## 5.11　モンテカルロシミュレーション Bootstrap の基礎　ver.0.1

「ブートストラップ」とは西洋人の履くブーツの突起しているつまみ皮の部分のことを指していて、具体的なイメージとしては、サンプル自身をそれでもって引き上げて自らリサンプリングすること。つまり、同じサンプルからデータをピックアップして自前でリサンプリングするものです。「靴紐」という一部の人の訳もありますが、shoelace のことではありませんので注意してください。Jackknife と同じような組ごとの bootstrap や残差の bootstrap を基本にして、果ては nested のものや multi scale のものまで数限りないバージョンが考案されてきています。ここでは基本的なものにしぼってプログラムします。

手計算

基本的な Bootstrap の仕組みを知るには、実際に手でやってみることが一番です。手もとに、研究者の方であればテストの答案、民間の方であれば名刺や抽選番号つきの官製年賀はがきがあるものとする。マークシートのカードであってもよい。表に２ケタのほどの数字が書かれているものとする。名刺の場合や官製年賀はがきの場合には電話番号か抽選番号の下２ケタを考える。仮に 10 枚からなるサンプルを考えよう。適当にシャッフルした後にまず１番目を引く、そして何か別の紙に数字をひかえる。それをまた元に戻してあらためてシャッフルしなおしてから２番目を引く。そしてその数字をひかえる。同じことを繰り返し、これを全体のサンプル数と同じ 10 番目まで引いて数字をひかえる。その時点で、10 回分の数字の統計量、例えば平均を計算してやって、その下に数字を出しておく。時間と労力はかかるが、もう１順重複を許してシャッフルした試行を繰り返して 10 回分の数字をひかえる。そして、平均を計算してやって、その下に数字を出す。これを例えば、時間があれば 100 回とか 1000 回繰り返してやる。実際には、そうしていると日が暮れるので、数十回繰り返して、その十数回分の平均の計算値を見てこれをそろばんか電卓で平均する。これが Bootstrap Mean 算出の基本です。

Indexing

上の手作業を計算機にやらせるために、実際に重複を許すシャッフルの代わりに、サンプルの 10 個の要素の順番にあらかじめ 1 から 10 までの index を割り当てていると考えて、１から 10 までの乱数をその都度引いて都合 10 回分乱数を引くものとする。その結果１番目に６の index 番号がきていれば、もとのサンプルの６番目の数、すなわち 42 を引いたことにして、次に 10 の index 番号がきていれば、もとのサンプルの 10 番目の数、すなわち 61 を引いたことにする。３番目には再び６の index 番号がきているので、再び 42 を重複して引いたものとする。この作業を 10 番目まで重複を許してコンピュータに乱数でもって擬似的にもとのサンプルから番号を引かせる。下のプログラムでは、その index 番号づけだけをやらせた部分で、内側の j ループで１番目から 10 番目までの index 番号による乱数割り当てを行なっていて、外側の i ループでこれを times 回繰り返している。この繰り返しは実際には数百回から数千回必要であるとされているが、下の例ではわかりやすいよう

に３回分だけ行なっている。技術的には、ループを回さなくても index=ceil(n*rndu(n,1)); の 1 行だけで済ますことが慣例になっている。これは一回一回１個だけの乱数を発生させて indexing をして、それを 10 回繰り返すのも、10 個まとめて列で乱数を発生させてもほとんど結果は変わらないことから簡易的にこうしている。なお、ceil ではなくて、round を用いて 0.5 を足す方法や、trunc や floor を使って１を足す方法もあるが同じことである。どれも、一様乱数[0,1)に何かをかけて伸ばしてやって、その整数部分を整数の乱数として取り出しているだけで、ランダムウオークで乱数で４つの方向や６つの方向に分けたプログラムと基本的には同じです。

プログラム

```
new; cls;
rndseed 911;
data={78,68,89,92,60,42,74,80,78,61};
times=3;
call bootindex(data,times);

proc(0)=bootindex(data,times);
    local n,i,j,index,resample,name;
    n=rows(data);
    i=1;
    do while i<=times;
        /* index=ceil(n*rndu(n,1)); */
        index=zeros(n,1);
        j=1;
        do while j<=n;
            index[j]=ceil(n*rndu(1,1));
            j=j+1;
        endo;
        resample=data[index];
        print/rz i;
        name={"no." "sample" "index" "resample"};
        print $name;;
        print/rz seqa(1,1,n)~data~index~resample;
        print;
        i=i+1;
    endo;
endp;
```

画面表示

1

| no. | sample | index | resample |
|---|---|---|---|
| 1 | 78 | 6 | 42 |
| 2 | 68 | 10 | 61 |
| 3 | 89 | 6 | 42 |
| 4 | 92 | 4 | 92 |
| 5 | 60 | 6 | 42 |
| 6 | 42 | 8 | 80 |
| 7 | 74 | 1 | 78 |
| 8 | 80 | 3 | 89 |
| 9 | 78 | 10 | 61 |
| 10 | 61 | 10 | 61 |

2

| no. | sample | index | resample |
|---|---|---|---|
| 1 | 78 | 2 | 68 |
| 2 | 68 | 7 | 74 |
| 3 | 89 | 6 | 42 |
| 4 | 92 | 10 | 61 |
| 5 | 60 | 1 | 78 |
| 6 | 42 | 2 | 68 |
| 7 | 74 | 3 | 89 |
| 8 | 80 | 2 | 68 |
| 9 | 78 | 4 | 92 |
| 10 | 61 | 6 | 42 |

3

| no. | sample | index | resample |
|---|---|---|---|
| 1 | 78 | 2 | 68 |
| 2 | 68 | 3 | 89 |
| 3 | 89 | 10 | 61 |
| 4 | 92 | 6 | 42 |
| 5 | 60 | 10 | 61 |
| 6 | 42 | 8 | 80 |
| 7 | 74 | 10 | 61 |

| | | | |
|---|---|---|---|
| 8 | 80 | 5 | 60 |
| 9 | 78 | 3 | 89 |
| 10 | 61 | 6 | 42 |

上の各表のように、サンプルのそれぞれの要素が１から 10 番目まで並んでいるものとします。そこで上のように一様乱数を発生させて１から 10 までの番号のうちどれかを毎回選択して計 10 番目まで 10 回引く。この番号にもとづいて、サンプルのもとの順番に照らしあわせてリサンプルを行なっている。Index の番号が sample の何番目かの番号に対応していて、その対応する実際の要素が resample のところの要素になっている。

Bootstrap サンプル

ここでは、Jackknife で行なったのと同じ９つのデータからなるサンプルを Bootstrap してその推定量を求める。平均を推定量にとってやって計算してみよう。

プログラム

```
new; cls;
rndseed 911;
data={78,68,89,92,60,42,74,80,78};
times=2000;
call bootmean(data,times);

proc bootmean(data,times);
    local n,mean,bmean,i,j,index,resample,meanbmean;
    n=rows(data);
    mean=meanc(data);
    bmean=zeros(times,1);
    i=1;
    do while i<=times;
        /* index=ceil(n*rndu(n,1)); */
        index=zeros(n,1);
        j=1;
        do while j<=n;
            index[j]=ceil(n*rndu(1,1));
            j=j+1;
        endo;
        resample=data[index];
        bmean[i]=meanc(resample);
        i=i+1;
```

```
    endo;
    meanbmean=meanc(bmean);
    print/lz "# of sample=" n;
    print/lz "sample mean=" mean;
    print "bootstrap mean" meanbmean;
    retp(bmean);
endp;
```

画面表示

```
# of sample= 9
sample mean= 73.444444
bootstrap mean          73.176333
```

上の例は、最も計算の簡単な推定量θにサンプル平均を考えたものです。その他に、標準偏差や Median なども推定量θに考えることも可能です。要するに、ある限られた数のデータからなるサンプルから繰り返しを許してランダムに引いた同じ数のデータからなるサンプルのコピーを数多く（この場合 2000 回）作成して、それぞれのサンプルの推定量θを計算して、それらの 2000 回分のサンプルのコピーの推定量の平均を考えたものです。勘違いをしてはいけないことは、Jackknife と同様に、冒頭でデータを乱数設定してもとのサンプルの分布を決めることはよくなされることですが、それはモンテカルロ部分であって、基本は限られた数のデータからなる１つの実際のサンプルから始めるということです。

上のプログラムをフォーマルにして、bootstrap された mean の各要素を小さいものから大きいもの順にソートしてやって、その 2.5%目と 97.5%目を見つけることによって、分布の形によらない 95%信頼区間を出してやろう。また、Jackknife と同様に、分散や Bias を計算し、Bias 修正済みの mean を出してみる。これからは、index=ceil(n*rndu(n,1));で一括して index を列として出していくことにします。また sorttheta=sortc(thetahat_i,1);によって、Bootstrap されたおのおののサンプルの推定量 thetahat_i を小さい方から大きい方に順にソートして、その結果を sorttheta とすることにします。この times×0.025 番目が分布によらない 95%信頼区間の下限に、times×0.975 番目がその上限になります。

プログラム

```
new; cls;
rndseed 911;
data={78,68,89,92,60,42,74,80,78};
times=2000; pp=0.05;
call boot(data,times,pp);
```

```
proc(3)=boot(data,times,pp);
    local n,thetahat,vartheta,thetahat_i,i,index,lowerb,upperb;
    local temp,bsthetahat,bias,corrected,bsvar,sorttheta;
    n=rows(data);
    /* Theta here is mean. */
    thetahat=meanc(data);
    vartheta=stdc(data)^2/n;
    thetahat_i=zeros(times,1);
    i=1;
    do while i<=times;
        index=ceil(n*rndu(n,1));
        temp=data[index];
        thetahat_i[i]=meanc(temp);
        i=i+1;
    endo;
    bsthetahat=meanc(thetahat_i);
    bias=bsthetahat-thetahat;
    corrected=thetahat-bias;
    bsvar=stdc(thetahat_i)^2;
    sorttheta=sortc(thetahat_i,1);
    lowerb=sorttheta[round(times*pp/2)];
    upperb=sorttheta[round(times*(1-pp/2))];
    print/lz "      # of sample=" n;
    print/lz "      sample mean=" thetahat;
    print/lz "sample variance=" vartheta;
    print/lz " bootstrap mean=" bsthetahat;
    print/lz "                bias=" bias;
    print/lz " bias-corrected=" corrected "[ thetahat-bias ]";
    print/lz "   bootstrap var=" bsvar;
    print/rz (1-pp)*100 " % Confidence Interval";
    print/lz "[      " lowerb ",      " upperb "]";
    retp(bsthetahat,bias,bsvar);
endp;
```

画面表示

```
     # of sample= 9
     sample mean= 73.444444
```

```
sample variance= 25.975309
 bootstrap mean= 73.176333
           bias=-0.26811111
 bias-corrected= 73.712556       [ thetahat-bias ]
  bootstrap var= 22.419017
              95  % Confidence Interval
[     63.111111       ,     81.666667       ]
```

Bootstrap OLS Estimates

**ペアサンプリングのケース**

この方法は、基本的に、Jackknife のやり方と同じで、ｙとｘ（ｘは複数の説明変数でも可能）をペアで考えて、そのペアを１つのもとの考えて Bootstrap するものです。基本的に Jackknife で行なったのと同じ手順でプログラムしています。index=ceil(n*rndu(n,1));によって作成した index にもとづいてｉループの中で、ｙの列とｘの行列をペアで bootstrap したものをその都度 OLS 回帰して係数を求め、times 回だけ順に bootb の変数の中に１行ずつ格納していって、最後に、それを列ごとに（複数の係数があるから複数列ある）平均してやって、bootstrap mean coefficient にしている。これを$\theta_{(.)}$とし、もとの OLS 係数と比べることによって、

$$Bias = \theta_{(.)} - \hat{\theta}$$

$$Corrected = \hat{\theta} - Bias$$

を計算します。

プログラム

```
new; cls;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];
y=data[.,1]; x=ones(28,1)~data[.,2];
call bootols(y,x,0.05,2000);



proc(0)=bootols(y,x,pp,times);
    local n,k,b,e,s2hat,varb,se,bootb,i,index,booty,bootx,lowols,upols;
    local bootmean,bootse,lowerb,upperb,lowersb,uppersb,sortb,bias,corrected;
/* OLS Estimates */
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
```

```
    e=y-x*b;
    s2hat=e'e/(n-k);
    varb=s2hat*inv(x'x);
    se=sqrt(diag(varb));
    lowols=b-cdftci(pp/2,n-k)*se;
    upols=b+cdftci(pp/2,n-k)*se;
/* Bootstrap Estimates */
    bootb=zeros(times,k);
    i=1;
    do while i<=times;
        index=ceil(n*rndu(n,1));
        booty=y[index];
        bootx=x[index,.];
        bootb[i,.]=( inv(bootx'bootx)*bootx'booty )';
        i=i+1;
    endo;
    bootmean=meanc(bootb);
    bootse=stdc(bootb);
    bias=bootmean-b;
    corrected=b-bias;
/* Assuming t-distribution */
    lowerb=b-cdftci(pp/2,n-k)*bootse;
    upperb=b+cdftci(pp/2,n-k)*bootse;
/* Without assuming distribution */
    sortb=zeros(times,k);
    i=1;
    do while i<=k;
        sortb[.,i]=sortc(bootb[.,i],1);
        i=i+1;
    endo;
    lowersb=sortb[round(times*pp/2),.]';
    uppersb=sortb[round(times*(1-pp/2)),.]';
/* Comment and Display */
    print "                                    OLS Estimates";
    print "     coefficients          se          lower          upper ";;
    print b~se~lowols~upols;
```

```
    print "s2=" s2hat;
    print "                                                    Bootstrap";
    print/rz "                                " (1-pp)*100 " % Confidence Interval";
    print "                                                    Assuming t-dist";
    print "     coefficients      se(bootstrap)          lower          upper ";;
    print b~bootse~lowerb~upperb;
    print "                                                    Bootstrap";
    print/rz "                                " (1-pp)*100 " % Confidence Interval";
    print "                                                      Without assuming dist";
    print "     coefficients      se(bootstrap)          lower          upper ";;
    print b~bootse~lowersb~uppersb;
    print;
    print "   bootstrap mean";
    print "     coefficients              bias     bias-corrected";;
    print bootmean~bias~corrected;
endp;
```

画面表示

```
                                OLS Estimates
    coefficients               se             lower             upper
       77.795198        22.037400        32.496674        123.09372
       52.009829        3.8317273        44.133600        59.886057
s2=        2738.5337

                                                  Bootstrap
                                          95  % Confidence Interval
                                                  Assuming t-dist
    coefficients    se(bootstrap)             lower             upper
       77.795198        21.479612        33.643224        121.94717
       52.009829        4.6994923        42.349884        61.669773
                                                  Bootstrap
                                          95  % Confidence Interval
                                                Without assuming dist
    coefficients    se(bootstrap)             lower             upper
       77.795198        21.479612        32.782281        119.11462
       52.009829        4.6994923        41.953535        60.120065


  bootstrap mean
```

| coefficients | bias | bias-corrected |
|---|---|---|
| 78.808450 | 1.0132513 | 76.781947 |
| 51.653268 | -0.35656106 | 52.366390 |

**パラメトリックモンテカルロのケース**

上のものの対極に OLS のβ's と s2 を用いて、ｙ*～N(Ｘβ, $s^2 I_n$)として、ｙ*を求めたうえで、このｙ*とｘとを OLS 回帰してｂ*$_{(1)}$をもとめ、この作業を b*$_{(times)}$まで繰り返すものです。Bootstrap というよりは Monte Carlo と言った方がよいでしょう。今度は、ｉループの中で、OLS で求められた s2 とβにもとづいて、s2 の分散の正規乱数を発生させて、ｙ*を生成した上で、これともとのｘとをあらためて OLS して times 回だけシミュレーションをするものです。分布を多分に仮定したものなので、もとのデータが正規分布にしたがっていなければいないほど、計算結果は、ほかの方法の結果とずれてきます。

プログラム

```
new; cls;
rndseed 911;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];
y=data[.,1]; x=ones(28,1)~data[.,2];
call paramonteols(y,x,0.05,2000);


proc(0)=paramonteols(y,x,pp,times);
    local n,k,b,e,s2hat,varb,se,bootb,i,lowols,upols,ystar,bias,corrected;
    local bootmean,bootse,lowerb,upperb,lowersb,uppersb,sortb;
/* OLS Estimates */
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    s2hat=e'e/(n-k);
    varb=s2hat*inv(x'x);
    se=sqrt(diag(varb));
    lowols=b-cdftci(pp/2,n-k)*se;
    upols=b+cdftci(pp/2,n-k)*se;
/* Bootstrap Estimates */
    bootb=zeros(times,k);
    i=1;
```

```
    do while i<=times;
        ystar=x*b+sqrt(s2hat)*rndn(n,1); /* ystar~N(xb,s2In) */
        bootb[i,.]=( inv(x'x)*x'ystar )';
        i=i+1;
    endo;
    bootmean=meanc(bootb);
    bootse=stdc(bootb);
    bias=bootmean-b;
    corrected=b-bias;
/* Assuming t-distribution */
    lowerb=b-cdftci(pp/2,n-k)*bootse;
    upperb=b+cdftci(pp/2,n-k)*bootse;
/* Without assuming distribution */
    sortb=zeros(times,k);
    i=1;
    do while i<=k;
        sortb[.,i]=sortc(bootb[.,i],1);
        i=i+1;
    endo;
    lowersb=sortb[round(times*pp/2),.]';
    uppersb=sortb[round(times*(1-pp/2)),.]';
/* Comment and Display */
    print "                                    OLS Estimates";
    print "     coefficients          se              lower           upper ";;
    print b~se~lowols~upols;
    print "s2=" s2hat;
    print "                                                Parametric Monte Carlo";
    print/rz "                                " (1-pp)*100 " % Confidence Interval";
    print "                                                      Assuming t-dist";
    print "     coefficients    se(Monte Carlo)         lower           upper ";;
    print b~bootse~lowerb~upperb;
    print "                                                Parametric Monte Carlo";
    print/rz "                                " (1-pp)*100 " % Confidence Interval";
    print "                                                 Without assuming dist";
    print "     coefficients    se(Monte Carlo)         lower           upper ";;
    print b~bootse~lowersb~uppersb;
```

```
    print;
    print "Parametric Monte";
    print "mean coefficients                bias     bias-corrected";;
    print bootmean~bias~corrected;
endp;
```

画面表示

```
                                OLS Estimates
    coefficients              se             lower             upper
       77.795198        22.037400        32.496674         123.09372
       52.009829        3.8317273        44.133600         59.886057
s2=        2738.5337

                                      Parametric Monte Carlo
                                    95  % Confidence Interval
                                         Assuming t-dist
    coefficients   se(Monte Carlo)           lower             upper
       77.795198        22.156551        32.251756         123.33864
       52.009829        3.8734037        44.047933         59.971724
                                      Parametric Monte Carlo
                                    95  % Confidence Interval
                                      Without assuming dist
    coefficients   se(Monte Carlo)           lower             upper
       77.795198        22.156551        33.535213         119.67557
       52.009829        3.8734037        44.495156         59.623951

Parametric Monte
mean coefficients             bias    bias-corrected
       77.205195       -0.59000364         78.385202
       52.107245       0.097416395         51.912412
```

**残差サンプリングのケース**

ペアワイズのブートストラップとパラメトリックでもってｙ*を生成してからモンテカルロを行なうものとの中間のようなものに、残差だけをブートストラップしてもとの OLS のβともとの順番のｘとからｙ*を生成してから OLS を行なうことを繰り返す残差再サンプリングの方法があります。この方法も、あらかじめ OLS した結果を用いて、ｉループの中で、ｙ*を生成した上で、これともとのｘの間で times 回だけ OLS するものです。こちらは、分布は仮定していませんが、残差の散らばりを bootstrap でならそうというものです。

この方法は、極めて回帰分析に特化したやり方です。線形に限定されたものではありませんが、回帰分析ではない場合には残差を計算できないので、この方法は適応できません。

プログラム

```
new; cls;
rndseed 911;
load data[29,2]=d:datafile1.txt;
data=data[2:29,.];
y=data[.,1]; x=ones(28,1)~data[.,2];
call booteols(y,x,0.05,2000);


proc(0)=booteols(y,x,pp,times);
    local n,k,b,e,s2hat,varb,se,bootb,i,index,lowols,upols,ystar,bias,corrected;
    local bootmean,bootse,lowerb,upperb,lowersb,uppersb,sortb,boote;
/* OLS Estimates */
    n=rows(x); k=cols(x);
    b=inv(x'x)*x'y;
    e=y-x*b;
    s2hat=e'e/(n-k);
    varb=s2hat*inv(x'x);
    se=sqrt(diag(varb));
    lowols=b-cdftci(pp/2,n-k)*se;
    upols=b+cdftci(pp/2,n-k)*se;
/* Bootstrap Estimates */
    bootb=zeros(times,k);
    i=1;
    do while i<=times;
        index=ceil(n*rndu(n,1));
        boote=e[index];
        ystar=x*b+boote;
        bootb[i,.]=( inv(x'x)*x'ystar )';
        i=i+1;
    endo;
    bootmean=meanc(bootb);
    bootse=stdc(bootb);
```

```
    bias=bootmean-b;
    corrected=b-bias;
/* Assuming t-distribution */
    lowerb=b-cdftci(pp/2,n-k)*bootse;
    upperb=b+cdftci(pp/2,n-k)*bootse;
/* Without assuming distribution */
    sortb=zeros(times,k);
    i=1;
    do while i<=k;
        sortb[.,i]=sortc(bootb[.,i],1);
        i=i+1;
    endo;
    lowersb=sortb[round(times*pp/2),.]';
    uppersb=sortb[round(times*(1-pp/2)),.]';
/* Comment and Display */
    print "                                    OLS Estimates";
    print "      coefficients              se              lower              upper ";;
    print b~se~lowols~upols;
    print "s2=" s2hat;
    print "                                                Bootstrap Residuals";
    print/rz "                                " (1-pp)*100 " % Confidence Interval";
    print "                                                      Assuming t-dist";
    print "      coefficients      se(bootstrap)          lower              upper ";;
    print b~bootse~lowerb~upperb;
    print "                                                Bootstrap Residuals";
    print/rz "                                " (1-pp)*100 " % Confidence Interval";
    print "                                                  Without assuming dist";
    print "      coefficients      se(bootstrap)          lower              upper ";;
    print b~bootse~lowersb~uppersb;
    print;
    print "   bootstrap mean";
    print "      coefficients              bias     bias-corrected";;
    print bootmean~bias~corrected;
endp;
```

画面表示

OLS Estimates

```
    coefficients              se           lower           upper
       77.795198       22.037400       32.496674       123.09372
       52.009829        3.8317273      44.133600       59.886057
s2=       2738.5337

                                        Bootstrap Residuals
                                    95  % Confidence Interval
                                          Assuming t-dist
    coefficients    se(bootstrap)           lower           upper
       77.795198       21.433473       33.738064       121.85233
       52.009829        3.7236469      44.355763       59.663895
                                        Bootstrap Residuals
                                    95  % Confidence Interval
                                       Without assuming dist
    coefficients    se(bootstrap)           lower           upper
       77.795198       21.433473       38.335442       121.05712
       52.009829        3.7236469      44.585537       59.177202

  bootstrap mean
    coefficients              bias   bias-corrected
       77.759390      -0.035808649        77.831007
       51.989242      -0.020587093        52.030416
```

上の３つの Bootstrap OLS は前半の i ループで times 回計算を行なう内部のところが違うだけです。その他の部分は基本的に同じです。OLS 計算ではその都度、計算結果が列で出てきますから、それを転置して行単位で格納していっていることに注意してください。その反対に、後半の i ループでは、列ごとに違う種類の係数が入っている行列を列ごとにソートしています。上の３つの結果は、どこの部分のサンプルを Bootstrap するかによって、かなりの違いが見られます。