

Calculating elasticities with PythonBiogeme

Michel Bierlaire

May 11, 2017

Report TRANSP-OR 170511

Transport and Mobility Laboratory

School of Architecture, Civil and Environmental Engineering

Ecole Polytechnique Fédérale de Lausanne

`transp-or.epfl.ch`

SERIES ON BIOGEME

1 Elasticities

Consider any 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

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

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

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

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 .

1.1 Point elasticities

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

$$\frac{\partial x_{ink}}{x_{ink}} = \frac{\partial x_{ipk}}{x_{ipk}} = \frac{\partial x_{ik}}{x_{ik}}, \quad (3)$$

where

$$x_{ik} = \frac{1}{N} \sum_{n=1}^N x_{ink}. \quad (4)$$

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

$$E_{x_{ink}}^{P_n(i)} = \frac{\partial P_n(i|x_n, C_n)}{\partial x_{ink}} \frac{x_{ink}}{P_n(i|x_n, C_n)}. \quad (5)$$

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

$$E_{x_{ik}}^{W_i} = \frac{\partial W_i}{\partial x_{ik}} \frac{x_{ik}}{W_i}. \quad (6)$$

Using (2), we obtain

$$E_{x_{ik}}^{W_i} = \frac{1}{N} \sum_{n=1}^{N_s} w_n \frac{\partial P_n(i|x_n, C_n)}{\partial x_{ik}} \frac{x_{ik}}{W_i}. \quad (7)$$

From (3), we obtain

$$E_{x_{ik}}^{W_i} = \frac{1}{N} \sum_{n=1}^{N_s} w_n \frac{\partial P_n(i|x_n, C_n)}{\partial x_{ink}} \frac{x_{ink}}{W_i} = \frac{1}{N} \sum_{n=1}^{N_s} w_n E_{x_{ink}}^{P_n(i)} \frac{P_n(i|x_n, C_n)}{W_i}, \quad (8)$$

where the second equation is derived from (5). Using (2) again, we obtain

$$E_{x_{ik}}^{W_i} = \sum_{n=1}^{N_s} E_{x_{ink}}^{P_n(i)} \frac{w_n P_n(i|x_n, C_n)}{\sum_{n=1}^{N_s} w_n P_n(i|x_n, C_n)}. \quad (9)$$

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 $w_n P_n(i|x_n, C_n)$.

1.2 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. As above, we assume that the relative change of the variable is the same for every individual in the population, that is

$$\frac{\Delta x_{ink}}{x_{ink}} = \frac{\Delta x_{ipk}}{x_{ipk}} = \frac{\Delta x_{ik}}{x_{ik}}, \quad (10)$$

where x_{ik} is defined by (4). The *disaggregate direct arc elasticity* of the model with respect to the variable x_{ink} is defined as

$$E_{x_{ink}}^{P_n(i)} = \frac{\Delta P_n(i|x_n, C_n)}{\Delta x_{ink}} \frac{x_{ink}}{P_n(i|x_n, C_n)}. \quad (11)$$

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

$$E_{x_{ik}}^{W_i} = \frac{\Delta W_i}{\Delta x_{ik}} \frac{x_{ik}}{W_i}. \quad (12)$$

The two quantities are also related by (9).

A Complete specification files

A.1 01simpleIntegral.py

```
1 #####
2 #
3 # File: 01simpleIntegral.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Sat Jul 25 11:41:13 2015
6 #
7 #####
8
9 from biogeme import *
10 from headers import *
11
12 integrand = exp(bioDraws('U'))
13 simulatedI = MonteCarlo(integrand)
14
15 trueI = exp(1.0) - 1.0
16
17 sampleVariance = \
18     MonteCarlo(integrand*integrand) - simulatedI * simulatedI
19 stderr = (sampleVariance / 200000.0)**0.5
20 error = simulatedI - trueI
21
22 simulate = {'01 Simulated Integral': simulatedI,
23             '02 Analytical Integral': trueI,
24             '03 Sample variance': sampleVariance,
25             '04 Std Error': stderr,
26             '05 Error': error}
27
28 rowIterator('obsIter')
29
30 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')
31 BIOGEME.OBJECT.PARAMETERS['NbrOfDraws'] = "5"
32 BIOGEME.OBJECT.PARAMETERS['RandomDistribution'] = "PSEUDO"
33 __rowId__ = Variable('__rowId__')
34 BIOGEME.OBJECT.EXCLUDE = __rowId__ >= 1
35 BIOGEME.OBJECT.DRAWS = { 'U': 'UNIFORM' }
```

A.2 02antithetic.py

```
1 #####
2 #
3 # File: 02antithetic.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Sat Jul 25 12:21:10 2015
6 #
```

```

7 #####
8
9 from biogeme import *
10 from headers import *
11
12 integrand = 0.5 * (exp(bioDraws('U')) + exp(1.0-bioDraws('U')))
13 simulatedI = MonteCarlo(integrand)
14
15 trueI = exp(1.0) - 1.0
16
17 sampleVariance = \
18     MonteCarlo(integrand*integrand) - simulatedI * simulatedI
19 stderr = (sampleVariance / 10000.0)**0.5
20 error = simulatedI - trueI
21
22 simulate = {'01_Simulated Integral': simulatedI,
23             '02_Analytical Integral': trueI,
24             '03_Sample variance': sampleVariance,
25             '04_Std Error': stderr,
26             '05_Error': error}
27
28 rowIterator('obsIter')
29
30 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')
31
32 BIOGEME.OBJECT.PARAMETERS['NbrOfDraws'] = "5"
33 __rowId__ = Variable('__rowId__')
34 BIOGEME.OBJECT.EXCLUDE = __rowId__ >= 1
35 BIOGEME.OBJECT.DRAWS = { 'U': 'UNIFORM' }

```

A.3 03controlVariate.py

```

1 #####
2 #
3 # File: 03controlVariate.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Sat Jul 25 12:24:25 2015
6 #
7 #####
8 #
9
10 from biogeme import *
11 from headers import *
12
13 integrand = exp(bioDraws('U'))
14 simulatedI = MonteCarloControlVariate(integrand, bioDraws('U'), 0.5)
15
16 trueI = exp(1.0) - 1.0
17

```

```

18 error = simulatedI - trueI
19
20 simulate = {'01_Simulated Integral': simulatedI,
21             '02_Analytical Integral': trueI,
22             '05_Error': error}
23
24 rowIterator('obsIter')
25
26 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')
27
28 BIOGEME.OBJECT.PARAMETERS['NbrOfDraws'] = "5"
29 __rowId__ = Variable('__rowId__')
30 BIOGEME.OBJECT.EXCLUDE = __rowId__ >= 1
31 BIOGEME.OBJECT.DRAWS = { 'U': 'UNIFORM' }

```

A.4 05normalMixtureTrueAnalytical.py

```

1 #####
2 #
3 # File: 05normalMixtureTrueAnalytical.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Sat Jul 25 18:50:11 2015
6 #
7 #####
8
9 from biogeme import *
10 from headers import *
11 from distributions import *
12 from loglikelihood import *
13
14 #Parameters
15 ASC_CAR = 0.137
16 ASC_TRAIN = -0.402
17 ASC_SM = 0
18 B_TIME = -2.26
19 B_TIME_S = 1.66
20 B_COST = -1.29
21
22 # Define a random parameter, normally distributed,
23 # designed to be used for integration
24 omega = RandomVariable('omega')
25 density = normalpdf(omega)
26 B_TIME_RND = B_TIME + B_TIME_S * omega
27
28 # Utility functions
29
30 #If the person has a GA (season ticket) her
31 #incremental cost is actually 0
32 #rather than the cost value gathered from the

```

```

33 # network data.
34 SMCOST = SMLCO * ( GA == 0 )
35 TRAIN_COST = TRAIN_CO * ( GA == 0 )
36
37 # For numerical reasons, it is good practice to scale the data to
38 # that the values of the parameters are around 1.0.
39 # A previous estimation with the unscaled data has generated
40 # parameters around -0.01 for both cost and time.
41 # Therefore, time and cost are multiplied by 0.01.
42
43 TRAIN_TT_SCALED = \
44     DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
45 TRAIN_COST_SCALED = \
46     DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
47 SMLTT_SCALED = DefineVariable('SMLTT_SCALED', SMLTT / 100.0)
48 SMLCOST_SCALED = DefineVariable('SMLCOST_SCALED', SMLCOST / 100)
49 CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
50 CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)
51
52 V1 = ASC_TRAIN + \
53     B_TIME_RND * TRAIN_TT_SCALED + \
54     B_COST * TRAIN_COST_SCALED
55 V2 = ASC_SM + \
56     B_TIME_RND * SMLTT_SCALED + \
57     B_COST * SMLCOST_SCALED
58 V3 = ASC_CAR + \
59     B_TIME_RND * CAR_TT_SCALED + \
60     B_COST * CAR_CO_SCALED
61
62
63 # Associate utility functions with the numbering of alternatives
64 V = {1: V1,
65      2: V2,
66      3: V3}
67
68 # Associate the availability conditions with the alternatives
69
70 CAR_AV_SP = DefineVariable('CAR_AV_SP', CAR_AV * ( SP != 0
71 ))
72 TRAIN_AV_SP = DefineVariable('TRAIN_AV_SP', TRAIN_AV * ( SP !=
73 0 ))
74
75 av = {1: TRAIN_AV_SP,
76      2: SMLAV,
77      3: CAR_AV_SP}
78
79 # The choice model is a logit, with availability conditions
80 integrand = bioLogit(V, av, CHOICE)

```



```

80
81 analyticalI = Integrate(integrand*density,'omega')
82 simulate = {'Analytical': analyticalI}
83
84 rowIterator('obsIter')
85
86 BIOGEME.OBJECT.PARAMETERS['decimalPrecisionForSimulation'] = "12"
87 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate,'obsIter')
88
89 __rowId__ = Variable('__rowId__')
90 BIOGEME.OBJECT.EXCLUDE = __rowId__ >= 1

```

A.5 06normalMixture.py

```

1 #####
2 #
3 # File: 06normalMixture.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Sat Jul 25 18:37:37 2015
6 #
7 #####
8
9 from biogeme import *
10 from headers import *
11 from loglikelihood import *
12 from statistics import *
13
14 #Parameters
15 ASC_CAR = 0.137
16 ASC_TRAIN = -0.402
17 ASC_SM = 0
18 B_TIME = -2.26
19 B_TIME_S = 1.66
20 B_COST = -1.29
21
22 # Define a random parameter, normally distributed,
23 # designed to be used for integration
24 omega = bioDraws('B_TIME_RND')
25 B_TIME_RND = B_TIME + B_TIME_S * omega
26
27 # Utility functions
28
29 #If the person has a GA (season ticket) her
30 #incremental cost is actually 0
31 #rather than the cost value gathered from the
32 # network data.
33 SML_COST = SML_CO * ( GA == 0 )
34 TRAIN_COST = TRAIN_CO * ( GA == 0 )
35

```

```

36 # For numerical reasons, it is good practice to scale the data to
37 # that the values of the parameters are around 1.0.
38 # A previous estimation with the unscaled data has generated
39 # parameters around -0.01 for both cost and time. Therefore, time and
40 # cost are multiplied by 0.01.
41
42 TRAIN_TT_SCALED = \
43     DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
44 TRAIN_COST_SCALED = \
45     DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
46 SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
47 SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
48 CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
49 CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)
50
51 V1 = ASC_TRAIN + \
52     B_TIME_RND * TRAIN_TT_SCALED + \
53     B_COST * TRAIN_COST_SCALED
54 V2 = ASC_SM + \
55     B_TIME_RND * SM_TT_SCALED + \
56     B_COST * SM_COST_SCALED
57 V3 = ASC_CAR + \
58     B_TIME_RND * CAR_TT_SCALED + \
59     B_COST * CAR_CO_SCALED
60
61 # Associate utility functions with the numbering of alternatives
62 V = {1: V1,
63      2: V2,
64      3: V3}
65
66 # Associate the availability conditions with the alternatives
67
68 CAR_AV_SP = DefineVariable('CAR_AV_SP', CAR_AV * ( SP != 0
69 ))
70 TRAIN_AV_SP = DefineVariable('TRAIN_AV_SP', TRAIN_AV * ( SP !=
71 0 ))
72
73 av = {1: TRAIN_AV_SP,
74       2: SMAV,
75       3: CAR_AV_SP}
76
77 # The choice model is a logit, with availability conditions
78 integrand = bioLogit(V, av, CHOICE)
79 simulatedI = MonteCarlo(integrand)
80
81 trueI = 0.637849835578
82
83 sampleVariance = \
84     MonteCarlo(integrand*integrand) - simulatedI * simulatedI

```

```

83 stderr = (sampleVariance / 200000.0)**0.5
84 error = simulatedI - trueI
85
86 simulate = {'01 Simulated Integral': simulatedI,
87             '02 Analytical Integral': trueI,
88             '03 Sample variance': sampleVariance,
89             '04 Std Error': stderr,
90             '05 Error': error}
91
92 rowIterator('obsIter')
93
94 BIOGEME_OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')
95
96 __rowId__ = Variable('__rowId__')
97 BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1
98
99 BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
100 BIOGEME_OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }

```

A.6 07normalMixtureAntithetic.py

```

1 #####
2 #
3 # File: 07normalMixtureAntithetic.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Sat Jul 25 19:14:42 2015
6 #
7 #####
8
9 from biogeme import *
10 from headers import *
11 from loglikelihood import *
12 from statistics import *
13
14 #Parameters
15 ASC_CAR = 0.137
16 ASC_TRAIN = -0.402
17 ASC_SM = 0
18 B_TIME = -2.26
19 B_TIME_S = 1.66
20 B_COST = -1.29
21
22 # Define a random parameter, normally distributed,
23 # designed to be used for integration,
24 # and its antithetic.
25 B_TIME_RND = B_TIME + B_TIME_S * bioDraws('B_TIME_RND')
26 B_TIME_RND_MINUS = B_TIME - B_TIME_S * bioDraws('B_TIME_RND')
27
28 # Utility functions

```

```

29
30 #If the person has a GA (season ticket) her
31 #incremental cost is actually 0
32 #rather than the cost value gathered from the
33 # network data.
34  $SM\_COST = SM\_CO * (GA == 0)$ 
35  $TRAIN\_COST = TRAIN\_CO * (GA == 0)$ 
36
37 # For numerical reasons, it is good practice to scale the data to
38 # that the values of the parameters are around 1.0.
39 # A previous estimation with the unscaled data has generated
40 # parameters around -0.01 for both cost and time.
41 # Therefore, time and cost are multiplied by 0.01.
42
43  $TRAIN\_TT\_SCALED = \backslash$ 
44    $DefineVariable('TRAIN\_TT\_SCALED', TRAIN\_TT / 100.0)$ 
45  $TRAIN\_COST\_SCALED = \backslash$ 
46    $DefineVariable('TRAIN\_COST\_SCALED', TRAIN\_COST / 100)$ 
47  $SM\_TT\_SCALED = DefineVariable('SM\_TT\_SCALED', SM\_TT / 100.0)$ 
48  $SM\_COST\_SCALED = DefineVariable('SM\_COST\_SCALED', SM\_COST / 100)$ 
49  $CAR\_TT\_SCALED = DefineVariable('CAR\_TT\_SCALED', CAR\_TT / 100)$ 
50  $CAR\_CO\_SCALED = DefineVariable('CAR\_CO\_SCALED', CAR\_CO / 100)$ 
51
52  $V1 = ASC\_TRAIN + \backslash$ 
53    $B\_TIME\_RND * TRAIN\_TT\_SCALED + \backslash$ 
54    $B\_COST * TRAIN\_COST\_SCALED$ 
55  $V2 = ASC\_SM + \backslash$ 
56    $B\_TIME\_RND * SM\_TT\_SCALED + \backslash$ 
57    $B\_COST * SM\_COST\_SCALED$ 
58  $V3 = ASC\_CAR + \backslash$ 
59    $B\_TIME\_RND * CAR\_TT\_SCALED + \backslash$ 
60    $B\_COST * CAR\_CO\_SCALED$ 
61
62  $V1\_MINUS = ASC\_TRAIN + \backslash$ 
63    $B\_TIME\_RND\_MINUS * TRAIN\_TT\_SCALED + \backslash$ 
64    $B\_COST * TRAIN\_COST\_SCALED$ 
65  $V2\_MINUS = ASC\_SM + \backslash$ 
66    $B\_TIME\_RND\_MINUS * SM\_TT\_SCALED + \backslash$ 
67    $B\_COST * SM\_COST\_SCALED$ 
68  $V3\_MINUS = ASC\_CAR + \backslash$ 
69    $B\_TIME\_RND\_MINUS * CAR\_TT\_SCALED + \backslash$ 
70    $B\_COST * CAR\_CO\_SCALED$ 
71
72 # Associate utility functions with the numbering of alternatives
73  $V = \{1: V1,$ 
74    $2: V2,$ 
75    $3: V3\}$ 
76
77  $V\_MINUS = \{1: V1\_MINUS,$ 

```

```

78         2: V2_MINUS,
79         3: V3_MINUS}
80
81 # Associate the availability conditions with the alternatives
82
83 CAR_AV_SP = DefineVariable('CAR_AV_SP', CAR_AV * ( SP != 0
84 ))
85 TRAIN_AV_SP = DefineVariable('TRAIN_AV_SP', TRAIN_AV * ( SP !=
86 0 ))
87
88 av = {1: TRAIN_AV_SP,
89       2: SM_AV,
90       3: CAR_AV_SP}
91
92 # The choice model is a logit, with availability conditions
93 integrand_plus = bioLogit(V, av, CHOICE)
94 integrand_minus = bioLogit(V_MINUS, av, CHOICE)
95 integrand = 0.5 * (integrand_plus + integrand_minus)
96 simulatedI = MonteCarlo(integrand)
97
98 trueI = 0.637849835578
99
100 sampleVariance = \
101     MonteCarlo(integrand*integrand) - simulatedI * simulatedI
102 stderr = (sampleVariance / 200000.0)**0.5
103 error = simulatedI - trueI
104
105 simulate = {'01 Simulated Integral': simulatedI,
106            '02 Analytical Integral': trueI,
107            '03 Sample variance': sampleVariance,
108            '04 Std Error': stderr,
109            '05 Error': error}
110
111 rowIterator('obsIter')
112
113 BIOGEME.OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')
114
115 __rowId__ = Variable('__rowId__')
116 BIOGEME.OBJECT.EXCLUDE = __rowId__ >= 1
117
118 BIOGEME.OBJECT.PARAMETERS['NbrOfDraws'] = "5"
119 BIOGEME.OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }

```

A.7 08normalMixtureControlVariate.py

```

1 #####
2 #
3 # File: 08normalMixtureControlVariate.py
4 # Author: Michel Bierlaire, EPFL

```

```

5 # Date: Sat Jul 25 18:50:11 2015
6 #
7 #####
8
9 from biogeme import *
10 from headers import *
11 from loglikelihood import *
12 from statistics import *
13
14 #Parameters
15 ASC_CAR = 0.137
16 ASC_TRAIN = -0.402
17 ASC_SM = 0
18 B_TIME = -2.26
19 B_TIME_S = 1.66
20 B_COST = -1.29
21
22 # Define a random parameter, normally distributed,
23 # designed to be used for Monte-Carlo simulation
24 B_TIME_RND = B_TIME + B_TIME_S * bioDraws('B_TIME_RND')
25
26 # Utility functions
27
28 #If the person has a GA (season ticket) her
29 #incremental cost is actually 0
30 #rather than the cost value gathered from the
31 # network data.
32 SM_COST = SM_CO * ( GA == 0 )
33 TRAIN_COST = TRAIN_CO * ( GA == 0 )
34
35 # For numerical reasons, it is good practice to scale the data to
36 # that the values of the parameters are around 1.0.
37 # A previous estimation with the unscaled data has generated
38 # parameters around -0.01 for both cost and time.
39 # Therefore, time and cost are multiplied by 0.01.
40
41 TRAIN_TT_SCALED = \
42     DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
43 TRAIN_COST_SCALED = \
44     DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
45 SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
46 SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
47 CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
48 CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)
49
50 V1 = ASC_TRAIN + \
51     B_TIME_RND * TRAIN_TT_SCALED + \
52     B_COST * TRAIN_COST_SCALED
53 V2 = ASC_SM + \

```

```

54     B.TIME_RND * SM.TT.SCALED + \
55     B.COST * SM.COST.SCALED
56 V3 = ASC_CAR + \
57     B.TIME_RND * CAR.TT.SCALED + \
58     B.COST * CAR.CO.SCALED
59
60 # Associate utility functions with the numbering of alternatives
61 V = {1: V1,
62      2: V2,
63      3: V3}
64
65 # Associate the availability conditions with the alternatives
66
67 CAR_AV.SP = DefineVariable('CAR_AV.SP', CAR_AV * ( SP != 0
68 ))
69 TRAIN_AV.SP = DefineVariable('TRAIN_AV.SP', TRAIN_AV * ( SP !=
70 0 ))
71
72 av = {1: TRAIN_AV.SP,
73       2: SM_AV,
74       3: CAR_AV.SP}
75
76 # The choice model is a logit, with availability conditions
77 integrand = bioLogit(V, av, CHOICE)
78
79 # Control variate
80
81 # Recycle the uniform draws used to generate the
82 #normal draws of B.TIME_RND
83 UNIFDRAW = bioRecycleDraws('B.TIME_RND')
84
85 # Utility function with the uniform draws instead of the normal.
86 VCV = ASC.TRAIN + \
87     (B.TIME + B.TIME.S * UNIFDRAW) * TRAIN.TT.SCALED + \
88     B.COST * TRAIN.COST.SCALED
89
90 # The analytical integral of exp(VCV) between 0 and 1
91 # is now calculated
92 VCV_ZERO = ASC.TRAIN + \
93     B.TIME * TRAIN.TT.SCALED + \
94     B.COST * TRAIN.COST.SCALED
95 VCV_ONE = ASC.TRAIN + \
96     (B.TIME + B.TIME.S) * TRAIN.TT.SCALED + \
97     B.COST * TRAIN.COST.SCALED
98 VCV_INTEGRAL = (exp(VCV.ONE) - exp(VCV.ZERO)) / \
99     (B.TIME.S * TRAIN.TT.SCALED)
100
101 simulatedI = MonteCarloControlVariate(integrand, \
102                                     exp(VCV), \

```

```

101                                     VCV_INTEGRAL)
102
103 trueI = 0.637849835578
104
105 error = simulatedI - trueI
106
107 simulate = {'01 Simulated Integral': simulatedI,
108             '02 Analytical Integral': trueI,
109             '05 Error': error}
110
111 rowIterator('obsIter')
112
113 BIOGEME_OBJECT.SIMULATE = Enumerate(simulate, 'obsIter')
114
115 __rowId__ = Variable('__rowId__')
116 BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1
117
118 BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
119 BIOGEME_OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }

```

A.8 11estimationNumerical.py

```

1 #####
2 #
3 # File: 11estimationNumerical.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Thu Jul 30 10:40:49 2015
6 #
7 #####
8
9 from biogeme import *
10 from headers import *
11 from distributions import *
12 from loglikelihood import *
13 from statistics import *
14
15 #Parameters to be estimated
16 # Arguments:
17 # 1 Name for report. Typically, the same as the variable
18 # 2 Starting value
19 # 3 Lower bound
20 # 4 Upper bound
21 # 5 0: estimate the parameter, 1: keep it fixed
22
23 ASC_CAR = Beta('ASC_CAR', 0, -10, 10, 0)
24 ASC_TRAIN = Beta('ASC_TRAIN', 0, -10, 10, 0)
25 ASC_SM = Beta('ASC_SM', 0, -10, 10, 1)
26 B_TIME = Beta('B_TIME', 0, -10, 10, 0)
27 B_TIME_S = Beta('B_TIME_S', 9, -10, 10, 0)

```



```

28 B_COST = Beta('B_COST',0,-10,10,0)
29
30 # Define a random parameter, normally distributed,
31 # designed to be used for simulation
32 omega = RandomVariable('omega')
33 density = normalpdf(omega)
34 B_TIME_RND = B_TIME + B_TIME_S * omega
35
36 # Utility functions
37
38 #If the person has a GA (season ticket) her
39 #incremental cost is actually 0
40 #rather than the cost value gathered from the
41 # network data.
42 SM_COST = SM_CO * ( GA == 0 )
43 TRAIN_COST = TRAIN_CO * ( GA == 0 )
44
45 # For numerical reasons, it is good practice to scale the data to
46 # that the values of the parameters are around 1.0.
47 # A previous estimation with the unscaled data has generated
48 # parameters around -0.01 for both cost and time.
49 # Therefore, time and cost are multiplied by 0.01.
50
51
52 TRAIN_TT_SCALED = \
53     DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
54 TRAIN_COST_SCALED = \
55     DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
56 SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
57 SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
58 CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
59 CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)
60
61 V1 = ASC_TRAIN + \
62     B_TIME_RND * TRAIN_TT_SCALED + \
63     B_COST * TRAIN_COST_SCALED
64 V2 = ASC_SM + \
65     B_TIME_RND * SM_TT_SCALED + \
66     B_COST * SM_COST_SCALED
67 V3 = ASC_CAR + \
68     B_TIME_RND * CAR_TT_SCALED + \
69     B_COST * CAR_CO_SCALED
70
71 # Associate utility functions with the numbering of alternatives
72 V = {1: V1,
73      2: V2,
74      3: V3}
75
76 # Associate the availability conditions with the alternatives

```

```

77 CAR_AV_SP = DefineVariable('CAR_AV_SP',CAR_AV * ( SP != 0
78 ))
79 TRAIN_AV_SP = DefineVariable('TRAIN_AV_SP',TRAIN_AV * ( SP !=
80 0 ))
81 av = {1: TRAIN_AV_SP,
82       2: SMLAV,
83       3: CAR_AV_SP}
84
85 # The choice model is a logit, with availability conditions
86 integrand = bioLogit(V,av,CHOICE)
87 prob = Integrate(integrand*density,'omega')
88 l = log(prob)
89
90 # Defines an iterator on the data
91 rowIterator('obsIter')
92
93 # Define the likelihood function for the estimation
94 BIOGEME.OBJECT.ESTIMATE = Sum(l,'obsIter')
95
96 # All observations verifying the following expression will not be
97 # considered for estimation
98 # The modeler here has developed the model only for work trips.
99 # Observations such that the dependent variable CHOICE is 0
100 # are also removed.
101 exclude = (( PURPOSE != 1 ) * ( PURPOSE != 3 ) + \
102            ( CHOICE == 0 )) > 0
103
104 BIOGEME.OBJECT.EXCLUDE = exclude
105
106 # Statistics
107
108 nullLoglikelihood(av,'obsIter')
109 choiceSet = [1,2,3]
110 cteLoglikelihood(choiceSet,CHOICE,'obsIter')
111 availabilityStatistics(av,'obsIter')
112
113 BIOGEME.OBJECT.PARAMETERS['RandomDistribution'] = "MLHS"
114
115 BIOGEME.OBJECT.PARAMETERS['optimizationAlgorithm'] = "BIO"
116 BIOGEME.OBJECT.FORMULAS['Train utility'] = V1
117 BIOGEME.OBJECT.FORMULAS['Swissmetro utility'] = V2
118 BIOGEME.OBJECT.FORMULAS['Car utility'] = V3

```

A.9 12estimationMonteCarlo.py

```

1 #####
2 #

```

```

3 # File: 12estimationMonteCarlo.py
4 # Author: Michel Bierlaire, EPFL
5 # Date: Thu Jul 30 18:33:34 2015
6 #
7 #####
8
9 from biogeme import *
10 from headers import *
11 from loglikelihood import *
12 from statistics import *
13
14 #Parameters to be estimated
15 # Arguments:
16 # 1 Name for report. Typically, the same as the variable
17 # 2 Starting value
18 # 3 Lower bound
19 # 4 Upper bound
20 # 5 0: estimate the parameter, 1: keep it fixed
21
22 ASC_CAR = Beta('ASC_CAR', 0, -10, 10, 0)
23 ASC_TRAIN = Beta('ASC_TRAIN', 0, -10, 10, 0)
24 ASC_SM = Beta('ASC_SM', 0, -10, 10, 1)
25 B_TIME = Beta('B_TIME', 0, -10, 10, 0)
26 B_TIME_S = Beta('B_TIME_S', 9, -10, 10, 0)
27 B_COST = Beta('B_COST', 0, -10, 10, 0)
28
29 # Define a random parameter, normally distributed, designed to be used
30 # for Monte-Carlo simulation
31 B_TIME_RND = B_TIME + B_TIME_S * bioDraws('B_TIME_RND')
32
33 # Utility functions
34
35 #If the person has a GA (season ticket) her incremental cost is actually 0
36 #rather than the cost value gathered from the
37 # network data.
38 SM_COST = SM_CO * ( GA == 0 )
39 TRAIN_COST = TRAIN_CO * ( GA == 0 )
40
41 # For numerical reasons, it is good practice to scale the data to
42 # that the values of the parameters are around 1.0.
43 # A previous estimation with the unscaled data has generated
44 # parameters around -0.01 for both cost and time. Therefore, time and
45 # cost are multiplied by 0.01.
46
47 TRAIN_TT_SCALED = DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
48 TRAIN_COST_SCALED = DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
49 SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
50 SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
51 CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)

```

```

52 CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)
53
54 V1 = ASC_TRAIN + B_TIME_RND * TRAIN_TT_SCALED + B_COST * TRAIN_COST_SCALED
55 V2 = ASC_SM + B_TIME_RND * SM_TT_SCALED + B_COST * SM_COST_SCALED
56 V3 = ASC_CAR + B_TIME_RND * CAR_TT_SCALED + B_COST * CAR_CO_SCALED
57
58 # Associate utility functions with the numbering of alternatives
59 V = {1: V1,
60      2: V2,
61      3: V3}
62
63 # Associate the availability conditions with the alternatives
64
65 CAR_AV_SP = DefineVariable('CAR_AV_SP', CAR_AV * ( SP != 0
66 ))
67 TRAIN_AV_SP = DefineVariable('TRAIN_AV_SP', TRAIN_AV * ( SP !=
68 0 ))
69
70 av = {1: TRAIN_AV_SP,
71       2: SMAV,
72       3: CAR_AV_SP}
73
74 # The choice model is a logit, with availability conditions
75 prob = bioLogit(V, av, CHOICE)
76 l = mixedloglikelihood(prob)
77
78 # Defines an iterator on the data
79 rowIterator('obsIter')
80
81 # Define the likelihood function for the estimation
82 BIOGEME.OBJECT.ESTIMATE = Sum(l, 'obsIter')
83
84 # All observations verifying the following expression will not be
85 # considered for estimation
86 # The modeler here has developed the model only for work trips.
87 # Observations such that the dependent variable CHOICE is 0 are also removed.
88 exclude = (( PURPOSE != 1 ) * ( PURPOSE != 3 ) + ( CHOICE == 0 )) > 0
89
90 BIOGEME.OBJECT.EXCLUDE = exclude
91
92 # Statistics
93
94 nullLoglikelihood(av, 'obsIter')
95 choiceSet = [1, 2, 3]
96 cteLoglikelihood(choiceSet, CHOICE, 'obsIter')
97 availabilityStatistics(av, 'obsIter')
98
99 BIOGEME.OBJECT.PARAMETERS['NbrOfDraws'] = "2000"
100 BIOGEME.OBJECT.PARAMETERS['RandomDistribution'] = "MLHS"

```

```
99
100 BIOGEME.OBJECT.PARAMETERS['optimizationAlgorithm'] = "BIO"
101 BIOGEME.OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }
102 BIOGEME.OBJECT.FORMULAS['Train utility'] = V1
103 BIOGEME.OBJECT.FORMULAS['Swissmetro utility'] = V2
104 BIOGEME.OBJECT.FORMULAS['Car utility'] = V3
```

References