AMS-512 Capital Markets and Portfolio Theory

Portfolio Optimization with Missing Data 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

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

Calendar Functions

```
In[•]:= xEndOfMonth[{y_, m_, d_}] := If[m == 12, {y, m, 31}, First@DayPlus[{y, m + 1, 1}, -1]];
```

```
In[•]:= xExpandCalendar[vxCalendar_, vxSeries_] := Sort[
          Join[
           vxSeries,
           Transpose[{#, Array[Missing[] &, Length[#]]}] &[
            Complement[vxCalendar, First /@ vxSeries]
           ]
          ]
         ];
```

Candidate Dataset

```
In[•]:= vsTickers = FinancialData["^DJI", "Members"]
        vsNames = FinancialData[#, "Name"] & /@ vsTickers;
        TableForm[{vsTickers, vsNames}^T, TableHeadings → {Range[Length[vsTickers]]}]
```

```
Out[•]= {NYSE:MMM, NYSE:AXP, NASDAQ:AAPL, NYSE:BA, NYSE:CAT, NYSE:CVX, NASDAQ:CSCO, NYSE:KO,
         NYSE:DIS, NYSE:DWDP, NYSE:XOM, NYSE:GS, NYSE:HD, NASDAQ:INTC, NYSE:IBM,
         NYSE:JNJ, NYSE:JPM, NYSE:MCD, NYSE:MRK, NASDAQ:MSFT, NYSE:NKE, NYSE:PFE,
         NYSE:PG, NYSE:TRV, NYSE:UNH, NYSE:UTX, NYSE:VZ, NYSE:V, NASDAQ:WBA, NYSE:WMT}
```

Out[•]//TableForm=

1	NYSE:MMM	3M Co
2	NYSE:AXP	American Express Co
3	NASDAQ:AAPL	Apple Inc
4	NYSE:BA	Boeing Co
5	NYSE:CAT	Caterpillar Inc
6	NYSE:CVX	Chevron Corp
7	NASDAQ:CSCO	Cisco Systems Inc
8	NYSE:KO	Coca-Cola Co
9	NYSE:DIS	The Walt Disney Co
10	NYSE:DWDP	Missing[NotAvailable]
11	NYSE:XOM	Exxon Mobil Corp
12	NYSE:GS	Goldman Sachs Group Inc
13	NYSE:HD	The Home Depot Inc
14	NASDAQ:INTC	Intel Corp
15	NYSE:IBM	International Business Machines Corp
16	NYSE:JNJ	Johnson & Johnson
17	NYSE:JPM	JPMorgan Chase & Co
18	NYSE:MCD	McDonald's Corp
19	NYSE:MRK	Merck & Co Inc
20	NASDAQ:MSFT	Microsoft Corp
21	NYSE:NKE	Nike Inc
22	NYSE:PFE	Pfizer Inc
23	NYSE:PG	Procter & Gamble Co
24	NYSE:TRV	The Travelers Companies Inc
25	NYSE:UNH	UnitedHealth Group Inc
26	NYSE:UTX	United Technologies Corp
27	NYSE:VZ	Verizon Communications Inc
28	NYSE:V	Visa Inc
29	NASDAQ:WBA	Walgreens Boots Alliance Inc
30	NYSE:WMT	Walmart Inc

```
In[•]:= mnDateRange = Join[{{2003, 12, 15}, {2020, 03, 31}}]
```

```
Out[•]= {{2003, 12, 15}, {2020, 3, 31}}
```

```
In[•]:= buffer = (Most[Last /@ Split[#, (#1[[1, 2]] == #2[[1, 2]]) &]]) & /@
            ((FinancialData[#, "Close", mnDateRange, Method → "Legacy"] &) /@ vsTickers);
```

```
In[•]:= buffer = FinancialData[#, "Close", mnDateRange, Method → "Legacy"] & /@ vsTickers;
```

```
In[•]:= Dimensions /@ buffer
```

```
Out[•]= {{4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2},
  {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4101, 2}, {4101, 2},
  {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2},
  {4102, 2}, {4102, 2}, {4102, 2}, {4102, 2}, {3031, 2}, {4097, 2}, {4102, 2}}
```

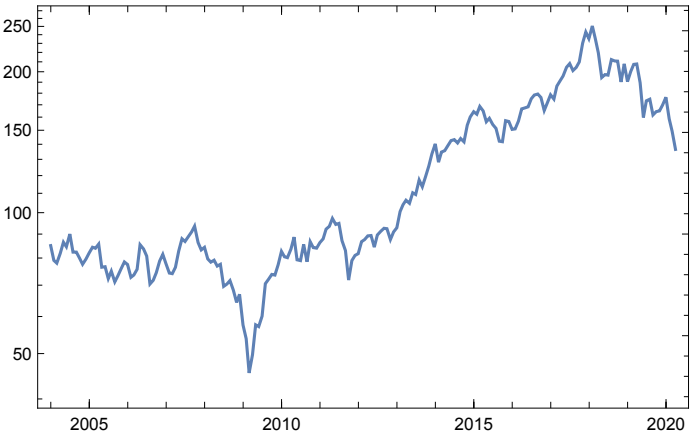
```
In[•]:= And @@ (FreeQ[#, _Missing] & /@ buffer)
```

```
Out[•]= True
```

```
In[•]:= monthly = (Last /@ Split[#, (#1[[1, 2]] == #2[[1, 2]]) &]) & /@ buffer;
```

```
In[•]:= DateListLogPlot[monthly[[1]]]
```

```
Out[•]=
```




```
In[•]:= vmxReturns = {xEndOfMonth /@ Rest[First /@ #], (Rest[#]/Most[#] - 1 &)[Last /@ #]}^T & /@ monthly;
```

```
In[•]:= Dimensions /@ vmxReturns
```

```
Out[•]= {{195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2},
  {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2},
  {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2},
  {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {144, 2}, {195, 2}, {195, 2}}
```

```
In[•]:= vxCalendar = Union[Flatten[#[[All, 1]] & /@ vmxReturns, 1]];
```

```
In[•]:= Length@vxCalendar
```

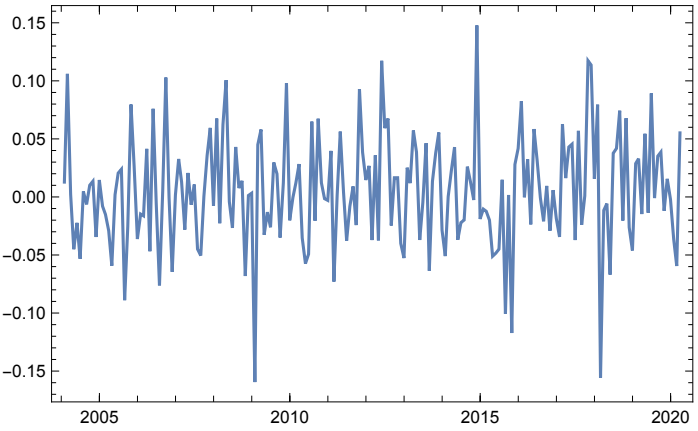
```
Out[•]= 195
```

In[•]:=
```
Short[vxCalendar, 10]
```

Out[•]//Short=
```
{{2004, 1, 31}, {2004, 2, 29}, {2004, 3, 31}, {2004, 4, 30},
 {2004, 5, 31}, {2004, 6, 30}, {2004, 7, 31}, {2004, 8, 31}, {2004, 9, 30},
 {2004, 10, 31}, {2004, 11, 30}, {2004, 12, 31}, {2005, 1, 31},
 {2005, 2, 28}, {2005, 3, 31}, {2005, 4, 30}, {2005, 5, 31}, {2005, 6, 30},
 {2005, 7, 31}, {2005, 8, 31}, {2005, 9, 30}, {2005, 10, 31}, {2005, 11, 30},
 <<149>>, {2018, 5, 31}, {2018, 6, 30}, {2018, 7, 31}, {2018, 8, 31},
 {2018, 9, 30}, {2018, 10, 31}, {2018, 11, 30}, {2018, 12, 31},
 {2019, 1, 31}, {2019, 2, 28}, {2019, 3, 31}, {2019, 4, 30}, {2019, 5, 31},
 {2019, 6, 30}, {2019, 7, 31}, {2019, 8, 31}, {2019, 9, 30}, {2019, 10, 31},
 {2019, 11, 30}, {2019, 12, 31}, {2020, 1, 31}, {2020, 2, 29}, {2020, 3, 31}}
```

In[•]:=
```
DateListPlot[vmxReturns[[30]]]
```

Out[•]=



In[•]:=
```
vmxRepaired = xExpandCalendar[vxCalendar, #] & /@ vmxReturns;
```

In[•]:=
```
Dimensions /@ vmxRepaired
```

Out[•]=
```
{{195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2},
 {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2},
 {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2},
 {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}, {195, 2}}
```

In[•]:=
```
Length[vmxRepaired]
```

Out[•]=
```
30
```

In[•]:=
```
dbReturns = {
    vxCalendar,
    vsTickers,
    Transpose[#[[All, 2]] & /@ vmxRepaired]
   };
```

In[•]:=
```
Dimensions /@ dbReturns
```

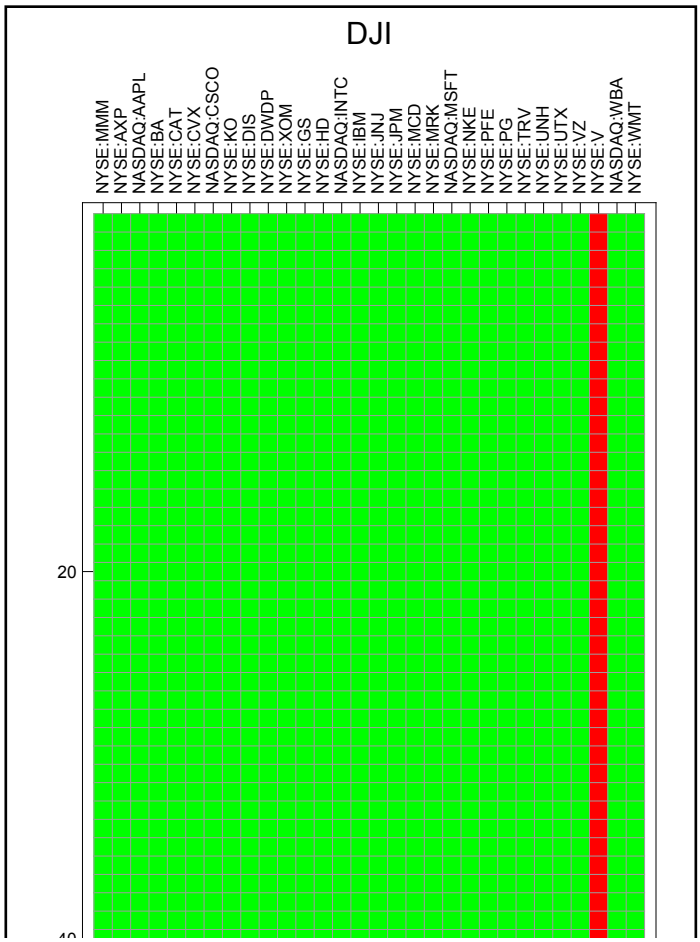
Out[•]=
```
{{195, 3}, {30}, {195, 30}}
```

```
In[•]:= Export[FileNameJoin[{NotebookDirectory[], "dbReturns.m"}], dbReturns]
Out[•]= /Users/robertjfrey/Documents/Work/Stony Brook
        University/AMS/QF/public_html/Instruction/Spring2020/AMS512/Class06/dbReturns.m
```

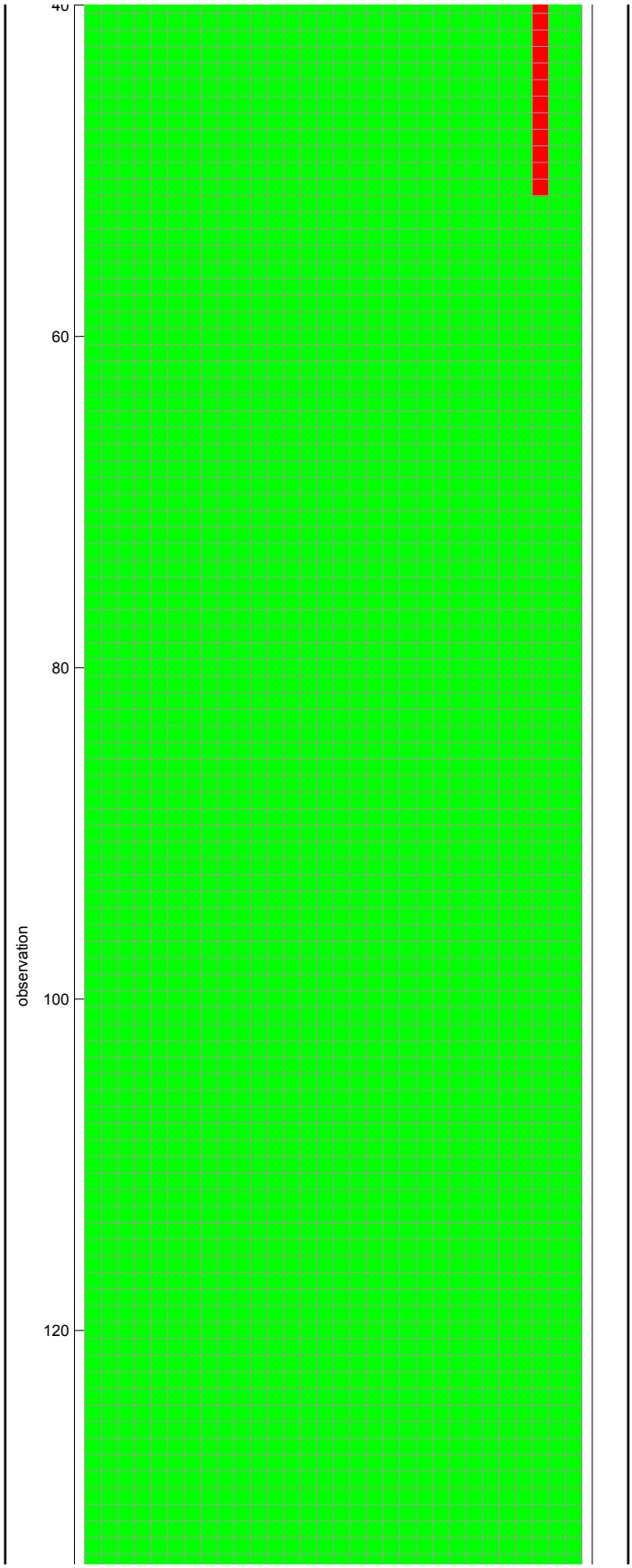
Missing Data Analysis

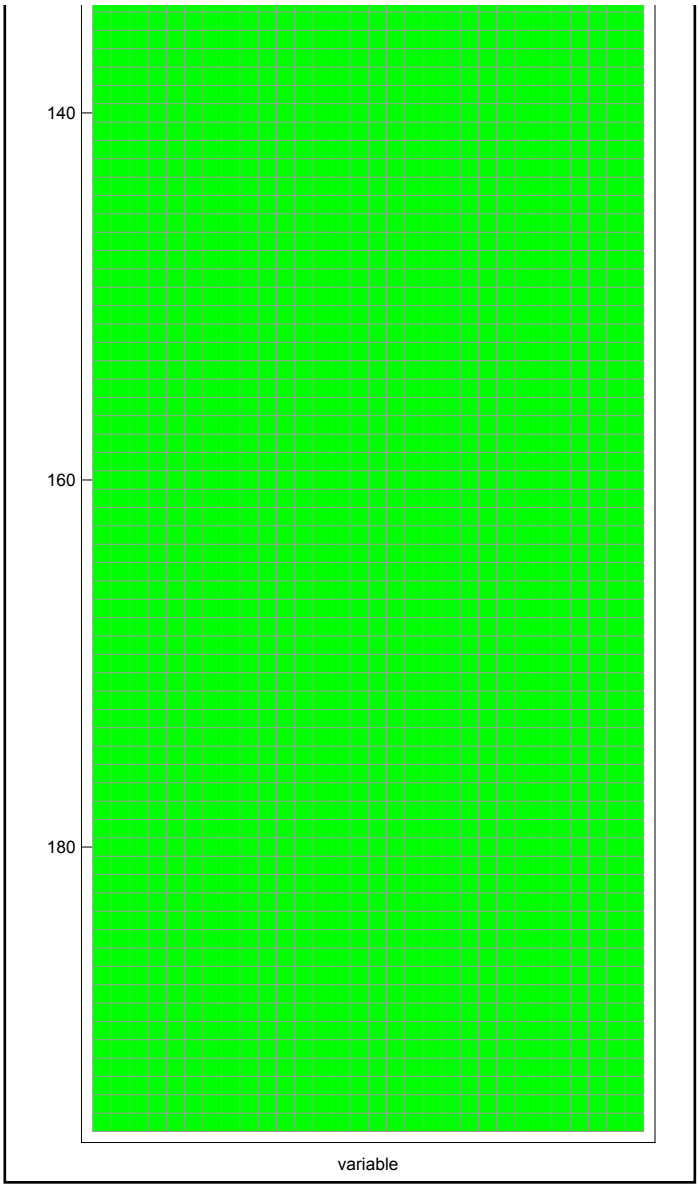
Missing Data Pattern

```
In[•]:= Framed@ArrayPlot[
          Map[Boole@FreeQ[#, _Missing] &, dbReturns[[3]], {2}],
          ImageSize → 350,
          Mesh -> True,
          ColorFunction → (If[# == 1, Green, Red] &),
          Frame → True ,
          FrameLabel → {{"variable", Style["DJI", FontSize → 16]}, {"observation", ""}},
          FrameTicks → {{Rest@Range[0, Length[dbReturns[[1]]], 20], None},
            {None, {Range[Length[#]], #}ᵀ &[Rotate[#, 90 Degree] & /@ dbReturns[[2]]]}}
        ]
```



Out[•]=





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

Mean and Covariance Estimation

In[]:=
```
? xMeanCovMissingMLE
```

Out[]=

Symbol

{{Mean, Cov}, {Completed, Cross}, LLHistory, RunID} =
    xMeanCovMissingMLE[dbReferenceReturns, opts] – MLE estimate of mean and cov with missing data.
  Uses XEM–algorithm (extrapolation, expectation, minimization).

dbReferenceReturns – db of reference returns; missing values indicated by Missing[].
Option names are strings:
  "InitialEstimate" – "Diagonal" (default), "CompleteObs", and "ImputedMean" or {Mean, Cov}.
    If a string, then defines how the intial estimate is produced;
    otherwise, intial starting values for the mean vector and covearniance matrix.
  "ToleranceGoal" – Rate of change in log likelihood, criterion for EM termination, default 10^–6.
  "MaxIterations" – Max number of iterations, criterion for EM termination, default 100.
  "Extrapolation" – True to use the X–Step (default); False to use standard EM algorithm.
  "CheckPoint" – Frequency at which state sufficient for restart saved; 0 (default) for no check–pointing.
  "Verbose" – True to print info on each iteration, default False.

Mean – Mean vector
Cov – Covariance matrix
Completed – Data with missing values replaced by expectations based on regression estimates.
Cross – Matrix for second order correction of cross products involving missing data.
LLHistory – History of log likelihood at each EM iteration.
RunID – Unique string representing the environment and time stamp of the run.

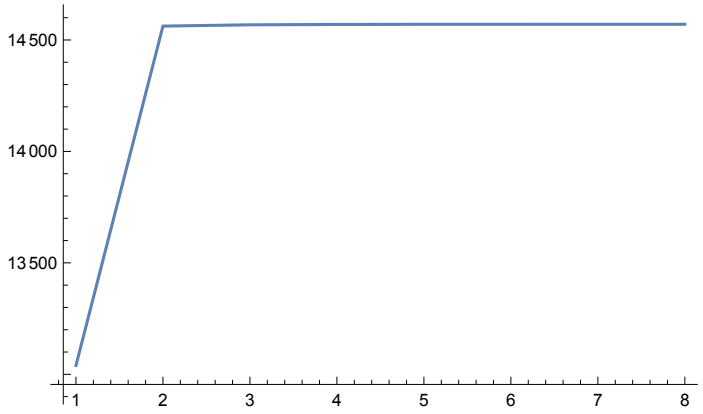
In[]:=
```
{{vnMean, mnCovariance}, {mnCompleted, mnCross}, vnLogLik, sRunID} =
  xMeanCovMissingMLE[dbReturns];
```

In[]:=
```
sRunID
```

Out[]=
```
robertjfrey_MacOSX-x86-64_0_20200331_20200412190222
```

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

Out[•]=




```
Mean[Last /@ #] & /@ vmxReturns // Length
```

30

```
Length@vnMean
```

30

```
In[•]:= TableForm[{vsTickers, vsNames, vnMean, Mean[Last /@ #] & /@ vmxReturns}^T,
          TableHeadings → {Automatic, {"Tickers", "Names", "Imputed μ", "Observed μ"}}
        ]
```

Out[•]//TableForm=

	Tickers	Names	Imputed μ	Observed μ
1	NYSE:MMM	3M Co	0.00401453	0.0040145
2	NYSE:AXP	American Express Co	0.0074671	0.0074671
3	NASDAQ:AAPL	Apple Inc	0.0311532	0.0311532
4	NYSE:BA	Boeing Co	0.0100985	0.0100985
5	NYSE:CAT	Caterpillar Inc	0.00934243	0.0093424
6	NYSE:CVX	Chevron Corp	0.00441164	0.0044116
7	NASDAQ:CSCO	Cisco Systems Inc	0.00520701	0.0052070
8	NYSE:KO	Coca-Cola Co	0.00385838	0.0038583
9	NYSE:DIS	The Walt Disney Co	0.00946225	0.0094622
10	NYSE:DWDP	Missing[NotAvailable]	0.00761465	0.0076146
11	NYSE:XOM	Exxon Mobil Corp	0.0012095	0.0012095
12	NYSE:GS	Goldman Sachs Group Inc	0.00593126	0.0059312
13	NYSE:HD	The Home Depot Inc	0.010385	0.010385
14	NASDAQ:INTC	Intel Corp	0.00518134	0.0051813
15	NYSE:IBM	International Business Machines Corp	0.0026758	0.0026758
16	NYSE:JNJ	Johnson & Johnson	0.00560399	0.0056039
17	NYSE:JPM	JPMorgan Chase & Co	0.00767116	0.0076711
18	NYSE:MCD	McDonald's Corp	0.0107814	0.0107814
19	NYSE:MRK	Merck & Co Inc	0.00469473	0.0046947
20	NASDAQ:MSFT	Microsoft Corp	0.0110627	0.0110627
21	NYSE:NKE	Nike Inc	0.0136097	0.0136097
22	NYSE:PFE	Pfizer Inc	0.00101316	0.0010131
23	NYSE:PG	Procter & Gamble Co	0.00494258	0.0049425
24	NYSE:TRV	The Travelers Companies Inc	0.00610687	0.0061068
25	NYSE:UNH	UnitedHealth Group Inc	0.0136886	0.0136886
26	NYSE:UTX	United Technologies Corp	0.00526605	0.0052660
27	NYSE:VZ	Verizon Communications Inc	0.00400314	0.0040031
28	NYSE:V	Visa Inc	0.0200631	0.0183666
29	NASDAQ:WBA	Walgreens Boots Alliance Inc	0.0039116	0.0039116
30	NYSE:WMT	Walmart Inc	0.00502602	0.0050260

```
In[•]:= TableForm[{vsTickers, vsNames, √Diagonal[mnCovariance],
          StandardDeviation[Last /@ #] & /@ vmxReturns}^T,
        TableHeadings → {None, {"Tickers", "Names", "Imputed σ", "Observed σ"}}
        ]
```

Out[•]//TableForm=

Tickers	Names	Imputed σ	Observed σ
NYSE:MMM	3M Co	0.0559226	0.0560666
NYSE:AXP	American Express Co	0.093359	0.0935993
NASDAQ:AAPL	Apple Inc	0.0954291	0.0956747
NYSE:BA	Boeing Co	0.0813718	0.0815812
NYSE:CAT	Caterpillar Inc	0.0890496	0.0892788
NYSE:CVX	Chevron Corp	0.0593854	0.0595383
NASDAQ:CSCO	Cisco Systems Inc	0.0736708	0.0738604
NYSE:KO	Coca-Cola Co	0.0443141	0.0444282
NYSE:DIS	The Walt Disney Co	0.0640501	0.0642149
NYSE:DWDP	Missing[NotAvailable]	0.152735	0.153128
NYSE:XOM	Exxon Mobil Corp	0.0558443	0.0559881
NYSE:GS	Goldman Sachs Group Inc	0.0851371	0.0853562
NYSE:HD	The Home Depot Inc	0.0604535	0.0606091
NASDAQ:INTC	Intel Corp	0.0703019	0.0704829
NYSE:IBM	International Business Machines Corp	0.0584531	0.0586035
NYSE:JNJ	Johnson & Johnson	0.0402448	0.0403484
NYSE:JPM	JPMorgan Chase & Co	0.0775206	0.0777201
NYSE:MCD	McDonald's Corp	0.0449545	0.0450703
NYSE:MRK	Merck & Co Inc	0.0635851	0.0637488
NASDAQ:MSFT	Microsoft Corp	0.0642051	0.0643704
NYSE:NKE	Nike Inc	0.0617527	0.0619117
NYSE:PFE	Pfizer Inc	0.0530175	0.0531539
NYSE:PG	Procter & Gamble Co	0.0420105	0.0421186
NYSE:TRV	The Travelers Companies Inc	0.0526409	0.0527764
NYSE:UNH	UnitedHealth Group Inc	0.0707714	0.0709535
NYSE:UTX	United Technologies Corp	0.0577773	0.057926
NYSE:VZ	Verizon Communications Inc	0.0500149	0.0501436
NYSE:V	Visa Inc	0.0651439	0.064385
NASDAQ:WBA	Walgreens Boots Alliance Inc	0.0741298	0.0743206
NYSE:WMT	Walmart Inc	0.0473991	0.0475211

Factor Model

In[]:=
```
? xFactorFitMLE
```

Out[]=

Symbol

{{FactorLoadings, ErrorMatrix}, LLHistory, BIC} = xFactorModelFitMLE[NumbObs,
CovMatrix, {InitLoadings, InitErrMatrix}, opts} – Fit an orthonormal factor model

NumbObs – Number of observations in dataset
CovMatrix – Covariance matrix
InitLoadings – Initialized factor loadings
InitErrMatrix – Initialized diagonal matrix of error variances

Option names are strings:
"ToleranceGoal" – Rate of change in log likelihood, criterion for EM termination (default 10^–8)
"MaxIterations" – Max number of iterations, criterion for EM termination (default 400)

FactorLoadings – Factor loading matrix
ErrorMatrix – Diagonal matrix of error variances
LLHistory – History of log likelihood at each EM iteration
BIC – Bayesian information criterion for the model

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

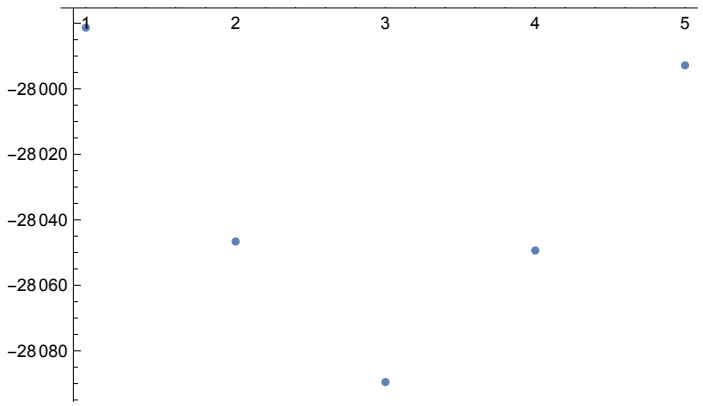
In[]:=
```
mnBIC = Table[{i, Last@xFactorFitMLE[Length[dbReturns[[1]]],
    mnCovariance, xInitializeFactorModel[mnCovariance, i]]}, {i, 1, 5}]
```

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[]=

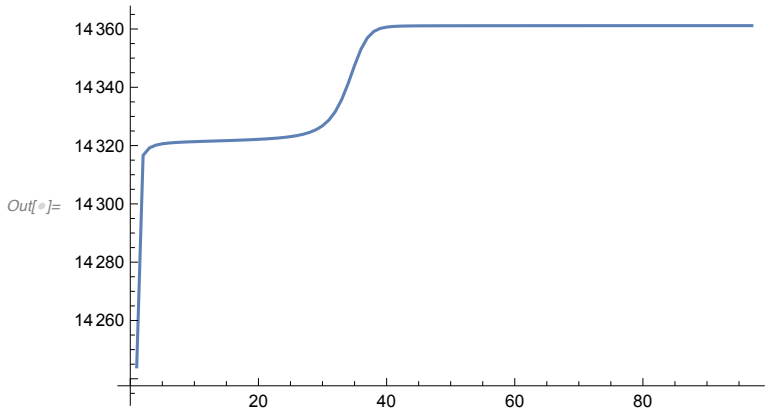


In[]:=
```
{{mnFactors, mnErrDiag}, vnLogLikHistory, nBIC} = xFactorFitMLE[
   Length[dbReturns[[1]]], mnCovariance, xInitializeFactorModel[mnCovariance, 3]];
```

Note how the log likelihood converges.

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

Out[•]=



Portfolio Optimization

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

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

```
Out[•]= 32.9343
```

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

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

```
Out[•]= 0.00665219
```

```
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

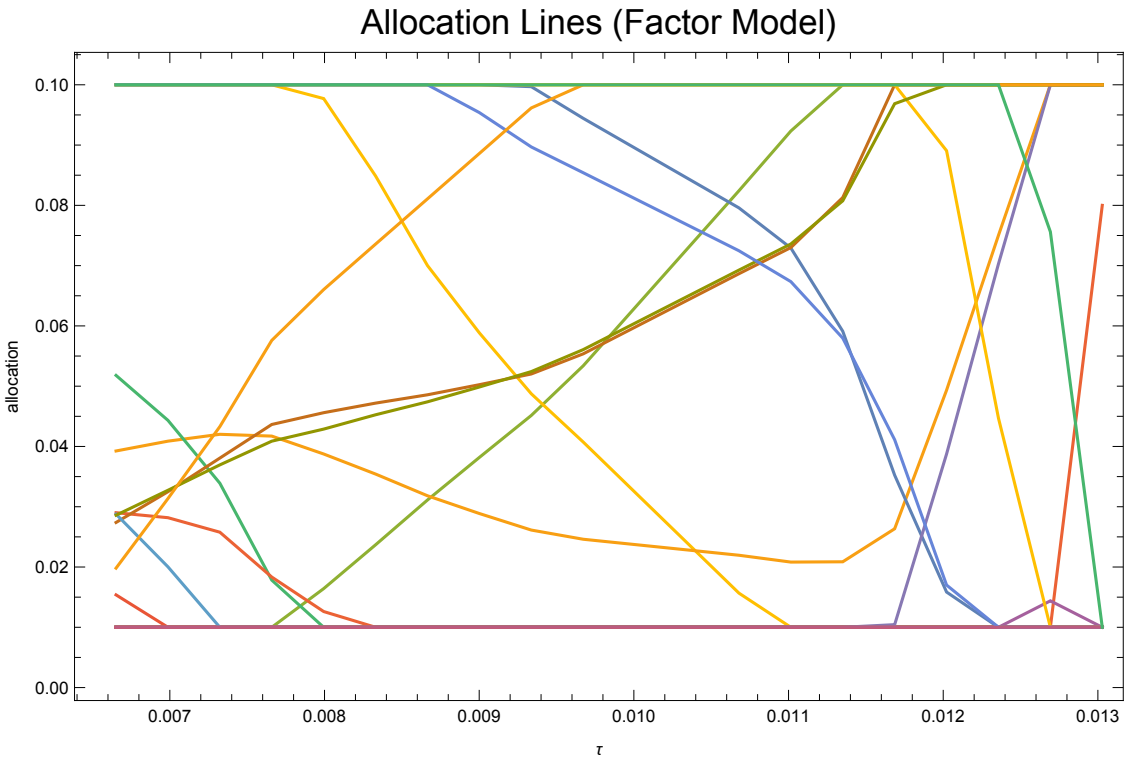
```
In[•]:= vnReturnTargetsF =
          Table[τ, {τ, nMinVarMeanF, nMaxRetMeanF, (nMaxRetMeanF - nMinVarMeanF) / 19}];

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

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

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

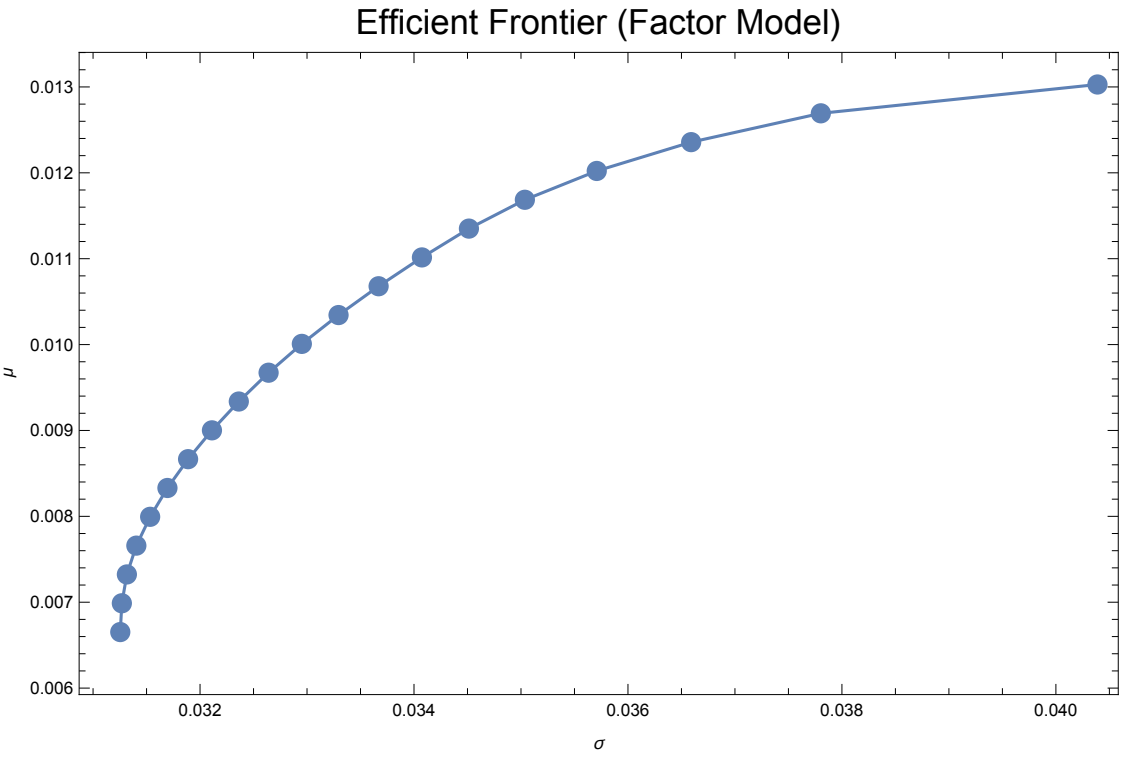
Out[•]=



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

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

Out[•]=



In[•]:=
```
iTangentF = First@First@Position[#, Max[#]] &[
  #.vnMean / Sqrt[#.mnModelCov.#] & /@ mnEfficientPortfoliosF]
```

Out[•]= 18

```
In[•]:= TableForm[{Chop[mnEfficientPortfoliosF[[iTangentF]], 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	0.0750232
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	0.0702856
Nike Inc	0.1
Pfizer Inc	0.01
Procter & Gamble Co	0.0446914
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

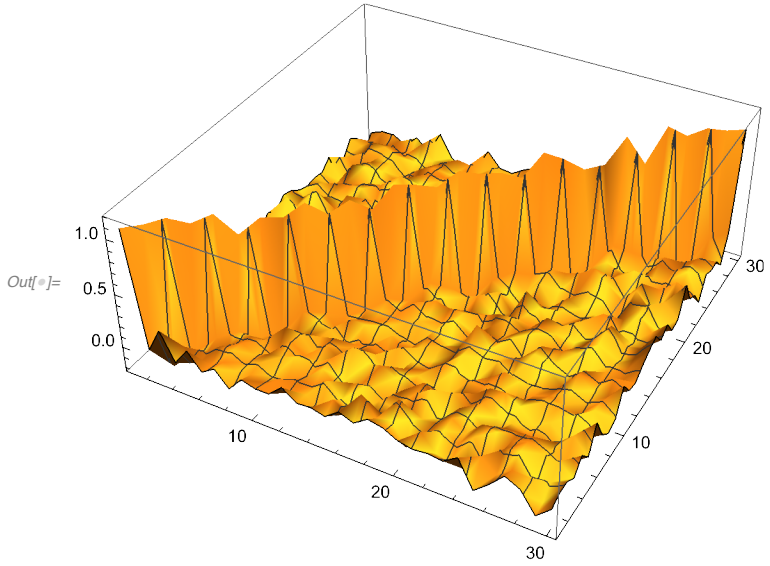
Denoising

```
In[•]:= sim =
          RandomVariate[MultinormalDistribution[Array[0. &, 30], IdentityMatrix[30]], 195];

In[•]:= simCov = Covariance[sim];
```

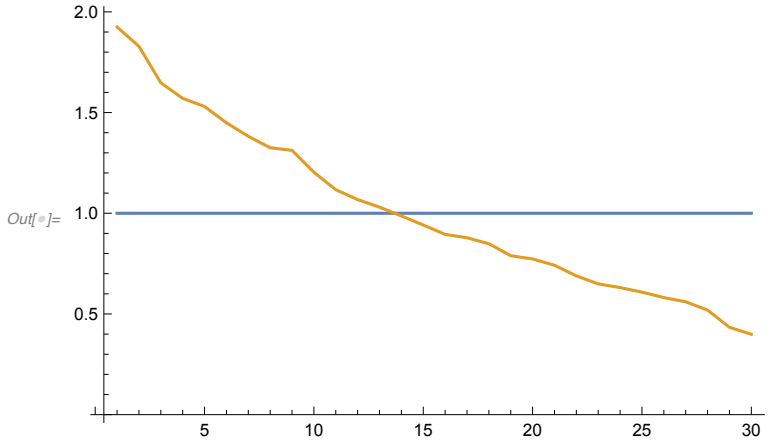
In[•]:=
```
ListPlot3D[simCov, PlotRange → All]
```

Out[•]=



In[•]:=
```
ListLinePlot[
 {Eigenvalues[IdentityMatrix[30]], Eigenvalues[simCov]}, PlotRange → All]
```

Out[•]=



Marchenko-Pastur Distribution

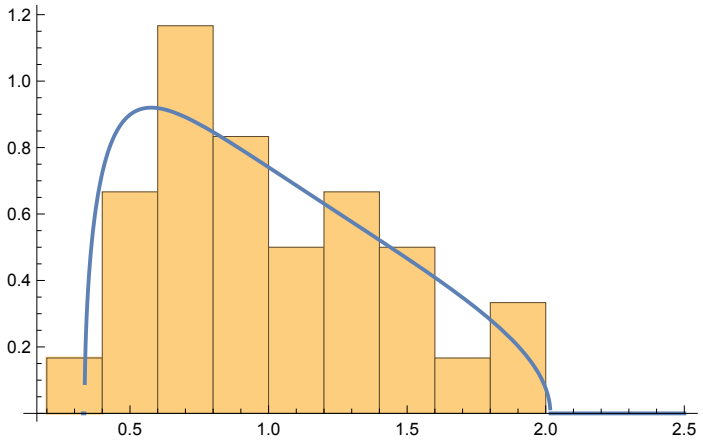
The Marchenko-Pastur distribution deals with the distribution of random matrices which follow a Wishart Distribution, *i.e.*, correlation and covariance matrices. (See https://en.wikipedia.org/wiki/Wishart_distribution and https : // en.wikipedia.org / wiki / Marchenko – Pastur_distribution.)

The parameter q is estimated by N/T where N is the dimension of the r.v. and T is the number of periods. The distribution is the limit at $N, T \to \infty$.

For the simulated data above the eigenvalues and Marchenko-Pastur distribution are plotted below

In[•]:=
```
Show[Histogram[Eigenvalues[simCov], 6, PDF, PlotRange → All],
 Plot[PDF[MarchenkoPasturDistribution[30 / 170], x],
  {x, 0.33, 2.5}, PlotStyle → Thick, PlotRange → All], PlotRange → All]
```

Out[•]=



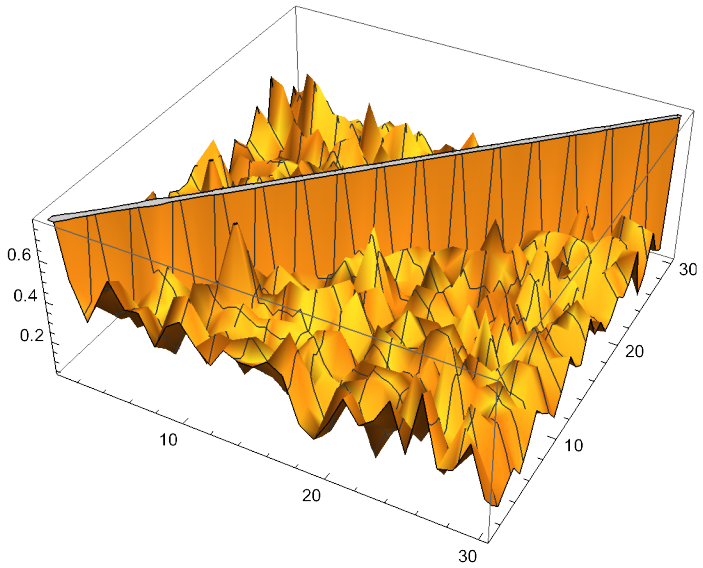
Denoising Using an MP-Threshold

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[#, #] &[Sqrt[Diagonal[mnCovariance]]]);
```

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

Out[•]=

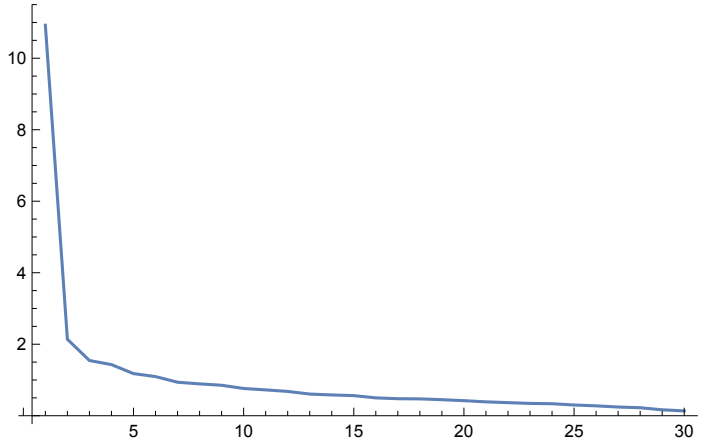


In[•]:=
```
vnEigen = Eigenvalues[mnCorrelation]
```

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}
```

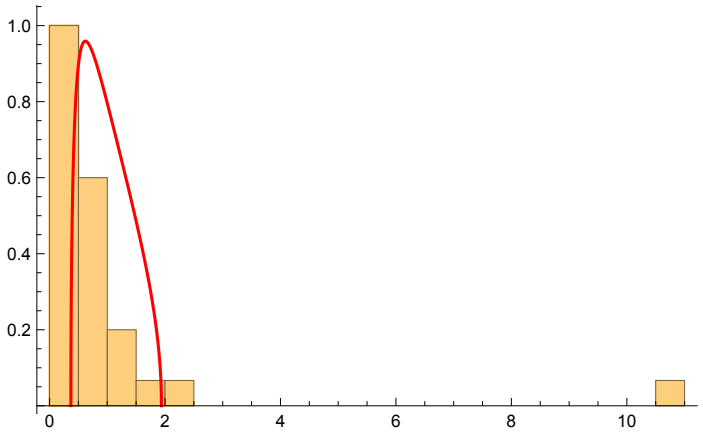
Experience has shown that at the upper edge of the distribution a "fuzz" of $N^{-2/3}$ can be added to adjust for the finite sample.

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}
```

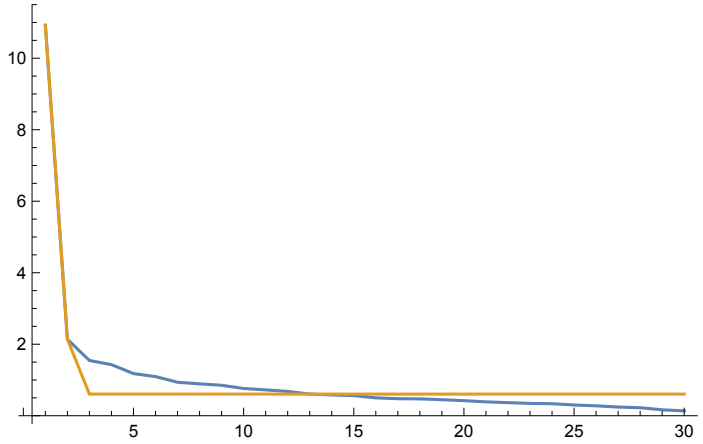
In[•]:=
```
Total[vnEigen]
Total[vnDenoisedEigen]
```

Out[•]=
```
30.
```

Out[•]=
```
30.
```

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

Out[•]=



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

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

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

Out[•]=
```
145.818
```

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

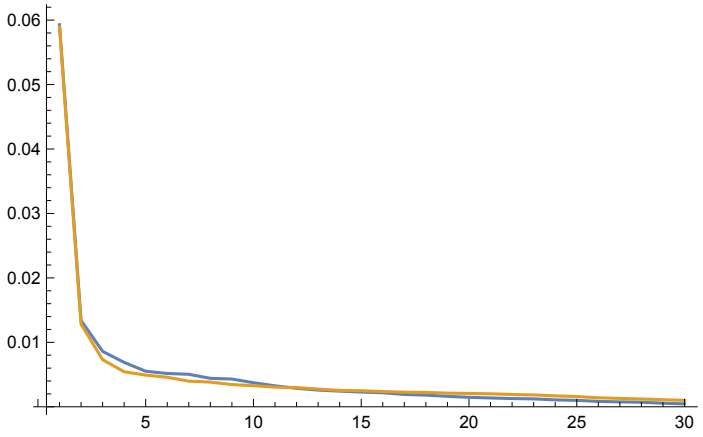
In[•]:=
```
mnDenoisedCov = mnDenoisedCor (KroneckerProduct[#, #] &[ √Diagonal[mnCovariance] ]);
```

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

Out[•]=
```
145.818
```

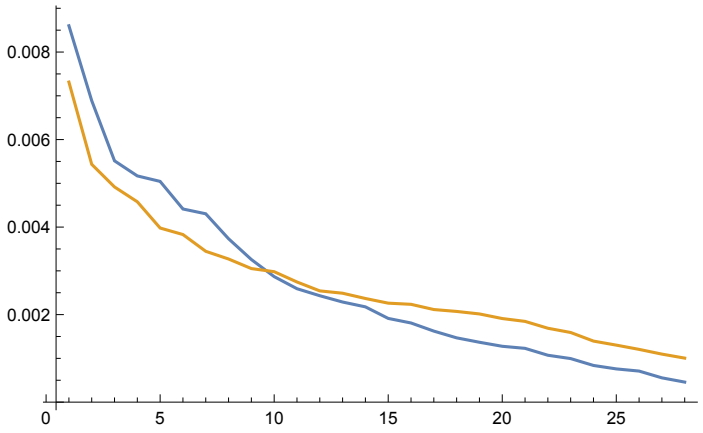
In[•]:=
```
ListLinePlot[
 {Eigenvalues[mnCovariance], Eigenvalues[mnDenoisedCov]}, PlotRange → All]
```

Out[•]=



In[•]:=
```
ListLinePlot[{Eigenvalues[mnCovariance][[3 ;;]],
  Eigenvalues[mnDenoisedCov][[3 ;;]]}, PlotRange → All]
```

Out[•]=



Homework

- Using the denoised covariance matrix above, repeat the sections “Factor Model” and “Portfolio Optimization” above using this result.
- Compare the efficient frontier based on the original estimates against those produced with the denoised estimate.