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 $P_n(i|x_n, \mathcal{C}_n)$ providing the probability that individual n chooses alternative i within the choice set $\mathcal{C}_n$, 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

$$N = \sum_{n=1}^{N_s} w_n. \tag{1}$$

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

$$W_i = \frac{1}{N} \sum_{n=1}^{N_s} w_n P_n(i|x_n, \mathcal{C}_n). \tag{2}$$

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

$$\frac{\partial x_{ink}}{x_{ink}} = \frac{\partial x_{ipk}}{x_{ipk}} = \frac{\partial x_{ik}}{x_{ik}}, \tag{3}$$

where

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

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

$$E^{P_n(i)}_{x_{ink}} = \frac{\partial P_n(i|x_n, \mathcal{C}_n)}{\partial x_{ink}} \frac{x_{ink}}{P_n(i|x_n, \mathcal{C}_n)}. \tag{5}$$

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

$$E^{W_i}_{x_{ik}} = \frac{\partial W_i}{\partial x_{ik}} \frac{x_{ik}}{W_i}. \tag{6}$$

Using (2), we obtain

$$E^{W_i}_{x_{ik}} = \frac{1}{N} \sum_{n=1}^{N_s} w_n \frac{\partial P_n(i|x_n, \mathcal{C}_n)}{\partial x_{ik}} \frac{x_{ik}}{W_i}. \tag{7}$$

From (3), we obtain

$$E^{W_i}_{x_{ik}} = \frac{1}{N} \sum_{n=1}^{N_s} w_n \frac{\partial P_n(i|x_n, \mathcal{C}_n)}{\partial x_{ink}} \frac{x_{ink}}{W_i} = \frac{1}{N} \sum_{n=1}^{N_s} w_n E^{P_n(i)}_{x_{ink}} \frac{P_n(i|x_n, \mathcal{C}_n)}{W_i}, \tag{8}$$

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

$$E^{W_i}_{x_{ik}} = \sum_{n=1}^{N_s} E^{P_n(i)}_{x_{ink}} \frac{w_n P_n(i|x_n, \mathcal{C}_n)}{\sum_{n=1}^{N_s} w_n P_n(i|x_n, \mathcal{C}_n)}. \tag{9}$$

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 normlaized version of $w_n P_n(i|x_n, \mathcal{C}_n)$.

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

$$\frac{\Delta x_{ink}}{x_{ink}} = \frac{\Delta x_{ipk}}{x_{ipk}} = \frac{\Delta x_{ik}}{x_{ik}}, \tag{10}$$

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

$$E^{P_n(i)}_{x_{ink}} = \frac{\Delta P_n(i|x_n, \mathcal{C}_n)}{\Delta x_{ink}} \frac{x_{ink}}{P_n(i|x_n, \mathcal{C}_n)}. \tag{11}$$

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

$$E_{x_{ik}}^{W_i} = \frac{\Delta W_i}{\Delta x_{ik}} \frac{x_{ik}}{W_i}. \tag{12}$$

The two quantities are also related by (9).

A Complete specification files

A.1 01simpleIntegral.py

```
#######################################
#
# File: 01simpleIntegral.py
# Author: Michel Bierlaire, EPFL
# Date: Sat Jul 25 11:41:13 2015
#
#######################################

from biogeme import *
from headers import *

integrand = exp(bioDraws('U'))
simulatedI = MonteCarlo(integrand)

trueI = exp(1.0) - 1.0

sampleVariance = \
  MonteCarlo(integrand*integrand) - simulatedI * simulatedI
stderr = (sampleVariance / 200000.0)**0.5
error = simulatedI - trueI

simulate = {'01 Simulated Integral': simulatedI,
            '02 Analytical Integral': trueI,
            '03 Sample variance': sampleVariance,
            '04 Std Error': stderr,
            '05 Error': error}

rowIterator('obsIter')

BIOGEME_OBJECT.SIMULATE = Enumerate(simulate,'obsIter')
BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
BIOGEME_OBJECT.PARAMETERS['RandomDistribution'] = "PSEUDO"
__rowId__ = Variable('__rowId__')
BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1
BIOGEME_OBJECT.DRAWS = { 'U': 'UNIFORM'}
```

A.2 02antithetic.py

```
#######################################
#
# File: 02antithetic.py
# Author: Michel Bierlaire, EPFL
# Date: Sat Jul 25 12:21:10 2015
#
```

```
#######################################

from biogeme import *
from headers import *

integrand = 0.5 * (exp(bioDraws('U')) + exp(1.0-bioDraws('U')))
simulatedI = MonteCarlo(integrand)

trueI = exp(1.0) - 1.0

sampleVariance = \
  MonteCarlo(integrand*integrand) - simulatedI * simulatedI
stderr = (sampleVariance / 10000.0)**0.5
error = simulatedI - trueI

simulate = {'01_Simulated Integral': simulatedI,
            '02_Analytical Integral': trueI,
            '03_Sample variance': sampleVariance,
            '04_Std Error': stderr,
            '05_Error': error}

rowIterator('obsIter')

BIOGEME_OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
__rowId__ = Variable('__rowId__')
BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1
BIOGEME_OBJECT.DRAWS = { 'U': 'UNIFORM'}
```

A.3 03controlVariate.py

```
#######################################
#
# File: 03controlVariate.py
# Author: Michel Bierlaire, EPFL
# Date: Sat Jul 25 12:24:25 2015
#
#######################################
#

from biogeme import *
from headers import *

integrand = exp(bioDraws('U'))
simulatedI = MonteCarloControlVariate(integrand,bioDraws('U'),0.5)

trueI = exp(1.0) - 1.0

```

```
error = simulatedI - trueI

simulate = {'01_Simulated Integral': simulatedI,
            '02_Analytical Integral': trueI,
            '05_Error': error}

rowIterator('obsIter')

BIOGEME_OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
__rowId__ = Variable('__rowId__')
BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1
BIOGEME_OBJECT.DRAWS = { 'U': 'UNIFORM'}
```

A.4 05normalMixtureTrueAnalytical.py

```
#######################################
#
# File: 05normalMixtureTrueAnalytical.py
# Author: Michel Bierlaire, EPFL
# Date: Sat Jul 25 18:50:11 2015
#
#######################################

from biogeme import *
from headers import *
from distributions import *
from loglikelihood import *

#Parameters
ASC_CAR = 0.137
ASC_TRAIN = -0.402
ASC_SM = 0
B_TIME = -2.26
B_TIME_S = 1.66
B_COST = -1.29

# Define a random parameter, normally distributed,
# designed to be used for integration
omega = RandomVariable('omega')
density = normalpdf(omega)
B_TIME_RND = B_TIME + B_TIME_S * omega

# Utility functions

#If the person has a GA (season ticket) her
#incremental cost is actually 0
#rather than the cost value gathered from the
```

```
# network data.
SM_COST = SM_CO * ( GA == 0 )
TRAIN_COST = TRAIN_CO * ( GA == 0 )

# For numerical reasons, it is good practice to scale the data to
# that the values of the parameters are around 1.0.
# A previous estimation with the unscaled data has generated
# parameters around -0.01 for both cost and time.
# Therefore, time and cost are multipled my 0.01.

TRAIN_TT_SCALED = \
  DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
TRAIN_COST_SCALED = \
  DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)

V1 = ASC_TRAIN + \
     B_TIME_RND * TRAIN_TT_SCALED + \
     B_COST * TRAIN_COST_SCALED
V2 = ASC_SM + \
     B_TIME_RND * SM_TT_SCALED + \
     B_COST * SM_COST_SCALED
V3 = ASC_CAR + \
     B_TIME_RND * CAR_TT_SCALED + \
     B_COST * CAR_CO_SCALED


# Associate utility functions with the numbering of alternatives
V = {1: V1,
     2: V2,
     3: V3}

# Associate the availability conditions with the alternatives

CAR_AV_SP = DefineVariable('CAR_AV_SP',CAR_AV * ( SP != 0 ))
TRAIN_AV_SP = DefineVariable('TRAIN_AV_SP',TRAIN_AV * ( SP != 0 ))

av = {1: TRAIN_AV_SP,
      2: SM_AV,
      3: CAR_AV_SP}

# The choice model is a logit, with availability conditions
integrand = bioLogit(V,av,CHOICE)

```

```

analyticalI = Integrate(integrand*density,'omega')
simulate = {'Analytical': analyticalI}

rowIterator('obsIter')

BIOGEME_OBJECT.PARAMETERS['decimalPrecisionForSimulation'] = "12"
BIOGEME_OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

__rowId__ = Variable('__rowId__')
BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1
```

A.5 06normalMixture.py

```
#######################################
#
# File: 06normalMixture.py
# Author: Michel Bierlaire, EPFL
# Date: Sat Jul 25 18:37:37 2015
#
#######################################

from biogeme import *
from headers import *
from loglikelihood import *
from statistics import *

#Parameters
ASC_CAR = 0.137
ASC_TRAIN = -0.402
ASC_SM = 0
B_TIME = -2.26
B_TIME_S = 1.66
B_COST = -1.29

# Define a random parameter, normally distributed,
# designed to be used for integration
omega = bioDraws('B_TIME_RND')
B_TIME_RND = B_TIME + B_TIME_S * omega

# Utility functions

#If the person has a GA (season ticket) her
#incremental cost is actually 0
#rather than the cost value gathered from the
# network data.
SM_COST = SM_CO * ( GA == 0 )
TRAIN_COST = TRAIN_CO * ( GA == 0 )

```

```
# For numerical reasons, it is good practice to scale the data to
# that the values of the parameters are around 1.0.
# A previous estimation with the unscaled data has generated
# parameters around -0.01 for both cost and time. Therefore, time and
# cost are multipled my 0.01.

TRAIN_TT_SCALED = \
  DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
TRAIN_COST_SCALED = \
  DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)

V1 = ASC_TRAIN + \
     B_TIME_RND * TRAIN_TT_SCALED + \
     B_COST * TRAIN_COST_SCALED
V2 = ASC_SM + \
     B_TIME_RND * SM_TT_SCALED + \
     B_COST * SM_COST_SCALED
V3 = ASC_CAR + \
     B_TIME_RND * CAR_TT_SCALED + \
     B_COST * CAR_CO_SCALED

# Associate utility functions with the numbering of alternatives
V = {1: V1,
     2: V2,
     3: V3}

# Associate the availability conditions with the alternatives

CAR_AV_SP =  DefineVariable('CAR_AV_SP',CAR_AV  * (  SP   !=  0  ))
TRAIN_AV_SP =  DefineVariable('TRAIN_AV_SP',TRAIN_AV  * (  SP   !=  0  ))

av = {1: TRAIN_AV_SP,
      2: SM_AV,
      3: CAR_AV_SP}

# The choice model is a logit, with availability conditions
integrand = bioLogit(V,av,CHOICE)
simulatedI = MonteCarlo(integrand)

trueI = 0.637849835578

sampleVariance = \
  MonteCarlo(integrand*integrand) - simulatedI * simulatedI
```

```
stderr = (sampleVariance / 200000.0)**0.5
error = simulatedI - trueI

simulate = {'01 Simulated Integral': simulatedI,
            '02 Analytical Integral': trueI,
            '03 Sample variance': sampleVariance,
            '04 Std Error': stderr,
            '05 Error': error}

rowIterator('obsIter')

BIOGEME_OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

__rowId__ = Variable('__rowId__')
BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1

BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
BIOGEME_OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }
```

A.6 07normalMixtureAntithetic.py

```
#######################################
#
# File: 07normalMixtureAntithetic.py
# Author: Michel Bierlaire, EPFL
# Date: Sat Jul 25 19:14:42 2015
#
#######################################

from biogeme import *
from headers import *
from loglikelihood import *
from statistics import *

#Parameters
ASC_CAR = 0.137
ASC_TRAIN = -0.402
ASC_SM = 0
B_TIME = -2.26
B_TIME_S = 1.66
B_COST = -1.29

# Define a random parameter, normally distributed,
# designed to be used for integration,
# and its antithetic.
B_TIME_RND = B_TIME + B_TIME_S * bioDraws('B_TIME_RND')
B_TIME_RND_MINUS = B_TIME - B_TIME_S * bioDraws('B_TIME_RND')

# Utility functions
```

```

#If the person has a GA (season ticket) her
#incremental cost is actually 0
#rather than the cost value gathered from the
# network data.
SM_COST = SM_CO * ( GA == 0 )
TRAIN_COST = TRAIN_CO * ( GA == 0 )

# For numerical reasons, it is good practice to scale the data to
# that the values of the parameters are around 1.0.
# A previous estimation with the unscaled data has generated
# parameters around -0.01 for both cost and time.
# Therefore, time and cost are multipled my 0.01.

TRAIN_TT_SCALED = \
  DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
TRAIN_COST_SCALED = \
  DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)

V1 = ASC_TRAIN + \
     B_TIME_RND * TRAIN_TT_SCALED + \
     B_COST * TRAIN_COST_SCALED
V2 = ASC_SM + \
     B_TIME_RND * SM_TT_SCALED + \
     B_COST * SM_COST_SCALED
V3 = ASC_CAR + \
     B_TIME_RND * CAR_TT_SCALED + \
     B_COST * CAR_CO_SCALED

V1_MINUS = ASC_TRAIN + \
           B_TIME_RND_MINUS * TRAIN_TT_SCALED + \
           B_COST * TRAIN_COST_SCALED
V2_MINUS = ASC_SM + \
           B_TIME_RND_MINUS * SM_TT_SCALED + \
           B_COST * SM_COST_SCALED
V3_MINUS = ASC_CAR + \
           B_TIME_RND_MINUS * CAR_TT_SCALED + \
           B_COST * CAR_CO_SCALED

# Associate utility functions with the numbering of alternatives
V = {1: V1,
     2: V2,
     3: V3}

V_MINUS = {1: V1_MINUS,
```

```
            2: V2_MINUS,
            3: V3_MINUS}

# Associate the availability conditions with the alternatives

CAR_AV_SP =  DefineVariable('CAR_AV_SP',CAR_AV  * (  SP   !=  0  ))
TRAIN_AV_SP =  DefineVariable('TRAIN_AV_SP',TRAIN_AV  * (  SP   !=  0  ))

av = {1: TRAIN_AV_SP,
      2: SM_AV,
      3: CAR_AV_SP}

# The choice model is a logit, with availability conditions
integrand_plus = bioLogit(V,av,CHOICE)
integrand_minus = bioLogit(V_MINUS,av,CHOICE)
integrand = 0.5 * (integrand_plus + integrand_minus)
simulatedI = MonteCarlo(integrand)

trueI = 0.637849835578

sampleVariance = \
  MonteCarlo(integrand*integrand) - simulatedI * simulatedI
stderr = (sampleVariance / 200000.0)**0.5
error = simulatedI - trueI

simulate = {'01 Simulated Integral': simulatedI,
            '02 Analytical Integral': trueI,
            '03 Sample variance': sampleVariance,
            '04 Std Error': stderr,
            '05 Error': error}

rowIterator('obsIter')

BIOGEME_OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

__rowId__ = Variable('__rowId__')
BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1

BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
BIOGEME_OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }
```

A.7 08normalMixtureControlVariate.py

```
#######################################
#
# File: 08normalMixtureControlVariate.py
# Author: Michel Bierlaire, EPFL
```

```
# Date: Sat Jul 25 18:50:11 2015
#
#########################################

from biogeme import *
from headers import *
from loglikelihood import *
from statistics import *

#Parameters
ASC_CAR = 0.137
ASC_TRAIN = -0.402
ASC_SM = 0
B_TIME = -2.26
B_TIME_S = 1.66
B_COST = -1.29

# Define a random parameter, normally distributed,
# designed to be used for Monte-Carlo simulation
B_TIME_RND = B_TIME + B_TIME_S * bioDraws('B_TIME_RND')

# Utility functions

#If the person has a GA (season ticket) her
#incremental cost is actually 0
#rather than the cost value gathered from the
# network data.
SM_COST = SM_CO * ( GA == 0 )
TRAIN_COST = TRAIN_CO * ( GA == 0 )

# For numerical reasons, it is good practice to scale the data to
# that the values of the parameters are around 1.0.
# A previous estimation with the unscaled data has generated
# parameters around -0.01 for both cost and time.
# Therefore, time and cost are multipled my 0.01.

TRAIN_TT_SCALED = \
  DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
TRAIN_COST_SCALED = \
  DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)

V1 = ASC_TRAIN + \
      B_TIME_RND * TRAIN_TT_SCALED + \
      B_COST * TRAIN_COST_SCALED
V2 = ASC_SM + \
```

```
      B_TIME_RND * SM_TT_SCALED + \
      B_COST * SM_COST_SCALED
V3 = ASC_CAR + \
      B_TIME_RND * CAR_TT_SCALED + \
      B_COST * CAR_CO_SCALED

# Associate utility functions with the numbering of alternatives
V = {1: V1,
      2: V2,
      3: V3}

# Associate the availability conditions with the alternatives

CAR_AV_SP =  DefineVariable('CAR_AV_SP',CAR_AV  * (  SP   !=  0  ))
TRAIN_AV_SP =  DefineVariable('TRAIN_AV_SP',TRAIN_AV  * (  SP   !=  0  ))

av = {1: TRAIN_AV_SP,
       2: SM_AV,
       3: CAR_AV_SP}

# The choice model is a logit, with availability conditions
integrand = bioLogit(V,av,CHOICE)

# Control variate

# Recycle the uniform draws used to generate the
#normal draws of B_TIME_RND
UNIFDRAW = bioRecycleDraws('B_TIME_RND')

# Utility function with the uniform draws instead of the normal.
VCV = ASC_TRAIN + \
       (B_TIME + B_TIME_S * UNIFDRAW) * TRAIN_TT_SCALED + \
       B_COST * TRAIN_COST_SCALED
# The analytical integral of exp(VCV) between 0 and 1
# is now calculated
VCV_ZERO = ASC_TRAIN + \
             B_TIME  * TRAIN_TT_SCALED + \
             B_COST * TRAIN_COST_SCALED
VCV_ONE = ASC_TRAIN + \
            (B_TIME + B_TIME_S ) * TRAIN_TT_SCALED + \
             B_COST * TRAIN_COST_SCALED
VCV_INTEGRAL = (exp(VCV_ONE) - exp(VCV_ZERO)) / \
                  (B_TIME_S * TRAIN_TT_SCALED)


simulatedI = MonteCarloControlVariate(integrand, \
                                       exp(VCV), \
```

```
                                                        VCV_INTEGRAL)

trueI = 0.637849835578

error = simulatedI - trueI

simulate = {'01 Simulated Integral': simulatedI,
            '02 Analytical Integral': trueI,
            '05 Error': error}

rowIterator('obsIter')

BIOGEME_OBJECT.SIMULATE = Enumerate(simulate,'obsIter')

__rowId__ = Variable('__rowId__')
BIOGEME_OBJECT.EXCLUDE = __rowId__ >= 1

BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "5"
BIOGEME_OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }
```

A.8 11estimationNumerical.py

```
########################################
#
# File: 11estimationNumerical.py
# Author: Michel Bierlaire, EPFL
# Date: Thu Jul 30 10:40:49 2015
#
########################################

from biogeme import *
from headers import *
from distributions import *
from loglikelihood import *
from statistics import *

#Parameters to be estimated
# Arguments:
#   1  Name for report. Typically, the same as the variable
#   2  Starting value
#   3  Lower bound
#   4  Upper bound
#   5  0: estimate the parameter, 1: keep it fixed

ASC_CAR = Beta('ASC_CAR',0,-10,10,0)
ASC_TRAIN = Beta('ASC_TRAIN',0,-10,10,0)
ASC_SM = Beta('ASC_SM',0,-10,10,1)
B_TIME = Beta('B_TIME',0,-10,10,0)
B_TIME_S = Beta('B_TIME_S',9,-10,10,0)
```

```
B_COST = Beta('B_COST',0,-10,10,0)

# Define a random parameter, normally distributed,
# designed to be used for  simulation
omega = RandomVariable('omega')
 density = normalpdf(omega)
B_TIME_RND = B_TIME + B_TIME_S * omega

# Utility functions

#If the person has a GA (season ticket) her
#incremental cost is actually 0
#rather than the cost value gathered from the
# network data.
SM_COST =  SM_CO   * (  GA   ==  0  )
TRAIN_COST =  TRAIN_CO   * (  GA   ==  0  )

# For numerical reasons, it is good practice to scale the data to
# that the values of the parameters are around 1.0.
# A previous estimation with the unscaled data has generated
# parameters around -0.01 for both cost and time.
# Therefore, time and cost are multipled my 0.01.


TRAIN_TT_SCALED = \
  DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
TRAIN_COST_SCALED = \
  DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)

V1 = ASC_TRAIN + \
      B_TIME_RND * TRAIN_TT_SCALED + \
      B_COST * TRAIN_COST_SCALED
V2 = ASC_SM + \
      B_TIME_RND * SM_TT_SCALED + \
      B_COST * SM_COST_SCALED
V3 = ASC_CAR + \
      B_TIME_RND * CAR_TT_SCALED + \
      B_COST * CAR_CO_SCALED

# Associate utility functions with the numbering of alternatives
V = {1: V1,
      2: V2,
      3: V3}

# Associate the availability conditions with the alternatives
```

```

CAR_AV_SP =  DefineVariable('CAR_AV_SP',CAR_AV  * (  SP   !=  0  ))
TRAIN_AV_SP =  DefineVariable('TRAIN_AV_SP',TRAIN_AV  * (  SP   !=  0  ))

av = {1: TRAIN_AV_SP,
      2: SM_AV,
      3: CAR_AV_SP}

# The choice model is a logit, with availability conditions
integrand = bioLogit(V,av,CHOICE)
prob = Integrate(integrand*density,'omega')
l = log(prob)

# Defines an itertor on the data
rowIterator('obsIter')

# Define the likelihood function for the estimation
BIOGEME_OBJECT.ESTIMATE = Sum(l,'obsIter')

# All observations verifying the following expression will not be
# considered for estimation
# The modeler here has developed the model only for work trips.
# Observations such that the dependent variable CHOICE is 0
# are also removed.
exclude = (( PURPOSE != 1 ) * (  PURPOSE   !=  3  ) + \
           ( CHOICE == 0 )) > 0

BIOGEME_OBJECT.EXCLUDE = exclude

# Statistics

nullLoglikelihood(av,'obsIter')
choiceSet = [1,2,3]
cteLoglikelihood(choiceSet,CHOICE,'obsIter')
availabilityStatistics(av,'obsIter')

BIOGEME_OBJECT.PARAMETERS['RandomDistribution'] = "MLHS"

BIOGEME_OBJECT.PARAMETERS['optimizationAlgorithm'] = "BIO"
BIOGEME_OBJECT.FORMULAS['Train utility'] = V1
BIOGEME_OBJECT.FORMULAS['Swissmetro utility'] = V2
BIOGEME_OBJECT.FORMULAS['Car utility'] = V3
```

A.9 12estimationMonteCarlo.py

```
########################################
#
```

```
# File: 12estimationMonteCarlo.py
# Author: Michel Bierlaire, EPFL
# Date: Thu Jul 30 18:33:34 2015
#
#########################################

from biogeme import *
from headers import *
from loglikelihood import *
from statistics import *

#Parameters to be estimated
# Arguments:
#   1  Name for report. Typically, the same as the variable
#   2  Starting value
#   3  Lower bound
#   4  Upper bound
#   5  0: estimate the parameter, 1: keep it fixed

ASC_CAR = Beta('ASC_CAR',0,-10,10,0)
ASC_TRAIN = Beta('ASC_TRAIN',0,-10,10,0)
ASC_SM = Beta('ASC_SM',0,-10,10,1)
B_TIME = Beta('B_TIME',0,-10,10,0)
B_TIME_S = Beta('B_TIME_S',9,-10,10,0)
B_COST = Beta('B_COST',0,-10,10,0)

# Define a random parameter, normally distirbuted, designed to be used
# for Monte-Carlo simulation
B_TIME_RND = B_TIME + B_TIME_S * bioDraws('B_TIME_RND')

# Utility functions

#If the person has a GA (season ticket) her incremental cost is actually 0
#rather than the cost value gathered from the
# network data.
SM_COST = SM_CO * ( GA == 0 )
TRAIN_COST = TRAIN_CO * ( GA == 0 )

# For numerical reasons, it is good practice to scale the data to
# that the values of the parameters are around 1.0.
# A previous estimation with the unscaled data has generated
# parameters around -0.01 for both cost and time. Therefore, time and
# cost are multipled my 0.01.

TRAIN_TT_SCALED = DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
TRAIN_COST_SCALED = DefineVariable('TRAIN_COST_SCALED', TRAIN_COST / 100)
SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_COST / 100)
CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
```

```
CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)

V1 = ASC_TRAIN + B_TIME_RND * TRAIN_TT_SCALED + B_COST * TRAIN_COST_SCALED
V2 = ASC_SM + B_TIME_RND * SM_TT_SCALED + B_COST * SM_COST_SCALED
V3 = ASC_CAR + B_TIME_RND * CAR_TT_SCALED + B_COST * CAR_CO_SCALED

# Associate utility functions with the numbering of alternatives
V = {1: V1,
     2: V2,
     3: V3}

# Associate the availability conditions with the alternatives

CAR_AV_SP =  DefineVariable('CAR_AV_SP',CAR_AV  * (  SP   !=  0  ))
TRAIN_AV_SP =  DefineVariable('TRAIN_AV_SP',TRAIN_AV  * (  SP   !=  0  ))

av = {1: TRAIN_AV_SP,
      2: SM_AV,
      3: CAR_AV_SP}

# The choice model is a logit, with availability conditions
prob = bioLogit(V,av,CHOICE)
l = mixedloglikelihood(prob)

# Defines an itertor on the data
rowIterator('obsIter')

# Define the likelihood function for the estimation
BIOGEME_OBJECT.ESTIMATE = Sum(l,'obsIter')

# All observations verifying the following expression will not be
# considered for estimation
# The modeler here has developed the model only for work trips.
# Observations such that the dependent variable CHOICE is 0 are also removed.
exclude = (( PURPOSE != 1 ) * (  PURPOSE   !=  3  ) + ( CHOICE == 0 )) > 0

BIOGEME_OBJECT.EXCLUDE = exclude

# Statistics

nullLoglikelihood(av,'obsIter')
choiceSet = [1,2,3]
cteLoglikelihood(choiceSet,CHOICE,'obsIter')
availabilityStatistics(av,'obsIter')

BIOGEME_OBJECT.PARAMETERS['NbrOfDraws'] = "2000"
BIOGEME_OBJECT.PARAMETERS['RandomDistribution'] = "MLHS"
```

```
BIOGEME_OBJECT.PARAMETERS['optimizationAlgorithm'] = "BIO"
BIOGEME_OBJECT.DRAWS = { 'B_TIME_RND': 'NORMAL' }
BIOGEME_OBJECT.FORMULAS['Train utility'] = V1
BIOGEME_OBJECT.FORMULAS['Swissmetro utility'] = V2
BIOGEME_OBJECT.FORMULAS['Car utility'] = V3
```

References