4.6 特殊関数とライブラリ CML[要 full version]　　ver. 0.1

引き続きパラメータに制約をかす場合のMaximum Log-Likelihoodを求めるCMLについての解説をします。少々使われる場面が限定されるのですが、パラメータ間にノンリニアな等号制約および不等号制約がある場合の設定の仕方について見てみます。まずは、非現実的なのですが、まずは、以前推定した標準Tobitを用いて、パラメータ間に次のような

AGE × ED ＋ 1 = －C18

ノンリニアな制約があって、その上でパラメータ推定をやってみます。
プログラム

```
new; cls;
library cml;
cmlset;
load data[51,7]=d:datafile6.txt;
data=data[2:51,.];

hour=data[.,2];
one=ones(50,1);
c18=data[.,4];
age=data[.,5];
age2=age^2;
ed1=data[.,6];
hi=data[.,7];
data=hour~one~c18~age~age2~ed1~hi;

_cml_Bounds = {-1e256   1e256,
                -1e256   1e256,
                -1e256   1e256,
                -1e256   1e256,
                -1e256   1e256,
                -1e256   1e256,
                0.0001   1e256};
fn eqproc(p)=p[3]*p[5]+p[2]+1;
_cml_EqProc=&eqproc;
_cml_ParNames = {"CONST","C18","AGE","AGE2","ED","HI","SIGMA"};
```

```
    _cml_Algorithm = 4;
    start={-1,0,0,0,0,0,1};
    call cmlprt(cml(data,0,&ll,start));

    proc ll(b,data);
        local y,x,s,beta;
        y=data[.,1];
        x=data[.,2:7];
        s=b[7];
        beta=b[1:6];
        retp( (y.==0).*ln(1-cdfn(x*beta/s))+ (y.>0).*lnpdfmvn(y-x*beta,s^2) );
    endp;
```

画面表示

```
return code =    0
normal convergence

Mean log-likelihood         -3.16617
Number of cases      50

Covariance of the parameters computed by the following method:
Inverse of computed Hessian

Parameters    Estimates     Std. err.    Gradient
-------------------------------------------------------------
CONST          -40.9837       21.0024     -0.0000
C18             -7.2916        3.0156     -0.0005
AGE              2.5455        1.6281     -0.0013
AGE2            -0.0402        0.0196     -0.0000
ED               2.4717        1.3188     -0.0013
HI               0.0001        0.0002     -0.0002
SIGMA           22.4104        3.0380      0.0000

Number of iterations    1800
Minutes to convergence      3.59400
```

上のように、パラメータ間のノンリニアの等号制約の設定には、左辺にすべてを移行した

陰関数のような形になるようＰベクトルの関数をfnまたはProcで作成したうえで、それをノンリニアの等号制約のグローバル変数にポインターのしるしをつけて設定します。上の制約の場合、すべてを左辺にもっていくと、

AGE × ED ＋ C18 ＋ 1 ＝ 0

となりますから、ｆｎ関数定義を用いて、

```
fn eqproc(p)=p[3]*p[5]+p[2]+1;
```

と表します。ここでなぜ関数の引数がＰで、そのほかの変数がp[3]やp[5]やp[2]になっているかと言いますと、_cml_ParNamesの推定パラメータ名設定のところで、もともとのパラメータのデフォルトは、P01とかP02とかいうふうに、Ｐのベクトルとして表現されてしたためです。GAUSSの内部計算では、P01とかP02といったパラメータを推定していて、これらに名前をつけるために、グローバル変数（文字列）_cml_ParNamesを設定していたわけです。したがって、この場合、ｐの１列目がCONSTに相当するもので、２列目がC18、さらに３列目がAGE、４列目がAGE2、５列目がED、６列目がHI、最終７列目がSIGMAということになります。これらのパラメータ間の関係を表したのが上のeqproc(p)という関数になります。なお、これはprocを用いて

```
proc eqproc(p);
    retp( p[3]*p[5]+p[2]+1 );
endp;
```

と書いても同じことです。プログラムの節のｆｎ１行関数定義のところでやったように、微分積分などに使う１行で書ける関数はprocを使って表現するまでもなく、ｆｎで１行で表現できるので、このようにしています。これらの関数をパラメータ間の等号制約として呼び出すには、この関数eqprocという関数名の前にポインタを表す＆のマークをつけて、

```
_cml_EqProc = &eqproc;
```

というふうに、グローバル変数_cml_EqProcの設定をします。なお、これらの制約は２乗の項があろうと何乗の項があろうと設定できますが、解けるか解けないかは、その関数の形とLog-Likelihoodとの間の関係に依存します。使える場面は限られてきますが、いざ使うとなると、手軽で興味深いグローバル変数の設定です。もちろん引数をP以外のもの、例えば微分積分でやったようにｘとできますが、ここではパラメータｘと混同が後のプログラム

でおきますから、Pを使います。

なお、この等号制約には、Jacobianを計算したものをさらに付け加えて計算を確実にする方法もあります。上のプログラムに加えて、次の２行

```
fn eqj(p)=0~1~p[5]~0~p[3]~0~0;
_cml_EqJacobian=&eqj;
```

を_cml_EqProcの行の次あたりに挿入します。なお、等号制約と同様に、eqjはprocで書いてもかまいません。ここで等号制約についてのJacobian、すなわち各パラメータでの１階の微分の行ベクトルには注意が必要です。上のノンリニアの等号制約には、p[3]とp[5]とp[2]しか存在しませんでしたが、このJacobianでは、ベクトルｐでの微分ということで、そべてのｐ、すなわち、p[1]からp[7]までの７つについてのそれぞれの微分を、横方向に行ベクトルとして表現する必要があります。また、微分はこのp[1]からp[7]までの１から７までの順序に従います。p[3]*p[5]+p[2]+1＝0について、p[1],p[2],…p[7]でのそれぞれの微分をして横方向のマージの演算記号～でくっつけあわせることになります。まず、p[1]については微分は０、p[2]については１、p[3]についてはp[5]が答えで、p[4]については０、p[5]についてはp[3]が答えで、p[6]とp[7]についてはその項がないので共に０となります。これらを行ベクトルにするために～でつなぎ合わせると、0~1~p[5]~0~p[3]~0~0　ということになります。これをｐの関数eqjとしてfnで１行定義したうえで、ポインタを表す＆のしるしをつけて、グローバル変数_cml_EqJacobianに設定します。計算結果は、この場合、等号制約式が複雑な形ではないので、上のJacobianなしのプログラムのものとほとんど同一になります。３つしかパラメータ制約がないといって、３つの微分をつなぎ合わせただけでは、行や列がマッチしないというエラーに直面することになります。すべてのパラメータ７つについての微分をこの１から７までの順序でする必要があります。

これと同様にして、異なるグローバル変数を同じように設定することにより、ノンリニアの不等号制約も設定できます。（ただし、かなりのケースで収束しないかしにくいケースがでてきます。）いま、次のような

$$AGE^2 + ED^2 + 1 >= -2 \times C18$$

という不等号制約がある場合のパラメータ推定をしてみます。グローバル変数は、等号制約のものに否定の意味のINをつけるだけでできます。CMLでは、＞＝の向きの制約がデフォルトになっています。（符号の向きが逆の場合には、リニアの場合の行列形式の制約設定と同じように、すべてのパラメータに－１をかけてやることによって、不等号の向きを逆転させます。）ここでは、上のような関係の不等号制約の設定は、Jacobianの計算では～でつなぎ合わせる時に、計算演算を含んでいれば、それぞれを丸括弧で包むことを忘れずに、

```
fn ineqproc(p)=p[3]^2+p[5]^2+2*p[2]+1;
_cml_IneqProc=&ineqproc;
fn ineqj(p)=0~2~(2*p[3])~0~(2*p[5])~0~0;
_cml_IneqJacobian=&ineqj;
```

というように書きます。上のような４行を、以前の等号制約のプログラムの等号制約部分とJacobianに相当する部分を消去して置きかえると、

画面表示

```
return code =    0
normal convergence

Mean log-likelihood         -3.16591
Number of cases      50

Covariance of the parameters computed by the following method:
Inverse of computed Hessian

Parameters    Estimates     Std. err.    Gradient
------------------------------------------------------------------
CONST          -47.4330       45.8227     -0.0000
C18             -7.1220        3.1961     -0.0000
AGE              2.8162        2.3679     -0.0000
AGE2            -0.0432        0.0273     -0.0000
ED               2.5322        1.3753     -0.0000
HI               0.0001        0.0002     -0.0001
SIGMA           22.4303        3.0444      0.0000

Number of iterations    1059
Minutes to convergence      2.50733
```

上のように、不等号制約に従ったパラメータの推定ができます。なお、定義する関数名とポインタで呼び出される関数名は一致している必要がありますが、等号制約および不等号制約、そしてそれらのJacobianを定義する関数名は任意です。等号制約同様、Jacobianをつければ計算は確かになりますが、どのような不等号制約も可能だからといって不等号制

約を設定しても、Log-Likelihoodを最大化する際に、これらはかなりきつい制約で、いつも解けるとはかぎりません。解ける場合は限られてきます。

ここで、特に注意したいことがあります。GARCHなどにも問題になりますが、パラメータ制約としてではなくて、Log-Likelihoodを定義するprocの内部で、パラメータに制約を加えたり、データの行数に変更を加えたり、中間の計算に制約を加えたり、負の数だからといって０に近い正の数に置き換えをする行為をしてはいけません。例えばGARCHでｈがすべて正の値であるはずだからといって、Log-Likelihoodを定義する「procの内部で」制限を加えたり置き換えをするのは適当ではありません。たいていの場合は、こうした補正した領域でLog-Likelihoodが最大になることは稀ですが、もしそこで最大値が見つかってしまったらそれは誤りです。「procの外部で」事後的に確認するアルゴリズムを書いてやるプログラムが正当なやり方であると思われます。もし、その確認に引っかかってしまったら、そこはローカルな極値である可能性が高く、あらためて別のスタートベクトルから始めて、違う場所の最大値を見つけることになります。もちろん、プログラムではなくて、コマンド言語では、ｈ全体の符号を正にするためにいろいろな追加的な制約を課していることがありありますが、これはGAUSSのプログラムのやり方とは違うので、おそらくエラーはないものだと思われますが、GAUSSのプログラムに移植する際、特にLog-Likelihoodのprocの内部で制限を加えるには注意が必要です。

CMLにはいろいろなstatistical inferenceをともなってパラメータの推定値を表示できます。バウンダリー条件のみがある標準Tobitにおいて、通常の信頼区間を求めるのならば、

プログラム（変更箇所）

```
{x,f,g,cov,retcode}=cml(data,0,&ll,start);
call cmlclprt(x,f,g,cmltlimits(x,cov),retcode);
```

とします。

また、WALD statisticsのinversionによってConstrained信頼区間を求めるのならば、

プログラム（変更箇所）

```
{x,f,g,cov,retcode}=cml(data,0,&ll,start);
call cmlclprt(x,f,g,cmlclimits(x,cov),retcode);
```

とすると結果は次のようになります。

画面表示

```
return code =     0
```

```
normal convergence

Mean log-likelihood        -3.16591
Number of cases     50
                                0.95 confidence limits
Parameters     Estimates     Lower Limit    Upper Limit    Gradient
-------------------------------------------------------------------
CONST          -47.4330      -139.8330       44.9670      -0.0000
C18             -7.1220       -13.5680       -0.6761      -0.0000
AGE              2.8162        -1.9586        7.5909      -0.0000
AGE2            -0.0432        -0.0983        0.0119      -0.0000
ED               2.5322        -0.2414        5.3058      -0.0000
HI               0.0001        -0.0002        0.0004      -0.0000
SIGMA           22.4303        16.2906       28.5700       0.0000

Number of iterations    1074
Minutes to convergence     2.14483
```

上の結果とこの場合ほとんど同じになりますが、Inverse Hessianを用いて計算した信頼区間は、Covariance Matrixの計算方法をquasi-maximum likelihoodのオプションに設定するためにグローバル変数_cml_CovParを３としたうえ、

プログラム（変更箇所）

```
_cml_CovPar=3;
{x,f,g,cov,retcode}=cml(data,0,&ll,start);
call cmlclprt(x,f,g,cmltlimits(x,_cml_HessCov),retcode);
```

とすることで計算できます。なお、グローバル変数_cml_CovParは

また、このInverse Hessianを用いて計算した場合のConstrained信頼区間は、cmltlimitsのｔをｃに変更して、cmlclimitsを用いて、

プログラム（変更箇所）

```
_cml_CovPar=3;
{x,f,g,cov,retcode}=cml(data,0,&ll,start);
call cmlclprt(x,f,g,cmlclimits(x,_cml_HessCov),retcode);
```

とすることで計算できます。なお、グローバル変数_cml_CovParは上の２つにおいては必ず３である必要があります。ちなみに、０がデフォルトで、iteration結果の最終information matrixのinverseから求めるCovariance Matrixとなっています。１はHessianのinverse、２は１階の微分の外積によるものです。上で設定した４つの信頼区間計算は、上の例では、不等号制約上でパラメータ解が決まっていないために、通常のMaxlikと同じ計算をしているので、すべて結果が同じになります。もし、パラメータの一部が、制約条件上で決まってくると、これらのパラメータの信頼区間は若干の違いが出てきます。

さらに、通常のWaldタイプの信頼区間ではなくて、likelihood ratioのinversionから計算するprofile likelihoodの信頼区間を求めるのには、次のように設定して計算をします。

プログラム（変更箇所）

```
{x,f,g,cov,retcode}=cml(data,0,&ll,start);
call cmlclprt(x,f,g,cmlpflclimits(x,f,data,0,&ll),retcode);
```

画面表示

```
return code =    0
normal convergence

Mean log-likelihood         -3.16591
Number of cases     50
                                0.95 confidence limits
Parameters    Estimates     Lower Limit   Upper Limit   Gradient
-----------------------------------------------------------------
CONST          -47.4330     -146.2462       40.9367     -0.0000
C18             -7.1220      -13.9683       -0.9374     -0.0000
AGE              2.8162       -1.6781        8.0404     -0.0000
AGE2            -0.0432       -0.1047        0.0078     -0.0000
ED               2.5322       -0.2588        5.3379     -0.0000
HI               0.0001       -0.0003        0.0004     -0.0000
SIGMA           22.4303       17.5442       29.9622      0.0000

Number of iterations    1074
Minutes to convergence     2.74817
```

上と同じパラメータ推定をWeighted Bayesian Bootstrapによって行ない、さらにその結果をbootstrapによる信頼区間とともに表示するのには、

プログラム（変更箇所）

```
_cml_BootFname="tobit";
{x,f,g,cov,retcode}=cmlboot(data,0,&ll,start);
call cmlclprt(x,f,g,cmlblimits(_cml_BootFname),retcode);
```

画面表示

```
return code =    0
normal convergence

Mean log-likelihood         -3.14952
Number of cases     50
                                0.95 confidence limits
Parameters    Estimates     Lower Limit   Upper Limit   Gradient
-------------------------------------------------------------------
CONST          -57.1415     -145.2310       -3.8253     -0.0000
C18             -6.5568      -14.4458       -2.4693     -0.0000
AGE              3.4539       -0.5019        6.9244     -0.0000
AGE2            -0.0503       -0.0939       -0.0155     -0.0000
ED               2.2270       -0.2350        3.9731     -0.0000
HI               0.0001       -0.0004        0.0003     -0.0000
SIGMA           21.0420       14.7670       26.5213      0.0000

Number of iterations    144
Minutes to convergence     0.05686
```

上のように表示されるまでアウトプットウインドウはしばらくの間、空白になります。ただし、あらかじめ_cml_BootFnameによって、Bootstrap信頼区間のパラメータが格納される場所を指定することが必要です。この格納されたデータセットの入ったグローバル変数_cml_BootFnameをもとにcmlblimitsによって、信頼区間が計算され、信頼区間の場合の表示フォーマットであるcmlclprtによって、パラメータとともにBootstrap信頼区間が画面表示されます。

このようにConstrained Maximum Likelihoodのケースには、パラメータに制約が課されるため、通常のstandard errorに依存するsymmetricな信頼区間ではなくて、制約にもとづいた信頼区間を計算する必要が出てきます。その信頼区間計算にいろいろな方法があるわけです。従来のstandard errrorから計算したｔ比などではなくて、実際の非対称な信頼区間が重要になってくるわけです。