Mathematical and Computational Statistics with a View Towards Finance and Risk Management - Assignment 2

Yannik Haller (12-918-645) Jaka Golob (20-716-791

16 10 2020

Information

This is our (Jaka Golob and Yannik Haller) solution to Assignment 2 of the course "Mathematical and Computational Statistics with a View Towards Finance and Risk Management". For a better overview, we decided to also write down the tasks in blue. In the first section we describe how we solved the tasks (pointing out the corresponding lines in the Matlab code). In the second section we show the graphs resulting from the Matlab program depicting the accumulated returns of all calculated portfolios together with a table containing the associated Sharpe Ratios, and in the last section we display the full Matlab code.

Univariate Collapsing Method (Long only)

Add two versions of the basic "UCM method" (long only): the non-parametric and the parametric UCM. (See the comments at the beginning of the matlab program more detailed explanation)

In this part we build up two different types of the univariate collapsing method (UCM) (see lines 107-155 of the code): the parametric UCM (see lines 125-134 of the code) and the non-parametric UCM (see lines 135-149 of the code). The basic idea of the UCM can be summarized in four steps:

1. Simulate a random combination of weights for assets in the portfolio and store them in a weighting vector.
2. Apply these weights to a fixed window of historical asset returns to calculate a pseudo historical sequence of portfolio returns.
3. Evaluate the performance of the portfolio resulting from this weighting vector according to predefined criteria.
4. Repeat the previous steps as often as possible and keep the best performing weight combination.

To implement this method in Matlab we start by generating a d-length sequence of random numbers (with d being the number of assets in the portfolio), which are uniformly distributed between 0 and 1. We then take the natural logarithm of these numbers and divide each of them by their total sum, such that they sum up to 1 in the end. This sequence then serves as the weighting vector for our assets to create a portfolio. In a next step we apply these weights to the assets to calculate a pseudo historical sequence of portfolio returns (i.e. Rp) for a window of 249 trading days. Thereafter, we use Rp to estimate the expected value and standard deviation of tomorrow's portfolio return, which then can be used to assess the performance of the portfolio. This estimation is done in two different ways: in the non-parametric approach we simply take the average and standard deviation of Rp as predictors, whereas in the parametric approach we fit a two component mixed-normal distribution on Rp and use the resulting MLE parameters to estimate the expected value and standard deviation of its distribution to predict the future portfolio characteristics. The estimates for the expected return and standard deviation together with the associated weighting vector then serve as our reference. We then sequentially try 100 different combinations of weights, for each of which we repeat the procedure described above and compare the resulting estimates to the reference. If a weighting vector appears to lead to a portfolio with higher mean and lower standard deviation than the reference, we choose it as our new reference. With this procedure, we end up choosing the best performing (according to our criteria) portfolio for this particular point in time. This procedure is then repeated for every day in our dataset, for which the asset returns of at least 249 previous trading days are known. Thereafter we compute cumulative returns for both, the parametric and non-parametric UCM and compare them to the Long only Max Sharpe Ratio portfolio (MSR) and the 1/N portfolio. We observe that (at least in some trials) we are able to outperform the MSR and 1/N portfolio using the UCM approaches. However, after several repetitions we can observe that the performance of the UCM can vary substantially (depending on the randomly generated weights). Hence we can conclude, that with the UCM we end up reaching a higher Sharpe-Ratio than the MSR and 1/N portfolio quite often (namely, we sometimes can produce sharp ratios around 0.9-1.0 which is quite good), but in only 20% of the trials we also achieve higher cumulative returns. Worth mentioning is also that the comparison to the 1/N method might be different if we account for transaction costs, as the UCM does daily re-balancing, while in the 1 /N method re-balancing is not necessary.

Mean-Variance (Long Only)

Make a function that does mean-variance LONG ONLY portfolio optimization using the **quadprog** function. This is based on the mixed normal, using its outputted mean and variance.

To solve this task we created the **PortMNS** and the **meanvar** function (see lines 236-263 of the code) and apply them (see lines 156-160 of the code) to perform a mean variance optimization for non-negative weights (no short selling). In a first step, the PortMNS function calculates the expected returns and variance-covariance matrix for the assets to consider by means of a 2 component MixN estimation on 249 past observations of the assets. To be able to pass the resulting variance-covariance matrix to the quadprog function we use within the meanvar function, we need to transform it such that it results in a symmetric matrix. Then we use these to apply the mean-variance optimization, which essentially means that we try to find the portfolio with the lowest predicted variance given a yearly minimum expected return (which we determined to be 10 %). To do so we incorporate the quadprog function, as mentioned above. This function minimizes an objective-function, which already has mean variance form and where we are solving for x (i.e. our weighting vector). We need to ensure that we calculate the variance-covariance matrix denoted by H, restrict x such that the sum of the weights is lower than 1, and guarantee that the expected portfolio return on a daily basis is at least as large as to lead to a yearly 10% return. For a given day, this function thus returns the weights, which produce the lowest expected variance corresponding to the predetermined minimum level of expected return. Finally, we compute the cumulative returns of this portfolio and include them in the plot to compare it to the other portfolios we calculated (i.e. non-parametric UCM, parametric UCM, MSR and 1/N). We observe that the Mean-Variance portfolio seems to perform slightly poorer than all the others.

Output Figures

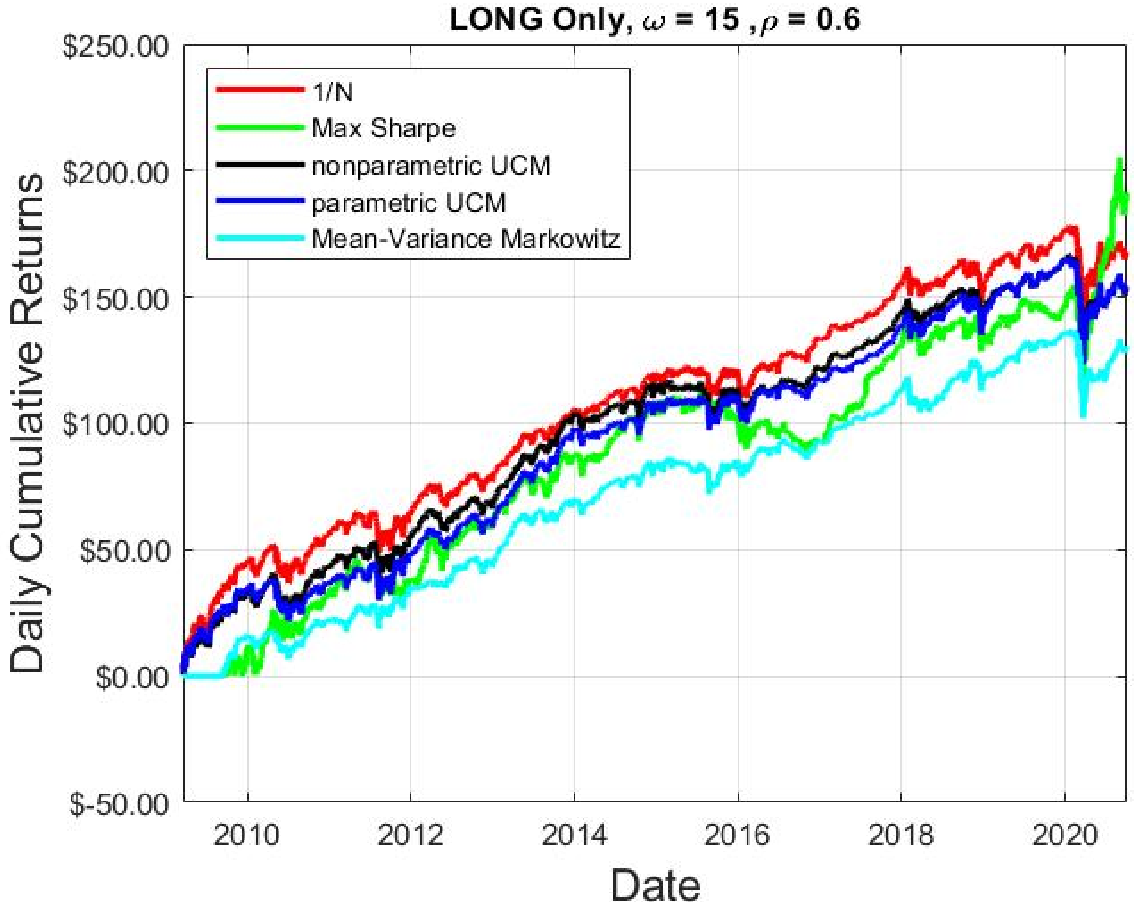

Figure 1: Accumulated Returns Long Only: Trial 1

Portfolio	Sharpe-Ratio
1/N	0.8313
Max Sharpe	0.7857
Non-Parametric UCM	0.8111
Parametric UCM	0.8024

Table 1: Sharpe Ratios Long Only: Trial 1

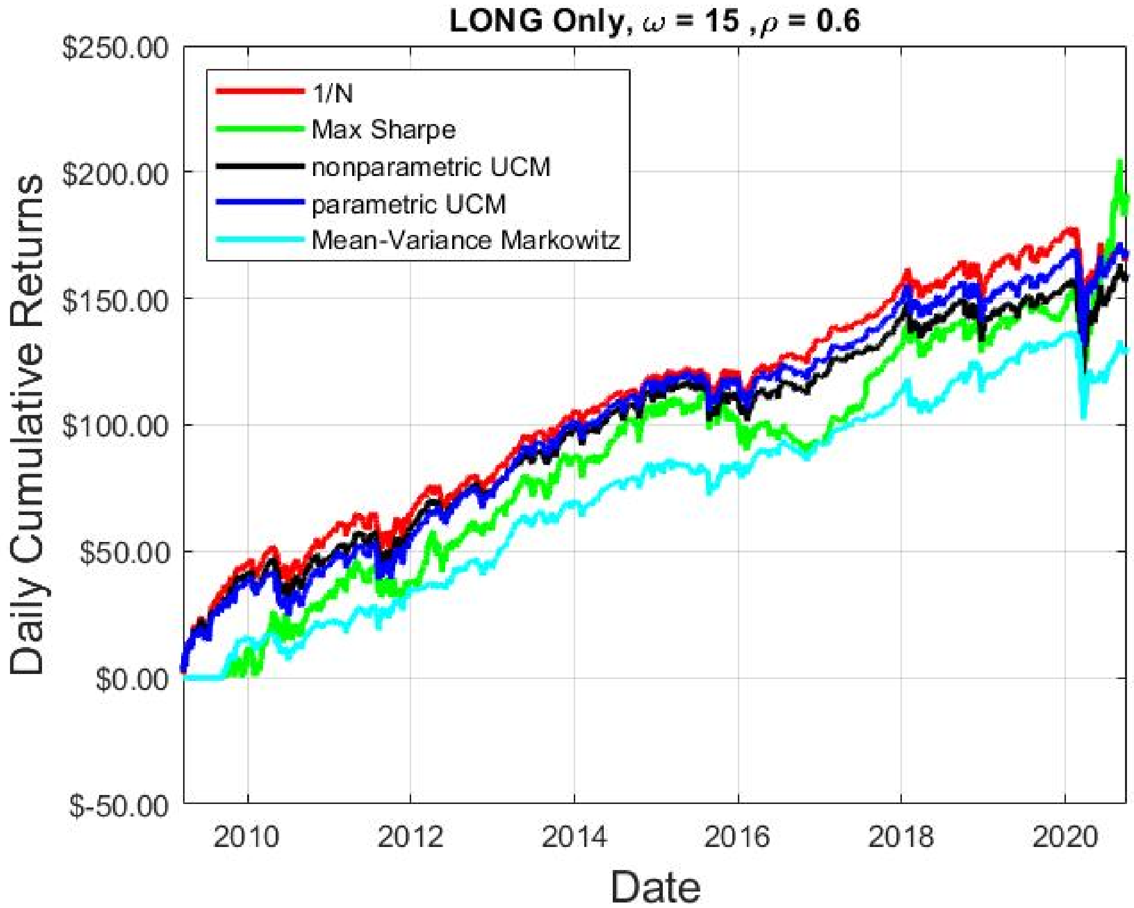

Figure 2: Accumulated Returns Long Only: Trial 2

Portfolio	Sharpe-Ratio
1/N	0.8313
Max Sharpe	0.7857
Non-Parametric UCM	0.8332
Parametric UCM	0.8842

Table 2: Sharpe Ratios Long Only: Trial 2

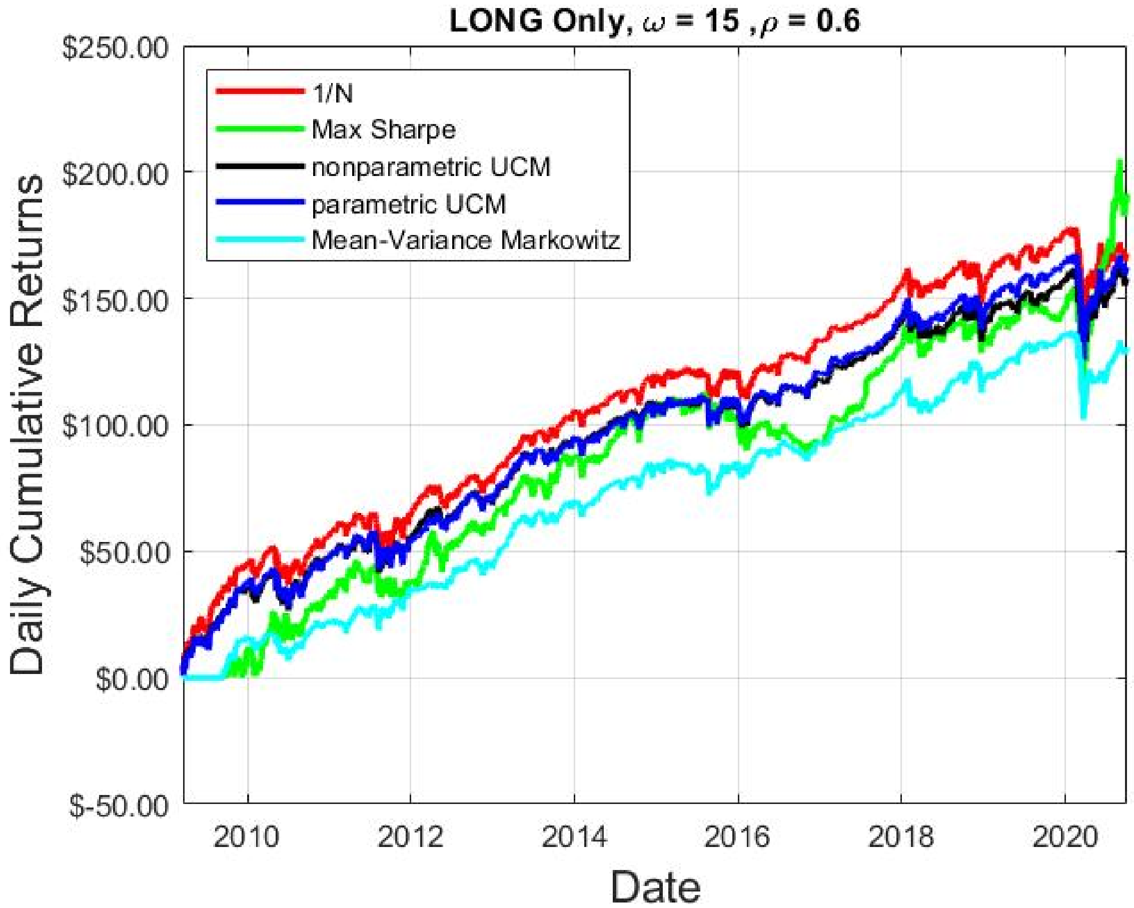

Figure 3: Accumulated Returns Long Only: Trial 3

Portfolio	Sharpe-Ratio
1/N	0.8313
Max Sharpe	0.7857
Non-Parametric UCM	0.8299
Parametric UCM	0.8439

Table 3: Sharpe Ratios Long Only: Trial 3

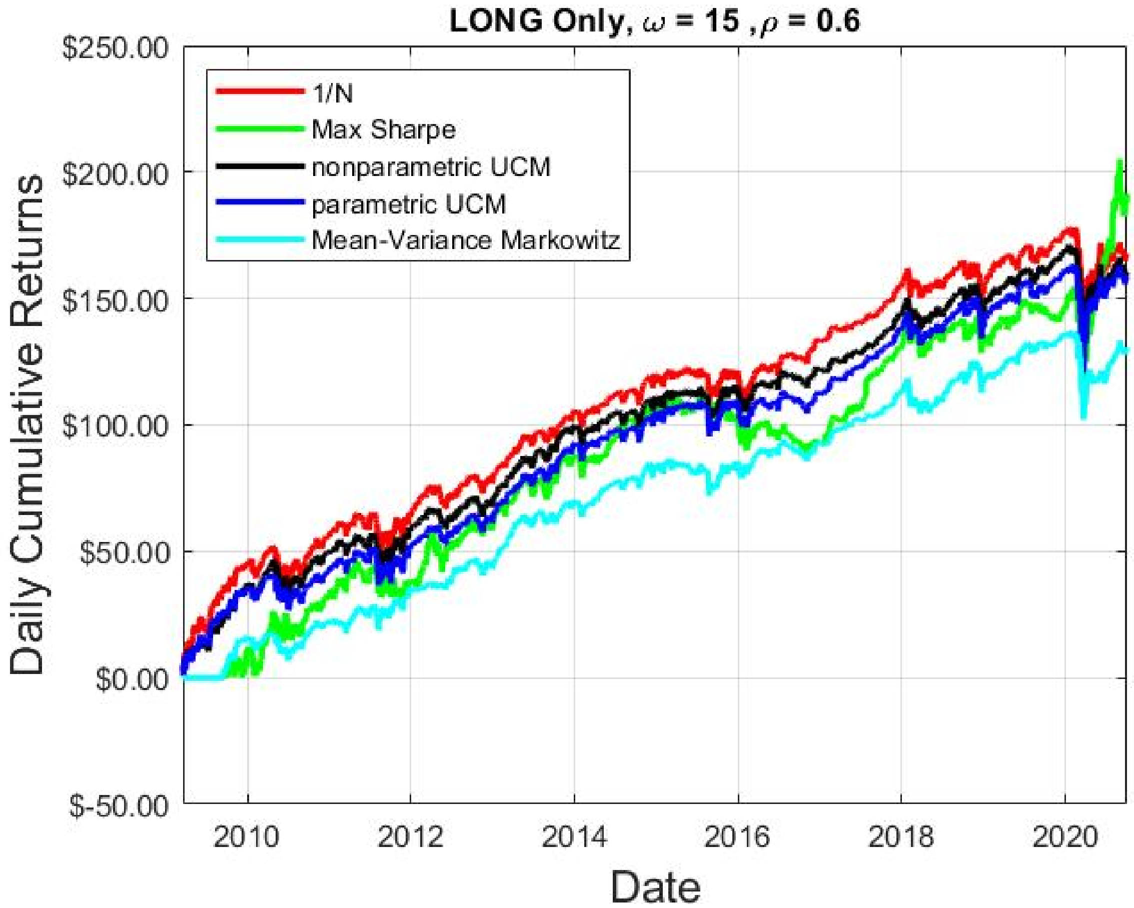

Figure 4: Accumulated Returns Long Only: Trial 4

Portfolio	Sharpe-Ratio
1/N	0.8313
Max Sharpe	0.7857
Non-Parametric UCM	0.8415
Parametric UCM	0.8190

Table 4: Sharpe Ratios Long Only: Trial 4

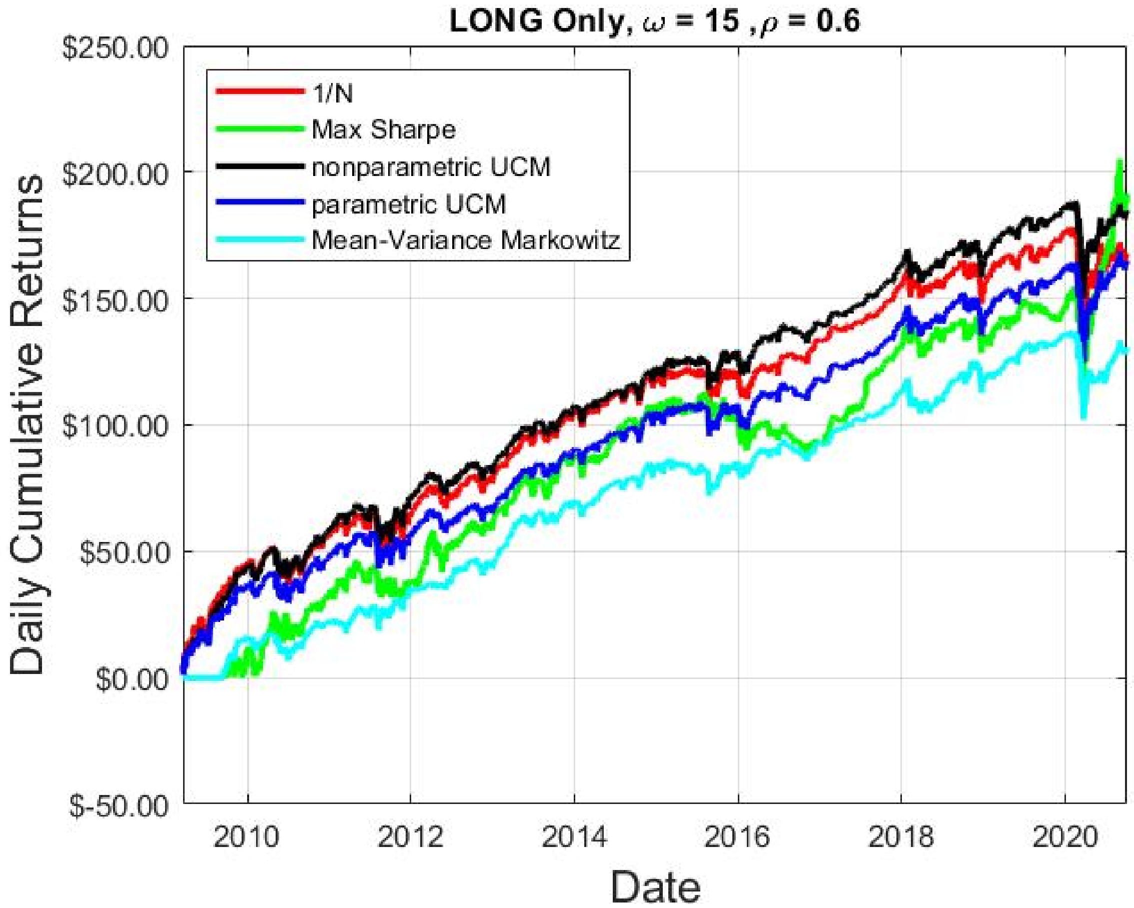

Figure 5: Accumulated Returns Long Only: Trial 5

Portfolio	Sharpe-Ratio
1/N	0.8313
Max Sharpe	0.7857
Non-Parametric UCM	0.9708
Parametric UCM	0.8606

Table 5: Sharpe Ratios Long Only: Trial 5

Since the figure showing the accumulated returns of the Max Sharpe Long-Short portfolio remains unchanged in every trial, we only display it once:

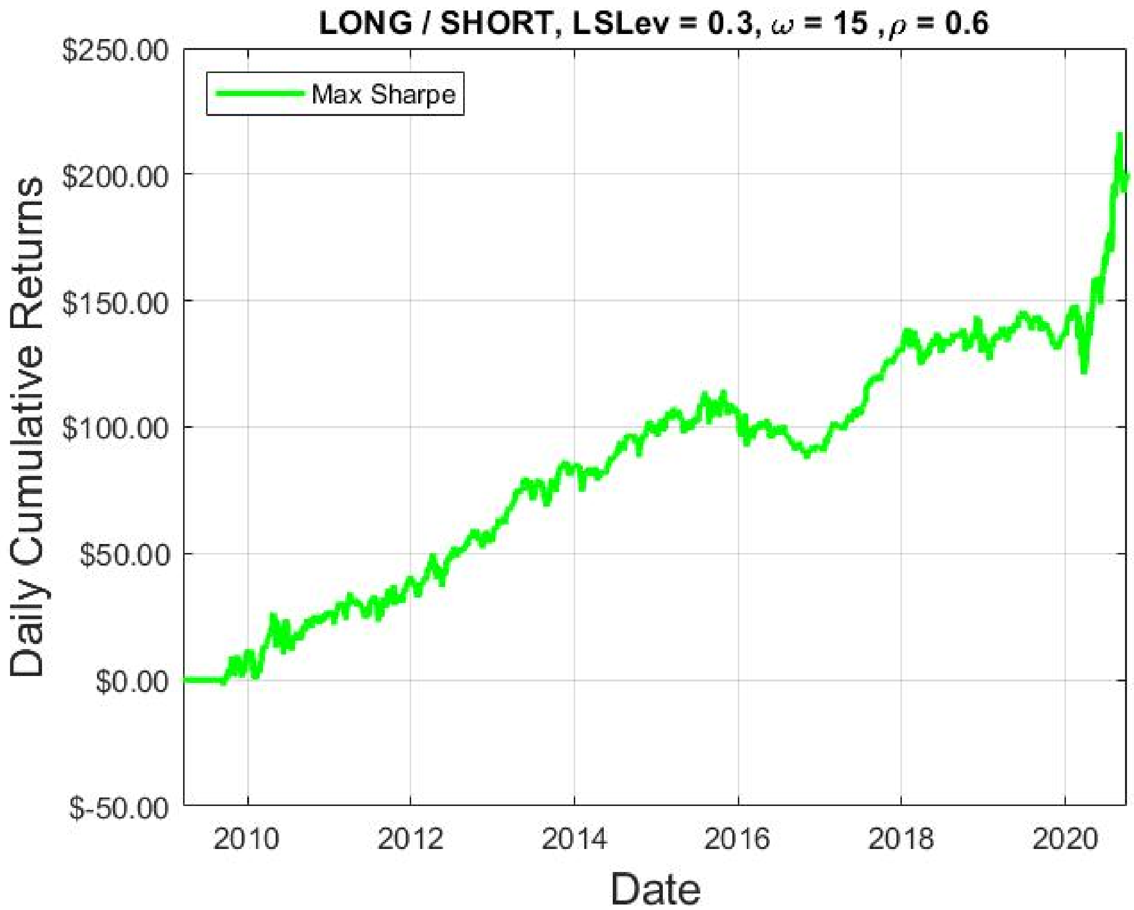

Figure 6: Accumulated Returns Long-Short

Portfolio	Sharpe-Ratio
Max Sharpe Long-Short	0.7841

Table 6: Sharpe Ratio Long-Short

Matlab Code

```
function MixNPortOptCompareOWNFINAL(RetMat, datevec, winlen, LSLev)
% Portfolio optimization on moving windows of length winlen:
% (A) LONG ONLY
%    1) MaxSharpe, based on the multivariate 2-comp MixN
%    2) nonparametric UCM MaxSharpe (UCM = univariate collapsing method)
%         where nonparametric means, simply take the sample mean and
%         std of the past winlen returns
%    3) parametric UCM MaxSharpe, where parametric means, use the mean
%         and sqrt(variance) from the univariate MixN
%    4) 1/N
% (B) LONG and SHORT
%    1) as in (1) above
%    NOTE: Shorting requires LSLev, Long-Short-Leverage, indicating how
%      much we can short, e.g., LSLev=0.3 indicates use 130/30 rule.
%
% INPUT
% RetMat (optional, default is DJIA-30 data) is a matrix of percentage log returns
% datevec (optional) is the datetime vector corresponding to the data
% winlen is length of moving window, default 250
%
% OUTPUT
% 2 labeled performance graphics of cusum returns (LONG, and LONG/SHORT)
%   such that the x and y axes are the same.

if nargin<1, RetMat=[]; end
if nargin<2, datevec=[]; end
if nargin<3, winlen=[]; end
if nargin<4, LSLev=0.3; end

if isempty(winlen), winlen=250; end

if 1==1 % these seem to work well
  omega=15; % used by MixNEMestimation; shrinkage prior strength
  rho=0.60; % used by MixNEMestimation; weighted likelihood
else % default IID MLE values
  omega=0; rho=1;
end
if isempty(RetMat)
  % load ('closePricesDJ30_20200810.mat') %#ok<LOAD>
  load ('closePricesDJ30_20201003.mat') %#ok<LOAD>
  % can use: summary(closePrices)   to see contents
  prices=closePrices.Variables;
  zeit=closePrices.Var1; clear closePrices
  % there are 30 stocks in there:
  %   AAPL, AXP, BA, CAT, CSCO, CVX, DIS, DOW, GS, HD,
  %   IBM, INTC, JNJ, JPM, KO, MCD, MMM, MRK, MSFT, NKE,
  %   PFE, PG, RTX, TRV, UNH, V, VZ, WBA, WMT, XOM

  % For DJIA, get at least 29 out of the 30 assets
  wstart=1; while sum(isnan(prices(wstart,:)))>1, wstart=wstart+1; end
  prices=prices(wstart:end,:);
```

```
  datevec=zeit(wstart:end);

  % make returns out of prices
  [Tmax, dmax]=size(prices); % Tmax gets overwritten later when using returns
  RetMat=zeros(Tmax-1,dmax);
  for i=1:dmax, u=prices(:,i); lr=log(u); rr=100*diff(lr); RetMat(:,i)=rr; end
  Tmax=Tmax-1; % lose one because conversion from price to return
  datevec=datevec((winlen+1):end);
end
if isempty(datevec), datevec=winlen:Tmax; end

% reserve memory
retseq1N=zeros(Tmax-winlen+1,1);
retseqSharpeLONG=zeros(Tmax-winlen+1,1);
retseqSharpeLS=zeros(Tmax-winlen+1,1);
%%%% reserve memory for UCM and Mean-Variance (MV)
retseqNPUCM=zeros(Tmax-winlen+1,1);
retseqPUCM=zeros(Tmax-winlen+1,1);
retseqMV=zeros(Tmax-winlen+1,1);


% The main FOR loop, over the windows of data
for wstart=1:(Tmax-winlen+1) % wstart is the time the window starts
  if mod(wstart,100)==0
    disp(['window start = ',int2str(wstart),' out of ',int2str(Tmax-winlen+1)])
  end
  wend=wstart+winlen-1; % end of the current window
  use=RetMat(wstart:wend,:);
  % remove assets with any missing returns
  badset=[]; for i=1:dmax, if any(isnan(use(:,i))), badset=union(badset,i); end, end
  okay=setdiff(1:dmax,badset);
  passretmat=RetMat(wstart:(wend-1),okay); % passed for parameter estimation
  retvecatt=RetMat(wend,okay); % returns vector at time t (to be used to evaluate performance)
  d=size(passretmat,2); if d ~= length(retvecatt), error('bad'), end
  % final check
  if any(any(isnan(passretmat))) || any(isnan(retvecatt)), error('bad'), end

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % LONG ONLY
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % The 1/N portfolio (obviously LONG Only)
  retseq1N(wstart)=retvecatt * ones(d,1)/d;

  % LONG Only MaxSharpe, based on the multivariate 2-comp MixN
  paramMMN=MixNEMestimation(passretmat,omega,rho); % Mult Mix Norm
  meanvecMMN = paramMMN.lam * paramMMN.mu1 ...  % Expected Return vector
           + (1-paramMMN.lam)* paramMMN.mu2;
  SigmaMMN = ...
      (   paramMMN.lam      * (paramMMN.Sig1 + paramMMN.mu1*paramMMN.mu1') ...
       + (1-paramMMN.lam) * (paramMMN.Sig2 + paramMMN.mu2*paramMMN.mu2') ) ...
```

```
        - meanvecMMN*meanvecMMN';
    w = maxSharpePortOptLongShort([], meanvecMMN, SigmaMMN, 0);
    retseqSharpeLONG(wstart)=retvecatt * w;

    % LONG Only using UCM (two ways)

    %%%% UCM
    % Define the required variables
    meanNPUCM = [];
    stdNPUCM = [];
    wNPUCM = [];
    meanPUCM = [];
    stdPUCM = [];
    wPUCM = [];

    for i=1:1e2
        % Create the weights via simulation
        r = unifrnd(0,1,d,1);
        w0 = log(r)/sum(log(r));
        % Create the pseudo-historical time series of asset returns
        Rp = passretmat*w0;

        %%% nonparametric UCM
        % Keep the weighting vector if the associated expected return is higher
        % and std is lower than from the 'old' weighting vector
        if isempty(meanNPUCM), meanNPUCM = mean(Rp); end
        if isempty(stdNPUCM), stdNPUCM = std(Rp); end
        if isempty(wNPUCM); wNPUCM = w0; end
        if (meanNPUCM < mean(Rp)) && (stdNPUCM > std(Rp))
            meanNPUCM = mean(Rp); stdNPUCM = std(Rp); wNPUCM = w0;
        end

        %%% parametric UCM
        % fit a univariate MixN estimation on Rp
        [paramPUCM, ESPUCM, SigmaPUCM]=MixNEMestimation(Rp, omega, rho);
        MixNmean = paramPUCM.lam * paramPUCM.mu1 ...    % Expected Return
                     + (1-paramPUCM.lam) * paramPUCM.mu2;
        % Keep the weighting vector if the associated expected return is higher
        % and std is lower than from the 'old' weighting vector
        if isempty(meanPUCM), meanPUCM = MixNmean; end
        if isempty(stdPUCM), stdPUCM = sqrt(SigmaPUCM); end
        if isempty(wPUCM); wPUCM = w0; end
        if (meanPUCM < MixNmean) && (stdPUCM > sqrt(SigmaPUCM))
            meanPUCM = MixNmean; stdPUCM = sqrt(SigmaPUCM); wPUCM = w0;
        end
    end

    % Calculate the return sequence for the PUCM portfolio
    retseqNPUCM(wstart) = retvecatt * wNPUCM;
    % Calculate the return sequence for the NPUCM portfolio
    retseqPUCM(wstart) = retvecatt * wPUCM;


    %%%% Mean-Variance Markowitz portfolio optimization
```

```
  % Calculate the optimal portfolio weights
  wMV = PortMNS(passretmat, omega, rho, 0.1);
  % Calculate the return sequence for the PUCM portfolio
  retseqMV(wstart) = retvecatt * wMV;

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % LONG SHORT
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % LONG/SHORT MaxSharpe, based on the multivariate 2-comp MixN
  w = maxSharpePortOptLongShort([], meanvecMMN, SigmaMMN, LSLev);
  retseqSharpeLS(wstart)=retvecatt * w;


end % for wstart=1:(Tmax-winlen+1)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Sharpe ratios
use=retseq1N;          sharpe_1N=sqrt(252)*mean(use)/std(use) %#ok<NASGU,NOPRT>
use=retseqSharpeLONG; sharpe_MaxSharpeLONG=sqrt(252)*mean(use)/std(use) %#ok<NASGU,NOPRT>
use=retseqNPUCM; sharpe_NPUCM=sqrt(252)*mean(use)/std(use) %#ok<NASGU,NOPRT>
use=retseqPUCM; sharpe_PUCM=sqrt(252)*mean(use)/std(use) %#ok<NASGU,NOPRT>
use=retseqSharpeLS; sharpe_MaxSharpeLONGSHORT=sqrt(252)*mean(use)/std(use) %#ok<NASGU,NOPRT>

% Plot the final results

% LONG Only
yyy1N=cumsum(retseq1N);
yyySharpeLONG=cumsum(retseqSharpeLONG);
yyyNPUCM=cumsum(retseqNPUCM);
yyyPUCM=cumsum(retseqPUCM);
yyyMV=cumsum(retseqMV);
fig1=figure;
  plot(...
      datevec,yyy1N,'r-', ...
      datevec,yyySharpeLONG,'g-', ...
      datevec,yyyNPUCM,'k-', ...
      datevec,yyyPUCM, 'b-', ...
      datevec,yyyMV, 'c-',...
      'linewidth',2)
  title(['LONG Only, \omega = ',int2str(omega),' ,\rho = ',num2str(rho)])
  legend(...
    '1/N', ...
    'Max Sharpe', ...
    'nonparametric UCM', ...
    'parametric UCM', ...
    'Mean-Variance Markowitz', ...
    'location','northwest')
  ylimLONG=ylim;
  ytickformat('usd')
  ylabel('Daily Cumulative Returns', 'fontsize',15)
```

```
  xlabel('Date','fontsize',15)
  grid


% LONG/SHORT
yyySharpeLS=cumsum(retseqSharpeLS);
fig2=figure;
  plot(...
      datevec,yyySharpeLS,'g-', ...
      'linewidth',2)
  title(['LONG / SHORT, LSLev = ',num2str(LSLev),', \omega = ',int2str(omega),' ,\rho = ',num2str(rho)])
  legend(...
    'Max Sharpe', ...
    'location','northwest')
  ylimLS=ylim;
  ytickformat('usd')
  ylabel('Daily Cumulative Returns', 'fontsize',15)
  xlabel('Date','fontsize',15)
  grid

% set y axis the same in each
ymin=min(ylimLONG(1), ylimLS(1));
ymax=max(ylimLONG(2), ylimLS(2));
figure(fig1), ylim([ymin ymax])
figure(fig2), ylim([ymin ymax])


% define the PortMNS and the meanvar function used for the MV portfolio
function w = PortMNS(data, omega, rho, tauAn)
if nargin < 2, omega = 15; end
if nargin < 3, rho = 0.60; end
if nargin < 4, tauAn = 0; end
% fit a Multivariate MixN on the data
paramMV=MixNEMestimation(data,omega,rho);
% calculate the expected returns and the associated covarinace matrix
mu = paramMV.lam * paramMV.mu1 ...
  + (1-paramMV.lam) * paramMV.mu2;
SigmaMixN = (paramMV.lam * (paramMV.Sig1 + paramMV.mu1*paramMV.mu1') ...
          + (1-paramMV.lam) * (paramMV.Sig2 + paramMV.mu2*paramMV.mu2') ) ...
          - mu*mu';
SigmaSym  = SymPDcovmatrix(SigmaMixN);
DEDR=100*((tauAn/100 + 1)^(1/250)-1); feas = max(mu)>=DEDR;
if feas
    w=meanvar(mu, SigmaSym, DEDR);
else
    w=zeros(length(mu),1);
end

function w = meanvar(mu, Sigma, tau)
opt = optimoptions('quadprog','Display', 'off');
d=length(mu); H = Sigma; f = zeros(d,1);
A = -mu'; B = -tau; LB = zeros(1,d); UB = ones(1,d); w0=UB'/d;
Aeq = ones(1,d); Beq = 1;  % to ensure that sum(w) = 1
```

```
zk_opt = quadprog(H,f,A,B,Aeq,Beq,LB,UB,w0,opt);
w = zk_opt(1:d)/sum(zk_opt);

function [w_opt, Var_opt] = maxSharpePortOptLongShort(w0, meanvec, Sigmat, longShort)
d = length(meanvec); % number of assets
if sum(meanvec>0)==0
  w_opt = zeros(d,1); Var_opt = 0; return
end
Sigmat  = SymPDcovmatrix(Sigmat);
if longShort==0 % long-only portfolio
    opt = optimoptions('quadprog','Display', 'off');
    LB = [zeros(1,d),-1e-12]; UB = ones(1,d);
    Aeq=[ones(1,d), -1; meanvec',0]; Beq = [0;1];
    Awuv = [-eye(d) ; zeros(1,d); eye(d)]; Bwuv = [-LB'; UB'];
    Awuvk =[Awuv,-Bwuv]; Bwuvk= zeros(size(Bwuv)); LB=[]; UB=[];
    Sigmat_zk = [[Sigmat, zeros(d,1)]; zeros(1,d+1)];
    zk_opt = quadprog(Sigmat_zk,zeros(d+1,1),Awuvk,Bwuvk,Aeq,Beq,LB,UB,[],opt);
    w_opt = zk_opt(1:d)./zk_opt(d+1);
    Var_opt = w_opt'*Sigmat*w_opt;
 elseif longShort~=0 % long-short portfolio e.g. longShort=0.3 corresponds to 130/30 portfolio
    opt = optimoptions('quadprog','Display', 'off','Algorithm','interior-point-convex','StepTolerance',0);
    LB = [-abs(longShort)*ones(d,1);zeros(2*d,1);-1e-12];     % longShort <= w
    UB = (1+abs(longShort))*ones(3*d,1);                       % [w,u,v] <= (1+longShort)
    Awuv = [zeros(1,d), ones(1,d), zeros(1,d);...
        -eye(d), zeros(d,d), zeros(d,d) ;...
         zeros(d,d), -eye(d), zeros(d,d) ;...
         zeros(d,d), zeros(d,d), -eye(d) ;...
         zeros(1,d), zeros(1,d), zeros(1,d);...
         eye(d), zeros(d,d), zeros(d,d) ;...
         zeros(d,d), eye(d), zeros(d,d) ;...
         zeros(d,d), zeros(d,d), eye(d) ;...
         ];
    Bwuv = [1+abs(longShort); -LB; UB]; Awuvk =[Awuv,-Bwuv];
    Bwuvk= zeros(size(Bwuv)); LB=[]; UB=[];
    Aeq = [eye(d),-1*eye(d),eye(d),zeros(d,1);...
         ones(1,d),zeros(1,d),zeros(1,d),-1;...
         meanvec',zeros(1,d),zeros(1,d),0];
    Beq = [zeros(d,1);0;1];
    Sigmat3d = [Sigmat,zeros(d,2*d);zeros(2*d,3*d)];
    Sigmat3dk = [[Sigmat3d,zeros(3*d,1)];zeros(1,3*d+1)];
    meanvec3dk = zeros(3*d+1,1);
    if ~isempty(w0)
      u0 =      w0(:) .* (w0(:)>0);
      v0 = -1*(w0(:) .* (w0(:)<0));
      wuvkinit=[w0(:);u0;v0;1];
      wuvk_opt = quadprog(Sigmat3dk,meanvec3dk,Awuvk,Bwuvk,Aeq,Beq,LB,UB,wuvkinit,opt) ;
    else
      wuvk_opt = quadprog(Sigmat3dk,meanvec3dk,Awuvk,Bwuvk,Aeq,Beq,LB,UB,[],opt) ;
```

```
    end
    w_opt =wuvk_opt(1:d)./wuvk_opt(3*d+1);
    Var_opt = w_opt'*Sigmat*w_opt;
end

function A = SymPDcovmatrix(A)
tol=1e-04; A=(A+A')/2;
[V,D]=eig(A); seig=diag(D); bad=find(seig<tol);
if ~isempty(bad), seig(bad)=tol; D=diag(seig); A=V*D*V'; end
A=(A+A')/2;

function [param, MixNES, MixNVariance]=MixNEMestimation (R,omega,rho,alpha,init,tol,maxit)
% Marc Paolella. Made for Jordan Oct 2020
% Estimates (univariate or multivariate) MixN
%   and returns the (possibly shrinkage and time-weighted) MLE
%   and, if dimension 1, also the predictive expected shortfall and
%   variance, where alpha dicates the probability level for ES
%
% See function mixnormEMm below for details of input and output
%
% Example: Simulate IID univariate MixN with parameters typical in finance,
%   and estimate the model.
%{
muTRUE  = [0.07 -0.03]; sigTRUE = [sqrt(1.34) sqrt(7.83)]; lamTRUE = [0.78 0.22];
T=1e4; R = mixnormsim(muTRUE,sigTRUE,lamTRUE,T);
[param, MixNES, MixNVariance] = MixNEMestimation(R)
% Now empircally check the ES and variance. It works... of course!
samplevariance = var(R)
qu=quantile(R,0.05); use=R(R<qu); sampleES=mean(use)
%}

if nargin<2, omega=0; end
if nargin<3, rho=1; end
if nargin<4, alpha=0.05; end
if nargin<5, init=[]; end
if nargin<6, tol=1e-6; end
if nargin<7, maxit=1e4; end

[n,p]=size(R);
if n<=p, error('Check the input time series R'), end

param = localmixnormEMm (R,omega,init,rho,tol,maxit);

if p==1
  % set up parameter vector as I require it, noting the sqrt() because
  %   this is for the univariate case, in which case, I model sigma1 and
  %   sigma2, and not their squares (matrices in the multivariate case)
  parampass = [param.mu1 param.mu2 sqrt(param.Sig1) sqrt(param.Sig2) param.lam];
  mu=[param.mu1 param.mu2];
  sig=[sqrt(param.Sig1) sqrt(param.Sig2)];
  lam=[param.lam 1-param.lam];
```

```
  VaRquantile = mixnormalquantile(parampass, alpha);
  t1 = (VaRquantile-mu(1))/sig(1); t1cdf=normcdf(t1); t1pdf=normpdf(t1);
  t2 = (VaRquantile-mu(2))/sig(2); t2cdf=normcdf(t2); t2pdf=normpdf(t2);
  numerator =   lam(1) * (-sig(1)*t1pdf + mu(1)*t1cdf) ...
              + lam(2) * (-sig(2)*t2pdf + mu(2)*t2cdf);
  MixNES = numerator/alpha;
  EY =  lam(1) * mu(1) ...
      + lam(2) * mu(2);
  EY2 = lam(1) * (mu(1)^2 + sig(1)^2) ...
      + lam(2) * (mu(2)^2 + sig(2)^2);
  MixNVariance = EY2 - EY^2;
else
  MixNES=[]; MixNVariance=[];
end

function [param,loglik,H1,crit,iter] = localmixnormEMm (y,omega,init,rho,tol,maxit)
% [param,loglik,H1,crit,iter] = mixnormEMm(y,omega,init,rho,tol,maxit)
%
% Estimates the parameters of the two-component p-variate mixed normal
%   distribution using the EM algorithm.
% y is nXp, the data.
% omega is the prior-strength is for the Hamilton quasi-Bayesian estimator.
%   pass 0 (default) for standard mle, no prior info,
%   pass a value >0 as the strength of the shrinkage prior
% init contains initial values as as structure, i.e.,
%    init.mu1, init.mu2, init.Sig1, init.Sig2, init.lam. Default is []
% rho indicates the weight for weighted likelihood:
%   Pass a scalar, then it is the "rho" for the hyperbolic weights, with a weight
%       of 1 yielding equally weighted (usual) likelihood,
%       and values less than 1 putting more weight on recent obs
%   Or pass vector [rho weightmeans weightsigmas weightlam]
%   where the latter 3 are booleans, and dictate of the mu_i, Sigma_i
%      and lambda_i are
%   Default is just the Sigma_i
% tol is required tolerance for each parameter to assume 'convergence'.
% maxit is maximum allowed number of iterations before giving up.
%
% param is a record with mu1, mu2, Sig1, Sig2, lam
% mu1 and mu2 are the 1Xp vectors of means of the two components
% Sig1 and Sig2 are the variance-covariance matrices,
% lam is the weight of the first component

if nargin < 6, maxit=1e4; end
if nargin < 5, tol=1e-6; end
if nargin < 4, rho=1; end
if nargin < 3, init=[]; end
if nargin < 2, omega=0; end

if length(rho)==1
  weightmeans=0; weightsigmas=1; weightlam=0;
else
  weightmeans=rho(2); weightsigmas=rho(3); weightlam=rho(4);
```

```
end

[n,p]=size(y);

% weighted likelihood
tvec=(1:n)'; likew=(n-tvec+1).^(rho(1)-1); likew=n*likew/sum(likew);

if 1==1 % based on typical financial data - see comment below
  s1=1.5; cov1=0.6; % variance and covariance for the prior on Sig1
  s2=10; cov2=4.6;
  m1=zeros(p,1); m2=-0.1*ones(p,1);
else % arbitrary
  s1=1; cov1=0.0; % variance and covariance for the prior on Sig1
  s2=1; cov2=0.0;
  m1=zeros(p,1); m2=zeros(p,1);
end
psig1=zeros(p,p); psig2=zeros(p,p);
for i=1:p, for j=1:p %#ok<ALIGN>
  if i==j, psig1(i,j)=s1; else, psig1(i,j)=cov1; end
  if i==j, psig2(i,j)=s2; else, psig2(i,j)=cov2; end
end, end
a1=2*omega; a2=omega/2; c1=20*omega; c2=20*omega; B1=a1*psig1; B2=a2*psig2;

% starting values
if isempty(init)
  mu1=m1; mu2=m2; Sig1=psig1; Sig2=5*psig2; lam=0.8;
else
  mu1=init.mu1; mu2=init.mu2; Sig1=init.Sig1; Sig2=init.Sig2; lam=init.lam;
end

wscheme=2; % just playing around with how the weighted likelihood
           % is used. See below how this is used.

iter = 0; crit=0; pdftol=1e-200; eigtol=1e-12;
new = [mu1 ; mu2 ; Sig1(:) ; Sig2(:) ; lam];
while 1
  iter=iter+1; old=new;
  %if iter==1000, disp(iter), end
  Sig1=(Sig1+Sig1')/2; Sig2=(Sig2+Sig2')/2; % sometimes off by a tiny amount

  [V,D] = eig(Sig1); dd=diag(D);
  if any(dd<eigtol), dd=max(dd,eigtol); D=diag(dd); Sig1=V*D*V'; end
  [V,D] = eig(Sig2); dd=diag(D);
  if any(dd<eigtol), dd=max(dd,eigtol); D=diag(dd); Sig2=V*D*V'; end

  Comp1=mvnpdf(y,mu1',Sig1); Comp1=max(Comp1,pdftol);
  Comp2=mvnpdf(y,mu2',Sig2); Comp2=max(Comp2,pdftol);
  mixn =lam*Comp1+(1-lam)*Comp2;
  H1=lam*Comp1./mixn; H2=1-H1;

  if weightmeans, G1=H1.*likew; G2=H2.*likew; else, G1=H1; G2=H2; end
  if wscheme==1, N1=sum(G1); N2=sum(G2); else, N1=sum(H1); N2=sum(H2); end
  rep1 = repmat(G1,1,p); rep2 = repmat(G2,1,p);
  mu1 = ( c1*m1 + sum( rep1 .* y )' ) / (c1 + N1);
```

```
  mu2 = ( c2*m2 + sum( rep2 .* y )' ) / (c2 + N2);

  if weightsigmas, G1=H1.*likew; G2=H2.*likew; else, G1=H1; G2=H2; end
  if wscheme==1, N1=sum(G1); N2=sum(G2); else, N1=sum(H1); N2=sum(H2); end
  rep1 = repmat(G1,1,p); rep2 = repmat(G2,1,p);
  ymm = y - repmat(mu1',n,1); ymmH = rep1 .* ymm; outsum1=ymmH'*ymm;
  Sig1 = (B1 + c1*(m1-mu1)*(m1-mu1)' + outsum1 ) / (a1+N1);
  ymm = y - repmat(mu2',n,1); ymmH = rep2 .* ymm; outsum2=ymmH'*ymm;
  Sig2 = (B2 + c2*(m2-mu2)*(m2-mu2)' + outsum2 ) / (a2+N2);

  if weightlam, G1=H1.*likew; else, G1=H1; end
  lam = mean(G1);

  new = [mu1 ; mu2 ; Sig1(:) ; Sig2(:) ; lam];
  crit = max (abs (old-new));
  if (crit < tol) || (iter >= maxit), break, end
end
loglik=sum(log(mixn));
param.mu1=mu1; param.mu2=mu2; param.Sig1=Sig1; param.Sig2=Sig2; param.lam=lam;


function q = mixnormalquantile(param, level)
mu1=param(1); mu2=param(2); sig1=param(3); sig2=param(4); lam=param(5);
themean=lam*mu1+(1-lam)*mu2;
mom2=lam*(mu1^2+sig1^2) + (1-lam)*(mu2^2+sig2^2); thevar=mom2-themean^2;
x0 = norminv(level, themean, sqrt(thevar)); % initial guess
opt=optimoptions('fminunc'); opt=optimoptions(opt,'Display','none');
q = fminunc(@(z) mnq(z, param, level), ...
      x0, opt);

function discrep = mnq(z, param, level)
mu1=param(1); mu2=param(2); sig1=param(3); sig2=param(4); lam=param(5);
thecdf = lam * normcdf(z, mu1, sig1) + (1-lam) * normcdf(z, mu2, sig2);
discrep = (thecdf - level)^2;
```