5.10　Jackknife ジャックナイフ　　ver.0.1

ここでは、サンプルのうちの１つ１つをその都度切り落としてその影響をみる分析方法であるジャックナイフ法についてプログラムしていきます。「ジャックナイフ」とは、その何枚も中に収納されているいろいろな形の刃を１枚 1 枚、くの字型に取り出して用途に応じて切る汎用ナイフの名前に由来にしていて、実際にサンプルのなかの１つ１つをカットして除いて分析する手法が実際のジャックナイフの形状や使い方によく似ていることからこう命名されました。実際、サンプリングを繰り返すことなくして、たった１つのサンプルから Bias や MSE を計算できるというまさに何にでも汎用できるジャックナイフのような存在です。

Jackknife Sample とその基本的なアルゴリズム

基本的には、サンプルが例えばｎ個あったとすると、その１番目を取り除いて推定量を計算し、またそれを戻して今度は２番目を取り除いて推定量を計算して、最終的にｎ番目まで取り除く作業を繰り返すものです。サンプリングをシミュレーションによって求めることはよくあることですが、サンプルから１つ１つの要素を切り落とす Jackknife の基本的アルゴリズムには乱数過程は関係ありません。したがって、モンテカルロシミュレーションとして Jackknife を取り上げるのには語弊がありますが、ここでは次にあげる Bootstrap との関連でシミュレーションの範疇として敢えて取り上げることにします。下の例では、サンプルの代表的な推定量としてまず平均を考えます。

プログラム

```
new; cls;
data={78,68,89,92,60,42,74,80,78};
call jackmean(data);

proc jackmean(data);
    local n,mean,jmean,i,index,temp,meanjmean;
    n=rows(data);
    mean=meanc(data);
    jmean=zeros(n,1);
    i=1;
    do while i<=n;
        index=zeros(n,1);
        index[i]=1;
        temp=delif(data,index);
```

```
        jmean[i]=meanc(temp);
        i=i+1;
    endo;
    meanjmean=meanc(jmean);
    print/lz "# of sample=" n;
    print/lz "sample mean=" mean;
    print "jackknife mean" jmean;
    print/lz "mean of jackknife mean=" meanjmean;
    retp(jmean);
endp;
```

画面表示

```
# of sample= 9
sample mean= 73.444444
jackknife mean
          72.875000
          74.125000
          71.500000
          71.125000
          75.125000
          77.375000
          73.375000
          72.625000
          72.875000
mean of jackknife mean= 73.444444
```

上のプログラムでは、index というｎ×１のあらかじめゼロによって確保されている領域に、ｉループによって、その都度ｉ番目に１を代入することによって、その番目だけが１でその他が０であるようなインデックス列ベクトルを作成し、それをもとに delif(data,index) によって論理で data のうち index が真である（すなわちこの場合１である）場合のデータの番目を delete して、それを(n-1)×１の temp というデータ列として、その推定量を求めて（ここでは平均を求めて）それをあらかじめ領域確保されてる jmean のｉ番目の行に格納していくプログラムです。その計ｎ個の Jackknife された推定量をデータ表示するとともに、その前後に、サンプルの実際の推定量と jackknife されたそれぞれの推定量のさらなる推定量（ここでは平均の値の平均）を表示させています。ここでは、それらの値に変化はないので、この場合の Bias はゼロと判断できます。上の例は、推定量として１変数の平均をとる場合ですが、もちろんその他の Median や分散、２変数以上の場合には相関係数

や回帰係数などにもあてはめて考えることができます。

手計算

上でやっている作業は、簡単なものであって、コンピュータを必要としないでも計算できます。例えば、マークシート用紙の裏に実際の総合得点を書いたものを例えば９枚用意し、その９枚からなるサンプルを考えて、上から下に机の上に並べます。まず、１番目の用紙を右にずらして取り除き、残る８枚の平均を電卓または暗算で計算し、それをどこかに書きとめておきます。次に、その右にずらしていたものをもとに戻してから、今度は１番目のものをずらして取り除き、残る８枚の平均を計算し、また書きとめます。これを９番目まで繰り返して、書きとめた９個の平均値をみて、さらにその平均値を計算します。これを、もとの９個全部のサンプルの平均値と比べてみます。そうすると、このサンプル平均の推定量に関しては、計算間違いがない限りそれらは完全に一致することがわかるでしょう。これが Jackknife の基本です。

ここで、単一変数の場合のサンプルの平均μも分散σ^2/nもこの Jackknife サンプルによって変化しないことをフォーマルに示しておきます。まったくおもしろくない結果が出るのですが、そのことがその後の分析の基礎と動機づけになります。いま、θ(i)をある推定量を計算するのにｉ番目のデータをサンプルから取り除いて計算された推定量、θ(.)をそれらのすべてを足し合わせて個数で割った平均として、

$$Bias = (n-1)(\hat{\theta}_{(.)} - \hat{\theta})$$

$$\tilde{\theta}_{corrected} = \hat{\theta} - Bias$$

$$Var_{jack} = \frac{n-1}{n}\sum_{i=1}^{n}[\hat{\theta}_{(i)} - \hat{\theta}_{(.)}]^2$$

を計算します。pseudo value 擬似値を計算し直して復元することもよくなされます。

$$\tilde{\theta}_i = \hat{\theta} - (n-1)(\hat{\theta}_{(i)} - \hat{\theta}) = n\hat{\theta} - (n-1)\hat{\theta}_{(i)} \qquad i = 1,2,...,n$$

プログラム

```
new; cls;
data={78,68,89,92,60,42,74,80,78};
call jack(data);

proc(3)=jack(data);
    local n,thetahat,vartheta,thetahat_i,i,index,temp;
    local jthetahat,bias,corrected,jvar,pseudo;
    n=rows(data);
```

```
    /* Theta here is mean. */
    thetahat=meanc(data);
    vartheta=stdc(data)^2/n;
    thetahat_i=zeros(n,1);
    i=1;
    do while i<=n;
        index=zeros(n,1);
        index[i]=1;
        temp=delif(data,index);
        thetahat_i[i]=meanc(temp);
        i=i+1;
    endo;
    jthetahat=meanc(thetahat_i);
    bias=(n-1)*(jthetahat-thetahat);
    corrected=thetahat-bias;
    jvar=((n-1)/n)*sumc((thetahat_i-jthetahat)^2);
    pseudo=n*thetahat-(n-1)*thetahat_i;
    print/lz "     # of sample=" n;
    print/lz "     sample mean=" thetahat;
    print/lz "sample variance=" vartheta;
    print/lz "jackknife mean=" jthetahat;
    print/lz "               bias=" bias;
    print/lz "bias-corrected=" corrected "[ thetahat-bias ]";
    print/lz " jackknife var=" jvar;
    print/rz "pseudo values:      [ n*thetahat-(n-1)*thetahat_i ]" pseudo;
    retp(jthetahat,bias,jvar);
endp;
```

画面表示

```
    # of sample= 9
    sample mean= 73.444444
sample variance= 25.975309
jackknife mean= 73.444444
              bias= 0
bias-corrected= 73.444444          [ thetahat-bias ]
 jackknife var= 25.975309
```

pseudo values:　[n*thetahat-(n-1)*thetahat_i]

78
68
89
92
60
42
74
80
78

上のように、Bias は０であって修正を必要としません。また、擬似値もそれぞれが元のサンプルのそれぞれの要素に一致します。単一変数の推定量を考える場合、この Jackknife によって、平均μも分散σ^2/nも変化がないことが示されました。

Jackknife によって Bias 補正された相関係数

上のように単一変数の場合は、Jackknife によってサンプル平均とサンプル分散は変化しないことを念頭において、そこを出発点にして、今度はサンプルが２つの変数の組からなるものとして、その平均と分散の代数的計算によって求められる相関係数を Jackknife によって求めて Bias を補正した相関係数を導き出しましょう。今度は Bias はゼロにはなりません。（平均や分散は smooth な推定量の典型ですが、相関係数や Median などはそうでないものの範疇に入りますが、ここではそれらの結果がうまくいくのかうまくいかないのかの違いについては深く立ち入らないことにします。）

プログラム

```
new; cls;
data1={78,68,89,92,60,42,74,80,78};
data2={52,55,48,40,92,90,60,70,65};
data=data1~data2;
call jack(data,0.05);

proc(3)=jack(data,pp);
    local n,thetahat,corr,thetahat_i,i,index,temp;
    local jthetahat,bias,corrected,jvar,lowerb,upperb;
    n=rows(data);
    /* Theta here is correlation coefficient. */
    corr=corrx(data);
    thetahat=corr[2,1];
```

```
    thetahat_i=zeros(n,1);
    i=1;
    do while i<=n;
        index=zeros(n,1);
        index[i]=1;
        temp=delif(data,index);
        corr=corrx(temp);
        thetahat_i[i]=corr[2,1];
        i=i+1;
    endo;
    jthetahat=meanc(thetahat_i);
    bias=(n-1)*(jthetahat-thetahat);
    corrected=thetahat-bias;
    jvar=((n-1)/n)*sumc((thetahat_i-jthetahat)^2);
    lowerb=thetahat-cdfni(1-pp/2)*sqrt(jvar);
    upperb=thetahat +cdfni(1-pp/2)*sqrt(jvar);
    print/lz "                                  # of sample=" n;
    print/lz "     sample correlation coefficient=" thetahat;
    print/lz "jackknife correlation coefficient=" jthetahat;
    print/lz "                                          bias=" bias;
    print/lz "           bias-corrected coefficient=" corrected;
    print/lz "                       jackknife variance=" jvar;
    print/rz (1-pp)*100 "% Confidence Interval    ";
    print "          [" lowerb upperb "     ]";
    retp(jthetahat,bias,jvar);
endp;
```

画面表示

```
                              # of sample= 9
    sample correlation coefficient=-0.84514297
jackknife correlation coefficient=-0.8441617
                                      bias= 0.0078501596
          bias-corrected coefficient=-0.85299313
                    jackknife variance= 0.0084539974
                   95 % Confidence Interval
        [         -1.0332032          -0.67278302       ]
```

上のプログラムでは、Bias 補正された相関係数を計算するとともに、さらに、もとの分布

が正規分布をしていると仮定した際の 95%信頼区間もあわせて表示させています。すなわち、ここでは、

$$\left(\hat{\theta} - 1.96 \times SE_{jack}, \hat{\theta} + 1.96 \times SE_{jack}\right)$$

を考えます。SE の部分は、ここでは Jackknife による分散から求めています。また、θ のところはハットつきのもとの完全サンプルの相関係数の推定値であって、Jackknife による相関係数でも補正されたものでもないことに注意してください。ここでは、もとのサンプルの相関係数が計算されていて、それに対する Jackknife 信頼区間を計算しているものだと考えてください。

Jackknife OLS Estimates

今度は、通常の多変数の OLS をするのに Jackknife をほどこしたものを考えます。

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 jackols(y,x);

proc(0)=jackols(y,x);
    local n,k,b,e,s2hat,varb,se,jackb,i,index,jacky,jackx;
    local jackmean,jackse,jackbias,corrected;
    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));
    print "OLS Estimates";
    print b';
    print "OLS Standard Errors";
    print se';
    print;

    jackb=zeros(n,k);
    i=1;
```

```
    do while i<=n;
        index=zeros(n,1);
        index[i]=1;
        jacky=delif(y,index);
        jackx=delif(x,index);
        jackb[i,.]=( inv(jackx'jackx)*jackx'jacky )';
        i=i+1;
    endo;
    print "Jackknife OLS Estimates";;
    print jackb;

    jackmean=meanc(jackb);
    print "Mean of Jackknife Estimates";
    print jackmean';

    jackse=sqrt( ((n-1)/n)*sumc((jackb-jackmean')^2) );
    print "Jackknife Standard Errors";
    print jackse';
    print;

    jackbias=(n-1)*(jackmean-b);
    print "Jackknife Bias";
    print jackbias';

    corrected=b-jackbias;
    print "Jackknife Bias-corrected Estimates";
    print corrected';
endp;
```

画面表示

```
OLS Estimates
        77.795198           52.009829
OLS Standard Errors
        22.037400           3.8317273

Jackknife OLS Estimates
```

77.195500	52.212228
76.331997	52.681190
76.827839	52.394101
92.568472	48.053803
78.754232	52.131027
77.502996	52.177612
79.700838	52.018721
76.129341	52.198248
79.169550	52.172564
70.926925	53.002898
84.905666	50.000349
74.718519	52.424841
73.150258	52.584153
77.241350	52.394565
82.246161	51.487521
73.236296	52.614765
69.202768	53.153872
74.036514	52.483274
82.579582	50.602870
75.185270	52.338088
79.635490	51.792582
78.061816	51.974908
77.078722	52.689766
76.745242	52.683357
76.153005	52.242082
80.359000	51.830158
79.380070	51.516799
80.547869	51.876491

Mean of Jackknife Estimates

77.841832	51.990458

Jackknife Standard Errors

22.875981	5.1615821

Jackknife Bias

1.2590996	-0.52299893

Jackknife Bias-corrected Estimates

76.536099 52.532828

さらに上と同じことに加えて、もとの OLS 係数が自由度(n－k)のｔ分布にしたがうものと仮定して、Jackknife による SE を利用して、その 95%信頼区間を出すプログラムに変更してみましょう。今度は、今までＺを考えていたところが自由度 n-k のｔ分布になります。

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 jackols(y,x,0.05);

proc(0)=jackols(y,x,pp);
    local n,k,b,e,s2hat,varb,se,jackb,i,index,jacky,jackx;
    local jackmean,jackse,jackbias,corrected,lowerb,upperb;
    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));
    jackb=zeros(n,2);
    i=1;
    do while i<=n;
        index=zeros(n,1);
        index[i]=1;
        jacky=delif(y,index);
        jackx=delif(x,index);
        jackb[i,.]=( inv(jackx'jackx)*jackx'jacky )';
        i=i+1;
    endo;
    jackmean=meanc(jackb);
    jackbias=(n-1)*(jackmean-b);
    corrected=b-jackbias;
    jackse=sqrt( ((n-1)/n)*sumc((jackb-jackmean')^2) );
    lowerb=b-cdftci(pp/2,n-k)*jackse;
```

```
    upperb=b+cdftci(pp/2,n-k)*jackse;
    print "                                                    Jackknife";
    print/rz "                                  " (1-pp)*100 " % Confidence Interval";
    print "     coefficients     se(jackknife)          lower          upper ";;
    print b~jackse~lowerb~upperb;
    print "              Bias";;
    print jackbias;
    print "   Bias-Corrected";;
    print corrected;
endp;
```

画面表示

		Jackknife	
		95 % Confidence Interval	
coefficients	se(jackknife)	lower	upper
77.795198	22.875981	30.772947	124.81745
52.009829	5.1615821	41.400045	62.619613
Bias			
1.2590996			
-0.52299893			
Bias-Corrected			
76.536099			
52.532828			

プログラムでは、ｔに Jackknife の SE をかけて True Parameter から差し引きすることによって信頼区間を計算するのですが、ｔのところの自由度は依然として全データを考えた自由度の n-k になることに注意してください。

delete-d Jackknife

これまでの Jackknife は delete-one に相当するもので、サンプルからただ１つのデータをカットしていくものでした。ここでは、その一般形のｄ個のデータをカットするものをプログラムします。ここで、ｄ個カットするのであれば、Jackknife Sample は、これまで n-1 の長さでｎ個あったものが、今度は n-d の長さで ${}_{n}C_{d}$ 個になります。下のプログラムでは、後半の combi01 の procedure が構造上メモリを著しく消費するのでサンプルのデータの個数は Light 版では９個以下までしかまわりませんので注意してください。なお、ｄ個ずつカットするそれぞれのグループのｉ番目を考えて、delete-one と同じように考えると

$$Var_{jack} = \frac{n-d}{d\binom{n}{d}} \sum [\hat{\theta}_{(i)} - \hat{\theta}_{(.)}]^2$$

$$Bias = (n-d)(\hat{\theta}_{(.)} - \hat{\theta})$$

ここで、$\binom{n}{d} = nCd$

と delete-one のものを変更する必要があります。下の場合は、ｄ＝３について考えます。

プログラム

```
new; cls;
data={78,68,89,92,60,42,74,80,78};
call jack(data,3);

proc(3)=jack(data,d);
    local n,thetahat,vartheta,thetahat_i,i,index,temp;
    local jthetahat,bias,corrected,jvar,total;
    n=rows(data); total=n!/((n-d)!*d!);
    /* Theta here is mean. */
    thetahat=meanc(data);
    vartheta=stdc(data)^2/n;
    thetahat_i=zeros(total,1);
    index=combi01(n,d);     /* print/rz index; */
    i=1;
    do while i<=total;
        temp=delif(data,index[i,.]');
        thetahat_i[i]=meanc(temp);
        i=i+1;
    endo;
    jthetahat=meanc(thetahat_i);
    bias=(n-d)*(jthetahat-thetahat);
    corrected=thetahat-bias;
    jvar=((n-d)/(d*total))*sumc((thetahat_i-jthetahat)^2);
    print/lz "     # of sample=" n;
    print/lz "     sample mean=" thetahat;
    print/lz "sample variance=" vartheta;
    print/lz " jackknife mean=" jthetahat;
```

```
    print/ld "                bias=" bias;
    print/lz "    bias-corrected=" corrected "[ thetahat-bias ]";
    print/lz "     jackknife var=" jvar;
    retp(jthetahat,bias,jvar);
endp;

/*
** Binary sequence matrix of nCr combination
** (C) Copyright 2002 Yosuke Amijima. All Rights Reserved.
**
** PROC COMBI01
**
** FORMAT
**          comb = combi01(n,r)
** INPUT
**          n - total number considered. We here think of nCr.
**          r - number of times taken from total number.
** OUTPUT
**          comb - nCr x n matrix(nCr row vectors of binary sequence).
** Remarks:
**          This algorithm is pretty simple and short, but it really takes
**          lots of memory to run. GAUSS Light would work only when n<=9.
*/
proc combi01(n,r);
    local comb,x,i,temp,index;
    comb=zeros(2^n,n);
    x=seqa(1,1,2^n);
    i=1;
    do while i<=n;
        temp=x;
        x=floor(x/2);
        comb[.,i]=temp-2*x;
        i=i+1;
    endo;
    index=zeros(2^n,1);
    i=1;
```

```
    do while i<=2^n;
        if sumc(comb[i,.]')==r;
            index[i]=1;
        endif;
        i=i+1;
    endo;
    comb=selif(comb,index);
    retp(comb);
endp;
```

画面表示

```
     # of sample= 9
     sample mean= 73.444444
sample variance= 25.975309
 jackknife mean= 73.444444
           bias= 0.00000000
 bias-corrected= 73.444444        [ thetahat-bias ]
  jackknife var= 25.975309
```

なお、うまく index がついているか確認するためには、index=combi01(n,d);の行のあとに続く　/* print/rz index; */　のところを /* */ をはずして画面表示させてみてください。

<u>delete-p percentage Jackknife</u>

同じことを delete-ｐパーセント Jackknife にするには、冒頭の部分で下のように input に個数ｄの代わりにパーセントの小数値ｐを代入して、procedure の内部でｐをｄに変換して、ｄをローカル変数に加えればできます。

プログラム

```
new; cls;
data={78,68,89,92,60,42,74,80,78};
call jackp(data,0.3);

proc jackp(data,p);
    local n,thetahat,vartheta,thetahat_i,i,index,temp;
    local jthetahat,bias,corrected,jvar,total,d;
    n=rows(data); d=round(n*p); total=n!/((n-d)!*d!);
    /* Theta here is mean. */
    thetahat=meanc(data);
    vartheta=stdc(data)^2/n;
```

```
    thetahat_i=zeros(total,1);
    index=combi01(n,d);
    i=1;
    do while i<=total;
        temp=delif(data,index[i,.]');
        thetahat_i[i]=meanc(temp);
        i=i+1;
    endo;
    jthetahat=meanc(thetahat_i);
    bias=(n-d)*(jthetahat-thetahat);
    corrected=thetahat-bias;
    jvar=((n-d)/(d*total))*sumc((thetahat_i-jthetahat)^2);
    print/lz "       # of sample=" n;
    print/lz "       sample mean=" thetahat;
    print/lz "sample variance=" vartheta;
    print/lz " jackknife mean=" jthetahat;
    print/ld "                bias=" bias;
    print/lz " bias-corrected=" corrected "[ thetahat-bias ]";
    print/lz "    jackknife var=" jvar;
    retp(jthetahat);
endp;
```

(以下に combi01 の procedure を置く　)

ここで、相関係数を求める計算に戻って、ｄを１から５まで変化させるにしたがって Bias および Jackknife Variance がどう変化するのかを見てみましょう。プログラム的には、冒頭部でｄを１から５まで combi01(data,d,pp)を DO ループでまわして呼び出します。

プログラム

```
new; cls;
data1={78,68,89,92,60,42,74,80,78};
data2={52,55,48,40,92,90,60,70,65};
data=data1~data2;
d=1;
do while d<=5;
    print/lz "delete" d;
    call jack(data,d,0.05);
    print;
```

```
    d=d+1;
endo;

proc jack(data,d,pp);
    local n,thetahat,corr,thetahat_i,i,index,temp,total;
    local jthetahat,bias,corrected,jvar,lowerb,upperb;
    n=rows(data); total=n!/((n-d)!*d!);
    /* Theta here is correlation coefficient. */
    corr=corrx(data);
    thetahat=corr[2,1];
    thetahat_i=zeros(total,1);
    index=combi01(n,d);
    i=1;
    do while i<=total;
        temp=delif(data,index[i,.]');
        corr=corrx(temp);
        thetahat_i[i]=corr[2,1];
        i=i+1;
    endo;
    jthetahat=meanc(thetahat_i);
    bias=(n-d)*(jthetahat-thetahat);
    corrected=thetahat-bias;
    jvar=((n-d)/(d*total))*sumc((thetahat_i-jthetahat)^2);
    lowerb=corrected-cdfni(1-pp/2)*sqrt(jvar);
    upperb=corrected+cdfni(1-pp/2)*sqrt(jvar);
    print/lz "                                # of sample=" n;
    print/lz "     sample correlation coefficient=" thetahat;
    print/lz "jackknife correlation coefficient=" jthetahat;
    print/lz "                                      bias=" bias;
    print/lz "           bias-corrected coefficient=" corrected;
    print/lz "                      jackknife variance=" jvar;
    print/rz (1-pp)*100 "% Confidence Interval    ";
    print "          [" lowerb upperb "     ]";
    retp(jthetahat);
endp;
endp;
```

(以下に combi01 の procedure を置く　)
画面表示

```
delete 1
                    # of sample= 9
   sample correlation coefficient=-0.84514297
jackknife correlation coefficient=-0.8441617
                            bias= 0.0078501596
       bias-corrected coefficient=-0.85299313
                jackknife variance= 0.0084539974
               95 % Confidence Interval
      [       -1.0332032       -0.67278302     ]

delete 2
                    # of sample= 9
   sample correlation coefficient=-0.84514297
jackknife correlation coefficient=-0.83935404
                            bias= 0.040522536
       bias-corrected coefficient=-0.88566551
                jackknife variance= 0.016363933
               95 % Confidence Interval
      [       -1.1363872       -0.63494380     ]

delete 3
                    # of sample= 9
   sample correlation coefficient=-0.84514297
jackknife correlation coefficient=-0.8291234
                            bias= 0.096117452
       bias-corrected coefficient=-0.94126042
                jackknife variance= 0.026395931
               95 % Confidence Interval
      [       -1.2596923       -0.62282851     ]

delete 4
                    # of sample= 9
   sample correlation coefficient=-0.84514297
jackknife correlation coefficient=-0.80977135
```

```
                          bias= 0.17685811
      bias-corrected coefficient=-1.0220011
               jackknife variance= 0.045771455
              95 % Confidence Interval
     [       -1.4413210       -0.60268120     ]

delete 5
                      # of sample= 9
   sample correlation coefficient=-0.84514297
jackknife correlation coefficient=-0.77338211
                          bias= 0.28704345
      bias-corrected coefficient=-1.1321864
               jackknife variance= 0.07891547
              95 % Confidence Interval
     [       -1.6827775       -0.58159534     ]
```

上のように、delete-d のｄの数が増すにしたがって Bias も Jackknife Variance も増加していることがわかります。

シミュレーションによる分析

Jackknife 自身は乱数過程とは無縁のものですが、データの設定に乱数過程を導入し今までどおりのシミュレーションをしてみます。上の場合と同じく２変数の相関係数についてプログラムします。乱数設定は、次のように

ｎ	Ｘ	u	Ｙ	有意水準	times
9	N(0,1)	N(0,1)	Ｘ＋u	5%	100

として、delete-d のｄの数が１からｎの半分の数まで増加するときの Bias と Jackknife 分散の変化を観察しグラフ化します。なお、Ｘと u は独立に標準正規分布にしたがいますから、Ｙ＝Ｘ＋u から得られるＹは、たいていの場合、Ｘと正の相関をしているはずです。具体的には、相関係数を求める計算をそれぞれのｄについて 100 回ずつ試行してその平均をとってやります。

プログラム

```
new; cls;
call jsim(0.05,9,100);

proc(0)=jsim(pp,n,times);
    local dmax,mbias,mjvar,i,x,u,y,data,j,jthetahat,bias,jvar;
    dmax=round(n/2);
```

```
    mbias=zeros(times,dmax); mjvar=zeros(times,dmax);
    i=1;
    do while i<=times;
        x=rndn(n,1);
        u=rndn(n,1);
        y=x+u;
        data=x~y;
        j=1;
        do while j<=dmax;
            {jthetahat,bias,jvar}=jack(data,j,pp);
            mbias[i,j]=bias; mjvar[i,j]=jvar;
            j=j+1;
        endo;
        i=i+1;
    endo;
    mbias=meanc(mbias); mjvar=meanc(mjvar);
    print/rz seqa(1,1,dmax)~mbias~mjvar;
    library pgraph;
    graphset;
    _plctrl=1;
    xlabel("Number of deleting: d");
    _plegctl=1;
    _plegstr="Bias¥000Jackknife Variance";
    xy(seqa(1,1,dmax),mbias~mjvar);
endp;

proc(3)=jack(data,d,pp);
    local n,thetahat,corr,thetahat_i,i,index,temp,total;
    local jthetahat,bias,corrected,jvar,lowerb,upperb;
    n=rows(data); total=n!/((n-d)!*d!);
    /* Theta here is correlation coefficient. */
    corr=corrx(data);
    thetahat=corr[2,1];
    thetahat_i=zeros(total,1);
    index=combi01(n,d);
    i=1;
```

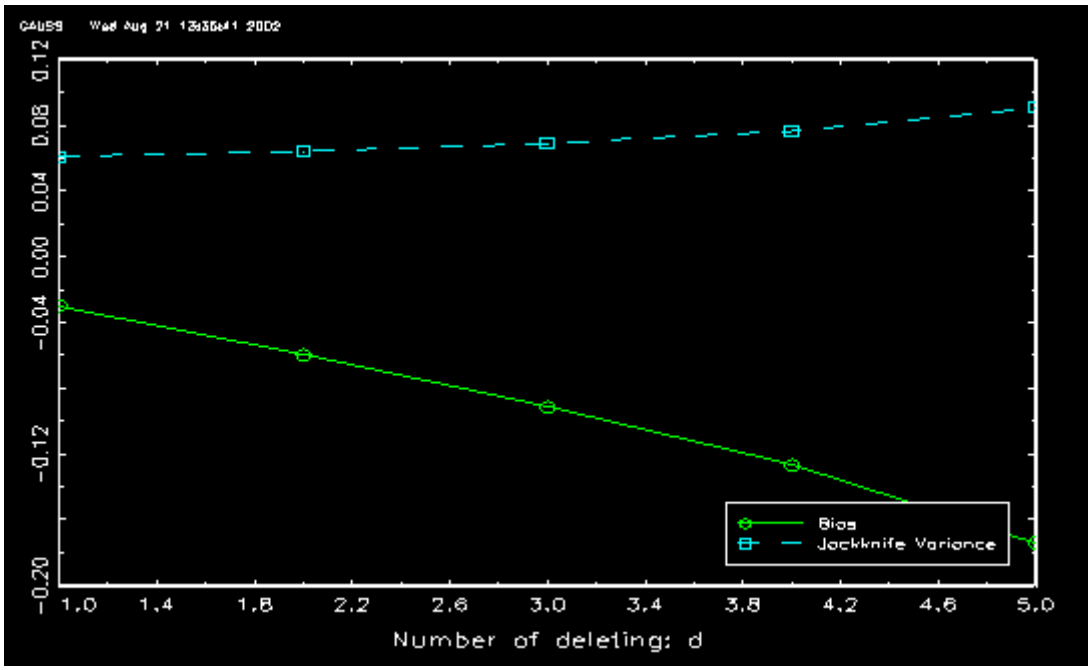
```
    do while i<=total;
        temp=delif(data,index[i,.]');
        corr=corrx(temp);
        thetahat_i[i]=corr[2,1];
        i=i+1;
    endo;
    jthetahat=meanc(thetahat_i);
    bias=(n-d)*(jthetahat-thetahat);
    corrected=thetahat-bias;
    jvar=((n-d)/(d*total))*sumc((thetahat_i-jthetahat)^2);
    lowerb=corrected-cdfni(1-pp/2)*sqrt(jvar);
    upperb=corrected+cdfni(1-pp/2)*sqrt(jvar);
    print/lz "                                # of sample=" n;
    print/lz "     sample correlation coefficient=" thetahat;
    print/lz "jackknife correlation coefficient=" jthetahat;
    print/lz "                                      bias=" bias;
    print/lz "           bias-corrected coefficient=" corrected;
    print/lz "                      jackknife variance=" jvar;
    print/rz (1-pp)*100 "% Confidence Interval    ";
    print "          [" lowerb upperb "     ]";
    retp(jthetahat,bias,jvar);
endp;
```

(以下に combi01 の procedure を置く　)

画面表示

1	-0.029857809	0.061024345
2	-0.060037034	0.064255864
3	-0.091432958	0.068964632
4	-0.12715172	0.076860778
5	-0.17398138	0.090904427

グラフ表示



上で示されているように delete-d のｄの数が増加するにつれて、「絶対値で」Bias も分散も増加していることがわかります（ここでは、相関係数はプラスになっているので以前の結果とは逆に Bias は負の値になっています)。

Delete-d Jackknife OLS

以下では定数項を含めて３説明変数の OLS についてデータを乱数設定して、delete-d のｄ＝３の場合の Jackknife OLS を試みます。メモリの制約から n＝９ですので定数項は大幅にぶれます。乱数設定は、

n	β_0	β_1	β_2	X_1	X_2	u	d
9	1	2	4	[0,5]	[0,5]	N(0,1)	3

として、delete-3 の Jackknife OLS を試みます。

プログラム

```
new; cls;
rndseed 1;
n=9;
beta={1,2,4}; x=ones(n,1)~5*rndu(n,1)~5*rndu(n,1); u=rndn(n,1);
y=x*beta+u;
call jackols(y,x,0.05,3);

proc(0)=jackols(y,x,pp,d);
    local n,k,b,e,s2hat,varb,se,jackb,i,index,jacky,jackx,total;
    local jackmean,jackse,jackbias,corrected,lowerb,upperb;
```

```
    n=rows(x); k=cols(x); total=n!/((n-d)!*d!);
    b=inv(x'x)*x'y;
    e=y-x*b;
    s2hat=e'e/(n-k);
    varb=s2hat*inv(x'x);
    se=sqrt(diag(varb));
    jackb=zeros(total,k);
    index=combi01(n,d);
    i=1;
    do while i<=total;
        jacky=delif(y,index[i,.]');
        jackx=delif(x,index[i,.]');
        jackb[i,.]=( inv(jackx'jackx)*jackx'jacky )';
        i=i+1;
    endo;
    jackmean=meanc(jackb);
    jackbias=(n-d)*(jackmean-b);
    corrected=b-jackbias;
    jackse=sqrt( ((n-d)/(d*total))*sumc((jackb-jackmean')^2) );
    lowerb=b-cdftci(pp/2,n-k)*jackse;
    upperb=b+cdftci(pp/2,n-k)*jackse;
    print "                                                    Jackknife";
    print/rz "                                  " (1-pp)*100 " % Confidence Interval";
    print "     coefficients     se(jackknife)          lower          upper ";;
    print b~jackse~lowerb~upperb;
    print "                Bias";;
    print jackbias;
    print "   Bias-Corrected";;
    print corrected;
endp;
```

画面表示(ｄ＝３のケース)

```
                                                    Jackknife
                                          95  % Confidence Interval
    coefficients     se(jackknife)              lower              upper
      0.67598558         1.3599924        -2.6517960         4.0037671
       1.6401645        0.56451718         0.25884071         3.0214883
```

4.3972466	0.52448253	3.1138841	5.6806091

Bias
-0.47882785
-0.41153929
0.59777658

Bias-Corrected
1.1548134
2.0517038
3.7994700