km_scaled',
51                                     distance_km / 5 )
52
53  male = DefineVariable('male', Gender == 1)
54  female = DefineVariable('female', Gender == 2)
55  unreportedGender = DefineVariable('unreportedGender', Gender == -1)
56
57  fulltime = DefineVariable('fulltime', OccupStat == 1)
58  notfulltime = DefineVariable('notfulltime', OccupStat != 1)
59
60  ### Definition of utility functions:
61  V_PT = BETA_TIME_FULLTIME * TimePT_scaled * fulltime + \
62         BETA_TIME_OTHER * TimePT_scaled * notfulltime + \
63         BETA_COST * MarginalCostPT_scaled
64  V_CAR = ASC_CAR + \
65         BETA_TIME_FULLTIME * TimeCar_scaled * fulltime + \
66         BETA_TIME_OTHER * TimeCar_scaled * notfulltime + \
67         BETA_COST * CostCarCHF_scaled
68  V_SM = ASC_SM + \
69         BETA_DIST_MALE * distance_km_scaled * male + \

```

```

70         BETA_DIST_FEMALE * distance_km_scaled * female + \
71         BETA_DIST_UNREPORTED * distance_km_scaled * unreportedGender
72
73
74 # Associate utility functions with the numbering of alternatives
75 V = {0: V_PT,
76      1: V_CAR,
77      2: V_SM}
78
79 # Associate the availability conditions with the alternatives.
80 # In this example all alternatives are available for each individual.
81 av = {0: 1,
82      1: 1,
83      2: 1}
84
85 ##### DEFINITION OF THE NESTS:
86 # 1: nests parameter
87 # 2: list of alternatives
88
89 NEST_NOCAR = Beta('NEST_NOCAR',1.52853,1,10,0,'NEST_NOCAR' )
90
91
92 CAR = 1.0 , [ 1]
93 NO_CAR = NEST_NOCAR , [ 0, 2]
94 nests = CAR, NO_CAR
95
96 # All observations verifying the following expression will not be
97 # considered for estimation
98 exclude = (Choice == -1)
99 BIOGEME.OBJECT.EXCLUDE = exclude
100
101 ##
102 ## This has been copied-pasted from the file 01nestedEstimation-param.py
103 ##
104 ## Code for the sensitivity analysis generated after the estimation of the model
105 names = ['ASC_CAR','ASC_SM','BETA_COST','BETA_DIST_FEMALE','BETA_DIST_MALE','BETA_DIST_UNREPORTED']
106 values = [[0.0100225,-0.0023271,0.00151986,0.00285251,0.00621963,0.00247439,0.02359]
107 vc = bioMatrix(9,names,values)
108 BIOGEME.OBJECT.VARCOVAR = vc
109
110
111
112 # The choice model is a nested logit
113 prob_pt = nested(V,av,nests,0)
114 prob_car = nested(V,av,nests,1)
115 prob_sm = nested(V,av,nests,2)
116
117 # Defines an iterator on the data
118 rowIterator('obsIter')

```

```

119
120 #Statistics
121 nullLoglikelihood(av,'obsIter')
122 choiceSet = [0,1,2]
123 cteLoglikelihood(choiceSet,Choice,'obsIter')
124 availabilityStatistics(av,'obsIter')
125
126 # Each weight is normalized so that the sum of weights is equal to the
127 # number of entries (1906).
128 # The normalization factor has been calculated during estimation
129 theWeight = Weight * 1906 / 0.814484
130
131
132 BIOGEME.OBJECT.STATISTICS['Gender: males'] = \
133     Sum(male,'obsIter')
134 BIOGEME.OBJECT.STATISTICS['Gender: females'] = \
135     Sum(female,'obsIter')
136 BIOGEME.OBJECT.STATISTICS['Gender: unreported'] = \
137     Sum(unreportedGender,'obsIter')
138 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = \
139     Sum(fulltime,'obsIter')
140 BIOGEME.OBJECT.STATISTICS['Sum of weights'] = \
141     Sum(Weight,'obsIter')
142 BIOGEME.OBJECT.STATISTICS['Number of entries'] = \
143     Sum(1-exclude,'obsIter')
144 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities PT'] = \
145     Sum(theWeight * prob_pt,'obsIter')
146 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities CAR'] = \
147     Sum(theWeight * prob_car,'obsIter')
148 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities SM'] = \
149     Sum(theWeight * prob_sm,'obsIter')
150
151 # Define the dictionary for the simulation.
152 simulate = {'Prob. car': prob_car,
153            'Prob. public transportation': prob_pt,
154            'Prob. slow modes': prob_sm,
155            'Revenue public transportation':
156            prob_pt * MarginalCostPT}
157
158 BIOGEME.OBJECT.WEIGHT = theWeight
159 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

```

A.3 03 nestedElasticities .py

```

1 ## File 03nestedElasticities.py
2 ## Simple nested logit model for the Optima case study
3 ## Calculation of direct point elasticities
4 ## Wed May 10 12:20:59 2017
5

```

```

6 from biogeme import *
7 from headers import *
8 from statistics import *
9 from nested import *
10
11 ### Three alternatives:
12 # CAR: automobile
13 # PT: public transportation
14 # SM: slow mode (walking, biking)
15
16 ### List of parameters and their estimated value.
17 ASC_CAR = Beta('ASC_CAR', 0.261291, -10000, 10000, 0, 'ASC_CAR' )
18 ASC_SM = Beta('ASC_SM', 0.0590204, -10000, 10000, 0, 'ASC_SM' )
19 BETA.TIME.FULLTIME = \
20   Beta('BETA_TIME_FULLTIME', -1.59709, -10000, 10000, 0, 'BETA_TIME_FULLTIME' )
21 BETA.TIME.OTHER = \
22   Beta('BETA_TIME_OTHER', -0.577362, -10000, 10000, 0, 'BETA_TIME_OTHER' )
23 BETA.DIST.MALE = \
24   Beta('BETA_DIST_MALE', -0.686327, -10000, 10000, 0, 'BETA_DIST_MALE' )
25 BETA.DIST.FEMALE = \
26   Beta('BETA_DIST_FEMALE', -0.83121, -10000, 10000, 0, 'BETA_DIST_FEMALE' )
27 BETA.DIST.UNREPORTED = \
28   Beta('BETA_DIST_UNREPORTED', -0.702974, -10000, 10000, 0, 'BETA_DIST_UNREPORTED' )
29 BETA.COST = \
30   Beta('BETA_COST', -0.716192, -10000, 10000, 0, 'BETA_COST' )
31
32 ### Definition of variables:
33 # For numerical reasons, it is good practice to scale the data to
34 # that the values of the parameters are around 1.0.
35
36 ### Warning: when calculation derivatives, the total formula must be
37 ### known to Biogeme. In this case, the use of
38 ### "DefineVariable" must be omitted, if the derivatives must be
39 ### calculated with respect to the original variables (as is often the
40 ### case)
41
42 # TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )
43 TimePT_scaled = TimePT / 200
44
45 #TimeCar_scaled = DefineVariable('TimeCar_scaled', TimeCar /
46 200 )
47 TimeCar_scaled = TimeCar / 200
48
49 #MarginalCostPT_scaled = DefineVariable('MarginalCostPT_scaled', MarginalCostPT
50 / 10 )
51 MarginalCostPT_scaled = MarginalCostPT / 10
52
53 #CostCarCHF_scaled = DefineVariable('CostCarCHF_scaled', CostCarCHF
54 / 10 )

```

```

52 CostCarCHF_scaled = CostCarCHF / 10
53
54 #distance_km_scaled = DefineVariable('distance_km_scaled', distance_km
55 / 5 )
56 distance_km_scaled = distance_km / 5
57
58 male = DefineVariable('male', Gender == 1)
59 female = DefineVariable('female', Gender == 2)
60 unreportedGender = DefineVariable('unreportedGender', Gender == -1)
61
62 fulltime = DefineVariable('fulltime', OccupStat == 1)
63 notfulltime = DefineVariable('notfulltime', OccupStat != 1)
64
65 ### Definition of utility functions:
66
67 V_PT = BETA.TIME.FULLTIME * TimePT_scaled * fulltime + \
68       BETA.TIME.OTHER * TimePT_scaled * notfulltime + \
69       BETA.COST * MarginalCostPT_scaled
70
71 V_CAR = ASC_CAR + \
72       BETA.TIME.FULLTIME * TimeCar_scaled * fulltime + \
73       BETA.TIME.OTHER * TimeCar_scaled * notfulltime + \
74       BETA.COST * CostCarCHF_scaled
75
76 V_SM = ASC_SM + \
77       BETA.DIST.MALE * distance_km_scaled * male + \
78       BETA.DIST.FEMALE * distance_km_scaled * female + \
79       BETA.DIST.UNREPORTED * distance_km_scaled * unreportedGender
80
81 # Associate utility functions with the numbering of alternatives
82 V = {0: V_PT,
83      1: V_CAR,
84      2: V_SM}
85
86 # Associate the availability conditions with the alternatives.
87 # In this example all alternatives are available for each individual.
88 av = {0: 1,
89       1: 1,
90       2: 1}
91
92 ### DEFINITION OF THE NESTS:
93 # 1: nests parameter
94 # 2: list of alternatives
95
96 NEST_NOCAR = Beta('NEST_NOCAR', 1.52853, 1, 10, 0, 'NEST_NOCAR' )
97
98 CAR = 1.0 , [ 1]
99 NO_CAR = NEST_NOCAR , [ 0, 2]
100 nests = CAR, NO_CAR

```



```

100 # All observations verifying the following expression will not be
101 # considered for estimation
102 exclude = (Choice == -1)
103 BIOGEME.OBJECT.EXCLUDE = exclude
104
105
106 ##
107 ## This has been copied-pasted from the file 0lnestedEstimation-param.py
108 ##
109 ## Code for the sensitivity analysis generated after the estimation of the model
110 names = ['ASC_CAR', 'ASC_SM', 'BETA_COST', 'BETA_DIST_FEMALE', 'BETA_DIST_MALE', 'BETA_I
111 values = [[0.0100225, -0.0023271, 0.00151986, 0.00285251, 0.00621963, 0.00247439, 0.02359
112 vc = bioMatrix(9, names, values)
113 #BIOGEME.OBJECT.VARCOVAR = vc
114
115
116
117 # The choice model is a nested logit
118 prob_pt = nested(V, av, nests, 0)
119 prob_car = nested(V, av, nests, 1)
120 prob_sm = nested(V, av, nests, 2)
121
122 elas_pt_time = Derive(prob_pt, 'TimePT') * TimePT / prob_pt
123 elas_pt_cost = Derive(prob_pt, 'MarginalCostPT') * MarginalCostPT / prob_pt
124 elas_car_time = Derive(prob_car, 'TimeCar') * TimeCar / prob_car
125 elas_car_cost = Derive(prob_car, 'CostCarCHF') * CostCarCHF / prob_car
126 elas_sm_dist = Derive(prob_sm, 'distance_km') * distance_km / prob_sm
127
128 # Defines an iterator on the data
129 rowIterator('obsIter')
130 #Statistics
131 nullLoglikelihood(av, 'obsIter')
132 choiceSet = [0, 1, 2]
133 cteLoglikelihood(choiceSet, Choice, 'obsIter')
134 availabilityStatistics(av, 'obsIter')
135
136 # Each weight is normalized so that the sum of weights is equal to the
137 # number of entries (1906)
138 # The normalization factor has been calculated during estimation
139
140 theWeight = Weight * 1906 / 0.814484
141 normalization_pt = 535.086
142 normalization_car = 1244.77
143 normalization_sm = 126.147
144
145 BIOGEME.OBJECT.STATISTICS['Gender: males'] = \
146     Sum(male, 'obsIter')
147 BIOGEME.OBJECT.STATISTICS['Gender: females'] = \
148     Sum(female, 'obsIter')

```

```

149 BIOGEME.OBJECT.STATISTICS['Gender: unreported'] = \
150     Sum(unreportedGender,'obsIter')
151 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = \
152     Sum(fulltime,'obsIter')
153 BIOGEME.OBJECT.STATISTICS['Sum of weights'] = \
154     Sum(Weight,'obsIter')
155 BIOGEME.OBJECT.STATISTICS['Number of entries'] = \
156     Sum(1-exclude,'obsIter')
157 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities PT'] = \
158     Sum(theWeight * prob_pt,'obsIter')
159 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities CAR'] = \
160     Sum(theWeight * prob_car,'obsIter')
161 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities SM'] = \
162     Sum(theWeight * prob_sm,'obsIter')
163 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = Sum(fulltime,'obsIter')
164
165 # Define the dictionary for the simulation.
166 simulate = {'Disag. Elast. PT - Time': elas_pt_time,
167             'Disag. Elast. PT - Cost': elas_pt_cost,
168             'Disag. Elast. Car - Time': elas_car_time,
169             'Disag. Elast. Car - Cost': elas_car_cost,
170             'Disag. Elast. Slow modes - Distance': elas_sm_dist,
171             'Agg. Elast. PT - Time': elas_pt_time * prob_pt / normalization_pt,
172             'Agg. Elast. PT - Cost': elas_pt_cost * prob_pt / normalization_pt,
173             'Agg. Elast. Car - Time': elas_car_time * prob_car / normalization_car,
174             'Agg. Elast. Car - Cost': elas_car_cost * prob_car / normalization_car,
175             'Agg. Elast. Slow modes - Distance': elas_sm_dist * prob_sm / normaliza
176 }
177
178 BIOGEME.OBJECT.WEIGHT = theWeight
179 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

```

A.4 04 nestedElasticities .py

```

1 ## File 04nestedElasticities.py
2 ## Simple nested logit model for the Optima case study
3 ## Calculation of cross point elasticities
4 ## Thu May 11 16:38:05 2017
5
6 from biogeme import *
7 from headers import *
8 from statistics import *
9 from nested import *
10
11 ### Three alternatives:
12 # CAR: automobile
13 # PT: public transportation
14 # SM: slow mode (walking, biking)
15

```

```

16 ### List of parameters and their estimated value.
17 ASC_CAR = Beta('ASC_CAR',0.261291,-10000,10000,0,'ASC_CAR' )
18 ASC_SM = Beta('ASC_SM',0.0590204,-10000,10000,0,'ASC_SM' )
19 BETA_TIME_FULLTIME = \
20   Beta('BETA_TIME_FULLTIME',-1.59709,-10000,10000,0,'BETA_TIME_FULLTIME' )
21 BETA_TIME_OTHER = \
22   Beta('BETA_TIME_OTHER',-0.577362,-10000,10000,0,'BETA_TIME_OTHER' )
23 BETA_DIST_MALE = \
24   Beta('BETA_DIST_MALE',-0.686327,-10000,10000,0,'BETA_DIST_MALE' )
25 BETA_DIST_FEMALE = \
26   Beta('BETA_DIST_FEMALE',-0.83121,-10000,10000,0,'BETA_DIST_FEMALE' )
27 BETA_DIST_UNREPORTED = \
28   Beta('BETA_DIST_UNREPORTED',-0.702974,-10000,10000,0,'BETA_DIST_UNREPORTED' )
29 BETA_COST = \
30   Beta('BETA_COST',-0.716192,-10000,10000,0,'BETA_COST' )
31
32
33 ###Definition of variables:
34 # For numerical reasons, it is good practice to scale the data to
35 # that the values of the parameters are around 1.0.
36
37 ### Warning: when calculation derivatives, the total formula must be
38 ### known to Biogeme. In this case, the use of
39 ### "DefineVariable" must be omitted, if the derivatives must be
40 ### calculated with respect to the original variables (as is often the
41 ### case)
42
43 # TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )
44 TimePT_scaled = TimePT / 200
45
46 #TimeCar_scaled = DefineVariable('TimeCar_scaled', TimeCar /
47 200 )
48 TimeCar_scaled = TimeCar / 200
49
50 #MarginalCostPT_scaled = DefineVariable('MarginalCostPT_scaled', MarginalCostPT
51 / 10 )
52 MarginalCostPT_scaled = MarginalCostPT / 10
53
54 #CostCarCHF_scaled = DefineVariable('CostCarCHF_scaled', CostCarCHF
55 / 10 )
56 CostCarCHF_scaled = CostCarCHF / 10
57
58 #distance_km_scaled = DefineVariable('distance_km_scaled', distance_km
59 / 5 )
60 distance_km_scaled = distance_km / 5
61
62 male = DefineVariable('male',Gender == 1)
63 female = DefineVariable('female',Gender == 2)
64 unreportedGender = DefineVariable('unreportedGender',Gender == -1)

```

```

61
62 fulltime = DefineVariable('fulltime',OccupStat == 1)
63 notfulltime = DefineVariable('notfulltime',OccupStat != 1)
64
65 ### Definition of utility functions:
66
67 V_PT = BETA.TIME_FULLTIME * TimePT_scaled * fulltime + \
68         BETA.TIME_OTHER * TimePT_scaled * notfulltime + \
69         BETA.COST * MarginalCostPT_scaled
70 V_CAR = ASC_CAR + \
71         BETA.TIME_FULLTIME * TimeCar_scaled * fulltime + \
72         BETA.TIME_OTHER * TimeCar_scaled * notfulltime + \
73         BETA.COST * CostCarCHF_scaled
74 V_SM = ASC_SM + \
75         BETA.DIST_MALE * distance_km_scaled * male + \
76         BETA.DIST_FEMALE * distance_km_scaled * female + \
77         BETA.DIST_UNREPORTED * distance_km_scaled * unreportedGender
78
79 # Associate utility functions with the numbering of alternatives
80 V = {0: V_PT,
81      1: V_CAR,
82      2: V_SM}
83
84 # Associate the availability conditions with the alternatives.
85 # In this example all alternatives are available for each individual.
86 av = {0: 1,
87       1: 1,
88       2: 1}
89
90 ### DEFINITION OF THE NESTS:
91 # 1: nests parameter
92 # 2: list of alternatives
93
94 NEST_NOCAR = Beta('NEST_NOCAR',1.52853,1,10,0,'NEST_NOCAR' )
95
96
97 CAR = 1.0 , [ 1]
98 NO_CAR = NEST_NOCAR , [ 0, 2]
99 nests = CAR, NO_CAR
100
101 # All observations verifying the following expression will not be
102 # considered for estimation
103 exclude = (Choice == -1)
104 BIOGEME.OBJECT.EXCLUDE = exclude
105
106
107 ##
108 ## This has been copied-pasted from the file 01nestedEstimation-param.py
109 ##

```

```

110 ## Code for the sensitivity analysis generated after the estimation of the model
111 names = ['ASC_CAR', 'ASC_SM', 'BETA_COST', 'BETA_DIST_FEMALE', 'BETA_DIST_MALE', 'BETA_I
112 values = [[0.0100225, -0.0023271, 0.00151986, 0.00285251, 0.00621963, 0.00247439, 0.02359
113 vc = bioMatrix(9, names, values)
114 #BIOGEME.OBJECT.VARCOVAR = vc
115
116
117
118 # The choice model is a nested logit
119 prob_pt = nested(V, av, nests, 0)
120 prob_car = nested(V, av, nests, 1)
121 prob_sm = nested(V, av, nests, 2)
122
123 elas_car_cost = Derive(prob_car, 'MarginalCostPT') * MarginalCostPT / prob_car
124 elas_car_time = Derive(prob_car, 'TimePT') * TimePT / prob_car
125 elas_pt_cost = Derive(prob_pt, 'CostCarCHF') * CostCarCHF / prob_pt
126 elas_pt_time = Derive(prob_pt, 'TimeCar') * TimeCar / prob_pt
127
128 # Defines an itertor on the data
129 rowIterator('obsIter')
130 #Statistics
131 nullLoglikelihood(av, 'obsIter')
132 choiceSet = [0, 1, 2]
133 cteLoglikelihood(choiceSet, Choice, 'obsIter')
134 availabilityStatistics(av, 'obsIter')
135
136 theWeight = Weight * 1906 / 0.814484
137 normalization_pt = 535.086
138 normalization_car = 1244.77
139 normalization_sm = 126.147
140
141 BIOGEME.OBJECT.STATISTICS['Gender: males'] = \
142     Sum(male, 'obsIter')
143 BIOGEME.OBJECT.STATISTICS['Gender: females'] = \
144     Sum(female, 'obsIter')
145 BIOGEME.OBJECT.STATISTICS['Gender: unreported'] = \
146     Sum(unreportedGender, 'obsIter')
147 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = \
148     Sum(fulltime, 'obsIter')
149 BIOGEME.OBJECT.STATISTICS['Sum of weights'] = \
150     Sum(Weight, 'obsIter')
151 BIOGEME.OBJECT.STATISTICS['Number of entries'] = \
152     Sum(1-exclude, 'obsIter')
153 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities PT'] = \
154     Sum(theWeight * prob_pt, 'obsIter')
155 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities CAR'] = \
156     Sum(theWeight * prob_car, 'obsIter')
157 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities SM'] = \
158     Sum(theWeight * prob_sm, 'obsIter')

```

```

159 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = Sum(fulltime, 'obsIter')
160
161 # Define the dictionary for the simulation.
162 simulate = {'Disag. Elast. PT - Time car': elas_pt_time,
163             'Disag. Elast. PT - Cost car': elas_pt_cost,
164             'Disag. Elast. Car - Time PT': elas_car_time,
165             'Disag. Elast. Car - Cost PT': elas_car_cost,
166             'Agg. Elast. Car - Cost PT': elas_car_cost * probab_car / normalization_c,
167             'Agg. Elast. Car - Time PT': elas_car_time * probab_car / normalization_c,
168             'Agg. Elast. PT - Cost car': elas_pt_cost * probab_pt / normalization_pt,
169             'Agg. Elast. PT - Time car': elas_pt_time * probab_pt / normalization_pt,
170
171 # Each weight is normalized so that the sum of weights is equal to the numer of en
172 BIOGEME.OBJECT.WEIGHT = theWeight
173 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')

```

A.5 05 nestedElasticities .py

```

1  ## File 05nestedElasticities.py
2  ## Simple nested logit model for the Optima case study
3  ## Calculation of direct arc elasticities
4  ## Thu May 11 16:38:05 2017
5
6  from biogeme import *
7  from headers import *
8  from statistics import *
9  from nested import *
10
11  ### Three alternatives:
12  # CAR: automobile
13  # PT: public transportation
14  # SM: slow mode (walking, biking)
15
16  ### List of parameters and their estimated value.
17  ASC_CAR = Beta('ASC_CAR', 0.261291, -10000, 10000, 0, 'ASC_CAR' )
18  ASC_SM = Beta('ASC_SM', 0.0590204, -10000, 10000, 0, 'ASC_SM' )
19  BETA.TIME.FULLTIME = \
20    Beta('BETA.TIME.FULLTIME', -1.59709, -10000, 10000, 0, 'BETA.TIME.FULLTIME' )
21  BETA.TIME.OTHER = \
22    Beta('BETA.TIME.OTHER', -0.577362, -10000, 10000, 0, 'BETA.TIME.OTHER' )
23  BETA.DIST.MALE = \
24    Beta('BETA.DIST.MALE', -0.686327, -10000, 10000, 0, 'BETA.DIST.MALE' )
25  BETA.DIST.FEMALE = \
26    Beta('BETA.DIST.FEMALE', -0.83121, -10000, 10000, 0, 'BETA.DIST.FEMALE' )
27  BETA.DIST.UNREPORTED = \
28    Beta('BETA.DIST.UNREPORTED', -0.702974, -10000, 10000, 0, 'BETA.DIST.UNREPORTED' )
29  BETA.COST = \
30    Beta('BETA.COST', -0.716192, -10000, 10000, 0, 'BETA.COST' )
31

```

```

32 ###Definition of variables:
33 # For numerical reasons, it is good practice to scale the data to
34 # that the values of the parameters are around 1.0.
35
36 ### Warning: when calculation derivatives, the total formula must be
37 ### known to Biogeme. In this case, the use of
38 ### "DefineVariable" must be omitted, if the derivatives must be
39 ### calculated with respect to the original variables (as is often the
40 ### case)
41
42 delta_dist = 1
43
44 # TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )
45 TimePT_scaled = TimePT / 200
46
47 #TimeCar_scaled = DefineVariable('TimeCar_scaled', TimeCar /
48 200 )
49 TimeCar_scaled = TimeCar / 200
50
51 #MarginalCostPT_scaled = DefineVariable('MarginalCostPT_scaled', MarginalCostPT
52 / 10 )
53 MarginalCostPT_scaled = MarginalCostPT / 10
54
55
56 #CostCarCHF_scaled = DefineVariable('CostCarCHF_scaled', CostCarCHF
57 / 10 )
58 CostCarCHF_scaled = CostCarCHF / 10
59
60 #distance-km_scaled = DefineVariable('distance-km_scaled', distance-km
61 / 5 )
62 distance_km_scaled = distance_km / 5
63 distance_km_scaled_after = (distance_km + delta_dist) / 5
64
65 male = DefineVariable('male', Gender == 1)
66 female = DefineVariable('female', Gender == 2)
67 unreportedGender = DefineVariable('unreportedGender', Gender == -1)
68
69 fulltime = DefineVariable('fulltime', OccupStat == 1)
70 notfulltime = DefineVariable('notfulltime', OccupStat != 1)
71
72 ### Definition of utility functions:
73
74 V_PT = BETA.TIME.FULLTIME * TimePT_scaled * fulltime + \
75         BETA.TIME.OTHER * TimePT_scaled * notfulltime + \
76         BETA.COST * MarginalCostPT_scaled
77
78 V_CAR = ASC_CAR + \
79         BETA.TIME.FULLTIME * TimeCar_scaled * fulltime + \
80         BETA.TIME.OTHER * TimeCar_scaled * notfulltime + \
81         BETA.COST * CostCarCHF_scaled
82
83 V_SM = ASC_SM + \

```

```

77     BETA_DIST_MALE * distance_km_scaled * male + \
78     BETA_DIST_FEMALE * distance_km_scaled * female + \
79     BETA_DIST_UNREPORTED * distance_km_scaled * unreportedGender
80
81 V_SM_after = ASC_SM + \
82     BETA_DIST_MALE * distance_km_scaled_after * male + \
83     BETA_DIST_FEMALE * distance_km_scaled_after * female + \
84     BETA_DIST_UNREPORTED * distance_km_scaled_after * unreportedGender
85
86
87 # Associate utility functions with the numbering of alternatives
88 V = {0: V_PT,
89      1: V_CAR,
90      2: V_SM}
91
92 V_after = {0: V_PT,
93            1: V_CAR,
94            2: V_SM_after}
95
96 # Associate the availability conditions with the alternatives.
97 # In this example all alternatives are available for each individual.
98 av = {0: one,
99       1: one,
100      2: one}
101
102 ### DEFINITION OF THE NESTS:
103 # 1: nests parameter
104 # 2: list of alternatives
105
106 NEST_NOCAR = Beta('NEST_NOCAR',1.52853,1,10,0,'NEST_NOCAR' )
107
108
109 CAR = 1.0 , [ 1]
110 NO_CAR = NEST_NOCAR , [ 0, 2]
111 nests = CAR, NO_CAR
112
113 # All observations verifying the following expression will not be
114 # considered for estimation
115 exclude = (Choice == -1)
116 BIOGEME_OBJECT.EXCLUDE = exclude
117
118
119 ##
120 ## This has been copied-pasted from the file 01nestedEstimation-param.py
121 ##
122 ## Code for the sensitivity analysis generated after the estimation of the model
123 names = ['ASC_CAR', 'ASC_SM', 'BETA_COST', 'BETA_DIST_FEMALE', 'BETA_DIST_MALE', 'BETA_DIST_UNREPORTED']
124 values = [[0.0100225, -0.0023271, 0.00151986, 0.00285251, 0.00621963, 0.00247439, 0.02359]
125 vc = bioMatrix(9, names, values)

```



```

126 BIOGEME.OBJECT.VARCOVAR = vc
127
128 # The choice model is a nested logit
129 prob_pt = nested(V,av,nests,0)
130 prob_car = nested(V,av,nests,1)
131 prob_sm = nested(V,av,nests,2)
132
133 prob_pt_after = nested(V_after,av,nests,0)
134 prob_car_after = nested(V_after,av,nests,1)
135 prob_sm_after = nested(V_after,av,nests,2)
136
137 elas_sm_dist = (prob_sm_after - prob_sm) * distance_km / (prob_sm * delta_dist)
138
139 # Defines an iterator on the data
140 rowIterator('obsIter')
141 #Statistics
142 nullLoglikelihood(av,'obsIter')
143 choiceSet = [0,1,2]
144 cteLoglikelihood(choiceSet,Choice,'obsIter')
145 availabilityStatistics(av,'obsIter')
146
147 theWeight = Weight * 1906 / 0.814484
148 normalization_pt = 535.086
149 normalization_car = 1244.77
150 normalization_sm = 126.147
151
152 BIOGEME.OBJECT.STATISTICS['Gender: males'] = \
153     Sum(male,'obsIter')
154 BIOGEME.OBJECT.STATISTICS['Gender: females'] = \
155     Sum(female,'obsIter')
156 BIOGEME.OBJECT.STATISTICS['Gender: unreported'] = \
157     Sum(unreportedGender,'obsIter')
158 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = \
159     Sum(fulltime,'obsIter')
160 BIOGEME.OBJECT.STATISTICS['Sum of weights'] = \
161     Sum(Weight,'obsIter')
162 BIOGEME.OBJECT.STATISTICS['Number of entries'] = \
163     Sum(1-exclude,'obsIter')
164 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities PT'] = \
165     Sum(theWeight * prob_pt,'obsIter')
166 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities CAR'] = \
167     Sum(theWeight * prob_car,'obsIter')
168 BIOGEME.OBJECT.STATISTICS['Normalization for elasticities SM'] = \
169     Sum(theWeight * prob_sm,'obsIter')
170 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = Sum(fulltime,'obsIter')
171
172
173 # Define the dictionary for the simulation.
174 simulate = {'Disag. Elast. SM - Distance': elas_sm_dist,

```

```

175         'Agg. Elast. SM - Distance': elas_sm_dist * prob_sm / normalization_sm
176
177 # Each weight is normalized so that the sum of weights is equal to the numer of en
178 BIOGEME.OBJECT.WEIGHT = theWeight
179 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')

```

A.6 06nestedWTP.py

```

1  ## File 06nestedWTP.py
2  ## Simple nested logit model for the Optima case study
3  ## Thu May 11 17:23:04 2017
4
5  from biogeme import *
6  from headers import *
7  from statistics import *
8  from nested import *
9
10 ### Three alternatives:
11 # CAR: automobile
12 # PT: public transportation
13 # SM: slow mode (walking, biking)
14
15 ### List of parameters and their estimated value.
16 ASC_CAR = Beta('ASC_CAR', 0.261291, -10000, 10000, 0, 'ASC_CAR' )
17 ASC_SM = Beta('ASC_SM', 0.0590204, -10000, 10000, 0, 'ASC_SM' )
18 BETA_TIME_FULLTIME = \
19     Beta('BETA_TIME_FULLTIME', -1.59709, -10000, 10000, 0, 'BETA_TIME_FULLTIME' )
20 BETA_TIME_OTHER = \
21     Beta('BETA_TIME_OTHER', -0.577362, -10000, 10000, 0, 'BETA_TIME_OTHER' )
22 BETA_DIST_MALE = \
23     Beta('BETA_DIST_MALE', -0.686327, -10000, 10000, 0, 'BETA_DIST_MALE' )
24 BETA_DIST_FEMALE = \
25     Beta('BETA_DIST_FEMALE', -0.83121, -10000, 10000, 0, 'BETA_DIST_FEMALE' )
26 BETA_DIST_UNREPORTED = \
27     Beta('BETA_DIST_UNREPORTED', -0.702974, -10000, 10000, 0, 'BETA_DIST_UNREPORTED' )
28 BETA_COST = \
29     Beta('BETA_COST', -0.716192, -10000, 10000, 0, 'BETA_COST' )
30
31 ### Definition of variables:
32 # For numerical reasons, it is good practice to scale the data to
33 # that the values of the parameters are around 1.0.
34
35 ### Warning: when calculation derivatives, the total formula must be
36 ### known to Biogeme. In this case, the use of
37 ### "DefineVariable" must be omitted, if the derivatives must be
38 ### calculated with respect to the original variables (as is often the
39 ### case)
40
41 # TimePT_scaled = DefineVariable('TimePT_scaled', TimePT / 200 )

```

```

42 TimePT_scaled = TimePT / 200
43
44 #TimeCar_scaled = DefineVariable('TimeCar_scaled', TimeCar /
45 200 )
46 TimeCar_scaled = TimeCar / 200
47
48 #MarginalCostPT_scaled = DefineVariable('MarginalCostPT_scaled', MarginalCostPT
49 / 10 )
50 MarginalCostPT_scaled = MarginalCostPT / 10
51
52 #CostCarCHF_scaled = DefineVariable('CostCarCHF_scaled', CostCarCHF
53 / 10 )
54 CostCarCHF_scaled = CostCarCHF / 10
55
56 #distance_km_scaled = DefineVariable('distance_km_scaled', distance_km
57 / 5 )
58 distance_km_scaled = distance_km / 5
59
60
61 male = DefineVariable('male', Gender == 1)
62 female = DefineVariable('female', Gender == 2)
63 unreportedGender = DefineVariable('unreportedGender', Gender == -1)
64
65 fulltime = DefineVariable('fulltime', OccupStat == 1)
66 notfulltime = DefineVariable('notfulltime', OccupStat != 1)
67
68 #### Definition of utility functions:
69 V_PT = BETA.TIME.FULLTIME * TimePT_scaled * fulltime + \
70       BETA.TIME.OTHER * TimePT_scaled * notfulltime + \
71       BETA.COST * MarginalCostPT_scaled
72 V_CAR = ASC.CAR + \
73       BETA.TIME.FULLTIME * TimeCar_scaled * fulltime + \
74       BETA.TIME.OTHER * TimeCar_scaled * notfulltime + \
75       BETA.COST * CostCarCHF_scaled
76 V_SM = ASC.SM + \
77       BETA.DIST.MALE * distance_km_scaled * male + \
78       BETA.DIST.FEMALE * distance_km_scaled * female + \
79       BETA.DIST.UNREPORTED * distance_km_scaled * unreportedGender
80
81 # It is advised to use the Derive operator, in order to take care
82 # automatically of the scaled variables.
83
84 WTP_PT.TIME = Derive(V_PT, 'TimePT') / Derive(V_PT, 'MarginalCostPT')
85 WTP_CAR.TIME = Derive(V_CAR, 'TimeCar') / Derive(V_CAR, 'CostCarCHF')
86
87 # All observations verifying the following expression will not be
88 # considered for estimation
89 exclude = (Choice == -1)
90 BIOGEME.OBJECT.EXCLUDE = exclude

```

```

87
88
89 ##
90 ## This has been copied-pasted from the file 0lnestedEstimation_param.py
91 ##
92 ## Code for the sensitivity analysis generated after the estimation of the model
93 names = ['ASC_CAR', 'ASC_SM', 'BETA_COST', 'BETA_DIST_FEMALE', 'BETA_DIST_MALE', 'BETA_I
94 values = [[0.0100225, -0.0023271, 0.00151986, 0.00285251, 0.00621963, 0.00247439, 0.02359
95 vc = bioMatrix(9, names, values)
96 BIOGEME.OBJECT.VARCOVAR = vc
97
98
99 # Defines an itertor on the data
100 rowIterator('obsIter')
101
102 theWeight = Weight * 1906 / 0.814484
103
104
105 BIOGEME.OBJECT.STATISTICS['Gender: males'] = \
106     Sum(male, 'obsIter')
107 BIOGEME.OBJECT.STATISTICS['Gender: females'] = \
108     Sum(female, 'obsIter')
109 BIOGEME.OBJECT.STATISTICS['Gender: unreported'] = \
110     Sum(unreportedGender, 'obsIter')
111 BIOGEME.OBJECT.STATISTICS['Occupation: full time'] = \
112     Sum(fulltime, 'obsIter')
113 BIOGEME.OBJECT.STATISTICS['Sum of weights'] = \
114     Sum(Weight, 'obsIter')
115 BIOGEME.OBJECT.STATISTICS['Number of entries'] = \
116     Sum(1-exclude, 'obsIter')
117
118 simulate = {'PT: Time': TimePT,
119             'PT: Value of time (CHF/min)': WTP_PT.TIME,
120             'PT: Value of time (CHF/h)': 60 * WTP_PT.TIME,
121             'Car: Time': TimeCar,
122             'Car: Value of time (CHF/min)': WTP_CAR.TIME,
123             'Car: Value of time (CHF/h)': 60 * WTP_CAR.TIME,
124             'Male': male,
125             'Full time': fulltime}
126
127 # Each weight is normalized so that the sum of weights is equal to the
128 # number of entries (1906).
129 BIOGEME.OBJECT.WEIGHT = theWeight
130 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')

```

References

- Atasoy, B., Glerum, A. and Bierlaire, M. (2013). Attitudes towards mode choice in switzerland, *disP - The Planning Review* **49**(2): 101–117.
- Bierlaire, M. (2016). Pythonbiogeme: a short introduction, *Technical Report TRANSP-OR 160706*, Transport and Mobility Laboratory, Ecole Polytechnique Fédérale de Lausanne.