{One variable program}

```
{$N+,E+}
Program Mix1;
Const Num   = 40;
      b0    = 0.912287;
      b1    = -0.00743;
      P     = 2;
      Times = 1000;

Type Data   = Array[1..Num,1..2] of real;
     Cap    = Array[1..Num] of real;
     Caps   = Array[1..Num] of double;
     Sing   = Array[1..2] of real;
     PM     = Array[1..2,1..Num] of real;
     Two    = Array[1..2,1..2] of real;
     Matrix = Array[1..Num,1..Num] of real;

Var Expo  : Data;
    VB    : Matrix;
    Prime : PM;
    ycap  : Caps;
    y,Err,UBr,xx : Cap;
    Brr,BRD,Br,Br1   : Sing;
    RmseMLE1,DevMLE1 : Real;
    Zeta,Beta,Elfa,std,mean    : Real;
    beta0,beta1,DevDF1,RmseDF1 : double;
   Row,Col,j,ratio,ch_howto   : Integer;
    DevRD1,RmseRD1,RmseSum,DevSum  : Double;
    HBr,InsHBr,Lamda,BetaR,BetaInv : Two;
```

```
    RmseSumMLE,DevSumMLE,RmseSumDF,DevSumDF,RmseSumRD,DevSumRD : Double;

Procedure Exponential(Zeta:real; Var Expo:Data);
Var x : real;
Begin
  randomize;
  For Row:=1 to Num do
  Begin
    x := (-Zeta)*ln(1-random);
    Expo[Row,2] := x;
    Expo[Row,1] := 1;
  End;
  For Row:=1 to Num Do
  Begin
    For Col:=1 to 2 Do
  End;
End;  {Procedure Exponential}

Procedure Start_Expo(Var Zeta:real);
Var ch_expo : integer;
Begin
  repeat
    Zeta := 0;
    writeln('  1 ==> Zeta = 0.5 ');
    writeln('  2 ==> Zeta = 1 ');
    writeln('  3 ==> Zeta = 2 ');
    write('  Enter Zeta : ');
    readln(ch_expo);
    case ch_expo of
        1 : Zeta:=0.5;
```

```
          2 : Zeta:=1;
          3 : Zeta:=2;
          else Begin
               writeln('Please enter number 1-3 only');
               End; {else}
      end; {case}
   until (ch_expo=1) or (ch_expo=2) or (ch_expo=3);
End; {Procedure Start_Expo}

Procedure Weibul(Beta,Elfa:real; Var Expo:Data);
Var a,b,x : real;
Begin
   randomize;
   a := 1/Elfa;
   For Row:=1 to Num do                                    {Find X with Exponential}
   begin
      b := -ln(1-random);
      x := Beta*Exp(a*ln(b));
      Expo[Row,2] := x;
      Expo[Row,1] := 1;
   end;
   For Row:=1 to Num Do
   Begin
      For Col:=1 to 2 Do
   End
End;   {Procedure Weibul}

Procedure Start_Weibul(Var Beta,Elfa:real);
Var ch_weibul,Fi : integer;
Begin
```

```
repeat
   Fi := 0;
   writeln('  1 ==> Fi = 1 ');
   writeln('  2 ==> Fi = 4 ');
   writeln('  3 ==> Fi = 8 ');
   writeln('  4 ==> Fi = 16 ');
   write('  Enter Fi of Weibul : ');
   readln(ch_weibul);
   case ch_weibul of
       1 : Fi := 1;
       2 : Fi := 4;
       3 : Fi := 8;
       4 : Fi := 16
       else begin
              writeln('Please enter number 1-4 only');
            end; {else}
   end; {case}
until (ch_weibul=1) or (ch_weibul=2) or (ch_weibul=3) or (ch_weibul=4);
Elfa := 4*Fi;
repeat
   Beta := 0;
   writeln('Select Beta for X of Weibul');
   writeln('  1 ==> Beta = (1/8)u ');
   writeln('  2 ==> Beta = (1/2)u ');
   writeln('  3 ==> Beta = 2u ');
   write('  Enter Beta : ');
   readln(ch_weibul);
   case ch_weibul of
       1 : Beta := (1/8)*Elfa;
       2 : Beta := (1/2)*Elfa;
```

```
        3 : Beta := 2*Elfa;
        else Begin
            writeln('Please enter number 1-3 only');
            End; {else}
    end; {case}
  until (ch_weibul=1) or (ch_weibul=2) or (ch_weibul=3);
End; {Procedure Start_Weibul}

Procedure Normal(std,mean:real; Var Expo:Data);
Var r1,r2,y1,y2,a,b : real;
   x1,x2 : real;
Begin
  randomize;
  For Row:=1 to Num do
  Begin
    r1 := random;
    r2 := random;
    a  := -2*ln(1-r1);
    b  := 2*pi*r2;
    y1 := sqrt(a)*cos(b);
    y2 := sqrt(a)*sin(b);
    x1 := y1*std+mean;
    x2 := y2*std+mean;
    if x1<x2 then Expo[Row,2] := x1
      else Expo[Row,2] := x2;
    Expo[Row,1] := 1;
  End;
  For Row:=1 to Num Do
  Begin
    For Col:=1 to 2 Do
```

```
  End;
End;   {Procedure Normal}

Procedure Start_Normal(Var std,mean:real);
Var ch_normal,Fi : integer;
Begin
  repeat
    Fi := 0;
    writeln('  1 ==> Fi = 1 ');
    writeln('  2 ==> Fi = 4 ');
    writeln('  3 ==> Fi = 8 ');
    writeln('  4 ==> Fi = 16 ');
    write('  Enter Fi of Normal : ');
    readln(ch_normal);
    writeln;
    case ch_normal of
        1 : Fi := 1;
        2 : Fi := 4;
        3 : Fi := 8;
        4 : Fi := 16
        else begin
               writeln('Please enter number 1-4 only');
             end; {else}
    end; {case}
  until (ch_normal=1) or (ch_normal=2) or (ch_normal=3) or (ch_normal=4);
  mean := 4*Fi;
  repeat
    Elfa := 0;
    writeln('Select Sigma for X of Normal');
    writeln('  1 ==> Sigma = (1/8)u ');
```

```
      writeln('  2 ==> Sigma = (1/2)u ');
      writeln('  3 ==> Sigma = 2u ')
      write('  Enter Sigma : ');
      readln(ch_normal);
      case ch_normal of
          1 : std := sqrt((1/8)*mean);
          2 : std := sqrt((1/2)*mean);
          3 : std := sqrt(2*mean);
          else Begin
              writeln('Please enter number 1-3 only');
              End; {else}
      end; {case}
   until (ch_normal=1) or (ch_normal=2) or (ch_normal=3);
End; {Procedure Start_Normal}

Procedure Howto;
Begin
   writeln('           Select x from');
   writeln('      Exponential  ==>  Press 1');
   writeln('      Weibul       ==>  Press 2');
   writeln('      Normal       ==>  Press 3');
   writeln('      Exit program ==>  Press 4');
   write('Enter Howto find x : ');
   readln(ch_howto);
   case ch_howto of
       1 : Start_Expo(Zeta);
       2 : Start_Weibul(Beta,Elfa);
       3 : Start_Normal(std,mean);
       4 : halt(1);
       else begin
```

```
            writeln('Please enter number 1-4 only');
          end; {else}
  end; {case}
End;

Procedure Error(Var Err:Cap);
Var x,a,b : real;
Begin
  a := -1;
  b := 1;
  randomize;
  For Row:=1 to Num do
  Begin
    x := (((b-a)*random)+a);
    Err[Row] := x ;
  end;
  For Row:=1 to Num do
End; {Error}

Procedure Find_Y(Expo:Data; Err:Cap; Var y:Cap);
Var a : real;
Begin
    For Row:=1 to Num Do
    Begin
      a := b0+(b1*Expo[Row,2])+Err[Row];
      y[Row] := Exp(a)/(1+Exp(a));
    End;
End; {Procedure Find_Y}
Procedure Select_Sort(y:Cap; Var xx:Cap);
Var i,j,k : integer;
```

```
    m :real;
Begin
    For Row:=1 to Num do
    xx[Row] := y[Row];
    For i:=1 to Num-1 do
    begin
      k := i;
      m := xx[i];
      For j:= i+1 to Num do
          If (xx[j]<m) then
          begin
            k := j;
            m := xx[j];
          end;
      xx[k] := xx[i];
      xx[i] := m;
    end;
End;

Procedure Select_Ratio(Var ratio:integer);
Begin
   Repeat
      writeln('  Choose ratio of y');
      writeln('  1 ==>  0.75 : 0.25');
      writeln('  2 ==>  0.80 : 0.20');
      writeln('  3 ==>  0.85 : 0.15');
      writeln('  4 ==>  0.90 : 0.10');
      writeln('  5 ==>  0.95 : 0.05');
      write('  Enter your ratio : ');
      readln(ratio);
```

```
  until (ratio=1) or (ratio=2) or (ratio=3) or (ratio=4) or (ratio=5);
End;   {Select_ratio}

Procedure Change_Y(ratio:integer; xx:Cap; Var y:Cap);
Var xi,yi,j,cnt0 : integer;
    hscore : real;
Begin
  Case ratio of
    1 : Begin
          xi := round(Num*0.75);
          yi := trunc(Num*0.25);
        End;
    2 : Begin
          xi := round(Num*0.80);
          yi := trunc(Num*0.20);
        End;
    3 : Begin
          xi := round(Num*0.85);
          yi := trunc(Num*0.15);
        End;
    4 : Begin
          xi := round(Num*0.90);
          yi := trunc(Num*0.10);
        End;
    5 : Begin
          xi := round(Num*0.95);
          yi := trunc(Num*0.05);
        End;
  End; {Case}
  Select_Sort(y,xx);
```

```
  cnt0 := 0;
  While cnt0<yi do
   For Row:=1 to yi Do
     For j:=1 to Num Do
       If (xx[Row]=y[j]) then
       begin
         y[j] := 0;
         cnt0 := cnt0+1;
       end;
  For Row:=1 to Num Do
   If (y[Row]<>0) then y[Row] := 1;
  For Row:=1 to Num Do
End;{Procedure Change_Y}

Procedure Count_y(y:Cap);
Var Count0,Count1 : integer;
Begin
    Count0 := 0;
    Count1 := 0;
    For Row:=1 to Num Do
      If y[Row]=1 then Count1 := Count1+1
        else Count0 := Count0+1;
End;{Procedure Count_y}

Procedure Find_UBr(Expo:Data; y:Cap; Br:Sing; Var UBr:Cap);
Var a,b : real;
Begin
  a := 0;
  b := 0;
  UBr[1] := 0;
```

```
  UBr[2] := 0;
  For Row:=1 to Num do
  begin
    a := Br[1]+(Br[2]*Expo[Row,2]);
    b := Exp(a)/(1+Exp(a));
    UBr[1] := UBr[1]+(y[Row]-b);
    UBr[2] := UBr[2]+(Expo[Row,2]*(y[Row]-b));
  end;
  For Row:=1 to 2 do
End;  {Find_UBr}

Procedure Find_HBr(Expo:Data; Br:Sing; Var HBr:Two);
Var a,b : real;
Begin
  a := 0;
  b := 0;
  HBr[1,1] := 0;
  HBr[1,2] := 0;
  HBr[2,1] := 0;
  HBr[2,2] := 0;
  For Row:=1 to Num do
  begin
    b := Br[1]+(Br[2]*Expo[Row,2]);
    a := Exp(b)/sqr(1+Exp(b));
    HBr[1,1] := HBr[1,1]+a;
    HBr[1,2] := HBr[1,2]+(Expo[Row,2]*a);
    HBr[2,1] := HBr[2,1]+(Expo[Row,2]*a);
    HBr[2,2] := HBr[2,2]+(sqr(Expo[Row,2])*a);
  end;
  For Row:=1 to 2 do
```

```
    For Col:=1 to 2 do
      HBr[Row,Col] := -(HBr[Row,Col]);
  For Row:=1 to 2 do
    For Col:=1 to 2 do
    begin
        if Col>Row Then writeln;
    end;
End;  {Find_HBr}

Procedure InvsHBr(HBr:Two; Var insHBr:Two);
Var det : real;
Begin
  det := 0;
  insHBr[1,1] := 0;
  insHBr[1,2] := 0;
  insHBr[2,1] := 0;
  insHBr[2,2] := 0;
  det := (HBr[1,1]*HBr[2,2])-(HBr[1,2]*HBr[2,1]);
  insHBr[1,1] := (1/det)*HBr[2,2];
  insHBr[1,2] := (1/det)*(-HBr[1,2]);
  insHBr[2,1] := (1/det)*(-HBr[2,1]);
  insHBr[2,2] := (1/det)*HBr[1,1];
  For Row:=1 to 2 do
    For Col:=1 to 2 do
    begin
      if Col>Row Then writeln;
    end;
End;  {InvsHBr}
Procedure MLE(UBr,y:Cap; insHBr:Two; Var Br1,Br:Sing);
Var i,k : integer;
```

```
    r : real;
    C,D : Cap;
Begin
  repeat
    k := 0;
    for Row:=1 to 2 do
    begin
      r := 0;
      for i:=1 to 2 do
        r := r+insHBr[Row,i]*UBr[i];
      C[Row] := r;
    end;
    For Row:=1 to 2 do
    For Row:=1 to 2 do
    begin
      Br1[Row] := Br[Row]-C[Row];
    end;
    If (abs(Br1[1])<0.01) or (abs(Br1[2])<0.01) or (abs(C[1])<0.01) or (abs(C[2])<0.01) then
    begin
      For Row:=1 to 2 do
      k := 1;
    end
    else
    begin
      For Row:=1 to 2 do
        Br[Row] := Br1[Row];
      Find_UBr(Expo,y,Br,UBr);
      Find_HBr(Expo,Br,HBr);
      InvsHBr(HBr,insHBr);
    end;
```

```
  Until k=1;
End;  {MLE}

Procedure YCap_MLE(Expo:Data; Br1:Sing; Var ycap:Caps);
Var a : real;
Begin
  For Row:=1 to Num Do
  Begin
    a := Br1[1]+(Br1[2]*Expo[Row,2]);
    ycap[Row] := Exp(a)/(1+Exp(a));
    If ycap[Row]=0 then ycap[Row] := 0.0000000001
      else if ycap[Row]=1 then ycap[Row] := 0.9999999999;
  End;
End; {Procedure Y_Cap}

Procedure RMSE_DEV_MLE1(Expo:Data; Br1:Sing; ycap:Caps; Var RmseMLE1,DevMLE1:real);
Var rm,sum,a,b,c,d : real;
   px : Cap;
Begin
  a := 0;
  b := 0;
  c := 0;
  d := 0;
  rm := 0;
  sum  := 0;
  For Row:=1 to Num do
  begin
    px[Row] := Exp(a)/(1+Exp(a));
    b := px[Row]-ycap[Row]; }
    b := y[Row] - ycap[Row];
```

```
      rm := rm+sqr(b);
      c := 1-ycap[Row];
      d := ycap[Row]*ln(ycap[Row]/c)+ln(c);
      sum := sum+d;
    end;
    RmseMLE1 := sqrt(rm/(Num-P));
    DevMLE1 := (-2)*sum;
    writeln('    RMSE is ',RmseMLE1:4:10);
    writeln('    Deviance is ',DevMLE1:4:10);
End;

Procedure ShowMLE1(RmseMLE1,DevMLE1:real; Var RmseSumMLE,DevSumMLE:double);
Begin
    RMSE_DEV_MLE1(Expo,Br1,ycap,RmseMLE1,DevMLE1);
    RmseSumMLE := RmseSumMLE+RmseMLE1;
    DevSumMLE := DevSumMLE+DevMLE1;
End;   {ShowMLE}

Procedure ShowSumMLE(RmseSumMLE,DevSumMLE:double);
Var RmseAvg,DevAvg : double;
Begin
    RmseAvg := RmseSumMLE/Times;
    DevAvg := DevSumMLE/Times;
    writeln('          MLE1 Total ',Times,' round');
    writeln('    Average of RMSE is ',RmseAvg:2:10);
    writeln('    Average of Deviance is ',DevAvg:2:10);
End;  {ShowSum}

Procedure DF(Var beta0,beta1:double; Expo:Data; y:Cap);
Var sum0,sum1,tsum0,tsum1,avg0,avg1,sd0,sd1 : real;
```

```
    a,b,c: double;
    sigma,zeta0,zeta1,mean0,mean1 : real;
    n0,n1 : integer;
Begin
  sum0  := 0;
  sum1  := 0;
  n1    := 0;
  n0    := 0;
  tsum1 := 0;
  tsum0 := 0;
  avg0  := 0;
  avg1  := 0;
  sigma := 0;
  mean0 := 0;
  mean1 := 0;
  zeta0 := 0;
  zeta1 := 0;
  sd0 := 0;
  sd1 := 0;
  For Row:=1 to Num Do
    If (y[Row]=1) then
    begin
      n1   := n1+1;                                          {Number of y=1}
      sum1 := sum1+Expo[Row,2];      {Find sum of x at y=1}
    end
    Else
    begin
      n0   := n0+1;                                          {Number of y=0}
      sum0 := sum0+Expo[Row,2];                              {Find sum of x at y=0}
    end;
```

```
If n0=0 then
begin
  avg0 := 0;                                        {Find mean of x at y=0}
  avg1 := sum1/n1;                                  {Find mean of x at y=1}
end
else if n1=0 then
    begin
      avg0 := sum0/n0;
      avg1 := 0;
    end
    else
    begin
      avg0 := sum0/n0;
      avg1 := sum1/n1;
    end;
For Row:=1 to Num Do
  If (y[Row]=1) then
     tsum1 := tsum1+(sqr(Expo[Row,2]-avg1))
  Else
     tsum0 := tsum0+(sqr(Expo[Row,2]-avg0));
If (n0<=1) and (n1<=1) then
begin
  sd0 := 0;
  sd1 := 0;
end
Else If n0<=1 then
begin
  sd0 := 0;
  sd1 := tsum1/(n1-1);
end
```

```
Else If n1<=1 then
begin
   sd0 := tsum0/(n0-1);
   sd1 := 0;
end
Else
begin
   sd0 := tsum0/(n0-1);
   sd1 := tsum1/(n1-1);
end;
sigma := (((n0-1)*sd0)+((n1-1)*sd1))/(n0+n1-2);
If sigma<=0 then sigma := 0;
 If (sum0=0) or (n0=0) then mean0 := 0
    else mean0 := sum0/n0;
 If (sum1=0) or (n1=0) then mean1 := 0
    else mean1 := sum1/n1;
 If (n0=0) then zeta0 := 0
    else zeta0 := n0/Num;
         zeta1 := 1-zeta0;
If zeta0=0 then a := 0
else a := zeta1/zeta0;
b := sqr(mean0);
c := sqr(mean1);
If a<>0 then
begin
   beta0 := ln(a)-(0.5*((c-b)/sigma));
   beta1 := (mean1-mean0)/sigma;
end
else
begin
```

```
      beta0 := 0.5*((c-b)/sigma);
      beta1 := (mean1-mean0)/sigma;
   end;
End; {Procedure DF}

Procedure YCap_DF(Expo:Data; beta0,beta1:double; Var ycap:Caps);
Var a : double
Begin
     For Row:=1 to Num Do
     Begin
        a := beta0+(beta1*Expo[Row,2]);
        ycap[Row] := Exp(a)/(1+Exp(a));
        If ycap[Row]=0 then ycap[Row] := 0.0000000001
          Else if ycap[Row]=1 then ycap[Row] := 0.9999999999;
     End;
End; {Procedure Y_Cap}

Procedure RMSE_DEV_DF1(ycap:Caps; Expo:Data; Var RmseDF1,DevDF1:double);
Var a,b,c,d : real;
    rm,sum : double;
    px  : Cap;
Begin
     rm := 0;
     sum := 0;
     For Row:=1 to Num Do
     Begin
        b := y[Row] - ycap[Row];
        rm := rm+sqr(b);
        c := 1-ycap[Row];
        d := ycap[Row]*ln(ycap[Row]/c)+ln(c);
```

```
      sum := sum+d;
    End;
    RmseDF1 := sqrt(rm/(Num-P));
    DevDF1 := (-2)*sum;
    writeln('    RMSE is ',RmseDF1:4:15);
    writeln('    Deviance is ',DevDF1:4:15);
End;   {RMSE_DEV}

Procedure ShowDF(RmseDF1,DevDF1:Double; ycap:Caps; Expo:Data; Var
RmseSumDF,DevSumDF:double);
Begin
  RMSE_DEV_DF1(ycap,Expo,RmseDF1,DevDF1);
  RmseSumDF := RmseSumDF+RmseDF1;
  DevSumDF  := DevSumDF+DevDF1;
End;

Procedure ShowSumDF(RmseSumDF,DevSumDF:double);
Var RmseAvg,DevAvg : double;
Begin
  RmseAvg := RmseSumDF/Times;
  DevAvg := DevSumDF/Times;
  writeln('    Average of RMSE is ',RmseAvg:2:10);
  writeln('    Average of Deviance is ',DevAvg:2:10);
End; {ShowSumDF}

Function low(C:Two):real;
Var min : real;
   a : integer;
Begin
  min := C[1,1];
```

```
  For a:=1 to 2 do
    If (C[a,a]<min) then
        min := C[a,a];
  low := min;
End; {Function low}

Procedure Find_Lamda(Expo:Data; Var Lamda:Two; Var Prime:PM);
Var C : Two;
   i : integer;
   r,lscore : real;
Begin
    For Row:=1 to Num Do                                    {Find transpose of x}
    Begin
       Prime[1,Row] := 1;
       Prime[2,Row] := Expo[Row,2];
    End;
    For Row:=1 to 2 do                                      {Find x multiply prime}
      For Col:=1 to 2 do
      begin
        r := 0.0;
        For i:=1 to Num do
          r := r+Prime[Row,i]*Expo[i,Col];
        C[Row,Col] := r;
      end;
    lscore := low(C);
    For Row:=1 to 2 do                                      {Make Lamda Matrix}
      For Col:=1 to 2 do
      begin
        If Row=Col then
          Lamda[Row,Col] := lscore
```

```
          Else
            Lamda[Row,Col] := 0;
        end;
     For Row:=1 to 2 do
        For Col:=1 to 2 do
        begin
          if Col>Row Then writeln;
        end;
End;   {Lamda}

Procedure Find_VB(Expo:Data; Prime:PM; Brr:Sing; Var VB:Matrix);
Var a : real;
   px : Cap;
Begin
   For Row:=1 to Num do
   begin
     a := Brr[1]+(Brr[2]*Expo[Row,2]);
     px[Row] := Exp(a)/(1+Exp(a));
   end;
   For Row:=1 to Num do
     For Col:=1 to Num do
       If Row=Col then VB[Row,Col] := px[Row]*(1-px[Row])
         Else VB[Row,Col] := 0;
   For Row:=1 to Num do
     For Col:=1 to Num do
     begin
       If Col=Num then writeln;
     end;
End;   {Find_VB}
```

```
Procedure Find_BR(Expo:Data; Prime:PM; VB:Matrix; Var BetaR:Two);
Var A : PM;
    k,p : real;
    i : integer;
Begin
  For Row:=1 to 2 do                                                    {Find prime*VB}
    For Col:=1 to Num do
    begin
      k:= 0;
      For i:=1 to Num do
        k := k+Prime[Row,i]*VB[i,Col];
      A[Row,Col] := k;
    end;
  For Row:=1 to 2 do                                                    {Find BetaR}
    For Col:=1 to 2 do
    begin
      p:= 0;
      For i:=1 to Num do
        p := p+A[Row,i]*Expo[i,Col];
      BetaR[Row,Col] := p;
    end;
  For Row:=1 to 2 Do
    For Col:=1 to 2 Do
    Begin
      If Col>Row Then writeln;
    End;
End;   {Find_BR}

Procedure Inv(BetaR:Two; Lamda:Two; Var BetaInv:Two);
Var det : real;
```

```
    Z : Two;
Begin
  For Row:=1 to 2 do
    For Col:=1 to 2 do
      Z[Row,Col] := BetaR[Row,Col]+(2*Lamda[Row,Col]);
  det := (Z[1,1]*Z[2,2])-(Z[1,2]*Z[2,1]);
  BetaInv[1,1] := Z[2,2]/det;
  BetaInv[1,2] := -(Z[1,2])/det;
  BetaInv[2,1] := -(Z[2,1])/det;
  BetaInv[2,2] := Z[1,1]/det;
  For Row:=1 to 2 Do
    For Col:=1 to 2 Do
    Begin
      If Col>Row Then writeln;
    End;
End;  {Inv}

Procedure Find_BRD(BetaInv,BetaR:Two; Expo:Data; Var BRD:Sing);
Var M : Two;
    t,u : real;
    i,k : integer;
Begin
  repeat
  k := 0;
  For Row:=1 to 2 Do
    For Col:=1 to 2 do
    begin
      t := 0;
      For i:=1 to 2 do
        t := t+BetaInv[Row,i]*BetaR[i,Col];
```

```
        M[Row,Col] := t;
      end;
    For Row:=1 to 2 do
    begin
      u := 0;
      For Col:=1 to 2 do
        u := u+M[Row,Col]*Br[Col];
      BRD[Row] := u;
    end;
    For Row:=1 to 2 Do
    If (abs(BRD[1])<1) or (abs(BRD[2])<1) then
    begin
      k := 1;
    end
    else
    begin
      For Row:=1 to 2 do
        Br[Row] := BRD[Row];
      Find_VB(Expo,Prime,Brr,VB);
      Find_BR(Expo,Prime,VB,BetaR);
      Inv(BetaR,Lamda,BetaInv);
    end; {if}
    Until k=1;
End;   {Find_BRD}
Procedure YCap_RD(Expo:Data; BRD:Sing; Var ycap:Caps);
Var a : real;
Begin
    For Row:=1 to Num Do
    Begin
      a := BRD[1]+(BRD[2]*Expo[Row,2]);
```

```
      ycap[Row] := Exp(a)/(1+Exp(a));
      If ycap[Row]=0 then ycap[Row] := 0.0000000001
        Else if ycap[Row]=1 then ycap[Row] := 0.9999999999;
   End;
End; {Procedure Y_Cap}

Procedure RD(ycap:Caps; Expo:Data; Var RmseRD1,DevRD1:double);
Var rm,sum,a,b,c,d : real;
    px  : Cap;
Begin
     rm := 0;
     a := 0;
     b := 0;
     c := 0;
     d := 0;
     sum := 0;
     For Row:=1 to Num Do
     Begin
       {a := b0+(b1*Expo[Row,2])+Err[Row];
       px[Row] := Exp(a)/(1+Exp(a));
       b := px[Row]-ycap[Row];}
       b := y[Row] - ycap[Row];
       rm := rm+sqr(b);
       c := 1-ycap[Row];
       d := ycap[Row]*ln(ycap[Row]/c)+ln(c);
       sum := sum+d;
     End;
     RmseRD1 := sqrt(rm/(Num-P));
     DevRD1 := (-2)*sum;
     writeln('  RMSE is ',RmseRD1:4:15);
```

```
    writeln('  Deviance is ',DevRD1:4:15);
End;   {RD}

Procedure ShowRD(RmseRD1,DevRD1:double; ycap:Caps; Expo:Data; Var
RmseSum,DevSum:double);
Begin
  RD(ycap,Expo,RmseRD1,DevRD1);
  RmseSum := RmseSum+RmseRD1;
  DevSum := DevSum+DevRD1;
End;   {ShowRD}

Procedure ShowSumRD(RmseSum,DevSum : Double);
Var RmseAvg,DevAvg : double;
Begin
  RmseAvg := RmseSum/Times;
  DevAvg := DevSum/Times;
  writeln('    Average of RMSE is ',RmseAvg:2:15);
  writeln('    Average of Deviance is ',DevAvg:2:15);
End; {ShowSum}

Procedure Mix_MLE1;
Begin
     Find_UBr(Expo,y,Br,UBr);
     Find_HBR(Expo,Br,HBr);
     InvsHBr(HBr,insHBr);
     MLE(UBr,y,insHBr,Br1,Br);
     YCap_MLE(Expo,Br1,ycap);
     ShowMLE1(RmseMLE1,DevMLE1,RmseSumMLE,DevSumMLE);
End;
```

```
Procedure Mix_DF1;
Begin
    DF(beta0,beta1,Expo,y);
    YCap_DF(Expo,beta0,beta1,ycap);
    ShowDF(RmseDF1,DevDF1,ycap,Expo,RmseSumDF,DevSumDF);
End;

Procedure Mix_RD1;
Begin
    Find_Lamda(Expo,Lamda,Prime);
    Find_VB(Expo,Prime,Brr,VB);
    Find_BR(Expo,Prime,VB,BetaR);
    Inv(BetaR,Lamda,BetaInv);
    Find_BRD(BetaInv,BetaR,Expo,BRD);
    YCap_RD(Expo,BRD,ycap);
    ShowRD(RmseRD1,DevRD1,ycap,Expo,RmseSumRD,DevSumRD);
End;
Begin   {Main_Mix1}
  {clrscr}
  ch_howto := 0;
  Howto;
  Select_ratio(ratio);
  For j:=1 to Times do
  begin
    case ch_howto of
        1 : Exponential(Zeta,Expo);
        2 : Weibul(Beta,Elfa,Expo);
        3 : Normal(std,mean,Expo);
    end;
    Error(Err);
```

```
      Find_Y(Expo,Err,y);
      Change_Y(ratio,xx,y);
      Count_y(y);
          {MLE}
      Br[1] := b0;
      Br[2] := b1;
      Mix_MLE1;
          {DF}
      Mix_DF1;
          {RD}
      Brr[1] := b0;
      Brr[2] := b1;
      Mix_RD1;
    end;
    ShowSumMLE(RmseSumMLE,DevSumMLE);
    ShowSumDF(RmseSumDF,DevSumDF);
    ShowSumRD(RmseSumRD,DevSumRD);
End.  {Main_Mix1}
```

{Two variable program}

```
{$N+,E+}
Program Ray_Mix2;
Const Num   = 40;
      b0    = 0.912287;
      b1    = -0.00743;
      b2    = 0.00574;
      P     = 3;
      Times = 1000;

Type Data   = Array[1..Num,1..3] of real;
     Cap    = Array[1..Num] of real;
     Caps   = Array[1..Num] of double;
     Sing   = Array[1..3] of real;
     PM     = Array[1..3,1..Num] of real;
     Two    = Array[1..2,1..2] of real;
     Three  = Array[1..3,1..3] of real;
     Matrix = Array[1..Num,1..Num] of real;

Var Expo  : Data;
    ycap  : Caps;
    VB    : Matrix;
    Prime : PM;
    Err,y,UBr,xx : Cap;
    Brr,BrD,Br,Br1 : Sing;
    Row,Col,j,ratio,ch_howto : Integer;
    Zeta,Beta,Elfa,std,mean  : Real;
    beta0,beta1,beta2        : Real;
    HBr,InsHBr,Lamda,BetaR,BetaInv : Three;
```

```
    RmseMLE2,DevMLE2,RmseDF2,DevDF2,RmseRD2,DevRD2 : Double;
    RmseSumMLE2,DevSumMLE2,RmseSumDF2,DevSumDF2,RmseSumRD2,DevSumRD2:Double;

Procedure N_E(std,mean,Zeta:real; Var Expo:Data);
Var r1,r2,y1,y2,a,b : real;
    x1,x2,x : real;
Begin
   randomize;
   a := 0;
   b := 0;
   For Row:=1 to Num do                                    {Find X1 with Normal}
   begin
      r1 := random;
      r2 := random;
      a  := -2*ln(1-r1);
      b  := 2*pi*r2;
      y1 := sqrt(a)*cos(b);
      y2 := sqrt(a)*sin(b);
      For Col:=2 to 2 do
      begin
         x1 := y1*std+mean;
         x2 := y2*std+mean;
         if x1<x2 then Expo[Row,Col] := x1
         else Expo[Row,Col] := x2;
      end;
      For Col:=3 to 3 do                                   {Find X2 with Exponential}
      begin
         x := (-Zeta)*ln(1-random);
         Expo[Row,Col] := x;
      end;
```

```
    Expo[Row,1] := 1;
  end;
  For Row:=1 to Num Do                                                    {Show X}
  Begin
    For Col:=1 to 3 Do
  End;
End;   {Procedure N_E}

Procedure Start_Expo(Var Zeta:real);
Var ch_expo : integer;
Begin
  repeat
    Zeta := 0;
    writeln('  1 ==> Zeta = 0.5 ');
    writeln('  2 ==> Zeta = 1 ');
    writeln('  3 ==> Zeta = 2 ');
    write('  Enter Zeta : ');
    readln(ch_expo);
    case ch_expo of
        1 : Zeta:=0.5;
        2 : Zeta:=1;
        3 : Zeta:=2;
        else Begin
            writeln('Please enter number 1-3 only');
            End; {else}
    end; {case}
  until (ch_expo=1) or (ch_expo=2) or (ch_expo=3);
End; {Procedure Start_Expo}

Procedure N_W(std,mean,Beta,Elfa:real; Var Expo:Data);
```

```
Var r1,r2,y1,y2,a,b,c,d : real;
    x1,x2,x : real;
Begin
   randomize;
   c := 1/Elfa;
   For Row:=1 to Num do
   begin
      r1 := random;
      r2 := random;
      a  := -2*ln(1-r1);
      b  := 2*pi*r2;
      y1 := sqrt(a)*cos(b);
      y2 := sqrt(a)*sin(b);
      For Col:=2 to 2 do                                          {Find X1 with Normal}
      begin
         x1 := y1*std+mean;
         x2 := y2*std+mean;
         if x1<x2 then Expo[Row,Col] := x1
         else Expo[Row,Col] := x2;
      end;
      For Col:=3 to 3 do        {Find X2 with Weibul}
      begin
         d := -ln(1-random);
         x := Beta*Exp(c*ln(d));
         Expo[Row,Col] := x;
      end;
      Expo[Row,1] := 1;
   end;
   For Row:=1 to Num Do         {Show X}
   Begin
```

```
    For Col:=1 to 3 Do
  End;
End;   {Procedure N_W}

Procedure Start_Weibul(Var Beta,Elfa:real);
Var ch_weibul,Fi : integer;
Begin
  repeat
    Fi := 0;
    writeln('  1 ==> Fi = 1 ');
    writeln('  2 ==> Fi = 4 ');
    writeln('  3 ==> Fi = 8 ');
    writeln('  4 ==> Fi = 16 ');
    write('  Enter Fi of Weibul : ');
    readln(ch_weibul);
    case ch_weibul of
        1 : Fi := 1;
        2 : Fi := 4;
        3 : Fi := 8;
        4 : Fi := 16
        else begin
               writeln('Please enter number 1-4 only');
             end; {else}
    end; {case}
  until (ch_weibul=1) or (ch_weibul=2) or (ch_weibul=3) or (ch_weibul=4);
  Elfa := 4*Fi;
  repeat
    Beta := 0;
    writeln('Select Beta for X of Weibul');
    writeln('  1 ==> Beta = (1/8)u ');
```

```
    writeln('  2 ==> Beta = (1/2)u ');
    writeln('  3 ==> Beta = 2u ');
    write('  Enter Beta : ');
    readln(ch_weibul);
    case ch_weibul of
        1 : Beta := (1/8)*Elfa;
        2 : Beta := (1/2)*Elfa;
        3 : Beta := 2*Elfa;
        else Begin
            writeln('Please enter number 1-3 only');
            End; {else}
    end; {case}
  until (ch_weibul=1) or (ch_weibul=2) or (ch_weibul=3);
End; {Procedure Start_Weibul}

Procedure E_W(Zeta,Beta,Elfa:real; Var Expo:Data);
Var u,v,a,b : real;
Begin
  randomize;
  u := 1/Elfa;
  For Row:=1 to Num do
  begin
    For Col:=2 to 2 do                                    {Find X1 with Exponential}
    begin
      a := (-Zeta)*ln(1-random);
      Expo[Row,Col] := a;
    end;
    For Col:=3 to 3 do                                    {Find X2 with Weibul}
    begin
      v := -ln(1-random);
```

```
        b := Beta*Exp(u*ln(v));
        Expo[Row,Col] := b;
      end;
      Expo[Row,1] := 1;
    end;
    For Row:=1 to Num Do
    Begin
      For Col:=1 to 3 Do
    End;
End;   {Procedure E_W}

Procedure Start_Normal(Var std,mean:real);
Var ch_normal,Fi : integer;
Begin
   repeat
      Fi := 0;
      writeln('  1 ==> Fi = 1 ');
      writeln('  2 ==> Fi = 4 ');
      writeln('  3 ==> Fi = 8 ');
      writeln('  4 ==> Fi = 16 ');
      write('  Enter Fi of Normal : ');
      readln(ch_normal);
      case ch_normal of
           1 : Fi := 1;
           2 : Fi := 4;
           3 : Fi := 8;
           4 : Fi := 16
           else begin
                  writeln('Please enter number 1-4 only');
                end; {else}
```

```
    end; {case}
  until (ch_normal=1) or (ch_normal=2) or (ch_normal=3) or (ch_normal=4);
  mean := 4*Fi;
  repeat
    Elfa := 0;
    writeln('Select Elfa for X of Normal');
    writeln('  1 ==> Sigma = (1/8)u ');
    writeln('  2 ==> Sigma = (1/2)u ');
    writeln('  3 ==> Sigma = 2u ');
    write('  Enter Sigma : ');
    readln(ch_normal);
    case ch_normal of
        1 : std := sqrt((1/8)*mean);
        2 : std := sqrt((1/2)*mean);
        3 : std := sqrt(2*mean);
        else Begin
            writeln('Please enter number 1-3 only');
            End; {else}
    end; {case}
  until (ch_normal=1) or (ch_normal=2) or (ch_normal=3);
End; {Procedure Start_Normal}

Procedure Howto;
Begin
  writeln('          Select X1 & X2 from');
  writeln('   Normal & Exponential  ==>  Press 1');
  writeln('   Normal & Weibul       ==>  Press 2');
  writeln('   Exponentail & Weibul  ==>  Press 3');
  writeln('       Exit program      ==>  Press 4');
  write('Enter your choice : ');
```

```
  readln(ch_howto);
  case ch_howto of
      1 : begin
            Start_Normal(std,mean);
            Start_Expo(Zeta);
          end;
      2 : begin
            Start_Normal(std,mean);
            Start_Weibul(Beta,Elfa);
          end;
      3 : begin
            Start_Expo(Zeta);
            Start_Weibul(Beta,Elfa);
          end;
      4 : halt(1);
      else begin
             writeln('Please enter number 1-4 only');
           end; {else}
  end; {case}
End;

Procedure Error(Var Err:Cap);
Var x,a,b : real;
Begin
  a := -1;
  b := 1;
  randomize;
  For Row:=1 to Num do
  Begin
    x := (((b-a)*random)+a);
```

```
      Err[Row] := x ;
   end;
   For Row:=1 to Num do
End; {Error}

Procedure Find_Y(Expo:Data; Err:Cap; Var y:Cap);
Var a : real;
Begin
     For Row:=1 to Num Do
     Begin
        a := b0+(b1*Expo[Row,2])+(b2*Expo[Row,3])+Err[Row];
        y[Row]:= Exp(a)/(1+Exp(a));
     End;
End; {Procedure Find_Y}

Procedure Select_Sort(y:Cap; Var xx:Cap);
Var i,j,k : integer;
    m :real;
Begin
    For Row:=1 to Num do
    xx[Row] := y[Row];
    For i:=1 to Num-1 do
    begin
      k := i;
      m := xx[i];
      For j:= i+1 to Num do
          If (xx[j]<m) then
          begin
             k := j;
             m := xx[j];
```

```
          end;
      xx[k] := xx[i];
      xx[i] := m;
    end;
End;

Procedure Select_Ratio(Var ratio:integer);
Begin
  Repeat
     writeln('  Choose ratio of y');
     writeln('  1 ==>  0.75 : 0.25');
     writeln('  2 ==>  0.80 : 0.20');
     writeln('  3 ==>  0.85 : 0.15');
     writeln('  4 ==>  0.90 : 0.10');
     writeln('  5 ==>  0.95 : 0.05');
     write('  Enter your ratio : ');
     readln(ratio);
  until (ratio=1) or (ratio=2) or (ratio=3) or (ratio=4) or (ratio=5);
End;   {Select_ratio}

Procedure Change_Y(ratio:integer; xx:Cap; Var y:Cap);
Var xi,yi,j,cnt0 : integer;
    hscore : real;
Begin
  Case ratio of
     1 : Begin
           xi := round(Num*0.75);
           yi := trunc(Num*0.25);
         End;
     2 : Begin
```

```
          xi := round(Num*0.80);
          yi := trunc(Num*0.20);
        End;
    3 : Begin
          xi := round(Num*0.85);
          yi := trunc(Num*0.15);
        End;
    4 : Begin
          xi := round(Num*0.90);
          yi := trunc(Num*0.10);
        End;
    5 : Begin
          xi := round(Num*0.95);
          yi := trunc(Num*0.05);
        End;
End; {Case}
 Select_Sort(y,xx);
cnt0 := 0;
While cnt0<yi do
  For Row:=1 to yi Do
     For j:=1 to Num Do
        If (xx[Row]=y[j]) then
        begin
          y[j] := 0;
          cnt0 := cnt0+1;
        end;
For Row:=1 to Num Do
  If (y[Row]<>0) then y[Row] := 1;
For Row:=1 to Num Do
  { writeln('y = ',y[Row]:2:4);}
```

```
End;{Procedure Change_Y}

Procedure Count_y(y:Cap);
Var Count0,Count1 : integer;
Begin
    Count0 := 0;
    Count1 := 0;
    For Row:=1 to Num Do
      If y[Row]=1 then Count1 := Count1+1
        else Count0 := Count0+1;
End;{Procedure Count_y}

Procedure Find_UBr(Expo:Data; y:Cap; Br:Sing; Var UBr:Cap);
Var a,b,c,d : real;
Begin
  a := 0;
  UBr[1] := 0;
  UBr[2] := 0;
  UBr[3] := 0;
  For Row:=1 to Num do
  begin
    a := Br[2]*Expo[Row,2];
    b := Br[3]*Expo[Row,3];
    c := Exp(Br[1]+a+b)/(1+Exp(Br[1]+a+b));
    d := y[Row]-c;
    UBr[1] := UBr[1]+d;
    UBr[2] := UBr[2]+(Expo[Row,2]*d);
    UBr[3] := UBr[3]+(Expo[Row,3]*d);
  end;
  For Row:=1 to 3 do
```

```
End;  {Find_UBr}


Procedure Find_HBr(Expo:Data; Br:Sing; Var HBr:Three);
Var a,b,c : real;
Begin
  a := 0;
  b := 0;
  c := 0;
  For Row:=1 to 3 do
    For Col:=1 to 3 do
      HBr[Row,Col] := 0;
  For Row:=1 to num do
  begin
    a := Br[2]*Expo[Row,2];
    b := Br[3]*Expo[Row,3];
    c := Exp(Br[1]+a+b)/sqr(1+Exp(Br[1]+a+b));
    HBr[1,1] := HBr[1,1]+c;
    HBr[1,2] := HBr[1,2]+(Expo[Row,2]*c);
    HBr[1,3] := HBr[1,3]+(Expo[Row,3]*c);
    HBr[2,1] := HBr[2,1]+(Expo[Row,2]*c);
    HBr[2,2] := HBr[2,2]+(sqr(Expo[Row,2])*c);
    HBr[2,3] := HBr[2,3]+(Expo[Row,2]*Expo[Row,3]*c);
    HBr[3,1] := HBr[3,1]+(Expo[Row,3]*c);
    HBr[3,2] := HBr[3,2]+(Expo[Row,2]*Expo[Row,3]*c);
    HBr[3,3] := HBr[3,3]+(sqr(Expo[Row,3])*c);
  end;
  For Row:=1 to 3 do
    For Col:=1 to 3 do
      HBr[Row,Col] := -(HBr[Row,Col]);
  For Row:=1 to 3 do
```

```
    For Col:=1 to 3 do
    begin
      if Col=3 Then
    end;
End;  {Find_HBr}

Procedure InvsHBr(HBr:Three; Var insHBr:Three);
Var det : double;
   adjt,adj : Three;
Begin
  det := 0;
  For Row:=1 to 3 do
    For Col:=1 to 3 do
    begin
      insHBr[Row,Col] := 0;
      adj[Row,Col] := 0;
    end;
  {Find adj^t}
  adjt[1,1] := (HBr[2,2]*HBr[3,3])-(HBr[2,3]*HBr[3,2]);
  adjt[1,2] := -((HBr[2,1]*HBr[3,3])-(HBr[2,3]*HBr[3,1]));
  adjt[1,3] := (HBr[2,1]*HBr[3,2])-(HBr[2,2]*HBr[3,1]);
  adjt[2,1] := -((HBr[1,2]*HBr[3,3])-(HBr[1,3]*HBr[3,2]));
  adjt[2,2] := (HBr[1,1]*HBr[3,3])-(HBr[1,3]*HBr[3,1]);
  adjt[2,3] := -((HBr[1,1]*HBr[3,2])-(HBr[1,2]*HBr[3,1]));
  adjt[3,1] := (HBr[1,2]*HBr[2,3])-(HBr[1,3]*HBr[2,2]);
  adjt[3,2] := -((HBr[1,1]*HBr[2,3])-(HBr[1,3]*HBr[2,1]));
  adjt[3,3] := (HBr[1,1]*HBr[2,2])-(HBr[1,2]*HBr[2,1]);
  det := (HBr[1,1]*adjt[1,1])+(HBr[1,2]*adjt[1,2])+(HBr[1,3]*adjt[1,3]);
  For Row:=1 to 3 Do                                          {Find Transport of adjt}
  begin
```

```
    adj[1,Row] := adjt[Row,1];
    adj[2,Row] := adjt[Row,2];
    adj[3,Row] := adjt[Row,3];
  end;
  For Row:=1 to 3 do                                          {Find Inverse of HBr}
    For Col:=1 to 3 do
      insHBr[Row,Col] := (1/det)*adj[Row,Col];
  For Row:=1 to 3 do
    For Col:=1 to 3 do
    begin
      if Col=3 Then
    end;
End;  {InvsHBr}

Procedure MLE2(insHBr:Three; UBr,y:Cap; Var Br1,Br:Sing);
Var i,k : integer;
   r : real;
   C : Cap;
Begin
  k := 0;
  repeat
    For Row:=1 to 3 do
    begin
      r := 0;
      For i:=1 to 3 do
        r := r+insHBr[Row,i]*UBr[i];
      C[Row] := r;
    end;
    For Row:=1 to 3 do
    For Row:=1 to 3 do
```

```
      begin
        Br1[Row] := Br[Row]-C[Row];
      end;
      If (abs(C[1])<0.01) or (abs(C[2])<0.01) or (abs(C[3])<0.01) or
        (abs(Br1[1])<0.01) or (abs(Br1[2])<0.01) or (abs(Br1[3])<0.01) then
      begin
        For Row:=1 to 3 do
        k := 1;
      end
      else
      begin
        For Row:=1 to 3 do
          Br[Row] := Br1[Row];
        Find_UBr(Expo,y,Br,UBr);
        Find_HBR(Expo,Br,HBr);
        InvsHBr(HBr,insHBr);
      end;
    Until k=1;
End;  {MLE}

Procedure YCap_MLE2(Expo:Data; Br1:Sing; Var ycap:Caps);
Var a : real;
Begin
    For Row:=1 to Num Do
    Begin
      a := Br1[1]+(Br1[2]*Expo[Row,2])+(Br1[3]*Expo[Row,3]);
      ycap[Row] := Exp(a)/(1+Exp(a));
      If ycap[Row]=0 then ycap[Row] := 0.0000000001
        else if ycap[Row]=1 then ycap[Row] := 0.9999999999;
    End;
```

```
End; {Procedure Y_Cap}

Procedure RMSE_DEV_MLE2(Expo:Data; Br1:Sing; ycap:Caps; Var RmseMLE2,DevMLE2:double);
Var rm,sum,a,b,c,d : real;
    px : Cap;
Begin
  a := 0;
  b := 0;
  c := 0;
  d := 0;
  rm  := 0;
  sum := 0;
  For Row:=1 to Num do
  begin
    b := y[Row] - ycap[Row];
    rm := rm+sqr(b);
    c := 1-ycap[Row];
    d := ycap[Row]*ln(ycap[Row]/c)+ln(c);
    sum := sum+d;
  end;
  RmseMLE2 := sqrt(rm/(Num-P));
  DevMLE2 := -2*sum;
  writeln('    RMSE is ',RmseMLE2:4:10);
  writeln('    Deviance is ',DevMLE2:4:10);
End;

Procedure ShowMLE2(RmseMLE2,DevMLE2:double; Expo:Data; Var
RmseSumMLE2,DevSumMLE2:double);
Begin
  RMSE_DEV_MLE2(Expo,Br1,ycap,RmseMLE2,DevMLE2);
```

```
  RmseSumMLE2 := RmseSumMLE2+RmseMLE2;
  DevSumMLE2  := DevSumMLE2+DevMLE2;
End;  {ShowMLE}

Procedure ShowSumMLE2(RmseSumMLE2,DevSumMLE2:double);
Var RmseAvg,DevAvg : double;
Begin
  RmseAvg := RmseSumMLE2/Times;
  DevAvg := DevSumMLE2/Times;
  writeln('    Average of RMSE is ',RmseAvg:2:10);
  writeln('    Average of Deviance is ',DevAvg:2:10);
End;  {ShowSum}

Procedure DF2(Var beta0,beta1,beta2:real; Expo:Data; y:Cap);
Var n0,n1,Row : integer;
    avg01,avg02,avg11,avg12,sd01,sd02,sd11,sd12 : real;
    sumy0,sumy1,zeta0,zeta1,ss0,ss1,Sol,det,r   : real;
    sum01,sum11,sum02,sum12,tsum01,tsum02,tsum11,tsum12 : real;
    sigma,invSigma,mean0,mean1,tmmean,mmean,pmean,s0,s1,C,Beta : Two;
Begin
  sum01 := 0;                                                    {Clear Data}
  sum02 := 0;
  sum11 := 0;
  sum12 := 0;
  n1    := 0;
  n0    := 0;
  tsum01:= 0;
  tsum02:= 0;
  tsum11:= 0;
  tsum12:= 0;
```

```
sd01  := 0;
sd02  := 0;
sd11  := 0;
sd12  := 0;
ss0   := 0;
ss1   := 0;
det   := 0;
For Row:=1 to 2 do
begin
  mean0[Row,1]  := 0;
  mean1[Row,1]  := 0;
  tmmean[1,Row] := 0;
  mmean[Row,1]  := 0;
  pmean[Row,1]  := 0;
  Beta[Row,1]   := 0;
  C[1,Row]      := 0;
  For Col:=1 to 2 do
  begin
    Sigma[Row,Col] := 0;
    invSigma[Row,Col] := 0;
  end;
end;
For Row:=1 to Num Do
  If (y[Row]=1) then
  begin
    n1   := n1+1;                                   {Number of y=1}
    sum11 := sum11+Expo[Row,2];                     {Find sum of X1 at y=1}
    sum12 := sum12+Expo[Row,3];                     {Find sum of X2 at y=1}
  end
  Else
```

```
  begin
    n0   := n0+1;                                   {Number of y=0}
    sum01 := sum01+Expo[Row,2];                     {Find sum of X1 at y=0}
    sum02 := sum02+Expo[Row,3];                     {Find sum of X2 at y=0}
  end;
If n0<=0 then
begin
  avg01 := 0;                                       {Find mean of X1 at y=0}
  avg02 := 0;                                       {Find mean of X2 at y=0}
  avg11 := sum11/n1;                                {Find mean of X1 at y=1}
  avg12 := sum12/n1;                                {Find mean of X2 at y=1}
end
else if n1<=0 then
    begin
      avg11 := 0;
      avg12 := 0;
      avg01 := sum01/n0;
      avg02 := sum02/n0;
    end
    else
    begin
      avg01 := sum01/n0;
      avg02 := sum02/n0;
      avg11 := sum11/n1;
      avg12 := sum12/n1;
    end;
     For Row:=1 to Num Do
  If (y[Row]=1) then
  begin
    tsum11 := tsum11+sqr(Expo[Row,2]-avg11);
```

```
      tsum12 := tsum12+sqr(Expo[Row,3]-avg12);
    end
    Else
    begin
      tsum01 := tsum01+sqr(Expo[Row,2]-avg01);
      tsum02 := tsum02+sqr(Expo[Row,3]-avg02);
    end;
  If n0<=1 then
  begin
    sd01 := 0;                                          {Find variane of X1 at y=0}
    sd02 := 0;                                          {Find variane of X2 at y=0}
    sd11 := tsum11/(n1-1);                              {Find variane of X1 at y=1}
    sd12 := tsum12/(n1-1);                              {Find variane of X2 at y=1}
  end
  else if n1<=1 then
  begin
    sd01 := tsum01/(n0-1);
    sd02 := tsum02/(n0-1);
    sd11 := 0;
    sd12 := 0;
  end
  else
  begin
    sd01 := tsum01/(n0-1);
    sd02 := tsum02/(n0-1);
    sd11 := tsum11/(n1-1);
    sd12 := tsum12/(n1-1);
  end;
  For Row:=1 to Num do
    If y[Row]=0 then ss0 := ss0+(Expo[Row,2]-avg01)*(Expo[Row,3]-avg02)
```

```
    Else ss1 := ss1+(Expo[Row,2]-avg11)*(Expo[Row,3]-avg12);
s0[1,1] := sd01;
s0[1,2] := ss0;
s0[2,1] := ss0;
s0[2,2] := sd02;
s1[1,1] := sd11;
s1[1,2] := ss1;
s1[2,1] := ss1;
s1[2,2] := sd12;
For Row:=1 to 2 do                                          {Find sigma}
  For Col:=1 to 2 do
  begin
    Sigma[Row,Col] := (((n0-1)*s0[Row,Col])+((n1-1)*s1[Row,Col]))/(n0+n1-2);
    If Sigma[Row,Col]<=0 then Sigma[Row,Col] := 0;
  end;
det := (Sigma[1,1]*Sigma[2,2])-(Sigma[1,2]*Sigma[2,1]);
invSigma[1,1] := (1/det)*Sigma[2,2];                        {Find inverse of sigma}
invSigma[1,2] := (1/det)*(-Sigma[1,2]);
invSigma[2,1] := (1/det)*(-Sigma[2,1]);
invSigma[2,2] := (1/det)*Sigma[1,1];
mean0[1,1] := avg01;                                            {Find U0}
mean0[2,1] := avg02;
mean1[1,1] := avg11;                                            {Find U1}
mean1[2,1] := avg12;
zeta0 := n0/Num;                                                {Find Zeta0}
zeta1 := 1-zeta0;                                               {Find Zeta1}
For Row:=1 to 2 do
begin
  mmean[Row,1] := mean1[Row,1]-mean0[Row,1];                    {Find U1-U0}
  pmean[Row,1] := mean1[Row,1]+mean0[Row,1];                    {Find U1+U0}
```

```
  end;
  tmmean[1,1] := mmean[1,1];                          {Find transpose of U1-U0}
  tmmean[1,2] := mmean[2,1];
  For Row:=1 to 2 do                                  {Find tmmean*tsigma}
  begin
    r := 0;
    For Col:=1 to 2 do
      r := r+tmmean[1,Col]*invSigma[Col,Row];
    C[1,Row] := r;
  end;
  r := 0;
  For Row:=1 to 2 do                                      {Find C*pmean}
    r := r+(C[1,Row]*pmean[Row,1]);
  Sol := r;
  If (zeta0<>0) and (zeta1<>0) then                       {Find Beta0}
    Beta0 := ln(zeta1/zeta0)-(0.5*Sol)
  else  Beta0 := (0.5*Sol);
  For Row:=1 to 2 do                                      {Find Beta1&Beta2}
  begin
    r := 0;
    For Col:=1 to 2 do
      r := r+(invSigma[Row,Col]*mmean[Col,1]);
    Beta[Row,1] := r;
  end;
  Beta1 := Beta[1,1];
  Beta2 := Beta[2,1];
End; {Procedure DF}

Procedure YCap_DF2(Expo:Data; beta0,beta1,beta2:Real; Var ycap:Caps);
Var a : real;
```

```
Begin
    For Row:=1 to Num Do
    Begin
       a := beta0+(beta1*Expo[Row,2])+(beta2*Expo[Row,3]);
       ycap[Row] := Exp(a)/(1+Exp(a));
       If ycap[Row]=0 then ycap[Row] := 0.0000000001
          else if ycap[Row]=1 then ycap[Row] := 0.9999999999;
    End;
End; {Procedure Y_Cap}

Procedure RMSE_DEV_DF2(ycap:Caps; Expo:Data; Var RmseDF2,DevDF2:double);
Var a,b,c,d : real;
   rm,sum : double;
   px  : Cap;
Begin
    rm := 0;
    sum := 0;
    For Row:=1 to Num Do
    Begin
       b := y[Row] - ycap[Row];
       rm := rm+sqr(b);
       c := 1-ycap[Row];
       d := ycap[Row]*ln(ycap[Row]/c)+ln(c);
       sum := sum+d;
    end;
    RmseDF2 := sqrt(rm/(Num-P));
    DevDF2 := (-2)*sum;
    writeln('    RMSE is ',RmseDF2:4:10);
    writeln('    Deviance is ',DevDF2:4:10);
End;   {RMSE_DEV}
```

```
Procedure ShowDF2(RmseDF2,DevDF2:Double; ycap:Caps; Expo:Data; Var
RmseSumDF2,DevSumDF2:double);
Begin
  RMSE_DEV_DF2(ycap,Expo,RmseDF2,DevDF2);
  RmseSumDF2 := RmseSumDF2+RmseDF2;
  DevSumDF2  := DevSumDF2+DevDF2;
End;

Procedure ShowSumDF2(RmseSumDF2,DevSumDF2:double);
Var RmseAvg,DevAvg : double;
Begin
  RmseAvg := RmseSumDF2/Times;
  DevAvg := DevSumDF2/Times;
  writeln('    Average of RMSE is ',RmseAvg:2:10);
  writeln('    Average of Deviance is ',DevAvg:2:10);
End; {ShowDF2}

Function low(C:Three):real;
Var min : real;
    a : integer;
Begin
  min := C[1,1];
  For a:=1 to 3 do
     If (C[a,a]<min) then
          min := C[a,a];
  low := min;
End; {Function low}

Procedure Find_Lamda(Expo:Data; Var Lamda:Three; Var Prime:PM);
Var C : Three;
```

```
    i : integer;
    r,lscore : real;
Begin
    For Row:=1 to Num Do                                    {Find transpose of x}
    Begin
        Prime[1,Row] := 1;
        Prime[2,Row] := Expo[Row,2];
        Prime[3,Row] := Expo[Row,3];
    End;
    For Row:=1 to 3 do                                      {Find x multiply prime}
      For Col:=1 to 3 do
      begin
        r := 0;
        For i:=1 to Num do
          r := r+Prime[Row,i]*Expo[i,Col];
        C[Row,Col] := r;
      end;
    lscore := low(C);
    For Row:=1 to 3 do                                          {Make Lamda Matrix}
      For Col:=1 to 3 do
      begin
        If Row=Col then
          Lamda[Row,Col] := lscore
        Else
          Lamda[Row,Col] := 0;
      end;
    For Row:=1 to 3 do
      For Col:=1 to 3 do
      begin
        if Col=3 Then
```

```
        end;
End;   {Lamda}

Procedure Find_VB(Expo:Data; Prime:PM; Brr:Sing; Var VB:Matrix);
Var a : real;
   px : Cap;
Begin
   For Row:=1 to Num do
   begin
      a := Brr[1]+(Brr[2]*Expo[Row,2])+(Brr[3]*Expo[Row,3]+Err[Row]);
      px[Row] := Exp(a)/(1+Exp(a));
   end;

   For Row:=1 to Num do
      For Col:=1 to Num do
      begin
         If Row=Col then
            VB[Row,Col] := px[Row]*(1-px[Row])
         Else
            VB[Row,Col] := 0;
      end;
   For Row:=1 to Num do
      For Col:=1 to Num do
      begin
         If Col=Num then writeln;
      end;
End;   {Find_VB}

Procedure Find_BetaR(Expo:Data; Prime:PM; VB:Matrix; Var BetaR:Three);
Var A : PM;
```

```
    r,s : real;
    i : integer;
Begin
  For Row:=1 to 3 do                                          {Find prime*VB}
    For Col:=1 to Num do
    begin
      r := 0;
      For i:=1 to Num do
        r := r+Prime[Row,i]*VB[i,Col];
      A[Row,Col] := r;
    end;

  For Row:=1 to 3 do                                          {Find BetaR}
    For Col:=1 to 3 do
    begin
      s := 0;
      For i:=1 to Num do
        s := s+A[Row,i]*Expo[i,Col];
      BetaR[Row,Col] := s;
    end;
  For Row:=1 to 3 Do
    For Col:=1 to 3 Do
    Begin
      If Col=3 Then writeln;
    End;
End;   {Find_BR}

Procedure Inv(BetaR:Three; Lamda:Three; Var BetaInv:Three);
Var det : real;
    Z,adjt,adj : Three;
```

```
Begin
  det := 0.0;
  For Row:=1 to 3 do
    For Col:=1 to 3 do
    begin
      Z[Row,Col] := 0;
      adjt[Row,Col] := 0;
      adj[Row,Col] := 0;
    end;
  For Row:=1 to 3 do
    For Col:=1 to 3 do
      Z[Row,Col] := BetaR[Row,Col]+(2*Lamda[Row,Col]);
  {Find adj^t}
  adjt[1,1] := (Z[2,2]*Z[3,3])-(Z[2,3]*Z[3,2]);
  adjt[1,2] := -((Z[2,1]*Z[3,3])-(Z[2,3]*Z[3,1]));
  adjt[1,3] := (Z[2,1]*Z[3,2])-(Z[2,2]*Z[3,1]);
  adjt[2,1] := -((Z[1,2]*Z[3,3])-(Z[1,3]*Z[3,2]));
  adjt[2,2] := (Z[1,1]*Z[3,3])-(Z[1,3]*Z[3,1]);
  adjt[2,3] := -((Z[1,1]*Z[3,2])-(Z[1,2]*Z[3,1]));
  adjt[3,1] := (Z[1,2]*Z[2,3])-(Z[1,3]*Z[2,2]);
  adjt[3,2] := -((Z[1,1]*Z[2,3])-(Z[1,3]*Z[2,1]));
  adjt[3,3] := (Z[1,1]*Z[2,2])-(Z[1,2]*Z[2,1]);
  det := (Z[1,1]*adjt[1,1])+(Z[1,2]*adjt[1,2])+(Z[1,3]*adjt[1,3]);
  For Row:=1 to 3 Do                                    {Find Transport of adjt}
   Begin
      adj[1,Row] := adjt[Row,1];
      adj[2,Row] := adjt[Row,2];
      adj[3,Row] := adjt[Row,3];
   End;
  For Row:=1 to 3 do                                    {Find Inverse}
```

```
      For Col:=1 to 3 do
        BetaInv[Row,Col] := (1/det)*adj[Row,Col];
    For Row:=1 to 3 Do
      For Col:=1 to 3 Do
      Begin
        If Col=3 Then
      End;
End;   {Inv}

Procedure RD2(Brr:Sing; BetaInv,BetaR:Three; Expo:Data; Var BrD:Sing);
Var M : Three;
    r,s : real;
    i,k : integer;
Begin
   k := 0;
   repeat
     For Row:=1 to 3 Do
       For Col:=1 to 3 do
       begin
         r := 0;
         For i:=1 to 3 do
           r := r+BetaInv[Row,i]*BetaR[i,Col];
         M[Row,Col] := r;
       end;
     For Row:=1 to 3 do
     begin
       s := 0;
       For Col:=1 to 3 do
         s := s+M[Row,Col]*Brr[Col];
       BrD[Row] := s;
```

```
      end;
      For Row:=1 to 3 Do
      If (abs(BrD[1])<1) or (abs(BrD[2])<1) or (abs(BrD[3])<1) then
      begin
      end
      else
      begin
        For Row:=1 to 3 do
          Brr[Row] := BrD[Row];
        Find_VB(Expo,Prime,Brr,VB);
        Find_BetaR(Expo,Prime,VB,BetaR);
        Inv(BetaR,Lamda,BetaInv);
      end; {if}
    Until k=1;
    WriteIn;
End;   {Find_BRD}

Procedure YCap_RD2(Expo:Data; BrD:Sing; Var ycap:Caps);
Var a : real;
Begin
    For Row:=1 to Num Do
    Begin
      a := BrD[1]+(BrD[2]*Expo[Row,2])+(BrD[3]*Expo[Row,3]);
      ycap[Row] := Exp(a)/(1+Exp(a));
      If ycap[Row]=0 then ycap[Row] := 0.0000000001
        else if ycap[Row]=1 then ycap[Row] := 0.9999999999;
    End;
End; {Procedure Y_Cap}

Procedure RMSE_DEV_RD2(ycap:Caps; Expo:Data; Var RmseRD2,DevRD2:double);
```

```
Var a,b,c,d : real;
    rm,sum : double;
    px  : Cap;
Begin
  a  := 0;
  b  := 0;
  c  := 0;
  d  := 0;
  rm := 0;
  sum := 0;
  For Row:=1 to Num Do
  Begin
    b := y[Row] - ycap[Row];
    rm := rm+sqr(b);
    c := 1-ycap[Row];
    d := ycap[Row]*ln(ycap[Row]/c)+ln(c);
    sum := sum+d;
  End;
  RmseRD2 := sqrt(rm/(Num-P));
  DevRD2 := (-2)*sum;
  writeln('   RMSE is ',RmseRD2:4:15);
  writeln('   Deviance is ',DevRD2:4:15);
End;   {RD}

Procedure ShowRD2(RmseRD2,DevRD2:double; ycap:Caps; Expo:Data; Var
RmseSumRD2,DevSumRD2:double);
Begin
  RMSE_DEV_RD2(ycap,Expo,RmseRD2,DevRD2);
  RmseSumRD2 := RmseSumRD2+RmseRD2;
  DevSumRD2 := DevSumRD2+DevRD2;
```

```
End;   {ShowRD}

Procedure ShowSumRD2(RmseSumRD2,DevSumRD2 : double);
Var RmseAvg,DevAvg : double;
Begin
  RmseAvg := RmseSumRD2/Times;
  DevAvg := DevSumRD2/Times;
  writeln('       RD2 Total ',Times,' rounds');
  writeln('  Average of RMSE is ',RmseAvg:2:20);
  writeln('  Average of Deviance is ',DevAvg:2:20);
End; {ShowSum}

Procedure Mix_MLE2;
Begin
    Find_UBr(Expo,y,Br,UBr);
    Find_HBR(Expo,Br,HBr);
    InvsHBr(HBr,insHBr);
    MLE2(insHBr,UBr,y,Br1,Br);
    YCap_MLE2(Expo,Br1,ycap);
    ShowMLE2(RmseMLE2,DevMLE2,Expo,RmseSumMLE2,DevSumMLE2);
End;   {Mix_MLE2}

Procedure Mix_DF2;
Begin
    DF2(beta0,beta1,beta2,Expo,y);
    YCap_DF2(Expo,beta0,beta1,beta2,ycap);
    ShowDF2(RmseDF2,DevDF2,ycap,Expo,RmseSumDF2,DevSumDF2);
End; {Mix_DF2}

Procedure Mix_RD2;
```

```
Begin
    Find_Lamda(Expo,Lamda,Prime);
    Find_VB(Expo,Prime,Brr, VB);
    Find_BetaR(Expo,Prime,VB,BetaR);
    Inv(BetaR,Lamda,BetaInv);
    RD2(Brr,BetaInv,BetaR,Expo,BrD);
    YCap_RD2(Expo,BrD,ycap);
    ShowRD2(RmseRD2,DevRD2,ycap,Expo,RmseSumRD2,DevSumRD2);
End;  {Mix_RD2}

Begin  {Main_Mix2}
  {clrscr;}
  ch_howto := 0;
  Howto;
  Select_ratio(ratio);
  For j:=1 to Times do
  begin
    case ch_howto of
        1 : N_E(std,mean,Zeta,Expo);
        2 : N_W(std,mean,Beta,Elfa,Expo);
        3 : E_W(Zeta,Beta,Elfa,Expo);
    end;
    Error(Err);
    Find_Y(Expo,Err,y);
    Change_Y(ratio,xx,y);
    Count_y(y);
{Use MLE2}
    Br[1] := b0;
    Br[2] := b1;
    Br[3] := b2;
```

```
      Mix_MLE2;
{Use DF2}
      Mix_DF2;
{Use RD2}
      Brr[1] := b0;
      Brr[2] := b1;
      Brr[3] := b2;
      Mix_RD2;
   end;
   ShowSumMLE2(RmseSumMLE2,DevSumMLE2);
   ShowSumDF2(RmseSumDF2,DevSumDF2);
   ShowSumRD2(RmseSumRD2,DevSumRD2);
End.   {Main_Mix2}
```