基礎ゼミ5

PythonBiogemeによる
多項ロジットモデルの推定

福田研究室
金子法子

16/5/2017

PythonBiogemeとは

Biogeme：
最尤推定を用いた離散選択モデルパラメータ推定ツール

Biogeme

BisonBiogeme

- 標準的な離散選択モデル
  (MNL,NL,CNL,Probit…)
- 形式に即した単純な言語

PythonBiogeme

- 多目的パラメータモデル
- モデルと尤度関数の使用はPython Programming 言語に基づく

使用するデータについて

◆Swissmetro.dat

GROUP	グループ
SURVEY	(0)電車，(1)車で行われた調査
SP	SP調査(1)に固定
ID	回答者ID
PURPOSE	目的(1)通勤(2)買物(3)仕事(4)私事(5)仕事から帰宅(6)買物から帰宅(7)出先から帰宅(8)私事から帰宅(9)その他
FIRST	First class traveler(0)no(1)yes
TICKET	搭乗券(0)None(1)往復半額券(2)片道半額券(3)往復普通券(4)片道往復券(5)半日券(6)定期券(7)学生・高齢者定期券(8)7時以降券(9)団体券(10)その他
WHO	払った人(0)不明(1)自分(2)雇用者(3)半分
LUGGAGE AGE	(0)なし(1)1(2)複数
MALE	(0)女性(1)男性
INCOME	年収(1000CHF)(0)(1)50以下(2)50～100(3)100以上(4)不明
GA	(1)GAあり(0)その他
ORIGIN	移動の始点

使用するデータについて

◆Swissmetro.dat

DEST	目的地
TRAIN_AV	列車利用可能ダミー
CAR_AV	車利用可能ダミー
SM_AV	SM利用可能ダミー
TRAIN_TT	電車移動時間
TRAIN_CO	電車費用
TRAIN_HE	列車間隔
SM_TT	SM移動時間
SM_CO	SM費用
SM_HE	SM間隔
SM_SEATS	(1)Airline seat (0)その他
CAR_TT	車移動時間
CAR_CO	車費用
CHOICE	(0)不明(1)列車(2)SM(3)車

推定するモデルについて

◆多項ロジットモデル（MNL)
選択肢3つの中から，誤差が第一種極値分布に従う
効用最大化を行うと考える

SwissMetro

Train

Car

推定するモデルについて

◆多項ロジットモデル(MNL)
選択肢3つの中から，誤差が第一種極値分布に従う
効用最大化を行うと考える

$$V_1 = V_{train} = ASC_{train} + B_{time} * Train_{TT\ scaled} + B_{cost} * Train_{cost\ scaled}$$

$$V_2 = V_{SM} = ASC_{SM} + B_{time} * SM_{TT\ scaled} + B_{cost} * SM_{cost\ scaled}$$

$$V_3 = V_{car} = ASC_{car} + B_{time} * car_{TT\ scaled} + B_{cost} * car_{cost\ scaled}$$

推定するモデルについて

◆多項ロジットモデル(MNL)
選択肢3つの中から，誤差が第一種極値分布に従う
効用最大化を行うと考える

$$V_1 = V_{train} = ASC_{train} + B_{time} * Train_{TT\ scaled} + B_{cost} * Train_{cost\ scaled}$$

$$V_2 = V_{SM} = ASC_{SM} + B_{time} * SM_{TT\ scaled} + B_{cost} * SM_{cost\ scaled}$$

$$V_3 = V_{car} = ASC_{car} + B_{time} * car_{TT\ scaled} + B_{cost} * car_{cost\ scaled}$$

ASC_{**}・B_{**}は推定されるパラメータ

$**_{TT\ scaled}$・$**_{cost\ scaled}$は変数

推定するモデルについて

◆選択確率式

$$P(i|\{1,2,3\};x,\beta) = \frac{y_i e^{V_i(x,\beta)}}{y_1 e^{V_1} + y_2 e^{V_2} + y_3 e^{V_3}}$$

y_i: 選択可能な時＝1，不可能な時＝0

◆対数尤度

$$LL = \sum_n \log P(i_n|\{1,2,3\};\beta)$$

i_n:個人nによって選択された選択肢

推定するコードについて

◆02logit.py

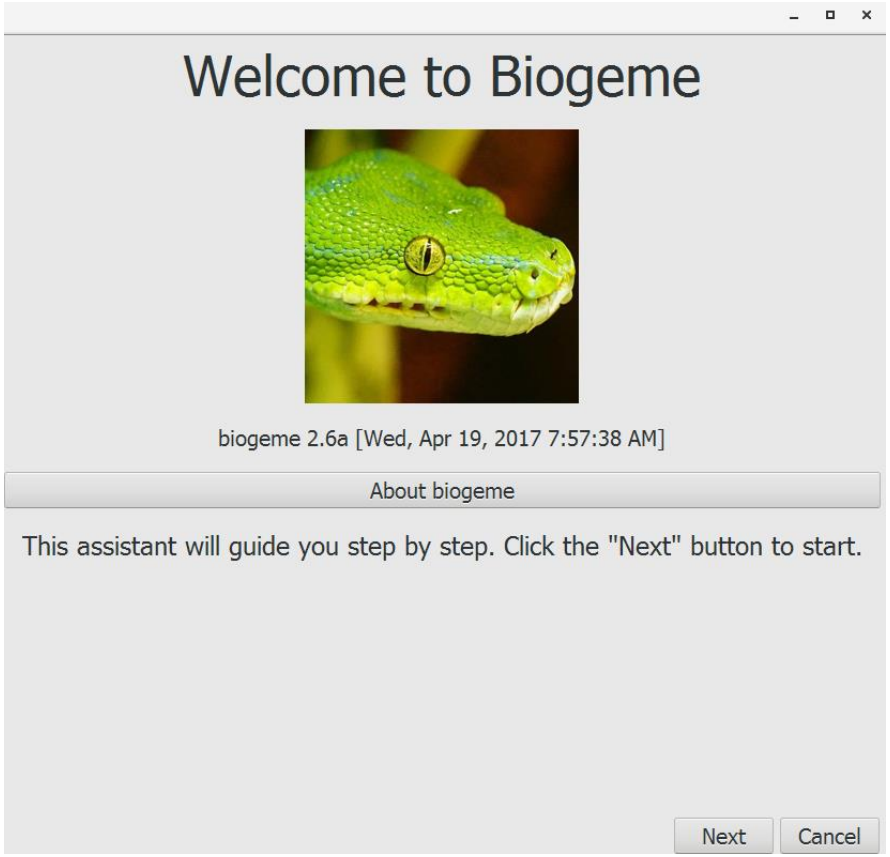
拡張子.py

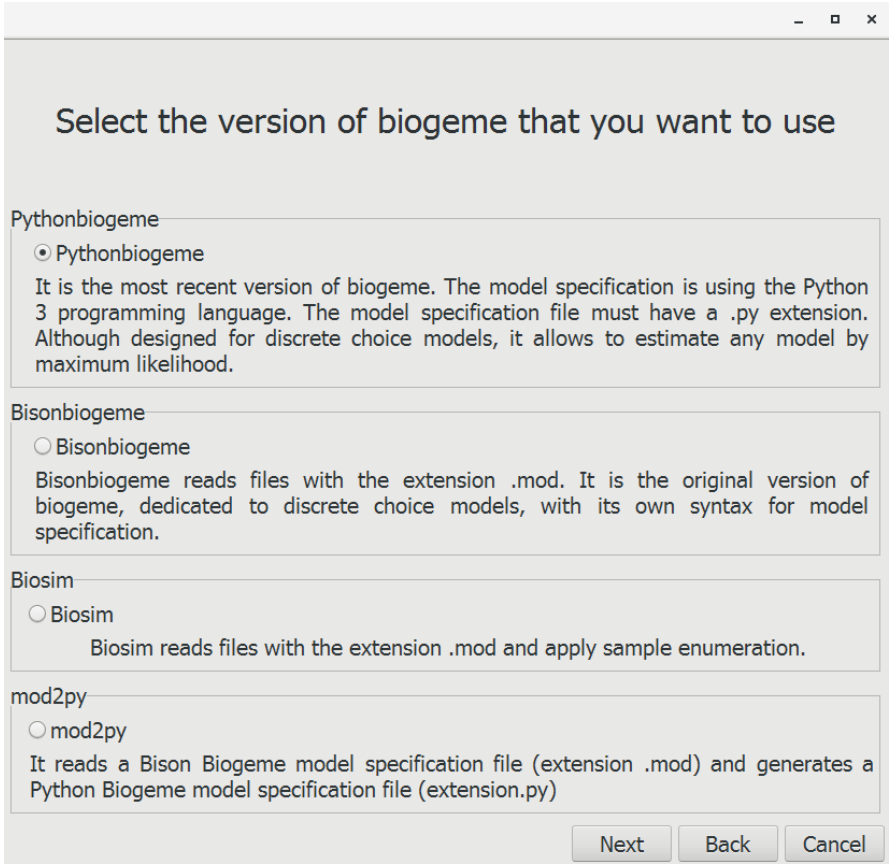
メモ帳などで開いてみると…

```
1  ##########################################↓
2  #↓
3  # @file 01logit.py↓
4  # @author: Michel Bierlaire, EPFL↓
5  # @date: Wed Dec 21 13:23:27 2011↓
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,'Car cte.')↓
23 ASC_TRAIN = Beta('ASC_TRAIN',0,-10,10,0,'Train cte.')↓
24 ASC_SM = Beta('ASC_SM',0,-10,10,1,'Swissmetro cte.')↓
25 B_TIME = Beta('B_TIME',0,-10,10,0,'Travel time')↓
26 B_COST = Beta('B_COST',0,-10,10,0,'Travel cost')↓
27 ↓
```

Pythonbiogemeの使い方

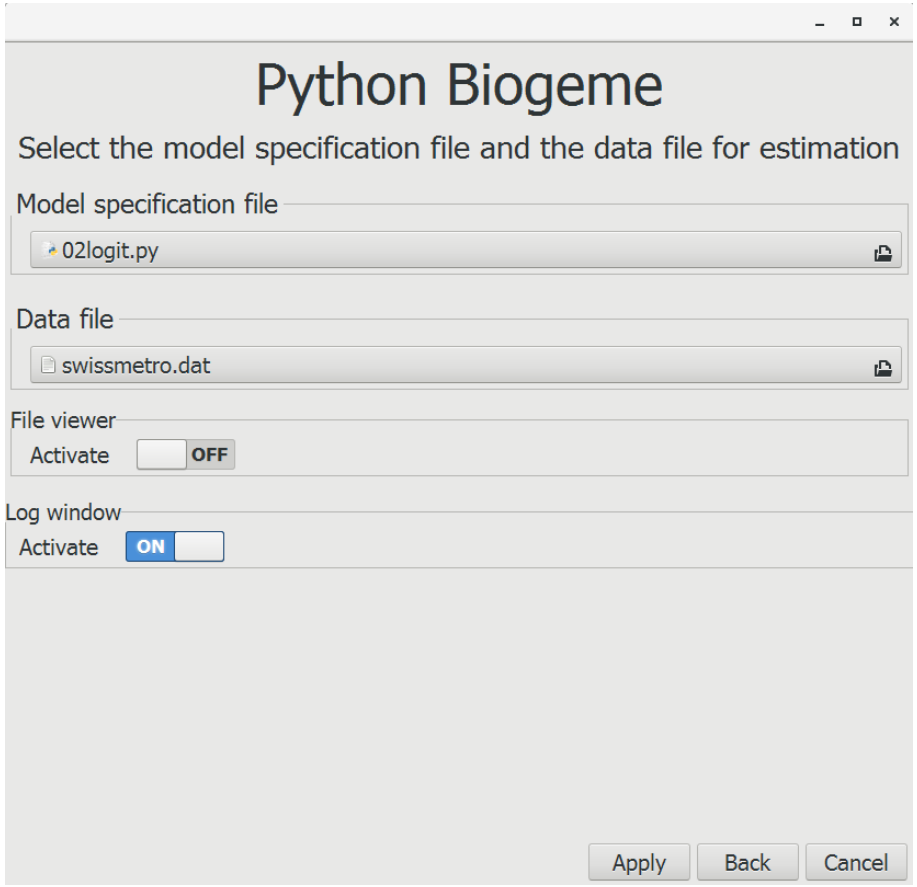
- BIOGEME/gtkguibiogeme_static.exe





Pythonbiogemeの使い方

- BIOGEME/gtkguibiogeme_static.exe



- Model specification 02logit.py
- Datafile swissmetro.dat

APPLY!!

推定するコードについて

◆02logit.py ～コメント文～

```
1 ##############################################
2 #
3 # @file 01logit.py
4 # @author: Michel Bierlaire, EPFL
5 # @date: Wed Dec 21 13:23:27 2011
6 #
7 ##############################################
8
```

#の後はコメント文として認識されて，コードには影響を及ぼさない
→python と統一された決まり

推定するコードについて

◆02logit.py ～ライブラリの読み込み～

```
 8 ↓
 9 from biogeme import *↓
10 from headers import *↓
11 from loglikelihood import *↓
12 from statistics import *↓
13 ↓
```

全部！！

必須のライブラリ

- biogeme･･･pythonbiogemeに必要なpython言語の拡張
- headers･･･dataの先頭行の読み込み

追加のライブラリ

- loglikelihood･･･最尤推定に必要なライブラリ
- statistics･･･統計に使う式のライブラリ

推定するコードについて

◆02logit.py ～パラメータの設定～

固定

```
ASC_CAR = Beta('ASC_CAR',0,-10,10,0,'Car cte.')↓
ASC_TRAIN = Beta('ASC_TRAIN',0,-10,10,0,'Train cte.')
ASC_SM = Beta('ASC_SM',0,-10,10,1,'Swissmetro cte.')↓
B_TIME = Beta('B_TIME',0,-10,10,0,'Travel time')↓
B_COST = Beta('B_COST',0,-10,10,0,'Travel cost')↓
```

ASC_CAR = Beta(‘ASC_CAR’ , 0 , -10 , 10 , 0 , ‘Car cte.’)

パラメータ名　初期値　上限・下限　固定(1)・非固定(0)　結果表示される時のパラメータ名

推定するコードについて

◆02logit.py ～変数の設定～

```
TRAIN_TT_SCALED = DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)
TRAIN_COST_SCALED = DefineVariable('TRAIN_COST_SCALED', TRAIN_CO / 100)
SM_TT_SCALED = DefineVariable('SM_TT_SCALED', SM_TT / 100.0)
SM_COST_SCALED = DefineVariable('SM_COST_SCALED', SM_CO / 100)
CAR_TT_SCALED = DefineVariable('CAR_TT_SCALED', CAR_TT / 100)
CAR_CO_SCALED = DefineVariable('CAR_CO_SCALED', CAR_CO / 100)
```

TRAIN_TT_SCALED
= DefineVariable('TRAIN_TT_SCALED', TRAIN_TT / 100.0)

変数定義の関数　　変数名　　スケール調整

推定するコードについて

◆02logit.py ～availabilityの設定～

```
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}↓
```

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

変数定義の関数　　変数名

$SP \neq 0$: CAR_AV

$SP = 0$: 0

推定するコードについて

◆02logit.py ～選択モデルの設定～

```
# The choice model is a logit, with availability conditions
logprob = bioLogLogit(V,av,CHOICE)
```

ロジットモデル選択
確率の対数をとる

bioLogLogit(効用関数，Availability,計算上の選択)

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

イテレータの設定

```
# DEfine the likelihood function for the estimation
BIOGEME_OBJECT.ESTIMATE = Sum(logprob,'obsIter')
```

推定変数の設定

推定するコードについて

◆02logit.py ～選択モデルの設定～

```
# Observations such that the dependent variable CHOICE is 0 are also removed.
exclude = (( PURPOSE != 1 ) * (  PURPOSE   !=  3  ) + ( CHOICE == 0 )) > 0
BIOGEME_OBJECT.EXCLUDE = exclude
```

目的が1,3(仕事目的)以外，選択が0(不明)をexcludeとして除外

推定するコードについて

◆02logit.py ～統計量を求める～

```
①  BIOGEME_OBJECT.PARAMETERS['optimizationAlgorithm'] = "BIO"
②  BIOGEME_OBJECT.FORMULAS['Train utility'] = V1
③  BIOGEME_OBJECT.FORMULAS['Swissmetro utility'] = V2
④  BIOGEME_OBJECT.FORMULAS['Car utility'] = V3
```

① BIOGEME_OBJECT.PARAMETERS[“optimizationAlgorithm”]=“BIO”

パラメータ推定の設定を
定義する関数

今回は最適化アルゴリズムを
BIOに設定

② ～④ BIOGEME_OBJECT.FORMULAS[“Car utility”]=“V3”

モデルの部分の選択

自動車の効用をV3と定義

推定するコードについて

◆02logit.py ～統計量を求める～

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

① nullLoglikelifood(利用可能な選択肢ID，データのイテレータ)
→ヌルログを計算する

$$L = -\sum_{n} \ln(J_n)$$

② cteLoglokelifood(選択肢，実際の選択，データのイテレータ)
→パラメータが1つ固定の場合の対数尤度

$$L = \sum_{i} n_i \ln(n_i) - nln(n)$$

③availabilityStatistics(availability,データのイテレータ)
→選択肢が利用可能である回数の統計値

結果の読み方

◆出力ファイルについて

① 02logit.html
HRML形式での推定結果
② 02logit_param.py
直接再使用できる推定値，分散・共分散のマトリクス
③ 02logit.log
実行中にpython biogemeから得られるログ
④ 02logit.tex
主な結果をtexで表記したもの
⑤ Hess.lis
最終値BHHH(Berndt Hall Hall and Hausman),2次導関数，ヘシアンマトリクス
⑥ Hessian.lis
各反復段階での対数尤度関数の逆行列(Matlab互換)
⑦ _parametersUsed.py
pythonbiogemeの実行で使用されたパラメータと使用されている値のリスト

結果の読み方

◆出力ファイルについて

Statistics

```
Alt. 1 available: 6768
   Alt. 1 chosen: 908
Alt. 2 available: 6768
   Alt. 2 chosen: 4090
Alt. 3 available: 5607
   Alt. 3 chosen: 1770
Cte loglikelihood (only for full choice sets): -6257.86
                Null loglikelihood: -6964.66
```

選択可能だった回数

選択した回数

パラメタ1つ固定の対数尤度

ヌルログ

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

結果の読み方

Estimation report

Number of estimated parameters:	4
Sample size:	6768
Excluded observations:	3960
Init log likelihood:	-6964.663
Final log likelihood:	-5590.464
Likelihood ratio test for the init. model:	2748.398
Rho-square for the init. model:	0.197
Rho-square-bar for the init. model:	0.197
Akaike Information Criterion:	11188.928
Bayesian Information Criterion:	11216.208
Final gradient norm:	+1.842e-004
Diagnostic:	Trust region algorithm with simple bounds (CGT2000): Convergence reached...
Iterations:	4
Data processing time:	00:00
Run time:	00:00
Nbr of threads:	2

初期尤度L^i

最終尤度L^i

尤度比検定 $-2(L^i - L^*)$

$\rho^2 = 1 - L^*/L^i$

$\overline{\rho^2} = 1 - L^*/L^i$

結果の読み方

◆出力ファイルについて

Estimated parameters

Click on the headers of the columns to sort the table [Credits]

Name	Value	Std err	t-test	p-value		Robust Std err	Robust t-test	p-value	
ASC_CAR	0.0735	0.0421	1.74	0.08	*	0.0593	1.24	0.22	*
ASC_TRAIN	-0.609	0.0546	-11.17	0.00		0.0800	-7.61	0.00	
B_COST	0.00635	0.00262	2.42	0.02		0.00221	2.88	0.00	
B_TIME	-1.16	0.0542	-21.32	0.00		0.0959	-12.06	0.00	

標準偏差 σ_k　t値 $t_k = \beta_k/\sigma_k$　p値

ロバスト推定による統計値

ロバスト推定：
誤差があるデータに対して
その誤差の影響を最小にすることを
目的とした理論

結果の読み方

◆出力ファイルについて

Correlation of coefficients

Click on the headers of the columns to sort the table [Credits]

Coefficient1	Coefficient2	Covariance	Correlation	t-test	p-value		Rob. cov.	Rob. corr.	Rob. t-test	p-value	
ASC_CAR	B_COST	3.12e-005	0.283	1.62	0.11	*	4.96e-005	0.379	1.15	0.25	*
ASC_TRAIN	B_TIME	-0.00216	-0.729	5.41	0.00		-0.00682	-0.888	3.20	0.00	
ASC_TRAIN	B_COST	2.29e-005	0.160	-11.36	0.00		3.67e-005	0.208	-7.73	0.00	
ASC_CAR	B_TIME	-0.00150	-0.655	14.00	0.00		-0.00478	-0.839	8.24	0.00	
B_COST	B_TIME	-1.54e-005	-0.108	21.30	0.00		-5.25e-005	-0.248	12.05	0.00	
ASC_CAR	ASC_TRAIN	0.00140	0.607	15.41	0.00		0.00388	0.818	14.69	0.00	

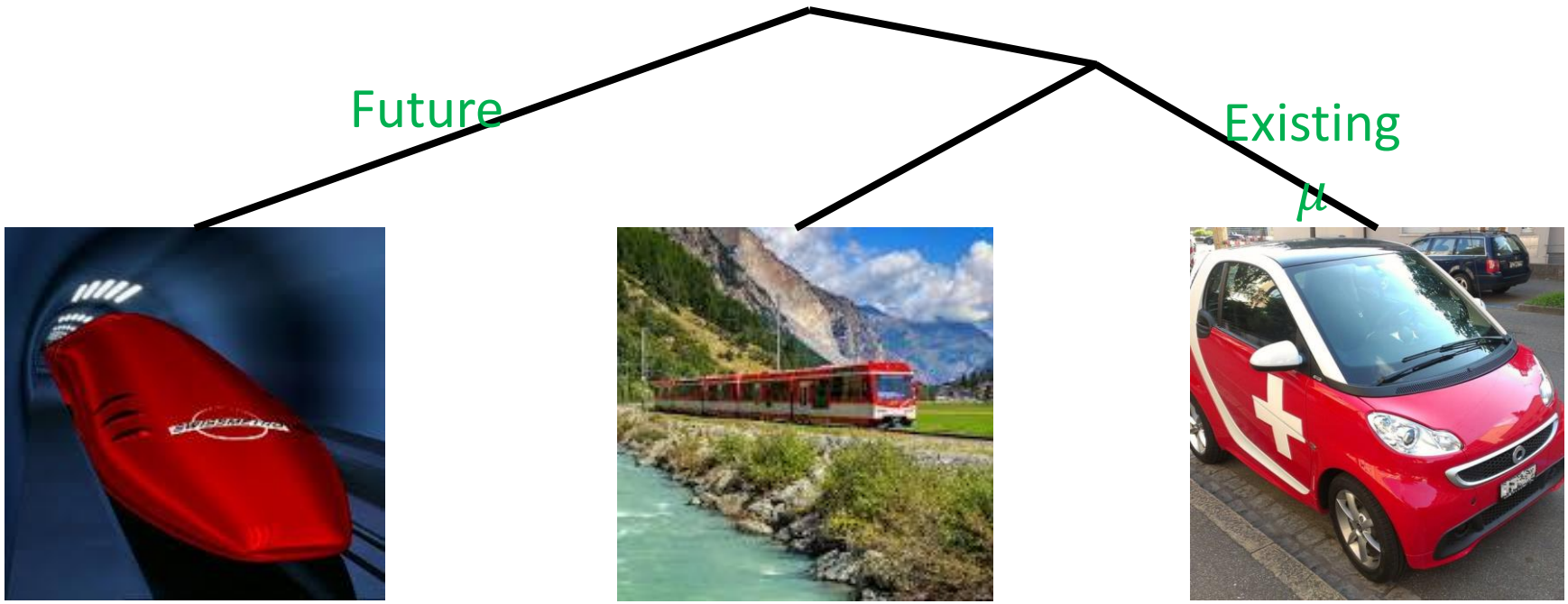
共分散　相関

t値　p値

ロバスト推定による
統計値

推定するモデルについて

◆NestedLogitモデル（MNL)
選択肢3つの中に以下のような相関があると仮定して
推定を行う



NLの推定

```
from biogeme import *
from headers import *
from nested import *
from loglikelihood import *
from statistics import *

#Parameters to be estimated
ASC_CAR = Beta('ASC_CAR',-0.167,-10,10,0)
ASC_TRAIN = Beta('ASC_TRAIN',-0.512,-10,10,0)
ASC_SM = Beta('ASC_SM',0,-10,10,1)
B_TIME = Beta('B_TIME',-0.899,-10,10,0)
B_COST = Beta('B_COST',-0.857,-10,10
MU = Beta('MU',2.05,1,10,0)

#GA
SM_COST =  SM_CO   * (  GA   ==  0  )
TRAIN_COST =  TRAIN_CO   * (  GA   ==  0  )

#scale
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)

#utility function
V1 = ASC_TRAIN + B_TIME * TRAIN_TT_SCALED + B_COST * TRAIN_COST_SCALED
V2 = ASC_SM + B_TIME * SM_TT_SCALED + B_COST * SM_COST_SCALED
V3 = ASC_CAR + B_TIME * CAR_TT_SCALED + B_COST * CAR_CO_SCALED
```

Nested Logit 用のライブラリのインポート

Nested Logit・相関を表すパラメーター

NLの推定

◆出力ファイルについて

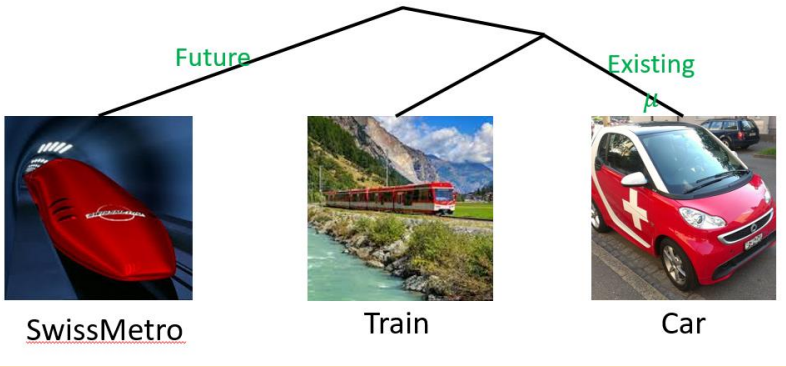
```
#
# Associate utility functions with the numbering of alternatives
V = [1: V1,
     2: V2,
     3: V3]

# 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]

#Definition of nests:
# 1: nests parameter
# 2: list of alternatives
existing = MU , [1,3]
future = 1.0 , [2]
nests = existing,future
```

Nested の設定



NLの推定

```
# The choice model is a nested logit, with availability conditions
logprob = lognested(V,av,nests,CHOICE)

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

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

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

NLの推定

Estimation report

Number of estimated parameters:	5
Sample size:	6768
Excluded observations:	3960
Init log likelihood:	-5236.901
Final log likelihood:	-5236.900
Likelihood ratio test for the init. model:	0.002
Rho-square for the init. model:	0.000
Rho-square-bar for the init. model:	-0.001
Akaike Information Criterion:	10483.800
Bayesian Information Criterion:	10517.900
Final gradient norm:	+2.309e-003
Diagnostic:	Trust region algorithm with simple bounds (CGT2000): Convergence reached...
Iterations:	1
Data processing time:	00:00
Run time:	00:01
Nbr of threads:	2

Estimated parameters

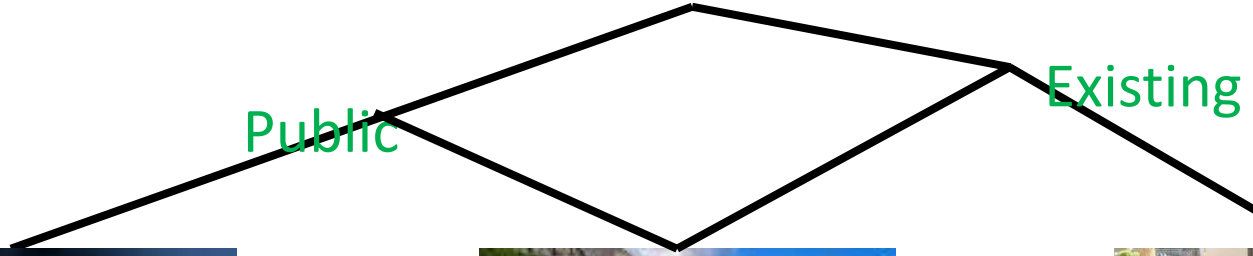
Click on the headers of the columns to sort the table [Credits]

Name	Value	Std err	t-test	p-value	Robust Std err	Robust t-test	p-value
ASC_CAR	-0.167	0.0371	-4.50	0.00	0.0545	-3.07	0.00
ASC_TRAIN	-0.512	0.0452	-11.33	0.00	0.0791	-6.47	0.00
B_COST	-0.857	0.0463	-18.51	0.00	0.0600	-14.27	0.00
B_TIME	-0.899	0.0570	-15.77	0.00	0.107	-8.39	0.00
MU	2.05	0.118	17.45	0.00	0.164	12.51	0.00

相関が認められることが分かる

(補足)その他のモデル

CNL



SwissMetro

Train

Car

```
#Definition of nests:↓
alpha_existing = {1: ALPHA_EXISTING,↓
                  2:0.0,↓
                  3:1.0}↓
↓
alpha_public = {1: ALPHA_PUBLIC,↓
                2: 1.0,↓
                3: 0.0}↓
```

（補足）その他のモデル

Probit

SwissMetro

Train

Car

```
P = [1: bioNormalCdf(V1-V3),
     3: bioNormalCdf(V3-V1)]
```