AMS-512 Capital Markets and Portfolio Theory

Mean-Variance Optimization

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

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

Set-Up & Code

In[139]:=
```
NotebookDirectory[]
```
Out[139]=
```
/Volumes/Files/Programming/Capital Markets and Portfolio Theory/1/
```

In[140]:=
```
FileNameJoin[{NotebookDirectory[], "QuadraticProgramming.m"}]
```
Out[140]=
```
/Volumes/Files/Programming/Capital
  Markets and Portfolio Theory/1/QuadraticProgramming.m
```

In[141]:=
```
Get[FileNameJoin[{NotebookDirectory[], "QuadraticProgramming.m"}]]
```

In[142]:=
```
xEfficientFrontier[mC_, vR_, {nMin_, nMax_, nIncr_}] := Module[
    {λ, mIC, mX, nC1, nC2, vC1, vC2, vU},
    mIC = Inverse[mC];
    vU = Table[1., {Length[vR]}];
    vC1 = mIC.vU;
    vC2 = mIC.vR;
    nC1 = vC1.vU;
    nC2 = vC2.vU;
    mX = Table[vX = λ vC2 + ((1 - λ nC2)/nC1) vC1, {λ, nMin, nMax, nIncr}];
    {{√(#.mC.#), vR.#} & /@ mX, mX}
  ];
```

Markowitz's Modern Portfolio Theory: Mean-Variance Portfolios

Modern Portfolio Theory or MPT is widely used as the basis for constructing portfolios. It assumes that asset returns can be modeled using a multivariate Normal distribution and

further assumes that risk is measured by a portfolio's variance (or equivalently standard deviation) of return and reward is measured by a portfolio's expected value of return .

Despite concerns about the reasonableness of these assumptions, MPT provides a framework for selecting portfolios based on a trade-off of risk and reward. MPT leads an investor to allocate capital based on the overall characteristics of the portfolio rather than allocating capital to individual assets on a case-by-case basis without considering how the performance of individual assets may interact.

See http://en.wikipedia.org/wiki/Modern_portfolio_theory.

The Markowitz Model

Portfolio Mean and Variance

Portfolio Mean -

$$\mu_P = \boldsymbol{\mu}^T \mathbf{x}$$

Portfolio Variance -

$$\sigma_P^2 = \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x}$$

Balancing Risk and Reward

Trade-off risk and reward, in what follows the set $\mathcal{S}$ is typically convex and frequently expressed as a system of linear constraints. The general form is the quadratic program:

$$\min \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} - \lambda \boldsymbol{\mu}^T \mathbf{x} \,\middle|\, \mathbf{x} \in \mathcal{S} \right\},\ 0 \le \lambda \le \infty$$

Minimum variance portfolio ($\lambda \to 0$):

$$\min \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} \,\middle|\, \mathbf{x} \in \mathcal{S} \right\}$$

Maximum return portfolio, which simplifies to a linear program ($\lambda \to \infty$):

$$\min \left\{ -\boldsymbol{\mu}^T \mathbf{x} \mid \mathbf{x} \in \mathcal{S} \right\}$$

Minimize risk at a target rate of return:

$$\min \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} \,\middle|\, \boldsymbol{\mu}^T \mathbf{x} \ge \tau \wedge \mathbf{x} \in \mathcal{S} \right\}$$

Maximize return at a target level of risk:

$$\max \left\{ \boldsymbol{\mu}^T \mathbf{x} \mid \mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} = q \wedge \mathbf{x} \in \mathcal{S} \right\}$$

Monte Carlo Simulation

Consider the case where capital is assumed to be equal to 1 (*i.e.*, $\mathbf{x}$ represents proportional allocation) and no short positions are allowed.

$$\mathcal{P}_{\text{Feasible}} = \left\{ \left\{ \mathbf{x}^T \, \mathbf{\Sigma} \, \mathbf{x} \, , \; \boldsymbol{\mu}^T \, \mathbf{x} \right\} \;\middle|\; \mathbf{x}^T \, \mathbb{1} = 1 \wedge \mathbf{x} \geq \mathbf{0} \right\} \right\}$$

The strategy is to simulate a large number of feasible portfolios and examine their behavior in $\{\sigma^2, \mu\}$-space.

A covariance matrix can be constructed by specifying the correlation and volatility separately. The strategy for this is based on the following:

$$\sigma_{x,y} = \rho_{x,y} \, \sigma_x \, \sigma_y$$

In[143]:=
```
vnMean = {0.05, 0.08, 0.10};
vnSigma = {0.07, 0.09, 0.10};
mnCor = {{1, 0.4, 0.6}, {0.4, 1, 0.4}, {0.6, 0.4, 1}};
mnCov = KroneckerProduct[vnSigma, vnSigma] mnCor;
Print["μ = ", MatrixForm[vnMean]]
Print["σ = ", MatrixForm[vnSigma]]
Print["C = ", MatrixForm[mnCor]]
Print["Σ = ", MatrixForm[mnCov]]
```

$$\mu = \begin{pmatrix} 0.05 \\ 0.08 \\ 0.1 \end{pmatrix}$$

$$\sigma = \begin{pmatrix} 0.07 \\ 0.09 \\ 0.1 \end{pmatrix}$$

$$\mathrm{C} = \begin{pmatrix} 1 & 0.4 & 0.6 \\ 0.4 & 1 & 0.4 \\ 0.6 & 0.4 & 1 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 0.0049 & 0.00252 & 0.0042 \\ 0.00252 & 0.0081 & 0.0036 \\ 0.0042 & 0.0036 & 0.01 \end{pmatrix}$$

In generating random feasible portfolios we will assume that short positions are not permitted, *i.e.*, $x_i \geq 0$. The simulation will produce 10,000 cases for analysis.

A straightforward way to sample from the set of feasible portfolios is:

In[151]:=
```
RandomVariate[UniformDistribution[], 2]
```

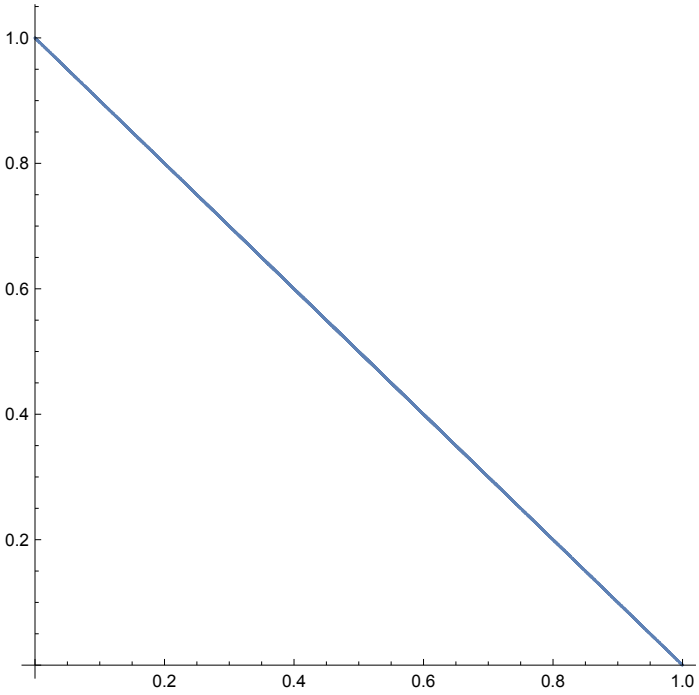
Out[151]=
```
{0.788575, 0.620039}
```

In[152]:=
```
# / Total[#] &[%]
```

Out[152]=
```
{0.559823, 0.440177}
```

In[153]:=
```
ListPlot[
 # / Total[#] & /@ RandomVariate[UniformDistribution[], {10 000, 2}], AspectRatio → 1]
```

Out[153]=



In[154]:=
```
iN = Length[vnMean];
iSimLength = 100 000;
xRandomPortfolio[n_] := # / Total[#] &[RandomVariate[UniformDistribution[], n]];
xPortSdevMean[x_, {μ_, Σ_}] := {√(x.Σ.x), μ.x};
```

In[158]:=
```
mnSamplePortfolios = Table[
   #/Total[#] &[RandomVariate[UniformDistribution[{0, 1}], iN]],
   {iSimLength}
  ];
```

In[159]:=
```
mnSamplePortfolios[[1 ;; 3]]
```

Out[159]=
```
{{0.338403, 0.152523, 0.509073},
 {0.423843, 0.325466, 0.25069}, {0.466135, 0.459536, 0.0743284}}
```

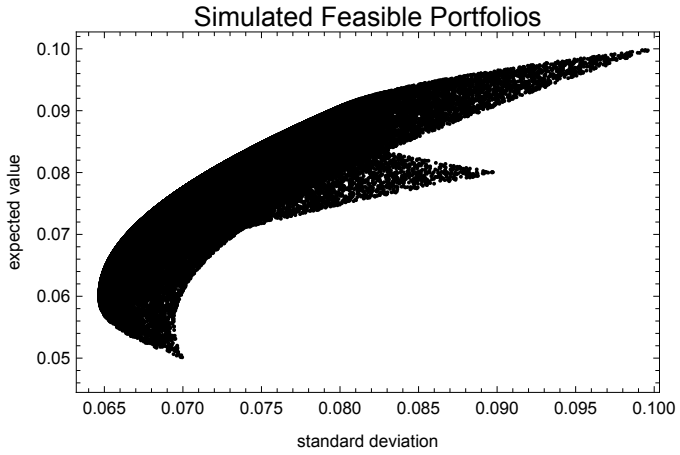
In[160]:=
```
vvnSdevMeanPts = xPortSdevMean[#, {vnMean, mnCov}] & /@ mnSamplePortfolios;
```

In[161]:=
```
Short[vvnSdevMeanPts, 10]
```

Out[161]//Short=
```
{{0.0748825, 0.0800294}, {0.0673941, 0.0722985},
  {0.0666858, 0.0675025}, {0.0671838, 0.0716242}, {0.0793827, 0.0898021},
  <<99 990>>, {0.0668622, 0.0614948}, {0.0808917, 0.0827483},
  {0.0649436, 0.0638531}, {0.074835, 0.0822427}, {0.0704304, 0.0701624}}
```

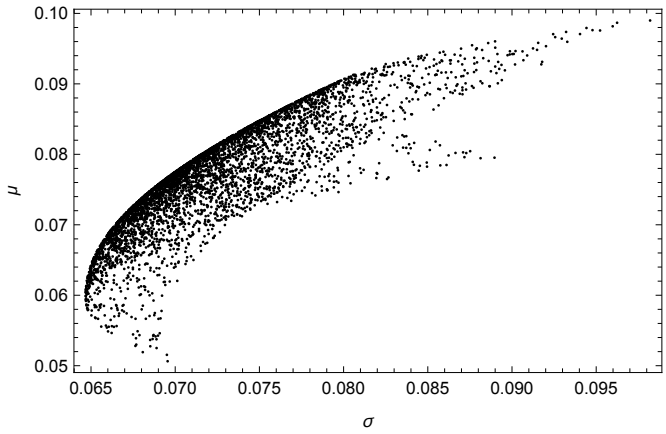
In[162]:=
```
ListPlot[
 vvnSdevMeanPts,
 Frame → True,
 FrameLabel → {
   "standard deviation",
   "expected value",
   Column[
    {
     Style["Portfolios in Variance-Mean Space", FontSize → 14],
     "w/empirical efficient set"
    },
    Center
   ]},
 PlotStyle → {PointSize → Small, Black},
 PlotLabel → Style["Simulated Feasible Portfolios", FontSize → 14],
 AxesLabel → {"σ", "μ"},
 ImageSize → 350
]
```

Out[162]=



Selecting a Set of Optimal Portfolios

Accepting for the moment that our definition of reward is the expected value of return and of risk the standard deviation of return, then clearly we wish to maximize a portfolio's expected return and minimize its standard deviation. While these conditions do not allow us to select a single optimal point from our feasible set they do allow us to restrict our solution to a set of values which are *Pareto optimal*. A solution that is subject to multiple objectives is Pareto optimal when increasing the value of one objective can only be accomplished by decreasing the value of another objective.



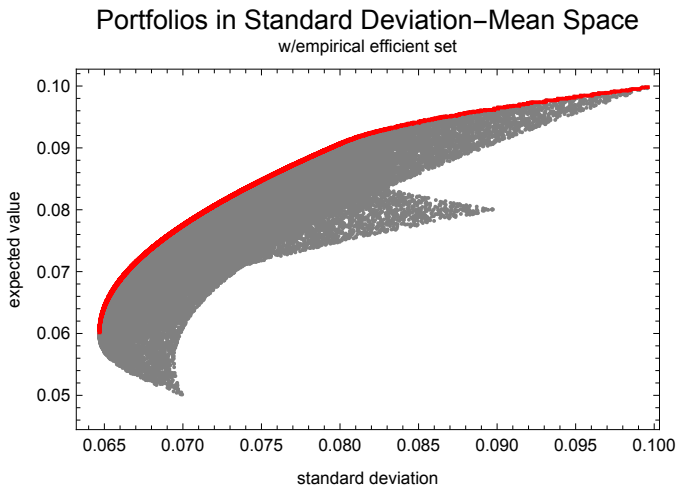
Click on image above to flip through different views.

Pareto optimal mean-standard deviation solutions are called *efficient*. The line of efficient mean-standard deviation solutions, shown in red above, is called the *efficient frontier*, and the portfolios associated with that set are called *efficient portfolios*.

Computing the Efficient Portfolios by Simulation

In[163]:=
```
Show[
 ListPlot[
  vvnSdevMeanPts,
  PlotStyle → {Gray, PointSize[Small]},
  PlotRange → All
 ],
 ListPlot[
  {First /@ #, Rest[FoldList[Max, -∞, Last /@ #]]}ᵀ &[Sort[vvnSdevMeanPts]],
  Joined → True,
  PlotStyle → {Thick, Red}
 ],
 Frame → True,
 FrameLabel → {
   "standard deviation",
   "expected value",
   Column[
    {
     Style["Portfolios in Standard Deviation-Mean Space", FontSize → 14],
     "w/empirical efficient set"
    },
    Center
   ]},
 ImageSize → 350
]
```

Out[163]=



Markowitz Portfolio Analysis - Analytical Solution for a

Simple Case

A portfolio optimization (See http://en.wikipedia.org/wiki/Portfolio_optimization) involves the solution of a mathematical program that seeks to create a portfolio that balances minimizing risk and maximizing reward subject to constraints reflecting the financial condition and policies of the portfolio holder.

Although the solution of a Markowitz Mean-Variance Portfolio Optimization must in general be solved numerically (See http://en.wikipedia.org/wiki/Quadratic_program), there are special cases which do admit an analytic solution. Such cases provide useful insights into the geometry of more general problems.

One approach to building a trade-off between risk and reward is to introduce a trade-off parameter $0 \le \lambda \le \infty$ in the following optimization

$$\mathcal{M} = \min \{\text{RISK} - \lambda \times \text{REWARD} \mid \text{PORFOLIO} \in \text{FEASIBLE SET}\}$$

Letting the vector $\mathbf{x}$ represent the allocation of total available capital, we can express the above in a more mathematical form as

$$\mathcal{M} = \min_{x} \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}\, \mathbf{x} - \lambda \boldsymbol{\mu}^T \mathbf{x} \,\middle|\, \mathbf{x} \in \mathcal{S} \right\}$$

Mathematical Development

A simple form of the mean-variance (Markowitz) model is

$$\mathcal{M} = \min_{x} \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}\, \mathbf{x} - \lambda \boldsymbol{\mu}^T \mathbf{x} \;\middle|\; \mathbb{1}^T \mathbf{x} = 1 \right\}$$

This is known more generally as a quadratic program (QP), *i.e.*, the optimization of a quadratic function subject to linear constraints.

As $\lambda \to 0$ $\mathcal{M}$ approaches a QP representing a minimum variance portfolio:

$$\min_{x} \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}\, \mathbf{x} - \lambda \boldsymbol{\mu}^T \mathbf{x} \;\middle|\; \mathbf{1}^T \mathbf{x} = 1 \right\} \xrightarrow{\lambda \to 0} \min_{x} \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}\, \mathbf{x} \;\middle|\; \mathbf{1}^T \mathbf{x} = 1 \right\}$$

As $\lambda \to \infty$ $\mathcal{M}$ approaches a linear program (LP) representing a maximum return portfolio:

$$\min_{x} \left\{ \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}\, \mathbf{x} - \lambda \boldsymbol{\mu}^T \mathbf{x} \;\middle|\; \mathbf{1}^T \mathbf{x} = 1 \right\} \xrightarrow{\lambda \to \infty} \max_{x} \left\{ \boldsymbol{\mu}^T \mathbf{x} \mid \mathbf{1}^T \mathbf{x} = 1 \right\}$$

In the above the vector $\mathbf{x}$ represents the proportional allocation of our capital; hence, the $\mathbb{1}^T \mathbf{x} = 1$ constraint. There is no restriction on the sign of $\mathbf{x}$. This means we are able to short positions, if necessary.

The Lagrangian (See http://en.wikipedia.org/wiki/Lagrange_multiplier) with multiplier ζ to price the capital constraint is

$$\mathcal{L}(\mathcal{M}) = \frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}\, \mathbf{x} - \lambda \boldsymbol{\mu}^T \mathbf{x} - \zeta \left(\mathbf{1}^T \mathbf{x} - 1 \right)$$

A necessary condition for optimality is that the gradient is equal to zero

$$\nabla \mathcal{L}(\mathcal{M}) = \boldsymbol{\Sigma}\, \mathbf{x} - \lambda \boldsymbol{\mu} - \zeta \mathbf{1} = \mathbf{0}$$

We assume the covariance matrix is of full rank and, hence, positive definite, and this means that the objective function is convex. Thus, the solution of the equation above is also sufficient to ensure global optimality.

Solving for **x** the solution is

$$\mathbf{x} = \lambda\, \Sigma^{-1}\, \boldsymbol{\mu} + \zeta\, \Sigma^{-1}\, \mathbf{1}$$

under the condition that the constraint on total capital holds.

Assume that λ is held fixed, what is the value of ζ ? In order for **x** to be an efficient portfolio, we know that ζ must be chosen so that the elements of **x** sum to 1.

$$1 = \mathbf{1}^T \mathbf{x} = \mathbf{1}^T \left(\lambda\, \Sigma^{-1}\, \boldsymbol{\mu} + \zeta\, \Sigma^{-1}\, \mathbf{1}\right) \Rightarrow \zeta = \frac{1 - \lambda\, \mathbf{1}^T\, \Sigma^{-1}\, \boldsymbol{\mu}}{\mathbf{1}^T\, \Sigma^{-1}\, \mathbf{1}}$$

Substituting ζ back into the solution for **x** we get the efficient frontier parameterized by λ.

$$\mathbf{x} = \lambda\, \Sigma^{-1}\, \boldsymbol{\mu} + \zeta\, \Sigma^{-1}\, \mathbf{1} = \lambda\, \Sigma^{-1}\, \boldsymbol{\mu} + \left(\frac{1 - \lambda\, \mathbf{1}^T\, \Sigma^{-1}\, \boldsymbol{\mu}}{\mathbf{1}^T\, \Sigma^{-1}\, \mathbf{1}}\right) \Sigma^{-1}\, \mathbf{1}$$

For $\lambda = 0$, we ignore the issue of return entirely and the solution realized is the minimum variance portfolio. (Rewrite the above solution with $\lambda = 0$ to see what this looks like.).

As $\lambda \to \infty$, we place less and less emphasis on risk and more on just maximizing return. If shorts are permitted as above, then additional capital will be raised by shorting lower return assets to invest in higher return ones.

Example

We'll reuse the same problem that we simulated above.

In[164]:=
```
xOptimalPortfolio[mnCov_, vnMean_, λ_] := Module[
    {mi, v1, vμ},
    mi = Inverse[mnCov];
    v1 = Total /@ mi;
    vμ = mi.vnMean;
    λ vμ + ((1 - λ Total[vμ]) / Total[v1]) v1
   ];
```

In[165]:=
```
xOptimalPortfolio[mnCov, vnMean, 0.03]
```

Out[165]=
```
{0.401346, 0.358316, 0.240338}
```

In[166]:=
```
Total[%]
```

Out[166]=
```
1.
```

In[167]:=
```
xOptimalPortfolio[mnCov, vnMean, 0.1]
```

Out[167]=
```
{-0.248046, 0.511081, 0.736966}
```

In[168]:=
```
Total[%]
```

Out[168]=
```
1.
```

In[169]:=
```
xOptimalPortfolio[mnCov, vnMean, 0.]
```

Out[169]=
```
{0.679656, 0.292846, 0.0274976}
```

In[170]:=
```
xOptimalPortfolio[mnCov, vnMean, 10.]
```

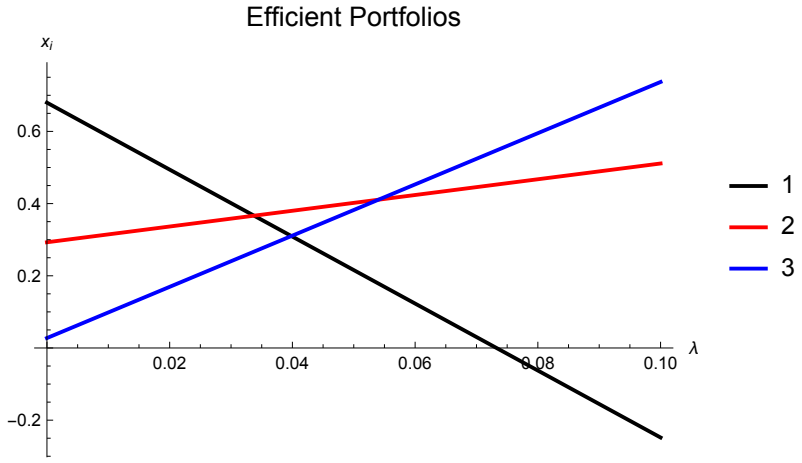
Out[170]=
```
{-92.0906, 22.1163, 70.9743}
```

In[171]:=
```
vuEfficientPortfolios = Table[
  {λ, xOptimalPortfolio[mnCov, vnMean, λ]},
  {λ, 0., 0.1, 0.01}
 ]
```

Out[171]=
```
{{0., {0.679656, 0.292846, 0.0274976}},
 {0.01, {0.586886, 0.314669, 0.0984444}}, {0.02, {0.494116, 0.336493, 0.169391}},
 {0.03, {0.401346, 0.358316, 0.240338}}, {0.04, {0.308575, 0.38014, 0.311285}},
 {0.05, {0.215805, 0.401963, 0.382232}}, {0.06, {0.123035, 0.423787, 0.453178}},
 {0.07, {0.0302646, 0.44561, 0.524125}}, {0.08, {-0.0625057, 0.467434, 0.595072}},
 {0.09, {-0.155276, 0.489257, 0.666019}}, {0.1, {-0.248046, 0.511081, 0.736966}}}
```

In[172]:=
```
ListLinePlot[
 Transpose[{First /@ vuEfficientPortfolios, #}] & /@
  Transpose[Last /@ vuEfficientPortfolios],
 PlotStyle → {{Black, Thick}, {Red, Thick}, {Blue, Thick}},
 PlotLabel → Style["Efficient Portfolios", FontSize → 14],
 AxesLabel → {"λ", "xᵢ"},
 PlotLegends → (ToString /@ Range[Length[vnMean]])
]
```

Out[172]=



In[173]:=
```
xOptimalPortfolio[mnCov, vnMean, 0.07325]
```

Out[173]=
```
{0.000114264, 0.452703, 0.547183}
```

In[174]:=
```
vuEfficientFrontier = Transpose[{First /@ vuEfficientPortfolios,
    xPortSdevMean[#, {vnMean, mnCov}] & /@ (Last /@ vuEfficientPortfolios)}]
```

Out[174]=
```
{{0., {0.0646821, 0.0601603}},
 {0.01, {0.0650061, 0.0643623}}, {0.02, {0.0659686, 0.0685643}},
 {0.03, {0.0675423, 0.0727664}}, {0.04, {0.0696858, 0.0769684}},
 {0.05, {0.0723484, 0.0811705}}, {0.06, {0.0754753, 0.0853725}},
 {0.07, {0.0790113, 0.0895746}}, {0.08, {0.0829041, 0.0937766}},
 {0.09, {0.0871059, 0.0979787}}, {0.1, {0.0915741, 0.102181}}}
```

In[175]:=
```
? Tooltip
```

Out[175]=

Symbol

Tooltip[*expr*, *label*] displays *label* as a tooltip while the mouse pointer is in the area where *expr* is displayed.

In[176]:=
```
Tooltip["This is a sentence.",
 Style["And a short one at that!", FontSize → 24, FontColor → Magenta]]
```

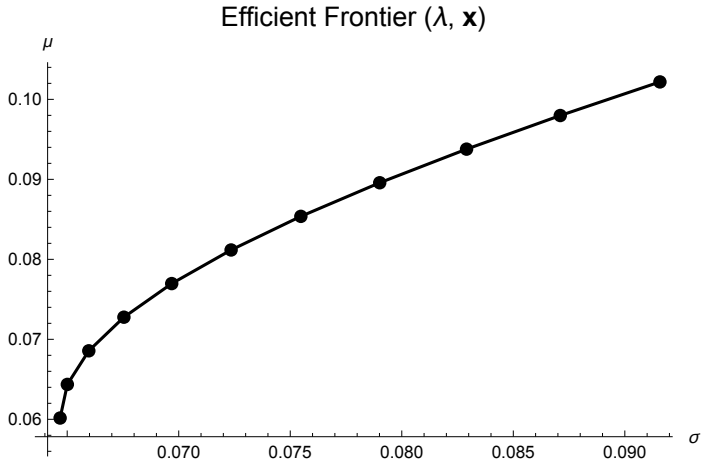
Out[176]=
```
This is a sentence.
```

In[177]:=
```
MapThread[f, {{a, b, c}, {x, y, z}, {U, V, W}}]
```

Out[177]=
```
{f[a, x, U], f[b, y, V], f[c, z, W]}
```

In[178]:=
```
ListPlot[
 MapThread[Tooltip, {Last /@ vuEfficientFrontier, vuEfficientPortfolios}],
 PlotStyle → {PointSize[Large], Black},
 PlotLabel → Style["Efficient Frontier (λ, x)", FontSize → 14],
 Joined → True, Mesh → All,
 AxesLabel → {"σ", "μ"}
]
```

Out[178]=



A Look at Real Data

Downloading and processing the data

The Import[ ] assumes that the file is in the same directory as this notebook. It contains monthly index data for the S & P 500 Total Return Index from {1899, 12} to {2015, 12}. The term "total return" in this instance includes the reinvestment of dividends back into the index so that it properly represents the total wealth from investing in the index. Although this index did not exist over this period, it is a reconstruction of it by Global Financial Data.

In[179]:=
```
NotebookDirectory[]
```

Out[179]=
```
/Volumes/Files/Programming/Capital Markets and Portfolio Theory/1/
```

In[180]:=
```
"/Users/mgomes/Documents/Techniques in
  Financial Modeling and Data Handling/Frey/AMS 512/1/"
```

Out[180]=
```
/Users/mgomes/Documents/Techniques in
  Financial Modeling and Data Handling/Frey/AMS 512/1/
```

In[181]:=
```
FileNameJoin[{NotebookDirectory[], "mxSP500Index.m"}]
```

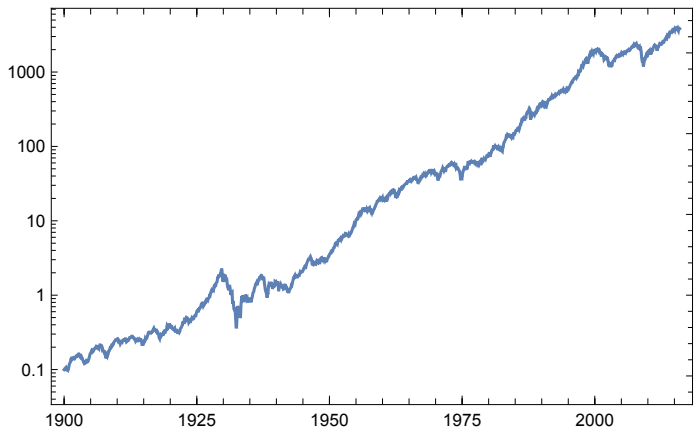
Out[181]=
```
/Volumes/Files/Programming/Capital Markets and Portfolio Theory/1/mxSP500Index.m
```

In[182]:=
```
mxSP500Index = Import[FileNameJoin[{NotebookDirectory[], "mxSP500Index.m"}]];
Short[mxSP500Index, 7]
```

Out[183]//Short=
```
{{{1899, 12, 31}, 0.0997052}, {{1900, 1, 31}, 0.101342},
  {{1900, 2, 28}, 0.103484}, {{1900, 3, 31}, 0.104631}, {{1900, 4, 30}, 0.106294},
  {{1900, 5, 31}, 0.101591}, {{1900, 6, 30}, 0.098891}, <<1380>>,
  {{2015, 7, 31}, 3895.8}, {{2015, 8, 31}, 3660.75}, {{2015, 9, 30}, 3570.17},
  {{2015, 10, 30}, 3871.33}, {{2015, 11, 30}, 3882.84}, {{2015, 12, 31}, 3821.6}}
```

In[184]:=
```
DateListLogPlot[mxSP500Index]
```

Out[184]=

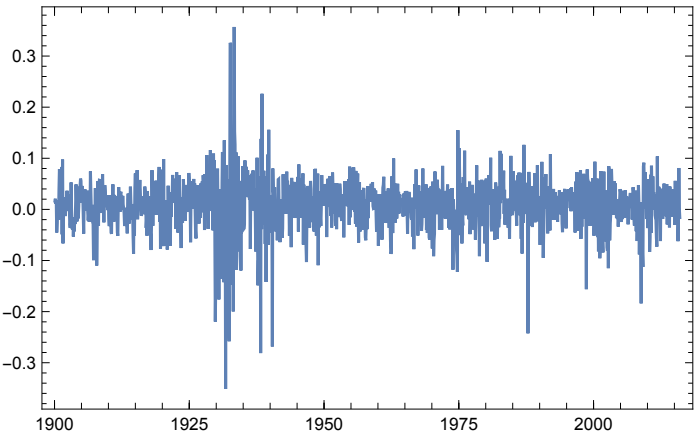


We can compute the log returns of the index by taking the first order differences of the log of the index. Note that we also have to drop the first date.

In[185]:=
```
mxSP500LogReturn =
  Transpose[{Rest[mxSP500Index[[All, 1]]], Differences@Log[mxSP500Index[[All, 2]]]}];
```

In[186]:=
```
DateListPlot[mxSP500LogReturn, Joined → True, PlotRange → All]
```

Out[186]=



In[187]:=
```
Dimensions[mxSP500LogReturn]
```

Out[187]=
```
{1392, 2}
```

Exploratory Data Analysis: Local Volatility

We will employ a simple and useful technique to compute the "local" volatility of the index over time. First note the computation of the variance:

$$\sigma_r{}^2 = \mathrm{E}\left[r^2\right] - \mu_r{}^2 = \left(\frac{1}{n}\sum_{i=1}^{n} r_i{}^2\right) - \mu_r{}^2$$

For r near zero, $\left(\sum_{i=1}^{n} r_i{}^2\right)/n >> \mu_r{}^2$. We can see this effect with our specific set of monthly S&P 500 log returns. First, $\left(\sum_{i=1}^{n} r_i{}^2\right)/n$:

In[188]:=
```
Mean[mxSP500LogReturn[[All, 2]]^2]
```

Out[188]=
```
0.002645
```

and, second, $\mu_r{}^2$:

In[189]:=
```
Mean[mxSP500LogReturn[[All, 2]]]^2
```

Out[189]=
```
0.0000574847
```

We see that $\left(\sum_{i=1}^{n} r_i{}^2\right)/n$ is nearly 50 times greater than $\mu_r{}^2$:

In[190]:=
```
%%
 %
```

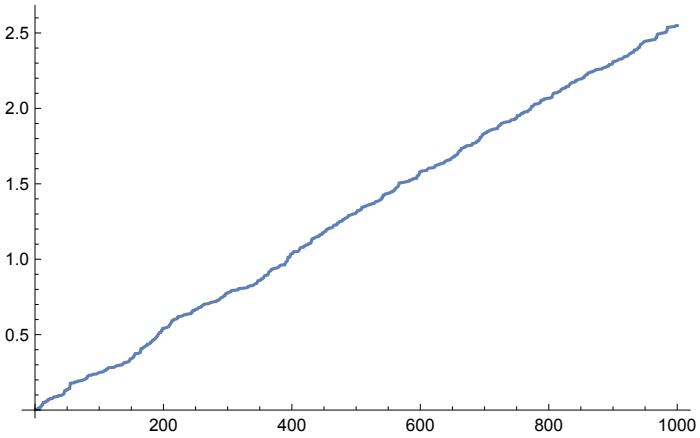
Out[190]=
```
46.0122
```

Thus, we can assume $\sigma_r{}^2 \approx E[r^2]$

$$\sigma_r{}^2 = \mathrm{E}\left[r^2\right] - \mu_r{}^2 \approx \left(\sum_{i=1}^{n} r_i{}^2\right)\Big/ n \implies n\,\sigma_r{}^2 \approx \sum_{i=1}^{n} r_i{}^2$$

In[191]:=
```
ListLinePlot[Accumulate@
  (RandomVariate[NormalDistribution[Mean[mxSP500LogReturn[[All, 2]]],
      StandardDeviation[mxSP500LogReturn[[All, 2]]]], 1000])^2]
```

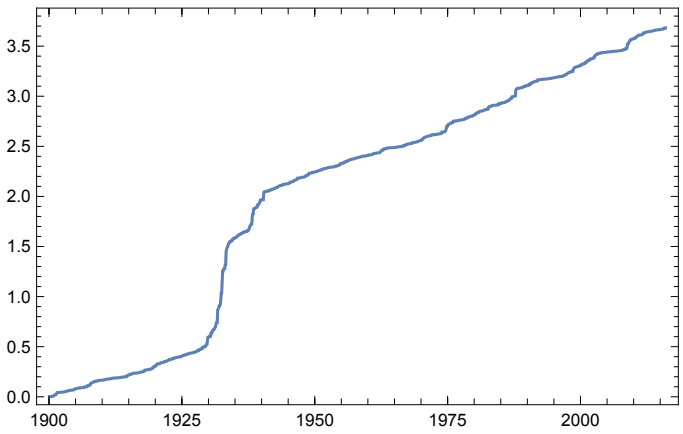
Out[191]=



Assuming that $-1 << r_i << 1$ (*i.e*, r_i is "near" zero), then one can use this as a quick exploratory data analysis technique to study return volatility over time: The cumulative sum of r^2 is approximately the variance of the random variable R. If the volatility of R varies with time, then it will show up in local changes in the slope.

In[192]:=
```
DateListPlot[
 Transpose[{mxSP500LogReturn[[All, 1]], Accumulate[mxSP500LogReturn[[All, 2]]^2]}],
 Joined → True, PlotRange → All, ImageSize → 350]
```

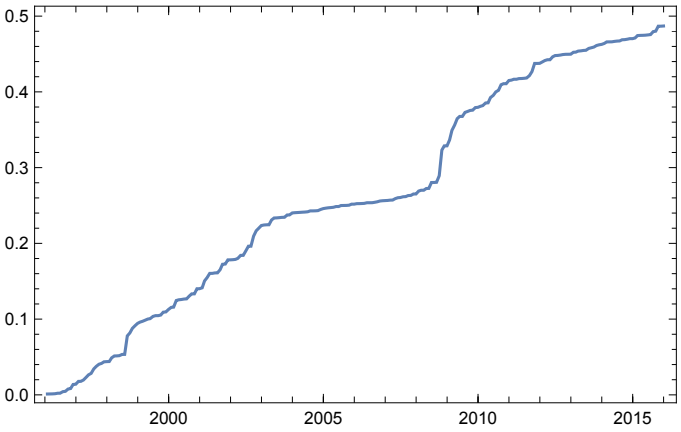
Out[192]=



Slicing notation specifies start, end, column.

In[193]:=
```
DateListPlot[Transpose[
  {mxSP500LogReturn[[-240 ;; -1, 1]], Accumulate[mxSP500LogReturn[[-240 ;; -1, 2]]^2]}],
 Joined → True, PlotRange → All, ImageSize → 350]
```

Out[193]=



Characterizing the return distribution

$$\begin{aligned}\mu &= \mathrm{E}[X]\\ \sigma^2 &= \mathrm{E}\left[(X-\mu)^2\right]\\ \text{skew} &= \mathrm{E}\left[(X-\mu)^3\right]/\sigma^3\\ \text{kurt} &= \mathrm{E}\left[(X-\mu)^4\right]/\sigma^4\end{aligned}$$

In[194]:=
```
Through[{Mean, StandardDeviation, Skewness, Kurtosis}[mxSP500LogReturn[[All, 2]]]]
```

Out[194]= {0.00758187, 0.0508859, −0.478152, 11.2052}

In[195]:=
```
Through[{Median, MedianDeviation}[mxSP500LogReturn[[All, 2]]]]
```

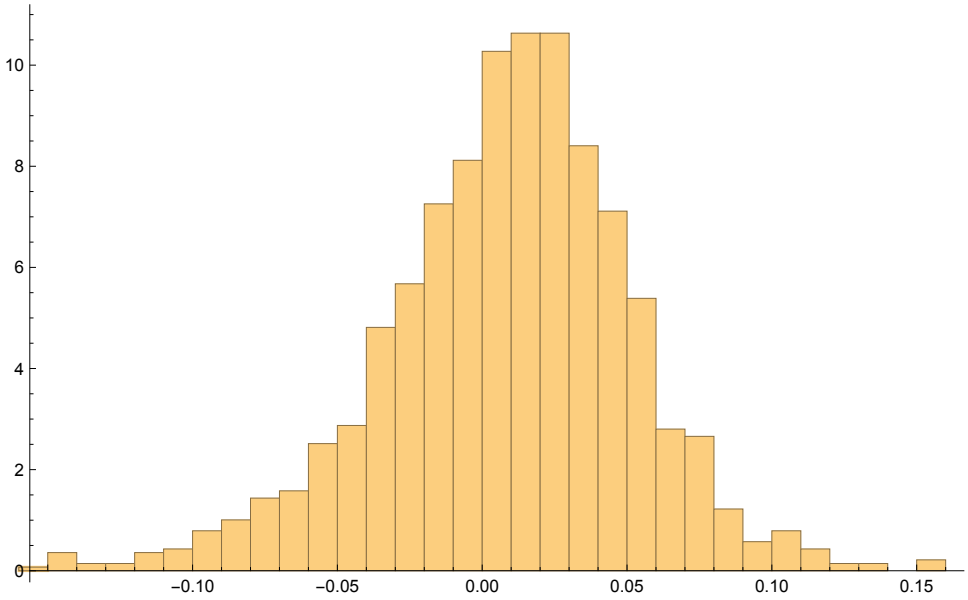
Out[195]= {0.0113887, 0.0267734}

In[196]:=
```
iSample = First[Dimensions[mxSP500LogReturn]]
vnNormedReturn = (# - Median[#])/MedianDeviation[#] &[mxSP500LogReturn[[All, 2]]];
```

Out[196]= 1392

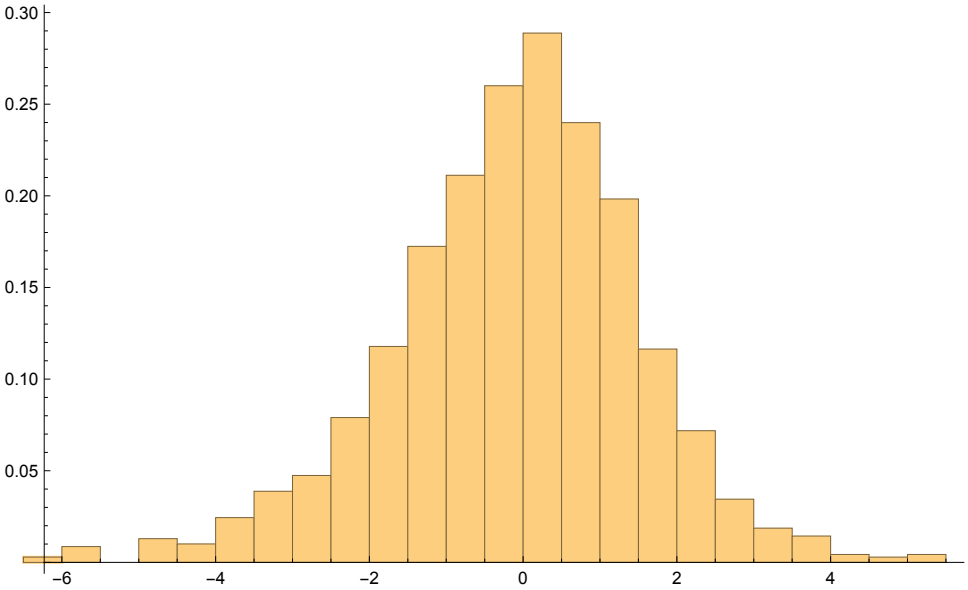
In[198]:=
```
Histogram[mxSP500LogReturn[[All, 2]], Automatic, "PDF", ImageSize → 500]
```

Out[198]=



In[199]:=
```
Histogram[vnNormedReturn, Automatic, "PDF", ImageSize → 500]
```

Out[199]=



Normal Distribution

In[200]:=
```
distSP500N =
 EstimatedDistribution[mxSP500LogReturn[[All, 2]], NormalDistribution[m, s]]
```

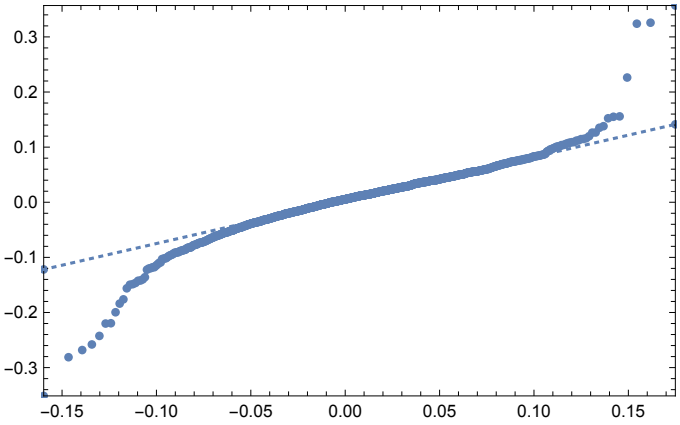
Out[200]=
```
NormalDistribution[0.00758187, 0.0508676]
```

In[201]:=
```
Through[{Mean, StandardDeviation, Skewness, Kurtosis}[distSP500N]]
```

Out[201]=
```
{0.00758187, 0.0508676, 0, 3}
```

In[202]:=
```
QuantilePlot[mxSP500LogReturn[All, 2], distSP500N,
 PlotRange → All, PlotStyle → {PointSize[Medium]}, ImageSize → 350]
```

Out[202]=



Student *t* Distribution

In[203]:=
```
distSP500T =
 EstimatedDistribution[mxSP500LogReturn[All, 2], StudentTDistribution[m, s, d]]
```

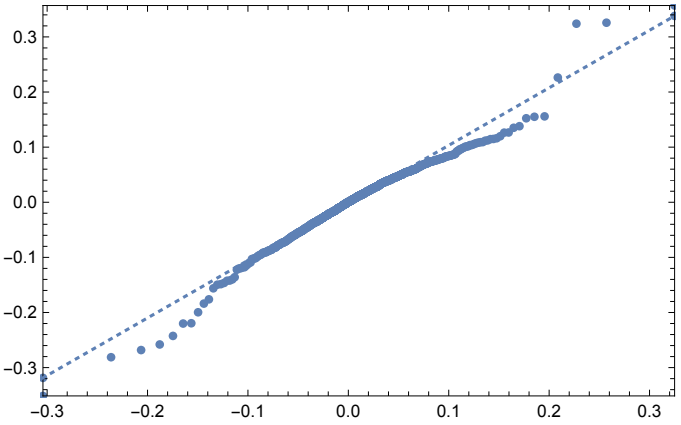
Out[203]=
```
StudentTDistribution[0.0103985, 0.0347629, 3.83284]
```

In[204]:=
```
Through[{Mean, StandardDeviation, Skewness, Kurtosis}[distSP500T]]
```

Out[204]=
```
{0.0103985, 0.0502706, 0, Indeterminate}
```

In[205]:=
```
QuantilePlot[mxSP500LogReturn[All, 2], distSP500T,
 PlotRange → All, PlotStyle → {PointSize[Medium]}, ImageSize → 350]
```

Out[205]=



Quadratic Programming

Consider adding a no short constraint to our original example.

$$\mathcal{M} = \min_{x} \left\{ \frac{1}{2} \mathbf{x}^T \mathbf{\Sigma}\, \mathbf{x} \;\middle|\; \boldsymbol{\mu}^T \mathbf{x} = r_{\text{targ}} \wedge 1^T \mathbf{x} = 1 \wedge \mathbf{x} \geq \mathbf{0} \right\}$$

We no longer have a direct analytical solution and must resort to numerical techniques. We will use the xQuadraticProgramming[] function.

In[206]:=
```
? xQuadraticProgramming
```

Out[206]=
```
Missing[UnknownSymbol, xQuadraticProgramming]
```

In[207]:=
```
mnCons = {vnMean, {1, 1, 1}};
xRhs = {{#, 0}, {1, 0}} &;
mnBounds = {{0, ∞}, {0, ∞}, {0, ∞}};
```

In[210]:=
```
xDisplayQP["Sample QP", {{"target", "capital"}, {"x1", "x2", "x3"}},
 {{}, mnCov}, mnCons, xRhs[τ], mnBounds]
```

Out[210]=
```
xDisplayQP[Sample QP, {{target, capital}, {x1, x2, x3}},
 {{}, {{0.0049, 0.00252, 0.0042}, {0.00252, 0.0081, 0.0036}, {0.0042, 0.0036, 0.01}}},
 {{0.05, 0.08, 0.1}, {1, 1, 1}}, {{τ, 0}, {1, 0}}, {{0, ∞}, {0, ∞}, {0, ∞}}]
```

In[211]:=

"Sample QP"						
		" "				
"Objective"	"Lin"	{}				
	"Quad"		"x1"	"x2"	"x3"	
		"x1"	0.00490000000000001`	0.00252`	0.00420000000000001`	
		"x2"	0.00252`	0.0081`	0.0036`	
		"x3"	0.00420000000000001`	0.0036`	0.0100000000000002`	
"Constraints"			"x1"	"x2"	"x3"	" " "RHS"
		"target"	0.05`	0.08`	0.1`	"==" τ
		"capital"	1	1	1	"==" 1
"Bounds"			"LO"	"UP"		
		"x1"	0	∞		
		"x2"	0	∞		
		"x3"	0	∞		

Out[211]=
```
{{{{{}}, {{{0.0049, 0.00252, 0.0042},
      {0.00252, 0.0081, 0.0036}, {0.0042, 0.0036, 0.01}}}}},
 {{{0.05, 0.08, 0.1, ==, τ}, {1, 1, 1, ==, 1}}}, {{{0, ∞}, {0, ∞}, {0, ∞}}}}
```

In[212]:=
```
xQuadraticProgramming[{{}, mnCov}, {{1, 1, 1}}, {{1, 0}}, mnBounds]
```

Out[212]=
```
xQuadraticProgramming[
 {{}, {{0.0049, 0.00252, 0.0042}, {0.00252, 0.0081, 0.0036}, {0.0042, 0.0036, 0.01}}},
 {{1, 1, 1}}, {{1, 0}}, {{0, ∞}, {0, ∞}, {0, ∞}}]
```

In[213]:=
```
nMinVarRet = vnMean.Last[%]
```

Out[213]=
```
{0., ∞}
```

In[214]:=
```
LinearProgramming[-vnMean, {{1, 1, 1}}, {{1, 0}}, mnBounds]
```

Out[214]=
```
{0., 0., 1.}
```

In[215]:=
```
nMaxRet = vnMean.%
```

Out[215]=
```
0.1
```

In[216]:=
```
mnNoShortEffPort = Table[

  Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]],
  {r, nMinVarRet, nMaxRet, (nMaxRet - nMinVarRet)/20}
 ]
```

Infinity: Indeterminate expression 0 (−∞) encountered.

Table: Iterator {r, {0., ∞}, 0.1, {0.005, −∞}} does not have appropriate bounds.

Infinity: Indeterminate expression 0 (−∞) encountered.

Table: Iterator {r, {0., ∞}, 0.1, {0.005, −∞}} does not have appropriate bounds.

Out[216]=
```
Table[Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]],
 {r, {0., ∞}, 0.1, {0.005, -∞}}]
```

In[217]:=
```
mnNoShortEffFront = {√(#.mnCov.#), vnMean.#} & /@ mnNoShortEffPort
```

Dot: Nonrectangular tensor encountered.

Dot: Nonrectangular tensor encountered.

Dot: Nonrectangular tensor encountered.

General: Further output of Dot::rect will be suppressed during this calculation.

Table: Non−list iterator ({√(#1.mnCov.#1), vnMean.#1} &)[{r, {0., ∞}, 0.1, {0.005, −∞}}] at position 2 does not evaluate to a real numeric value.

Out[217]=
```
Table[({√(#1.mnCov.#1), vnMean.#1} &)[
  Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]]],
 ({√(#1.mnCov.#1), vnMean.#1} &)[{r, {0., ∞}, 0.1, {0.005, -∞}}]]
```

In[218]:=
```
ListPlot[mnNoShortEffFront, Joined → True,
 PlotStyle → {Red, Thick}, PlotLabel → "Efficient Frontier (No Shorts)\n",
 AxesLabel → {"σ", "μ"}, PlotRange → All, ImageSize → 350]
```

ListPlot: Table[({√(#1.mnCov.#1), vnMean.#1} &)[Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]]], ({√(#1.mnCov.#1), vnMean.#1} &)[{r, {0., ∞}, 0.1, {0.005, −∞}}]] is not a list of numbers or pairs of numbers.

Out[218]=
```
ListPlot[Table[({√(#1.mnCov.#1), vnMean.#1} &)[
   Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]]],
  ({√(#1.mnCov.#1), vnMean.#1} &)[{r, {0., ∞}, 0.1, {0.005, -∞}}]],
 Joined → True, PlotStyle → {■, Thickness[Large]},
  PlotLabel → Efficient Frontier (No Shorts)
 , AxesLabel → {σ, μ}, PlotRange → All, ImageSize → 350]
```

In[219]:=
```
mnNoShortEffPort.vnMean
```

Out[219]=
```
Table[Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]],
 {r, {0., ∞}, 0.1, {0.005, -∞}}].{0.05, 0.08, 0.1}
```

In[220]:=
```
ListLinePlot[
 Transpose[{mnNoShortEffPort.vnMean, #}] & /@
  Transpose[mnNoShortEffPort],
 PlotStyle → {{Black, Thick}, {Red, Thick}, {Blue, Thick}},
 PlotLabel → Style["Efficient Portfolios (No Shorts)", FontSize → 14],
 AxesLabel → {"τ", "xᵢ"},
 PlotLegends → (ToString /@ Range[Length[vnMean]])
]
```

Transpose: The first two levels of {Table[Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]], {r, {0., ∞}, 0.1, {0.005, −∞}}].{0.05, 0.08, 0.1}, Table[Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]], {r, {0., ∞}, 0.1, ≪1≫}] } cannot be transposed.

ListLinePlot: Transpose[Transpose[{Table[Last[xQuadraticProgramming[{≪2≫}, mnCons, xRhs[≪1≫], mnBounds]], {r, {0., DirectedInfinity[≪1≫]}, 0.1, {0.005, DirectedInfinity[≪1≫]}}].{0.05, 0.08, 0.1}, Table[Last[xQuadraticProgramming[≪1≫]], {≪1≫}]}]] is not a list of numbers or pairs of numbers.

Out[220]=
```
ListLinePlot[Transpose[Transpose[
   {Table[Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]],
     {r, {0., ∞}, 0.1, {0.005, -∞}}].{0.05, 0.08, 0.1},
    Table[Last[xQuadraticProgramming[{{}, mnCov}, mnCons, xRhs[r], mnBounds]],
     {r, {0., ∞}, 0.1, {0.005, -∞}}]}]]],
 PlotStyle → {{■, Thickness[Large]}, {■, Thickness[Large]}, {■, Thickness[Large]}},
 PlotLabel → Efficient Portfolios (No Shorts),
 AxesLabel → {τ, xᵢ},
 PlotLegends → {1, 2, 3}]
```