```
% MATLAB file MBLE.m %

function [theta,v,err,Ohat,H]=MBLE(S,df,nn,theta0,v0,funn,vfree,pr,bd,thrs,btol,maxiter,varargin)

% Maximum Beta Likelihood with bias adjustment
% Code last updated Dec 2013.

% Electronic Supplementary Material to
% QUANTIFYING ADVENTITIOUS ERROR IN A COVARIANCE STRUCTURE AS A RANDOM EFFECT
% Psychometrika, 80(3), 619-624
% by Hao Wu (Boston College, hao.wu.5@bc.edu) and Michael W. Browne (The Ohio State University)

% The actual function being minimized is discrepancy function over n.
% The parameter list is n*v followed by covariance structure parameters.
% Hessian element Hvv takes positive approximation when negative.

%%%%%% List of Input Arguments %%%%%%%%%%%%%%%%%%%
% df is the degrees of freedom.
%   For unadjusted discrepancy function, use df=p(p+1)/2.
% S is the sample covariance matrix.
% nn is the df of the Wishart distribution (S|Sigma).
% theta0 is the initial value of structure parameters in the model
% v0 is the initial value of v=1/m
% fun is a function handle of the Covariance structure ...
%   of the form [O,err,R,dOdx,iRdOdx]=fun(theta,...)
%   input and output arguments are
%   theta: a row vector of parameters
%   O: the covariance structure
%   [R,err]=chol(O);
%   dOdx: the derivative of O, a p^2 by q matrix
%   iRdOdx=(R'\otimesR')\dOdx.
% vfree denotes whether m is fixed or not:
%   0 means 1/m is treated as fixed at v0;
%   else means it is not fixed.
% pr prints iteration details:
%   0:  none;
%   1:  only F, Res.Cos, cond.#.H, NPB and NEC;
%   2:  only parameter values;
```

```
%   3:  both
% bd is a matrix of length(theta) by 2,
%     giving boundary conditions for parameters
% thrs is the stopping threshold
% btol is the boundary threshold matrix.
%   It can be input as a scalar, a column vector or a 2-columned matrix.
% maxiter is the maximum iterations allowed.
% varargin passes arguments to fun, the function handle.

%%%%%% List of Output Arguments %%%%%%%%%%%%%%%%%%%
% theta is the parameter estimate
% v is the estimate of 1/m, negative values allowed
% err>0 means nonconvergence
% Ohat is the implied covariance matrix
% H is the approximate Hessian for the covariance structure parameters

%%%%%% List of Screen Outputs %%%%%%%%%%%%%%%%%%%%
% #: iteration number
% F: discrepancy function value (per sample size)
% Res.Cos: residual cosine
% cond.#.H: conditional number of Hessian (unconstrained parameters only)
% NPB: number of parameters on boundary
% NEC: number of effective constraints
% parameters: parameter values, the first one is v.


global CS p s n vfr detS fun other alpha coef

n=nn;
vfr=vfree;
fun=funn;
other=varargin;

if nargin<7, error('Too few arguments.');end
s=length(theta0); % number of parameters in the covariance structure.
if nargin<8 || all(pr~=0:3),    pr=1;end
if nargin<9 || isempty(bd),    bd=kron([-Inf,Inf],ones(s,1));end
if nargin<10 || isempty(thrs),    thrs=.1^7; end
```

```
if nargin<11 || isempty(btol),    btol=.1^8; end
if nargin<12 || isempty(maxiter),    maxiter=200; end

if all(size(btol)==[1,1])
     btol=btol*ones(s,2);
elseif size(btol,2)==1
     btol=btol*[1,1];
end

if df<0
    error('degrees of freedom must be positive.');
end
[p,p1]=size(S); % p is the number of manifest variables.
if p~=p1|| ~all(all(isreal(S)))
    error('The covariance matrix should be real square matrix.');
elseif any([s,2]~=size(bd))|| any([s,2]~=size(btol))|| any(size(theta0)~=[1,s])
    error('The size of one or more input matrices is wrong.');
end


btol=[[0,0.1^8];btol];   % including v
bd=[[0,1/(p-1)];bd];
bd=[bd(:,1)+btol(:,1),bd(:,2)-btol(:,2)];
inc0=true(s+1,1);
inc0(1)=true*vfr;
ubd=~true(s+1,1);
lbd=ubd;

%-------------------------------------------------
% Initial screening

if any(any(S~=S'))
    display('Covariance matrix not symmetric. Lower half used.');
    S=tril(S);
    S=S+triu(S',1);
end
[CS,err]=chol(S);
if err>0,    error('Covariance matrix not positive definite.'); end
```

```
alpha=2*df/p/(p+1); % The adjustment factor.
detS=prod(diag(CS))^2;
coef=[p*(p+1)/2,p*(2*p^2+3*p-1)/12,(p-1)*p*(p+1)*(p+2)/24,p*(6*p^4+15*p^3-10*p^2-30*p+3)/360];
theta=[v0,theta0]';

if any(theta>bd(:,2)) || any(theta<bd(:,1))
        error('initial value on or out of bounds.');
end
[~,err]=feval(fun,theta(2:end)',other{:});
if err~=0
    error('initial value yields non-positive definite covariance');
end

switch pr
        case 0
        case 1
            display(' #       F         Res.Cos     cond.#.H    NPB NEC');
        case 2
            display(' #       F        parameters');
        case 3
            display(' #       F         Res.Cos     cond.#.H    NPB NEC     parameters');
end


%---------------------------------
% iterations
i=-1;
inc=inc0;
crit=Inf;
d=0;
f=Inf;
invH=zeros(s+1,s+1);
while (crit>thrs&&f>thrs&&i<=maxiter&&any(inc))
    i=i+1;
    theta1=theta+d;
    ubd1=find(theta1>bd(:,2)); % locations of new effective bounds
    lbd1=find(theta1<bd(:,1));
```

```
bd1=[lbd1;ubd1];
if ~isempty(bd1) % constrain the proposed value inside the bound
    bds=[bd(lbd1,1);bd(ubd1,2)];
    if pr~=0
        display('back to the boundary.');
    end
    [r,ind]=min((bds-theta(bd1))./d(bd1));  % proportions of distance to the bound
    d=d*r;
    theta1=theta+d;
    theta1(bd1(ind))=bds(ind);
end
while 1 % guarantee the discrepancy function is smaller
    [f1,err,g,H]=FgH(theta1,f);
    if err~=0 && pr==0
        d=d/2;
        theta1=theta+d;
        continue;
    elseif err~=0
        display('step halved.')
        d=d/2;
        theta1=theta+d;
        continue;
    end
    break;
end
theta=theta1;
f=f1;
d=zeros(s+1,1);
ubd(inc0)=(theta(inc0)==bd(inc0,2)); % current effective upper bounds
lbd(inc0)=(theta(inc0)==bd(inc0,1)); % current effective lower bounds
invH(inc0,inc0)=inv(H(inc0,inc0));
d(inc0)=-invH(inc0,inc0)*g(inc0); % unconstrained search direction
d(1)=d(1)/n;
inc=inc0;
if any(d(ubd)>0) ||any(d(lbd)<0) % out of boundary
    lambda=zeros(s+1,1);
    effbd=~true(s+1,1);
    effbd(ubd)=true;
```

```
    effbd(lbd)=true;
    lambda(effbd)=invH(effbd,effbd)\d(effbd); % would-be lagrange multiplier
    [rmbd,ind]=max((-lambda.*ubd+lambda.*lbd).*effbd); % removable bounds
    while  rmbd>0 % some effective bounds can be removed
        effbd(ind)=false; % remove inequality constraints;
        lambda(effbd)=invH(effbd,effbd)\d(effbd); % would-be lagrange multiplier
        [rmbd,ind]=max((-lambda.*ubd+lambda.*lbd).*effbd); % removable bounds
    end
    inc(effbd)=false;
    d(inc)=-H(inc,inc)\g(inc);
    d(1)=d(1)/n;
    d(~inc)=0;
end

condH=cond(H(inc,inc));
crit=max(g(inc).*g(inc)./diag(H(inc,inc))/f);
switch pr
    case 0
    case 1
        str1=sprintf('%3d',i);
        str2=sprintf('%0.5e',f);
        str3=sprintf('%0.5e',crit);
        str4=sprintf('%0.5e',condH);
        str5=sprintf('%2d',sum(ubd)+sum(lbd));
        str6=sprintf('%2d',s+1-sum(inc));
        display([str1,' ',str2,' ',str3,' ',str4,' ',str5,'  ',str6]);
    case 2
        str1=sprintf('%3d',i);
        str2=sprintf('%0.5e',f);
        str7=sprintf('%+15.5e',theta);
        display([str1,' ',str2,' ',str7]);
    case 3
        str1=sprintf('%3d',i);
        str2=sprintf('%0.5e',f);
        str3=sprintf('%0.5e',crit);
        str4=sprintf('%0.5e',condH);
        str5=sprintf('%2d',sum(ubd)+sum(lbd));
        str6=sprintf('%2d',s+1-sum(inc));
```

```
            str7=sprintf('%+15.5e',theta);
            display([str1,' ',str2,' ',str3,' ',str4,' ',str5,'  ',str6,'  ',str7]);
    end
end

%---------------------
% output
if i<=maxiter
    v=theta(1);
    if vfr && v==0
        v=-g(1)/df;
    end
    theta=theta(2:end);
    H=H(2:end,2:end);
    Ohat=feval(fun,theta',other{:});
    err=0;
elseif pr~=0
    display('maximum numbers of iterations reached without convergence.');
    err=1;
    v=NaN;
    H=nan(s,s);
    theta=nan(s,1);
    Ohat=[];
else
    err=1;
    v=NaN;
    H=nan(s,s);
    theta=nan(s,1);
    Ohat=[];
end

%---------------------------------
function [f,err,g,H]=FgH(theta,f0)
% discrepancy function is 1/n of (-2)-log-likelihood-ratio.
global CS p s n vfr detS fun alpha coef other

v=theta(1);
theta=theta(2:end);
```

```
fac=1+n*v;
I=eye(p);

[~,err,R]=feval(fun,theta',other{:}); % R'R=O
if  err~=0;
    [f,g,H]=deal([]);
    err=1;
    return;
end
detO=prod(diag(R))^2;

A=CS/R;
iOS=A'*A;
iOaveOS=(I+n*v*iOS)/fac; %[mI+n(R'\S/R)]/(m+n)
detiOaveOS=det(iOaveOS); % det of [mI+n(R'\S/R)]/(m+n)
if v==0
    f=log(detO/detS)+trace(iOS)-p;
elseif (p==1 && v<=0.02)|| (p>1 && (p-1)*v<0.02)
    m=1/v;
    c1=[log(1+n*v)/n,1/m/(m+n),(2*m+n)/m^2/(m+n)^2,(3*m*(m+n)+n^2)/m^3/(m+n)^3]';
    f=alpha*coef*c1-log(detS/detO)+(1+m/n)*log(detiOaveOS);
else
    m=1/v;
    i=1:p;
    f=alpha/n*(2*sum(gammaln((m-i+1)/2)-gammaln((m-i+n+1)/2))-m*p*log(m/2)+(m+n)*p*log((m+n)/2)-n*p)...
            +log(detO/detS)+(1+m/n)*log(detiOaveOS);
end
f=max([0,f]);

if isnan(f)||isinf(f)
    err=2;
    g=[];
    H=[];
    return;
elseif f>f0
    err=3;
    g=[];
    H=[];
```

```
    return;
end
    %--------------------------------
    % gradient and Hessian
dif=iOS-I;
WLS=sum(dif(:).^2); % trace[(iOS-I)^2]
B=(I-iOS)/iOaveOS/fac; %m/n*[inv(iOaveOS)-I]=m/n*(I-iOaveOS)/iOaveOS = (iOS-I)/fac/iOaveOS

if isempty(theta)
    dFdx=[];
    d2Fdxdx=[];
else
    [~,~,~,~,iRdOdx]=feval(fun,theta',other{:}); % a p by ps matrix
    dFdx=B(:)'*iRdOdx;
    d2Fdxdx=iRdOdx'*iRdOdx/fac;

end

if vfr==0
    dFdv=0;
    d2Fdv2=1;
elseif v==0
    dFdv=alpha*coef(1)-n*WLS/2;
    d2Fdv2=alpha*(-n*coef(1)+2*coef(2))+n^2*(WLS-2/3*sum(sum((dif*dif).*dif)));
elseif (p==1 && v<=0.02)|| (p>1 && (p-1)*v<0.02)

c1=[m/(m+n),(2*m+n)/(m+n)^2,2*(3*m*(m+n)+n^2)/m/(m+n)^3,3*(m*m*(4*m+6*n)+n^2*(4*m+n))/m^2/(m+n)^4]';
    dF2dm=log(detiOaveOS)+v*n*trace(B);
    dFdv=alpha*coef*c1-m^2/n*dF2dm;
    c1=[-n/(n*v+1)^2,2/(n*v+1)^3,2*(3*m^3+(m+n)*(n*n+3*m*(m+n)))/(m+n)^4,6*(2*m^4+(n+m)*(n^3-
2*m*m*n+4*m*(m+n)*(m+n)))/m/(m+n)^5]';
    d2Fdv2=alpha*coef*c1-m^2*n/(n+m)*sum(B(:).^2)+2*m^3*dF2dm/n;
elseif m>=p-1
    i=1:p;
    dcdm=sum(psi((m-i+1)/2)-psi((m-i+n+1)/2)+log(1+n*v));
    dF2dm=log(detiOaveOS)+v*n*trace(B);
    dFdv=-m^2/n*(alpha*dcdm+dF2dm);
    d2Fdv2=alpha/n*(m^4/2*sum((psi(1,(m-i+1)/2)-2*v)-(psi(1,(m-i+n+1)/2)-2/(m+n)))+2*m^3*dcdm)...
```

```
        -m^2*n/(n+m)*sum(B(:).^2)+2*m^3*dF2dm/n;
end
g=[dFdv/n;dFdx'];
ind=(d2Fdv2>0);
d2Fdv2=d2Fdv2*ind+(1-ind)*(2*coef(2)/fac+(-
alpha*n*coef(1)+2*(coef(1)*n+coef(2)))/fac/fac+(alpha*2*coef(2)+p*(p+3))/fac^3);
H=[d2Fdv2/n^2,zeros(1,s);zeros(s,1),d2Fdxdx];
err=0;
```