AMS-512 Capital Markets and Portfolio Theory

Homework Solution:
Portfolio Optimization with Missing Data, Denoising, and Factor Models

Robert J. Frey, Research Professor
Stony Brook University, Applied Mathematics and Statistics

frey@ams.sunysb.edu
http://www.ams.sunysb.edu/~frey

Set-Up

Packages

These packages contain the code for quadratic programming, estimating mean and covariance with missing values, and fitting an orthonormal factor model.

```
In[•]:= Get[FileNameJoin[{NotebookDirectory[], "QuadraticProgramming.m"}]]
Get[FileNameJoin[{NotebookDirectory[], "MeanCovMissingMLE.m"}]]
Get[FileNameJoin[{NotebookDirectory[], "FactorFitMLE.m"}]]
```

Data

It is not necessary to rebuild the returns data. We will import the data that was computed and exported in Missing-Values02.nb.

```
dbReturns = Import[FileNameJoin[{NotebookDirectory[], "dbReturns.m"}]];
```

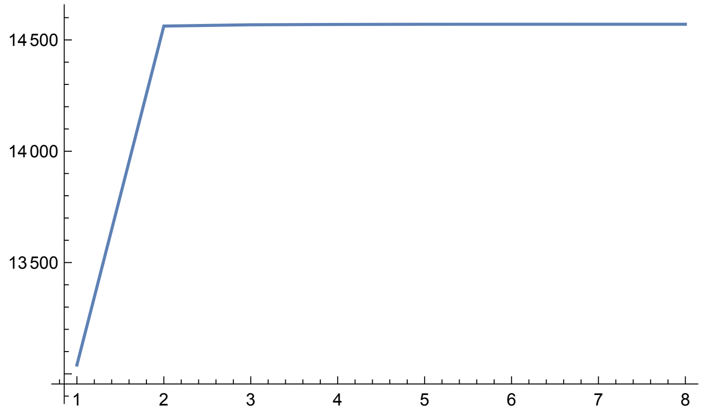
Mean and Covariance Estimation

The estimation of the mean vector and covariance matrix is:

```
In[•]:= {{vnMean, mnCovariance}, {mnCompleted, mnCross}, vnLogLik, sRunID} =
  xMeanCovMissingMLE[dbReturns];
```

In[∘]:=
```
ListPlot[vnLogLik, Joined → True, PlotRange → All]
```

Out[∘]=



Factor Model I (No Denoising)

The first step is to fit the factor model with out denoising the correlation matrix. We can use the BIC to pick the order of the factor model:

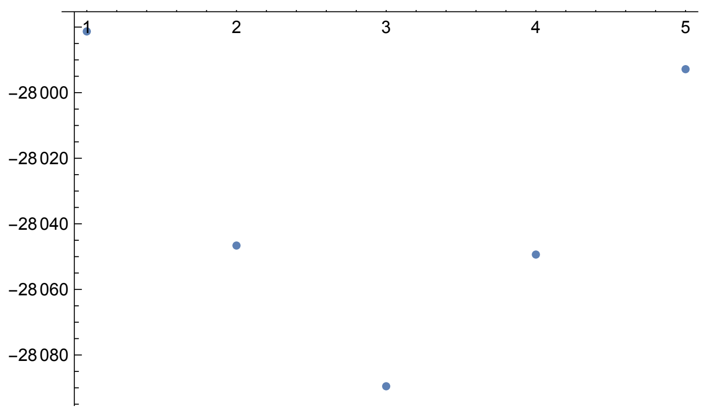
In[∘]:=
```
mnBIC = Table[{i, Last@xFactorFitMLE[Length[dbReturns⟦1⟧],
      mnCovariance, xInitializeFactorModel[mnCovariance, i]]}, {i, 1, 5}]
```

Out[∘]= {{1, -27 981.3}, {2, -28 046.6}, {3, -28 089.5}, {4, -28 049.4}, {5, -27 992.8}}

A model order (number of factors) of three looks best:

In[∘]:=
```
ListPlot[mnBIC]
```

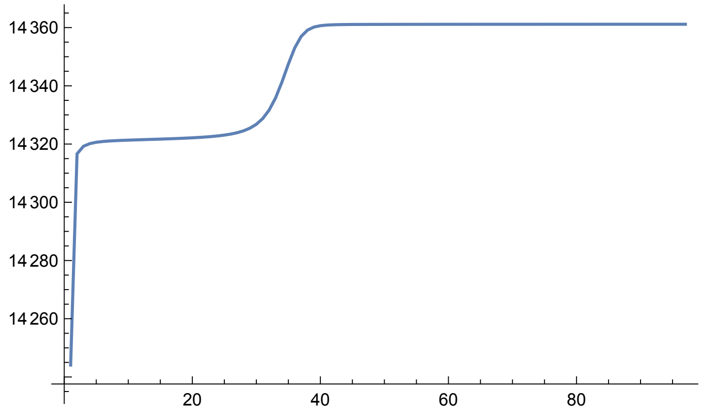
Out[∘]=



In[∘]:=
```
{{mnFactors, mnErrDiag}, vnLogLikHistory, nBIC} = xFactorFitMLE[
   Length[dbReturns⟦1⟧], mnCovariance, xInitializeFactorModel[mnCovariance, 3]];
```

Note how the log likelihood converges.

In[•]:=
```
ListPlot[vnLogLikHistory, Joined → True, PlotRange → All]
```

Out[•]=


Portfolio Optimization I (No Denoising)

We next compute the efficient portfolios and frontier using the mean vector and covariance matrix built from the factor model:

In[•]:=
```
mnModelCov = mnFactors.(mnFactorsᵀ) + mnErrDiag;
```

In[•]:=
```
Det[mnModelCov] / Det[mnCovariance]
```

Out[•]=
```
32.9343
```

The constraints, right-hand sides, bounds are below. Note that the bounds are set to $0.01 \le x_i \le 0.10$; *i.e.*, the positions are between 1% and 10%. Note that the right-hand side specification contains a symbolic τ which will be set when we iterate over the target returns.

In[•]:=
```
mnConstraints = {
    Array[1. &, Length[dbReturns[[2]]]],
    vnMean
   };
mnRHS = {{1., 0}, {τ, 1}};
mnBounds = Table[{0.01, 0.1}, {Length[dbReturns[[2]]]}];
```

The minimum variance portfolio and its associated mean are computed by dropping the target return constraint:

In[•]:=
```
vnMinVarPortfolioF = Last@xQuadraticProgramming[{{}, mnModelCov},
     {Array[1. &, Length[dbReturns[[2]]]]}, {{1., 0.}}, mnBounds, PrecisionGoal → 9];
nMinVarMeanF = vnMean.vnMinVarPortfolioF
```

Out[•]=
```
0.00665219
```

The maximum return portfolio and its associated mean are computed by solving the linear program:

In[•]:=
```
vnMaxRetPortfolioF = LinearProgramming[-vnMean,
    {Array[1. &, Length[dbReturns[[2]]]]}, {{1., 0.}}, mnBounds];
nMaxRetMeanF = vnMean.vnMaxRetPortfolioF
```

Out[•]=
```
0.0130284
```

In[ ]:=
```
TableForm[Chop[{vnMinVarPortfolioF, vnMaxRetPortfolioF}ᵀ, 0.0001],
 TableHeadings → {vsNames, {"MinVar", "MaxRet"}}]
```

Out[ ]//TableForm=

	MinVar	MaxRet
3M Co	0.0100011	0.01
American Express Co	0.0100001	0.01
Apple Inc	0.0100004	0.1
Boeing Co	0.0100002	0.08
Caterpillar Inc	0.0100001	0.01
Chevron Corp	0.0100012	0.01
Cisco Systems Inc	0.0100004	0.01
Coca-Cola Co	0.0999985	0.01
The Walt Disney Co	0.0100003	0.01
Missing[NotAvailable]	0.0100001	0.01
Exxon Mobil Corp	0.0153878	0.01
Goldman Sachs Group Inc	0.0100002	0.01
The Home Depot Inc	0.0392478	0.1
Intel Corp	0.0100006	0.01
International Business Machines Corp	0.0517902	0.01
Johnson & Johnson	0.0999988	0.01
JPMorgan Chase & Co	0.0100002	0.01
McDonald's Corp	0.099999	0.1
Merck & Co Inc	0.0290125	0.01
Microsoft Corp	0.0100014	0.1
Nike Inc	0.0273861	0.1
Pfizer Inc	0.028777	0.01
Procter & Gamble Co	0.0999995	0.01
The Travelers Companies Inc	0.010001	0.01
UnitedHealth Group Inc	0.0285905	0.1
United Technologies Corp	0.0100003	0.01
Verizon Communications Inc	0.099999	0.01
Visa Inc	0.0198053	0.1
Walgreens Boots Alliance Inc	0.0100007	0.01
Walmart Inc	0.0999997	0.01

The efficient frontier will be computed at the following 20 target returns starting with the min variance and ending with the max return.

In[ ]:=
```
vnReturnTargetsF =
 Table[τ, {τ, nMinVarMeanF, nMaxRetMeanF, (nMaxRetMeanF - nMinVarMeanF) / 19}]
```

Out[ ]=
```
{0.00665219, 0.00698778, 0.00732337, 0.00765896, 0.00799455, 0.00833014,
 0.00866573, 0.00900132, 0.00933691, 0.0096725, 0.0100081, 0.0103437, 0.0106793,
 0.0110149, 0.0113504, 0.011686, 0.0120216, 0.0123572, 0.0126928, 0.0130284}
```

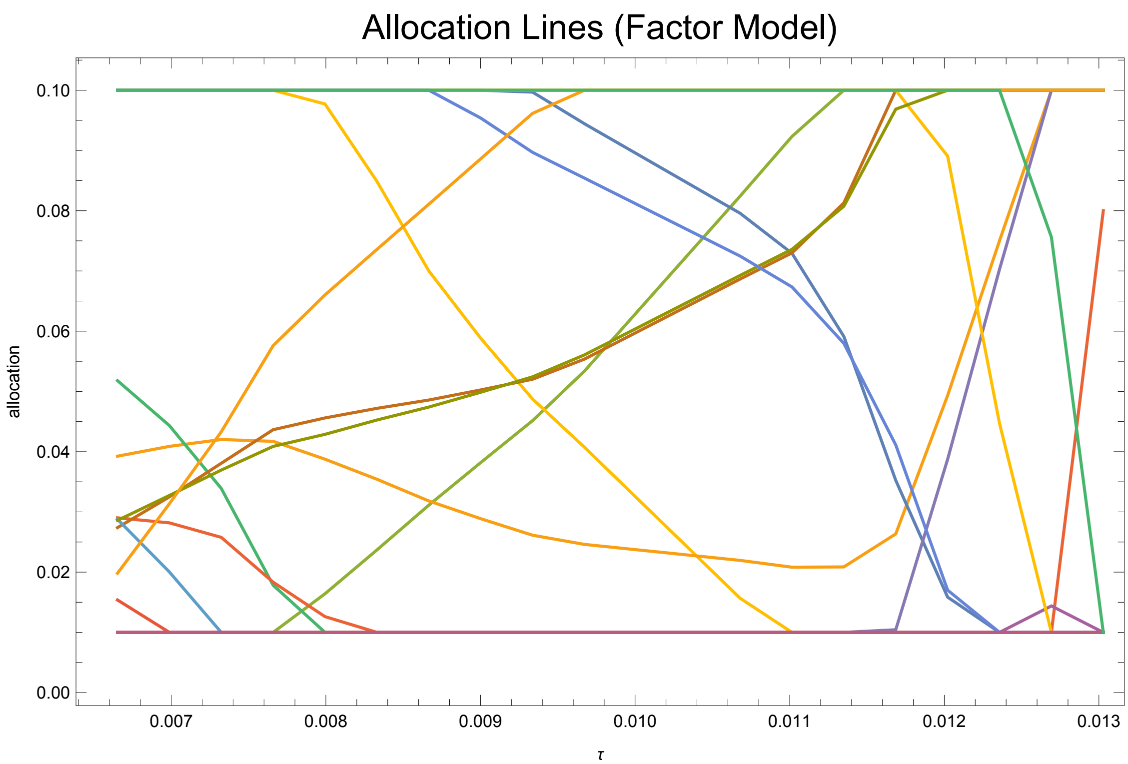
In[ ]:=
```
mnEfficientPortfoliosF = Table[
   Last@xQuadraticProgramming[{{}, mnModelCov}, mnConstraints, mnRHS,
     mnBounds, AccuracyGoal → 9, PrecisionGoal → 12, MaxIterations → 2000],
   {τ, vnReturnTargetsF}
  ];
```

The allocations across the efficient frontier are a line in $\mathbb{R}^{30}$ which we can plot on a two dimensional graph by projecting each component of the line onto the y – axis.

In[ ]:=
```
vmnAllocationLinesF = {vnReturnTargetsF, #}ᵀ & /@ Transpose[mnEfficientPortfoliosF];
```

In[•]:=
```
ListLinePlot[MapThread[Tooltip, {vmnAllocationLinesF, dbReturns[[2]]}],
  PlotRange → All, Frame → True, FrameLabel → {{"allocation", ""},
    {"τ", Style["Allocation Lines (Factor Model)", FontSize → 18]}}, ImageSize → 600]
```

Out[•]=

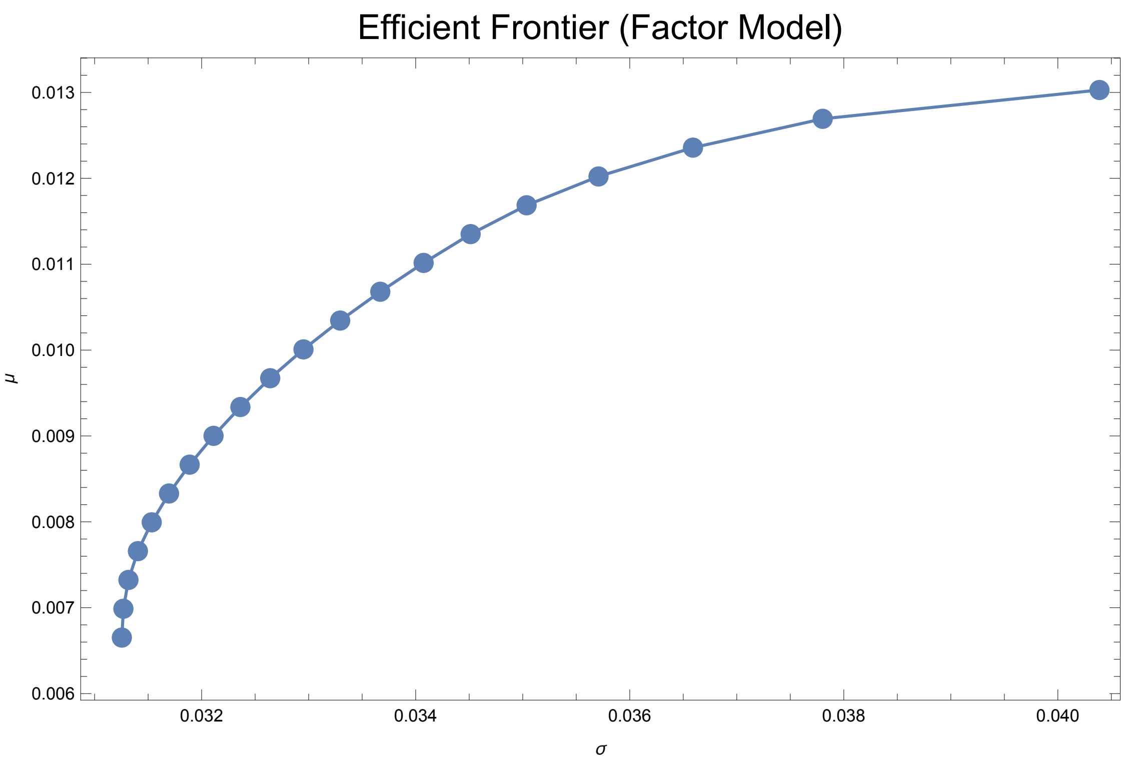


Given the allocations across the efficient portfolios we can compute and plot each $\{\sigma, \mu\}$ – pair. Note that we have set a tooltip so that hovering the mouse over each $\{\sigma, \mu\}$ – pair will show the associated portfolio allocations.

In[•]:=
```
mnEfficientFrontierF = {√(#.mnModelCov.#), #.vnMean} & /@ mnEfficientPortfoliosF;
```

In[•]:=
```
ListLinePlot[MapThread[Tooltip, {mnEfficientFrontierF, mnEfficientPortfoliosF}],
  Mesh → All, PlotRange → All, Frame → True, FrameLabel → {{"μ", ""}, {"σ",
     Style["Efficient Frontier (Factor Model)", FontSize → 18]}}, ImageSize → 600]
```

Out[•]=



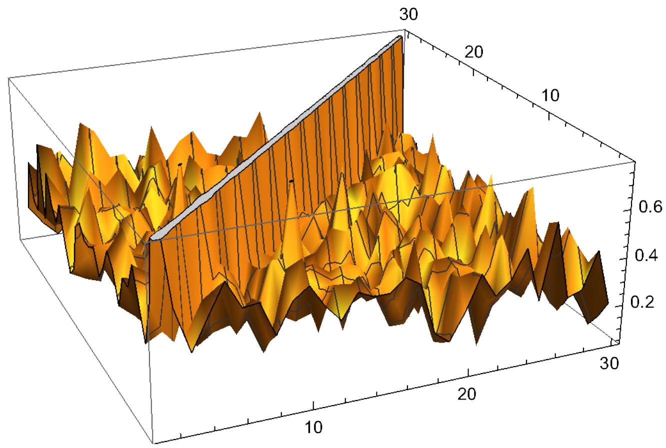
Denoising the Correlation Matrix

Denoising Using an MP-Threshold

Here will examine the estimates from dbReturns starting with the raw covariance estimated with xMeanCovMissingMLE. Note that $\rho_{i,j} = \sigma_{i,j}/\sigma_i\,\sigma_j$.

In[•]:=
```
mnCorrelation = mnCovariance / (KroneckerProduct[#, #] &[√Diagonal[mnCovariance]]);
```

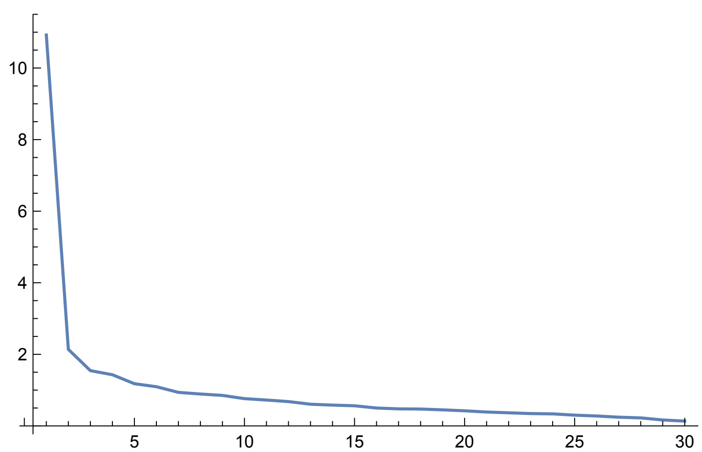
In[•]:=
```
ListPlot3D[mnCorrelation]
```

Out[•]= 

In[•]:=
```
vnEigen = Eigenvalues[mnCorrelation]
```

Out[•]=
```
{10.9219, 2.13899, 1.54447, 1.42987, 1.17877, 1.09485, 0.936727, 0.891062,
 0.852202, 0.761979, 0.722757, 0.679724, 0.606094, 0.581295, 0.56236,
 0.499533, 0.476253, 0.471183, 0.448413, 0.421802, 0.3878, 0.366427, 0.344494,
 0.336029, 0.300302, 0.27689, 0.243467, 0.224005, 0.16572, 0.134608}
```

In[•]:=
```
ListLinePlot[vnEigen, PlotRange → All]
```

Out[•]= 

The M-P parameter q is estimated by

In[•]:=
```
{iT, iN} = Dimensions[dbReturns⟦3⟧]
nQ = N[iN / iT]
```

Out[•]=
```
{195, 30}
```

Out[•]=
```
0.153846
```

The M-P range is

In[•]:=
```
vnQ = N[(1 - {1, -1} √nQ)^2]
```

Out[•]=
```
{0.369382, 1.93831}
```

Experience has shown that at the upper edge of the distribution a “fuzz” of $N^{-2/3}$ can be added to adjust for the finite sample.

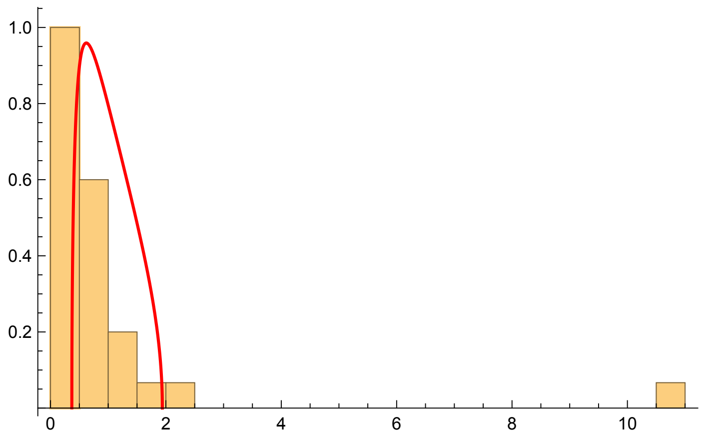
In[•]:=
```
nCutOff = Max[vnQ] + iN^-2/3
```

Out[•]=
```
2.04189
```

A plot of the eigenvalues and Marchenko-Pastur distribution for this case is

In[•]:=
```
Show[
 Histogram[vnEigen, Automatic, PDF, PlotRange → All],
 Plot[Evaluate@PDF[MarchenkoPasturDistribution[nQ], x],
  {x, Min@vnQ, Max@vnQ}, PlotStyle → {Red}]
]
```

Out[•]=



Define the cut-index as the index of the first eigenvalue to fall below the cut-off.

In[•]:=
```
iCutIndex = First@First@Position[nCutOff < # & /@ vnEigen, False]
```

Out[•]=
```
3
```

We can finally denoise the correlation matrix by setting the eigenvalues below the cut-off, preserving the trace.

In[•]:=
```
vnDenoisedEigen = Join[vnEigen[[1 ;; iCutIndex - 1]],
  Table[Total[vnEigen[[iCutIndex ;;]]] / Length[vnEigen[[iCutIndex ;;]]],
   {Length[vnEigen] - iCutIndex + 1}]]
```

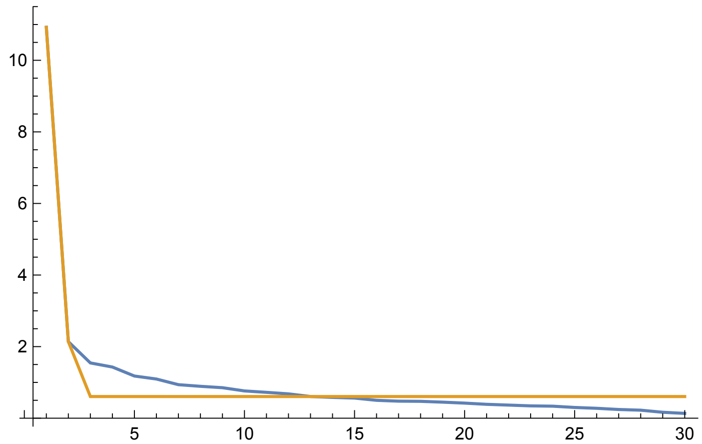
Out[•]=
```
{10.9219, 2.13899, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967,
 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967,
 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967,
 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967, 0.604967}
```

In[•]:=
```
Total[vnEigen]
Total[vnDenoisedEigen]
```

Out[•]=
```
30.
```

Out[•]=
```
30.
```

In[•]:=
```
ListLinePlot[{vnEigen, vnDenoisedEigen}, PlotRange → All]
```

Out[•]=


In[•]:=
```
vmnSVD = SingularValueDecomposition[mnCorrelation];
```

In[•]:=
```
Dimensions /@ vmnSVD
```

Out[•]= {{30, 30}, {30, 30}, {30, 30}}

In[•]:=
```
mnDenoisedCor = vmnSVD⟦1⟧.DiagonalMatrix[vnDenoisedEigen].Transpose@vmnSVD⟦3⟧;
```

In[•]:=
```
Det[mnDenoisedCor] / Det[mnCorrelation]
```

Out[•]= 145.818

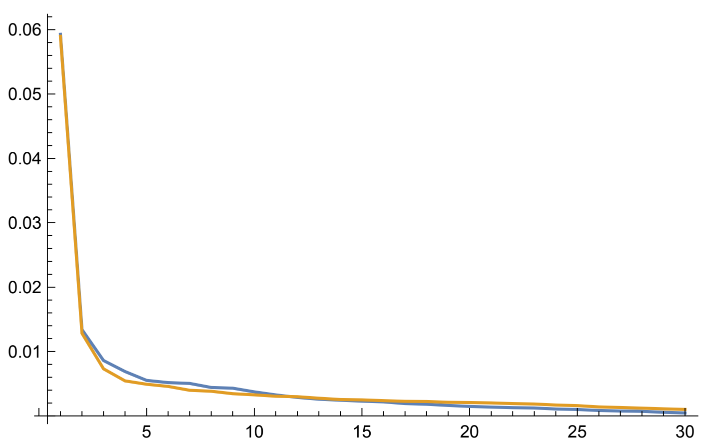
We can now reconstruct the denoised covariance: $\sigma_{i,j} = \rho_{i,j}\,\sigma_i\,\sigma_j$.

In[•]:=
```
mnDenoisedCov = mnDenoisedCor (KroneckerProduct[#, #] &[√Diagonal[mnCovariance]]);
```

In[•]:=
```
Det[mnDenoisedCov] / Det[mnCovariance]
```

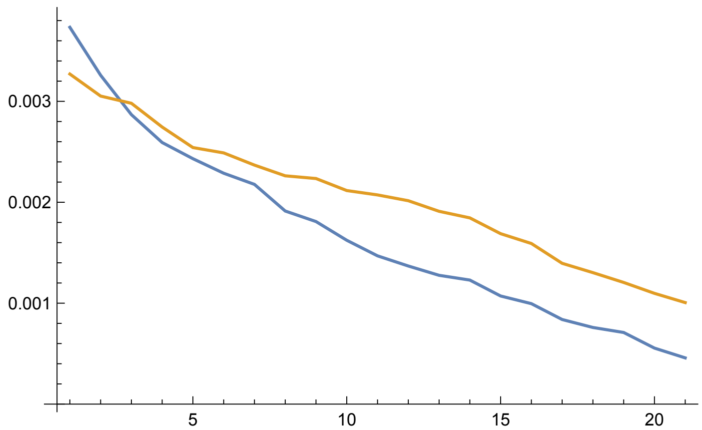
Out[•]= 145.818

In[•]:=
```
ListLinePlot[
 {Eigenvalues[mnCovariance], Eigenvalues[mnDenoisedCov]}, PlotRange → All]
```

Out[•]=


In[∘]:=
```
ListLinePlot[{Eigenvalues[mnCovariance]⟦10 ;;⟧,
  Eigenvalues[mnDenoisedCov]⟦10 ;;⟧}, PlotRange → All]
```

Out[∘]=



Factor Model II (Denoised Correlation)

We can use the BIC to pick the order of the factor model:

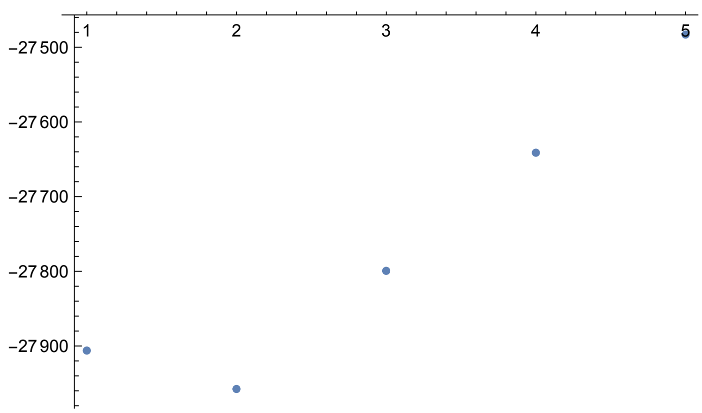
In[∘]:=
```
mnBIC = Table[{i, Last@xFactorFitMLE[Length[dbReturns⟦1⟧],
     mnDenoisedCov, xInitializeFactorModel[mnDenoisedCov, i]]}, {i, 1, 5}]
```

Out[∘]=
```
{{1, -27 906.1}, {2, -27 957.6}, {3, -27 799.4}, {4, -27 641.2}, {5, -27 483.}}
```

A model order (number of factors) of TWO looks best:

In[∘]:=
```
ListPlot[mnBIC]
```

Out[∘]=

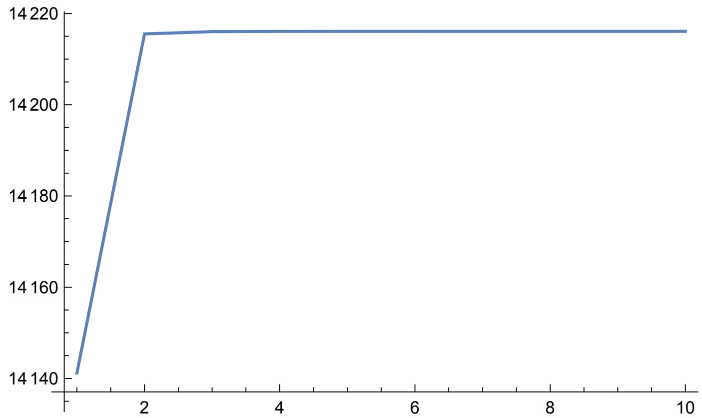


In[∘]:=
```
{{mnDenoisedFactors, mnDenoisedErrDiag}, vnDenoisedLogLikHistory, nDenoisedBIC} =
  xFactorFitMLE[Length[dbReturns⟦1⟧], mnDenoisedCov,
   xInitializeFactorModel[mnDenoisedCov, 2]];
```

Note how the log likelihood converges.

In[∘]:=
```
ListPlot[vnDenoisedLogLikHistory, Joined → True, PlotRange → All]
```

Out[∘]=



Portfolio Optimization II (Denoised Correlation)

In[∘]:=
```
mnDenoisedModelCov = mnDenoisedFactors.(mnDenoisedFactorsᵀ) + mnDenoisedErrDiag;
```

In[∘]:=
```
Det[mnDenoisedModelCov] / Det[mnCovariance]
```

Out[∘]=
```
146.248
```

In[∘]:=
```
mnConstraints = {
    Array[1. &, Length[dbReturns⟦2⟧]],
    vnMean
   };
mnRHS = {{1., 0}, {τ, 1}};
mnBounds = Table[{0.01, 0.1}, {Length[dbReturns⟦2⟧]}];
```

In[∘]:=
```
vnDenoisedMinVarPortfolioF = Last@xQuadraticProgramming[{{}, mnDenoisedModelCov},
     {Array[1. &, Length[dbReturns⟦2⟧]]}, {{1., 0.}}, mnBounds, PrecisionGoal → 9];
nDenoisedMinVarMeanF = vnMean.vnDenoisedMinVarPortfolioF
```

Out[∘]=
```
0.0066492
```

In[∘]:=
```
vnDenoisedMaxRetPortfolioF = LinearProgramming[-vnMean,
    {Array[1. &, Length[dbReturns⟦2⟧]]}, {{1., 0.}}, mnBounds];
nDenoisedMaxRetMeanF = vnMean.vnDenoisedMaxRetPortfolioF
```

Out[∘]=
```
0.0130284
```

In[∘]:=
```
TableForm[Chop[{vnDenoisedMinVarPortfolioF, vnDenoisedMaxRetPortfolioF}ᵀ, 0.0001],
  TableHeadings → {vsNames, {"MinVar", "MaxRet"}}]
```

Out[∘]//TableForm=

	MinVar	MaxRet
3M Co	0.0100011	0.01
American Express Co	0.0100001	0.01
Apple Inc	0.0100003	0.1
Boeing Co	0.0100002	0.08
Caterpillar Inc	0.0100001	0.01
Chevron Corp	0.0100006	0.01
Cisco Systems Inc	0.0100004	0.01
Coca-Cola Co	0.0999986	0.01
The Walt Disney Co	0.0100003	0.01
Missing[NotAvailable]	0.0100001	0.01
Exxon Mobil Corp	0.0100018	0.01
Goldman Sachs Group Inc	0.0100002	0.01
The Home Depot Inc	0.0400631	0.1
Intel Corp	0.0100006	0.01
International Business Machines Corp	0.049196	0.01
Johnson & Johnson	0.0999989	0.01
JPMorgan Chase & Co	0.0100002	0.01
McDonald's Corp	0.0999988	0.1
Merck & Co Inc	0.0280227	0.01
Microsoft Corp	0.0100013	0.1
Nike Inc	0.0364861	0.1
Pfizer Inc	0.0321341	0.01
Procter & Gamble Co	0.0999995	0.01
The Travelers Companies Inc	0.0100009	0.01
UnitedHealth Group Inc	0.0339698	0.1
United Technologies Corp	0.0100004	0.01
Verizon Communications Inc	0.0999987	0.01
Visa Inc	0.0101244	0.1
Walgreens Boots Alliance Inc	0.0100007	0.01
Walmart Inc	0.0999998	0.01

In[∘]:=
```
vnDenoisedReturnTargetsF = Table[τ, {τ, nDenoisedMinVarMeanF,
   nDenoisedMaxRetMeanF, (nDenoisedMaxRetMeanF - nDenoisedMinVarMeanF) / 19}]
```

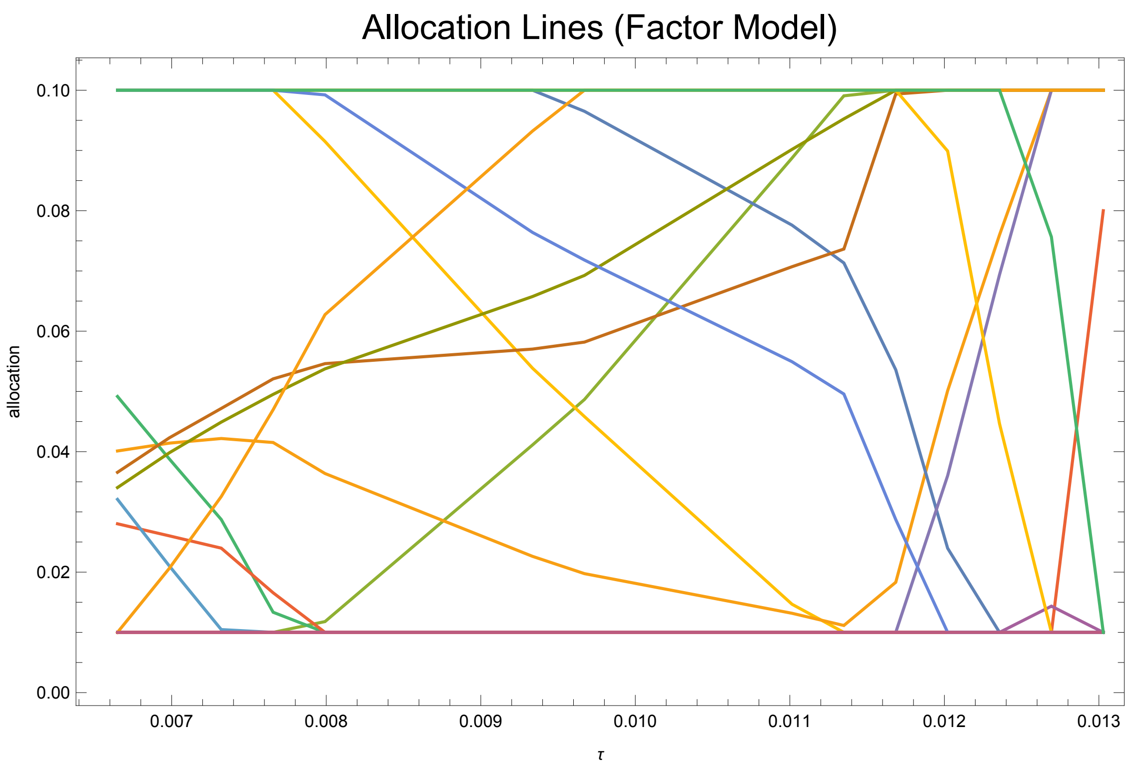
Out[∘]=
```
{0.0066492, 0.00698495, 0.00732069, 0.00765644, 0.00799219, 0.00832794, 0.00866368,
 0.00899943, 0.00933518, 0.00967093, 0.0100067, 0.0103424, 0.0106782,
 0.0110139, 0.0113497, 0.0116854, 0.0120212, 0.0123569, 0.0126927, 0.0130284}
```

In[∘]:=
```
mnDenoisedEfficientPortfoliosF = Table[
   Last@xQuadraticProgramming[{{}, mnDenoisedModelCov}, mnConstraints, mnRHS,
     mnBounds, AccuracyGoal → 9, PrecisionGoal → 12, MaxIterations → 2000],
   {τ, vnDenoisedReturnTargetsF}
  ];
```

In[∘]:=
```
vmnDenoisedAllocationLinesF =
  {vnDenoisedReturnTargetsF, #}ᵀ & /@ Transpose[mnDenoisedEfficientPortfoliosF];
```

In[∘]:=
```
ListLinePlot[MapThread[Tooltip, {vmnDenoisedAllocationLinesF, dbReturns[[2]]}],
  PlotRange → All, Frame → True, FrameLabel → {{"allocation", ""},
   {"τ", Style["Allocation Lines (Factor Model)", FontSize → 18]}}, ImageSize → 600]
```

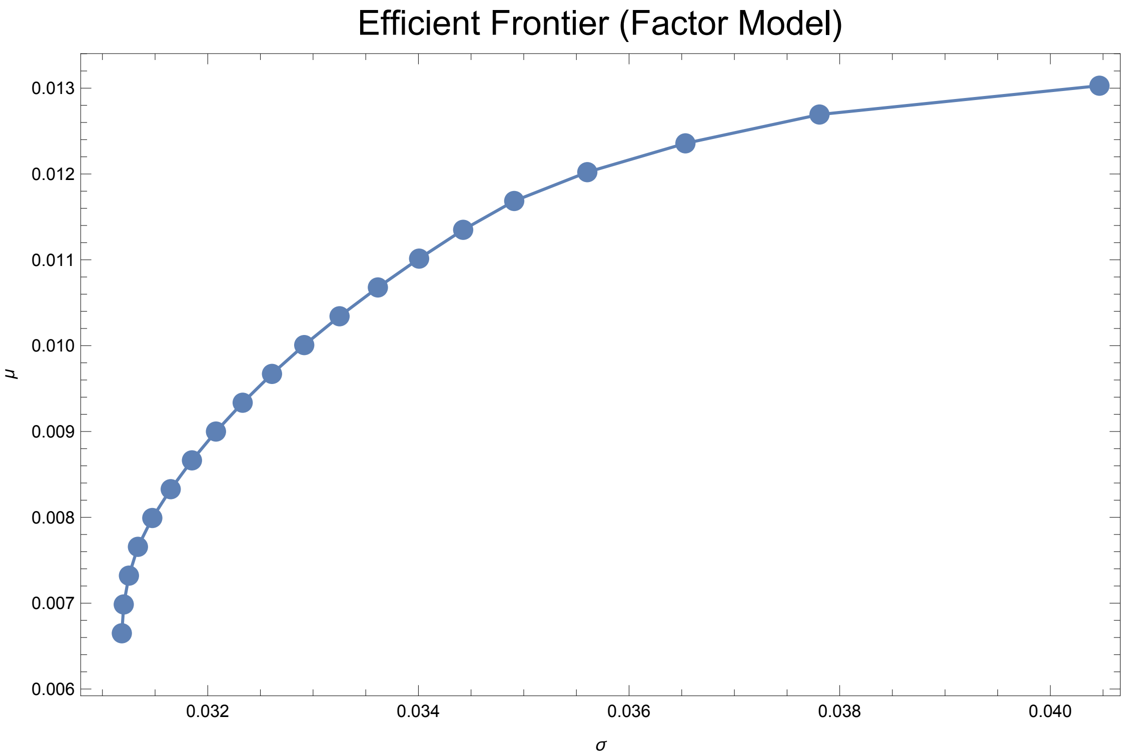
Out[∘]=



In[∘]:=
```
mnDenoisedEfficientFrontierF =
  {√(#.mnDenoisedModelCov.#), #.vnMean} & /@ mnDenoisedEfficientPortfoliosF;
```

In[•]:=
```
ListLinePlot[MapThread[Tooltip,
  {mnDenoisedEfficientFrontierF, mnDenoisedEfficientPortfoliosF}],
 Mesh → All, PlotRange → All, Frame → True, FrameLabel → {{"μ", ""}, {"σ",
    Style["Efficient Frontier (Factor Model)", FontSize → 18]}}, ImageSize → 600]
```

Out[•]=



In[•]:=
```
iDenoisedTangentF = First@First@Position[#, Max[#]] &[
  #.vnMean / √(#.mnDenoisedModelCov.#) & /@ mnDenoisedEfficientPortfoliosF]
```

Out[•]= 18

In[∘]:=
```
TableForm[{Chop[mnDenoisedEfficientPortfoliosF[[iDenoisedTangentF]], 10^-4]}^T,
 TableHeadings → {vsNames, {"Tangent Portfolio"}}]
```

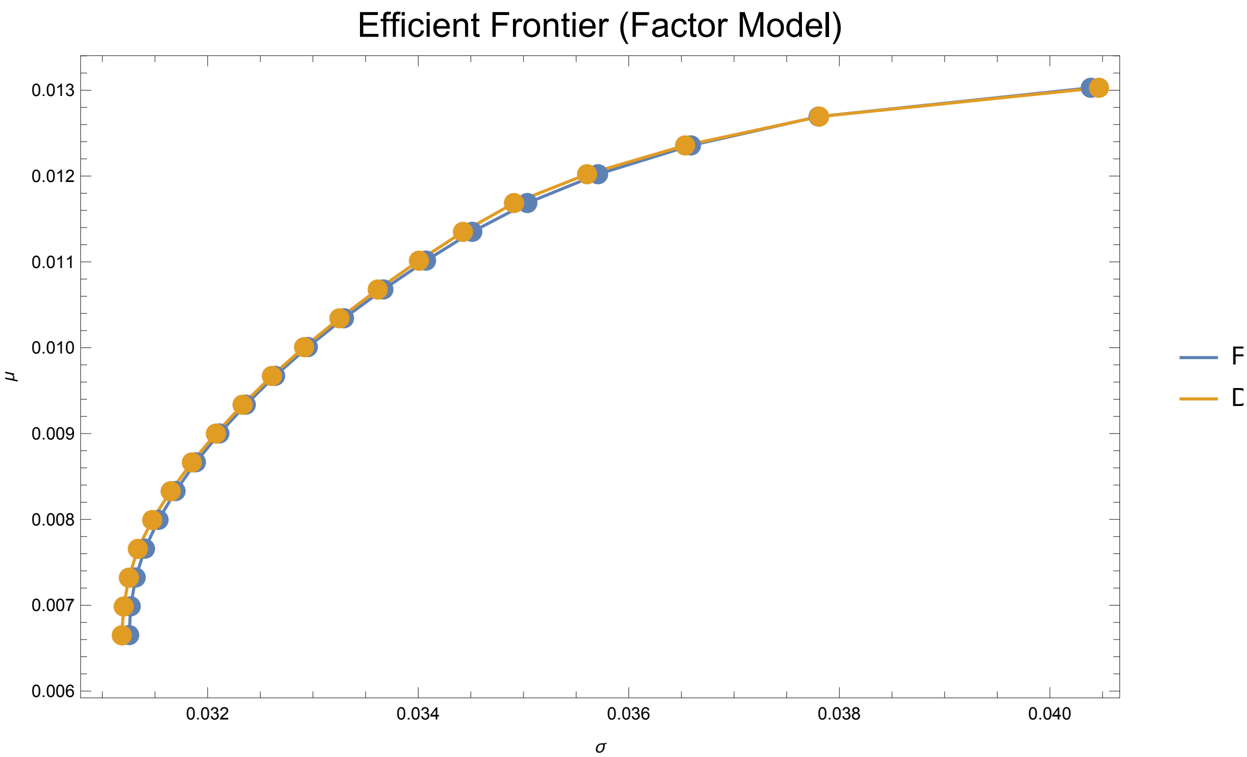
Out[∘]//TableForm=

	Tangent Portfolio
3M Co	0.01
American Express Co	0.01
Apple Inc	0.1
Boeing Co	0.01
Caterpillar Inc	0.01
Chevron Corp	0.01
Cisco Systems Inc	0.01
Coca-Cola Co	0.01
The Walt Disney Co	0.01
Missing[NotAvailable]	0.01
Exxon Mobil Corp	0.01
Goldman Sachs Group Inc	0.01
The Home Depot Inc	0.075989
Intel Corp	0.01
International Business Machines Corp	0.01
Johnson & Johnson	0.01
JPMorgan Chase & Co	0.01
McDonald's Corp	0.1
Merck & Co Inc	0.01
Microsoft Corp	0.0693753
Nike Inc	0.1
Pfizer Inc	0.01
Procter & Gamble Co	0.0446359
The Travelers Companies Inc	0.01
UnitedHealth Group Inc	0.1
United Technologies Corp	0.01
Verizon Communications Inc	0.01
Visa Inc	0.1
Walgreens Boots Alliance Inc	0.01
Walmart Inc	0.1

Comparison of Efficient Frontier with and without Denoising Correlation

In[∘]:=
```
ListLinePlot[
 {MapThread[Tooltip, {mnEfficientFrontierF, mnEfficientPortfoliosF}], MapThread[
   Tooltip, {mnDenoisedEfficientFrontierF, mnDenoisedEfficientPortfoliosF}]},
 Mesh → All, PlotRange → All, Frame → True, FrameLabel →
  {{"μ", ""}, {"σ", Style["Efficient Frontier (Factor Model)", FontSize → 18]}},
 PlotLegends → {"Factor Model", "Denoised Factor Model"}, ImageSize → 600]
```

Out[∘]=



In[∘]:=
```
TableForm[
 {Chop[mnEfficientPortfoliosF[[1]], 10^-4], Chop[mnDenoisedEfficientPortfoliosF[[1]],
    10^-4], Chop[mnEfficientPortfoliosF[[1]], 10^-4] -
    Chop[mnDenoisedEfficientPortfoliosF[[1]], 10^-4]}^T,
 TableHeadings → {vsNames, {"Min Var", "Denoised Min Var", "Difference"}}]
```

Out[∘]//TableForm=

	Min Var	Denoised Min Var	Difference
3M Co	0.01	0.01	0.
American Express Co	0.01	0.01	0.
Apple Inc	0.01	0.01	0.
Boeing Co	0.01	0.01	0.
Caterpillar Inc	0.01	0.01	0.
Chevron Corp	0.01	0.01	0.
Cisco Systems Inc	0.01	0.01	0.
Coca-Cola Co	0.1	0.1	0.
The Walt Disney Co	0.01	0.01	0.
Missing[NotAvailable]	0.01	0.01	0.
Exxon Mobil Corp	0.0153761	0.01	0.0053761
Goldman Sachs Group Inc	0.01	0.01	0.
The Home Depot Inc	0.0392513	0.0401173	-0.00086597
Intel Corp	0.01	0.01	0.
International Business Machines Corp	0.0517889	0.0491647	0.00262415
Johnson & Johnson	0.1	0.1	0.
JPMorgan Chase & Co	0.01	0.01	0.
McDonald's Corp	0.1	0.1	0.
Merck & Co Inc	0.0290125	0.0280251	0.000987402
Microsoft Corp	0.01	0.01	0.
Nike Inc	0.0273906	0.0365716	-0.00918103
Pfizer Inc	0.0287718	0.0320829	-0.00331109
Procter & Gamble Co	0.1	0.1	0.
The Travelers Companies Inc	0.01	0.01	0.
UnitedHealth Group Inc	0.0285942	0.0340385	-0.00544434
United Technologies Corp	0.01	0.01	0.
Verizon Communications Inc	0.1	0.1	0.
Visa Inc	0.0198148	0.01	0.00981476
Walgreens Boots Alliance Inc	0.01	0.01	0.
Walmart Inc	0.1	0.1	0.

In[∘]:=
```
TableForm[{Chop[mnEfficientPortfoliosF[[iTangentF]], 10^-4],
   Chop[mnDenoisedEfficientPortfoliosF[[iDenoisedTangentF]], 10^-4],
   Chop[mnEfficientPortfoliosF[[iTangentF]], 10^-4] -
    Chop[mnDenoisedEfficientPortfoliosF[[iDenoisedTangentF]], 10^-4]}ᵀ,
 TableHeadings → {vsNames, {"Tangent", "Denoised Tangent", "Difference"}}]
```

Out[∘]//TableForm=

	Tangent	Denoised Tangent	Difference
3M Co	0.01	0.01	0.
American Express Co	0.01	0.01	0.
Apple Inc	0.1	0.1	0.
Boeing Co	0.01	0.01	0.
Caterpillar Inc	0.01	0.01	0.
Chevron Corp	0.01	0.01	0.
Cisco Systems Inc	0.01	0.01	0.
Coca-Cola Co	0.01	0.01	0.
The Walt Disney Co	0.01	0.01	0.
Missing[NotAvailable]	0.01	0.01	0.
Exxon Mobil Corp	0.01	0.01	0.
Goldman Sachs Group Inc	0.01	0.01	0.
The Home Depot Inc	0.0750232	0.075989	-0.00096587
Intel Corp	0.01	0.01	0.
International Business Machines Corp	0.01	0.01	0.
Johnson & Johnson	0.01	0.01	0.
JPMorgan Chase & Co	0.01	0.01	0.
McDonald's Corp	0.1	0.1	0.
Merck & Co Inc	0.01	0.01	0.
Microsoft Corp	0.0702856	0.0693753	0.0009103
Nike Inc	0.1	0.1	0.
Pfizer Inc	0.01	0.01	0.
Procter & Gamble Co	0.0446914	0.0446359	0.000055576
The Travelers Companies Inc	0.01	0.01	0.
UnitedHealth Group Inc	0.1	0.1	0.
United Technologies Corp	0.01	0.01	0.
Verizon Communications Inc	0.01	0.01	0.
Visa Inc	0.1	0.1	0.
Walgreens Boots Alliance Inc	0.01	0.01	0.
Walmart Inc	0.1	0.1	0.

In[∘]:=
```
TableForm[
 {Chop[mnEfficientPortfoliosF[[-1]], 10^-4], Chop[mnDenoisedEfficientPortfoliosF[[-1]],
    10^-4], Chop[mnEfficientPortfoliosF[[-1]], 10^-4] -
    Chop[mnDenoisedEfficientPortfoliosF[[-1]], 10^-4]}^T,
 TableHeadings → {vsNames, {"Min Var", "Denoised Min Var", "Difference"}}]
```

Out[∘]//TableForm=

	Min Var	Denoised Min Var	Difference
3M Co	0.01	0.01	0.
American Express Co	0.01	0.01	0.
Apple Inc	0.1	0.1	0.
Boeing Co	0.0800001	0.0800001	-1.38778 × 1(
Caterpillar Inc	0.01	0.01	0.
Chevron Corp	0.01	0.01	0.
Cisco Systems Inc	0.01	0.01	0.
Coca-Cola Co	0.01	0.01	0.
The Walt Disney Co	0.01	0.01	0.
Missing[NotAvailable]	0.01	0.01	0.
Exxon Mobil Corp	0.01	0.01	0.
Goldman Sachs Group Inc	0.01	0.01	0.
The Home Depot Inc	0.1	0.1	0.
Intel Corp	0.01	0.01	0.
International Business Machines Corp	0.01	0.01	0.
Johnson & Johnson	0.01	0.01	0.
JPMorgan Chase & Co	0.01	0.01	0.
McDonald's Corp	0.1	0.1	0.
Merck & Co Inc	0.01	0.01	0.
Microsoft Corp	0.1	0.1	0.
Nike Inc	0.1	0.1	0.
Pfizer Inc	0.01	0.01	0.
Procter & Gamble Co	0.01	0.01	0.
The Travelers Companies Inc	0.01	0.01	0.
UnitedHealth Group Inc	0.1	0.1	0.
United Technologies Corp	0.01	0.01	0.
Verizon Communications Inc	0.01	0.01	0.
Visa Inc	0.1	0.1	0.
Walgreens Boots Alliance Inc	0.01	0.01	0.
Walmart Inc	0.01	0.01	0.