1. WEIGHTS CONSTRAINED MVO

OVER-ARCHING GOALS

1. Review key aspects of mean-variance optimal portfolio mathematical theory, including details on:
   a. Global minimum variance portfolio and renewed interest in it
   b. Linear efficient frontier
2. Illustrate estimation error with bootstrap resampling
3. Show in detail how to use solve.QP and Rglpk_solve_LP functions to compute MVO portfolios under frequently used weights constraints
4. Examples of efficient frontiers under weight constraints with view toward subsequent computing exercises

1.1 MVO THEORY BASICS

This section provides a brief review of basic mean-variance portfolio optimization (MVO) with linear equality constraints. These type of MVO problems have analytic solutions that can easily be obtained using the method of Lagrange multipliers, and as such have a long history in academic coverage of portfolio optimization. There are three main types of such problems:

1. **Investment in both cash and risky assets: portfolio weights in risky assets sum to less than one**
2. **Full-investment in risky assets: portfolio weights sum to one**
3. **Benchmark relative active portfolios and dollar neutral portfolios, where weights sum to zero**

Since applied portfolio optimization problems typically involve linear (and sometimes nonlinear) inequality constrains, analytic solutions with equality constraints are mainly of conceptual use and as reference points for performance loss caused due to inequality constraints and penalty terms.

Since active portfolio management is a very substantial topic that we discuss in a later chapter, we defer discussion of type three MVO with linear equality constraints to that chapter and concentrate on the first two types in this section.

Asset Returns $\mathbf{r}_t = \left(r_{t1},...,r_{tn}\right)', \quad t = 1,...,T$

r_{ti} = arithmetic return for asset i at time t

Portfolio P Weights $\mathbf{w} = (w_1, w_2, \cdots, w_n)'$

The weights represent fractions of wealth invested in risky assets.

Portfolio Returns $r_{P,t} = \sum_{i=1}^{n} w_i r_{ti} \quad \rightarrow \quad r_P = \sum_{i=1}^{n} w_i r_i$

(often drop the time subscript t)

Returns Mean Vector & Covariance Matrix

Returns matrix

$$\mathbf{R} = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{T1} & \cdots & r_{Tn} \end{pmatrix}$$

Mean returns vector (assumed time invariant)

$$\begin{aligned} \boldsymbol{\mu} &= E(\mathbf{r}_t) \\ &= (E(r_{t1}), E(r_{t2}), \cdots, E(r_{tn}))' \\ &= (\mu_1, \mu_2, \cdots, \mu_n)' \end{aligned}$$

Pairwise Covariances (assumed time invariant)

$$\Sigma_{ij} = E\Big[(r_{ti} - \mu_i)(r_{tj} - \mu_j)\Big], \quad i, j = 1, \cdots, n$$
$$= E(r_{ti} r_{tj}) - \mu_i \mu_j$$

Covariance Matrix

$$\mathbf{\Sigma} = E\Big[(\mathbf{r}_t - \boldsymbol{\mu})(\mathbf{r}_t - \boldsymbol{\mu})'\Big] \quad n \times n$$
$$= E(\mathbf{r}_t \mathbf{r}_t') - \boldsymbol{\mu}\boldsymbol{\mu}'$$

It is assumed throughout unless otherwise noted that the returns covariance matrix $\mathbf{\Sigma}$ is positive definite, hence non-singular. Under this assumption the inverse covariance matrix $\mathbf{\Sigma}^{-1}$ exists.

Correlations

$$\rho_{ij} = \frac{\text{cov}(r_i, r_j)}{\text{var}^{1/2}(r_i) \cdot \text{var}^{1/2}(r_j)}, \qquad i, j = 1, \cdots, n$$

$$= \frac{\Sigma_{ij}}{\Sigma_{ii}^{1/2} \cdot \Sigma_{jj}^{1/2}}$$

$$= \frac{\Sigma_{ij}}{\sigma_i \cdot \sigma_j}$$

Correlation Matrix

$$\mathbf{Corr} = diag(\sigma_i^{-1}) \cdot \mathbf{\Sigma} \cdot diag(\sigma_i^{-1}) \qquad n \times n$$

Portfolio Returns $$r_P = \sum_{i=1}^{n} w_i r_i = \mathbf{w}'\mathbf{r}$$

Portfolio Mean Returns $$\mu_P = E\left(r_P\right) = E(\mathbf{w}'\mathbf{r}) = \mathbf{w}'E(\mathbf{r}) = \mathbf{w}'\boldsymbol{\mu}$$

Portfolio Variance

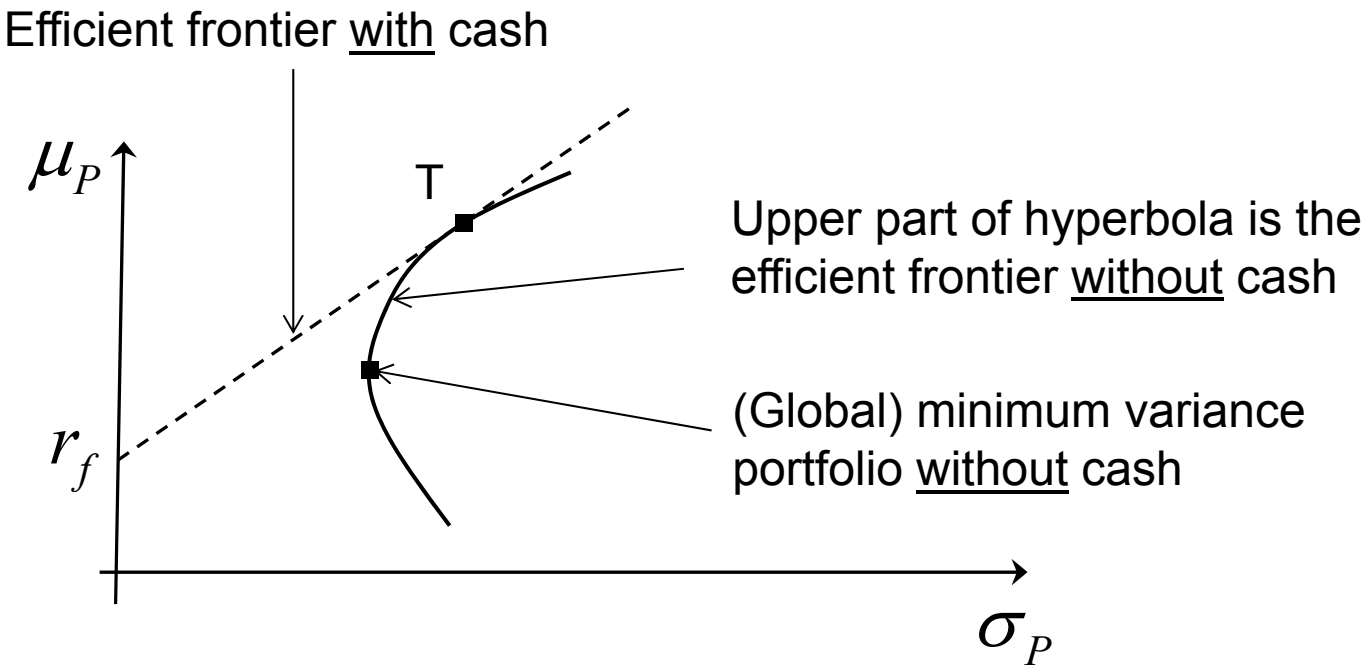
$$\begin{aligned}
\sigma_P^2 &= \mathrm{var}(r_P) \\
&= \mathrm{var}(\mathbf{w}'\mathbf{r}) \\
&= E\left(\mathbf{w}'\left(\mathbf{r}-\boldsymbol{\mu}\right)\right)^2 \\
&= E(\mathbf{w}'\left(\mathbf{r}-\boldsymbol{\mu}\right)\left(\mathbf{r}-\boldsymbol{\mu}\right)'\mathbf{w}) \\
&= \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}
\end{aligned}$$

Portfolio Volatility $$\sigma_P = \left(\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}\right)^{1/2}$$

The Efficient Frontiers

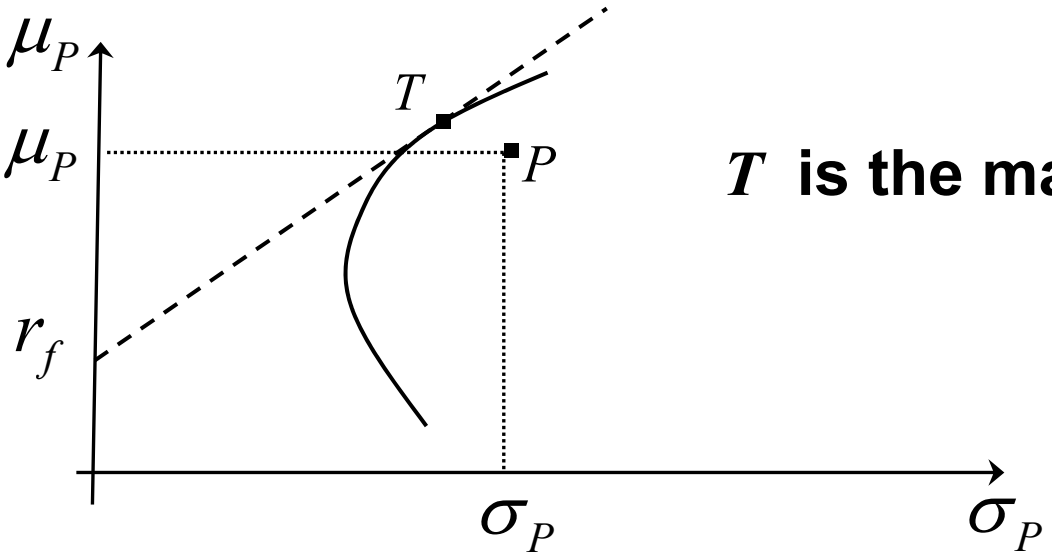
The Problem: Minimize portfolio variance for each possible mean return constraint for two distinct cases: (1) risky assets only, and (2) risky assets plus "cash" (i.e., risk-free investment)

The Solution: The "efficient frontiers"



The Sharpe Ratio

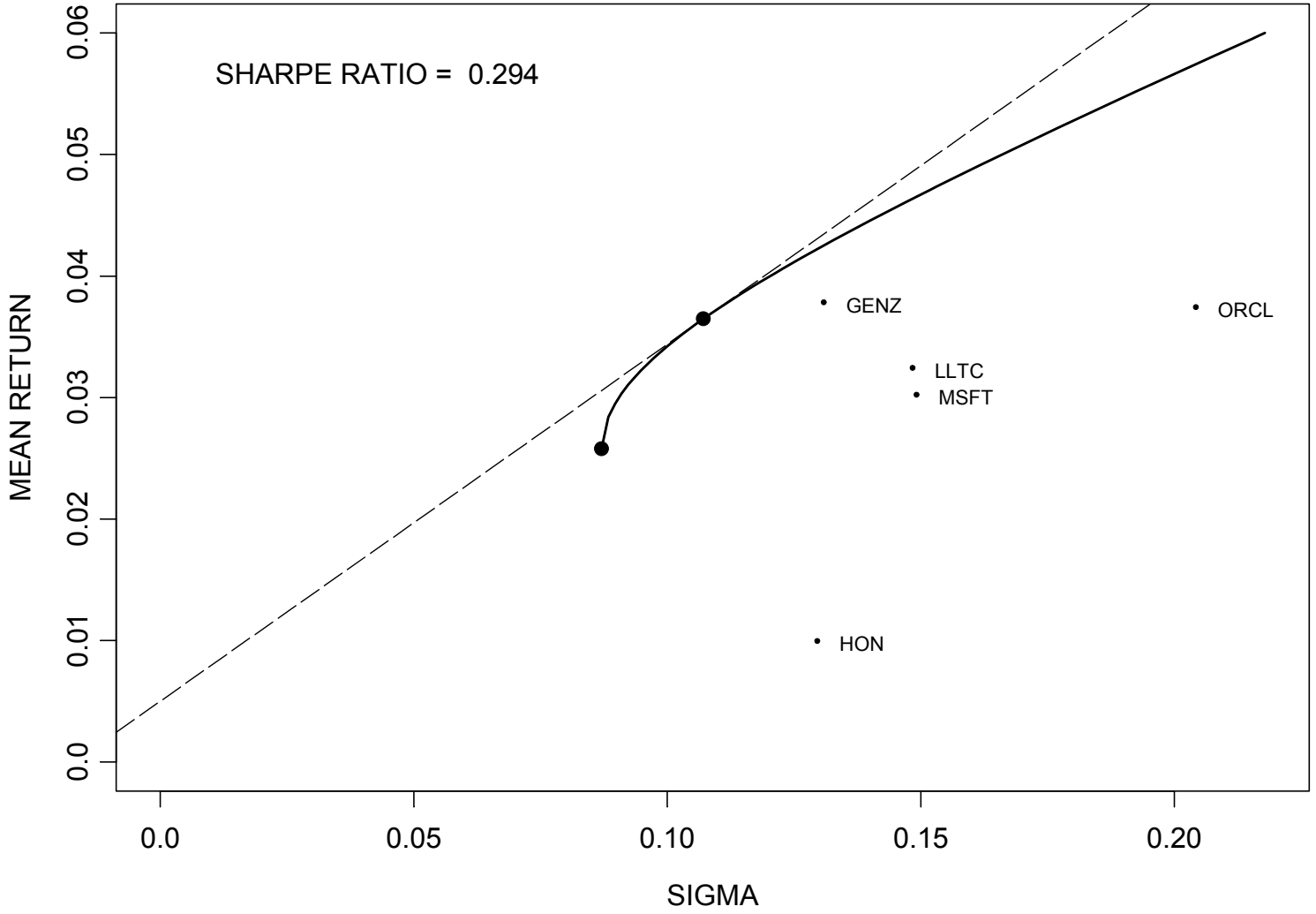
For any portfolio P: $$SR_P = \frac{\mu_P - r_f}{\sigma_P} = \text{Sharpe Ratio}$$



T is the maximum SR portfolio

NOTE: Subsequently we develop a simple formula for SR_T

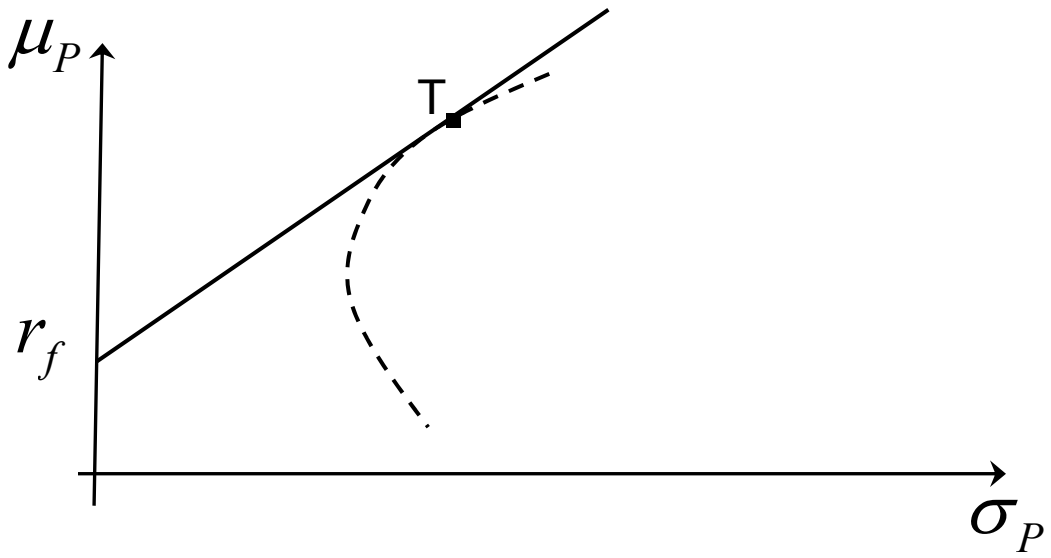
EFFICIENT FRONTIER



```
$w.mv:
      GENZ        HON        LLTC       MSFT        ORCL
 0.3081789 0.3753527 0.08328089 0.1456418 0.08754582
$mu.mv:
[1] 0.02577599
$sigma.mv:
[1] 0.08702582
$w.t:
      GENZ          HON      LLTC       MSFT       ORCL
 0.5548272 -0.03873531 0.233836 0.1492129 0.1008592
$mu.t:
[1] 0.03648147
$sigma.t:
          [,1]
[1,] 0.107126
```

The Separation Theorem

The portfolio T is determined without knowledge of the investor's risk preferences. You only need to know the asset returns mean vector $\boldsymbol{\mu}$, the covariance matrix $\boldsymbol{\Sigma}$, and the risk-free rate r_f.



Intuitively, your risk aversion determines where you operate along the straight line. Shortly we find out just how.

Equilibrium, Market Portfolio and CAPM

We have not made use of any equilibrium assumption or market portfolio assumption. In fact:

1. Under equilibrium it can be argued that T is the market portfolio M
2. The equilibrium and market portfolio result is needed to establish the capital asset pricing model (CAPM)

For discussion of the above see Luenberger (1998). *Investment Science*, Sections 7.1-7.3.

No-Cash MVO Portfolios

MVO portfolios with risky assets only, i.e., no cash, can be obtained by one of the following two equivalent methods, using a full-investment constraint that the portfolio weights sum to one, but with no restriction on short-selling:

Method 1: Minimize variance (risk) for each mean return target

Method 2: Maximize quadratic utility

Both methods use Lagrange multipliers to hand the constraints (two in Method 1 and one in Method 2).

Method 1 leads to the entire hyperbolic frontier whose shape and formula is given on the next slide. Derivation is given in 1A.1.

Method 2 leads only to the efficient frontier (why?). See 1A.2.

No-Cash Global Min Variance Portfolio

Here we assume full-investment in risky assets (no cash in portfolio), and have a very simple solution. Recent research shows that this is a surprisingly good portfolio (see subsequent references and discussion)

$$\min_{\mathbf{w}} \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w} \qquad \text{subject to} \qquad \mathbf{w}'\mathbf{1} = 1$$

Langrangian: $$L(\mathbf{w}) = \frac{1}{2}\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w} + \lambda\left(1 - \mathbf{w}'\mathbf{1}\right)$$

Set $\frac{d}{d\mathbf{w}} L(\mathbf{w}) = 0$ and apply constraint to get:

$$\mathbf{w}_{GMV} = \frac{\boldsymbol{\Sigma}^{-1}\mathbf{1}}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}$$

Knowledge of mean returns is not required, except as nuisance parameters in $\boldsymbol{\Sigma}$!

GMV Portfolio Mean Return and Volatility

$$\mu_{GMV} = \mathbf{w}'_{GMV}\boldsymbol{\mu} = \frac{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}$$

N.B. μ_{GMV} can be positive or negative!

$$\sigma_{GMV} = \sqrt{\mathbf{w}'_{GMV}\boldsymbol{\Sigma}\mathbf{w}_{GMV}} = \frac{1}{\sqrt{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}}$$

Research Results on the GMV Portfolio

Clarke, de Silva and Thorley, S. (2006). Minimum-Variance Portfolios in the U.S. Equity Market, *Jour. of Portfolio Management*, Fall, pp. 10-24.

- Monthly portfolio rebalancing 1968 through 2004 (456 months)
- Training window is one year of trailing daily excess returns
- 1,000 largest market cap stocks for each rebalance period
- Shrink 1,000 x 1,000 covariance matrix using Bayesian shrinkage toward two-parameter covariance matrix (Ledoit & Wolf, 2004)
- Market is the cap-weighted portfolio of the 1,000 stocks (approx. R1,000)
- GMV portfolio is long-only with upper bound of 3% on weights.

Partial Results (annnualized using excess returns, avg. T-bill rate = 5.95):

Portfolio	Mean Return	Volatility	Sharpe Ratio
Market	5.6%	15.4%	.36
GMV	6.5%	11.7%	.55

Will return to study the approach and results more carefully later on.

See also:

Scherer, B. (2011). A New Look at Minimum Variance Investing. *Journal of Empirical Finance*.

Clarke, de Silva and Thorley, S. (2011). Minimum-Variance Portfolio Composition, *Jour. of Portfolio Management*, Fall, 37, No. 2, 10-24.

Haugen, and Baker, N. (1991). The Efficient Market Inefficiency of Capitalization Weighted Stock Portfolios, *Jour. of Portfolio Management*, Spring, 35-40.

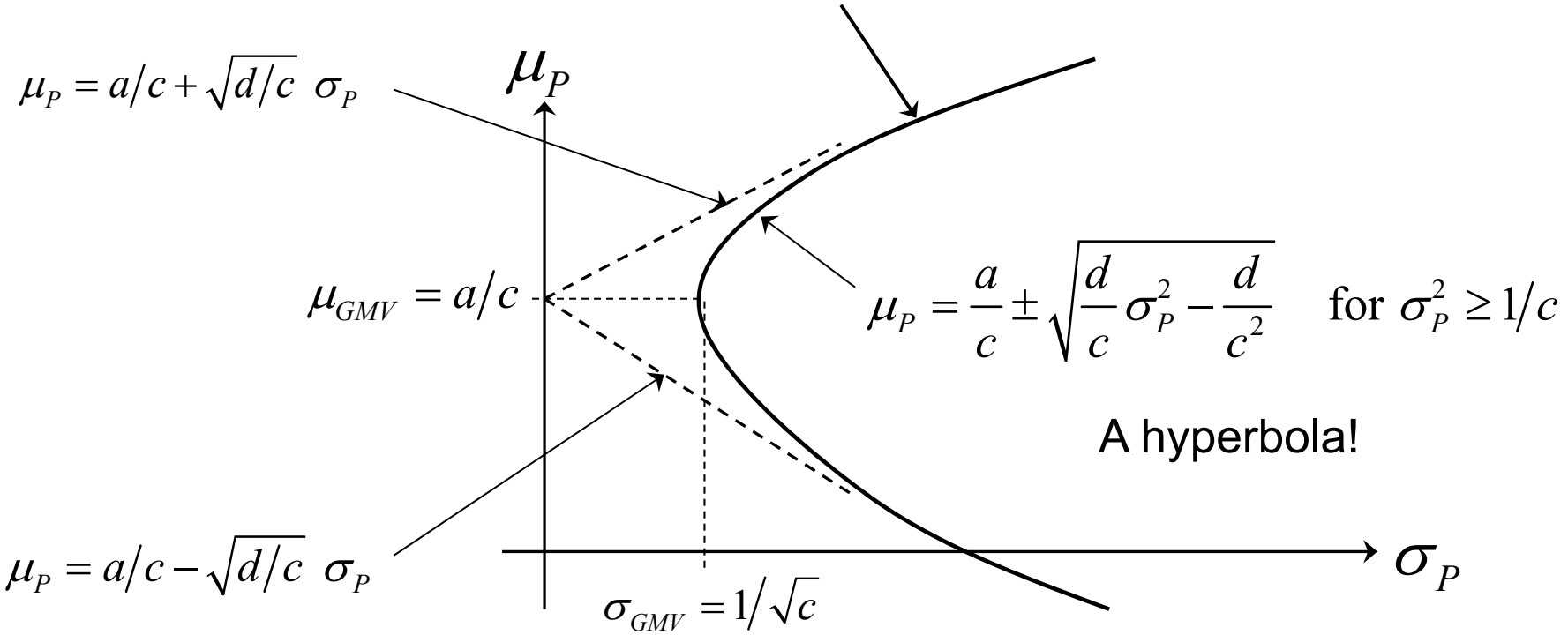
No-Cash Efficient Frontier

$$a = \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{1} \qquad b = \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$$

$$c = \mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1} \qquad d = bc - a^2$$

No restriction on short-selling!

The upper part is the "**Efficient Frontier**"



Risky Assets & Cash MVO Portfolios

A frequent approach to this problem is to first derive the efficient frontier with risky assets only (the upper part of the hyperbola) and then argue more or less heuristically that when cash is added you end up with the straight line that is tangent to efficient frontier of risky assets only. This approach is often confusing.

On the other hand rigorous derivations can be obtained by two methods analogous to those used for the efficient frontier with no cash, only now the sum of weights in the risky assets is less than one:

Method 1: Minimize variance (risk) for each mean return target

Method 2: Maximize quadratic utility

Method 2 is simpler and leads directly to formulas for the maximum Sharpe ratio and risk aversion along the linear efficient frontier.

Quadratic Utility Optimality with Cash

Maximize $$\underbrace{w_0 r_f + \mathbf{w}'\boldsymbol{\mu}}_{\mu_P} - \frac{1}{2}\lambda \underbrace{\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}}_{\sigma_P^2} \qquad 0 < \lambda < \infty$$

Subject to: $w_0 + \mathbf{w}'\mathbf{1} = 1$ No short-selling restriction!

In terms of excess returns $\boldsymbol{\mu}_e = \boldsymbol{\mu} - \mathbf{1} r_f$ the problem becomes:

Maximize $$\underbrace{r_f + \mathbf{w}'\boldsymbol{\mu}_e}_{\mu_P} - \frac{1}{2}\lambda \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}$$

Solution: $$\mathbf{w}_{opt} = \lambda^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e \qquad w_0 = 1 - \mathbf{w}'_{opt}\mathbf{1}$$

Case 1: An all cash position results as $\lambda \to \infty$

Case 2: As $\lambda \to 0$ the portfolio weights become unbounded in absolute value.

Question: In Case 2 what happens to the investment w_0 in cash?

Optimal portfolio mean excess return:

$$\mu_{P,e} = \mathbf{w}'_{opt}\boldsymbol{\mu}_e = \lambda^{-1}\boldsymbol{\mu}_e\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e > 0$$

Note that $\lambda^{-1} = \dfrac{\mu_{P,e}}{\boldsymbol{\mu}_e\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e}$ and substitute into $\mathbf{w}_{opt} = \lambda^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e$.

The result is $$\mathbf{w}_{opt} = \frac{\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e}{\boldsymbol{\mu}_e\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e} \cdot \mu_{P,e}$$

which gives $$\sigma^2_{opt} = \mathbf{w}'_{opt}\boldsymbol{\Sigma}\mathbf{w}_{opt}$$

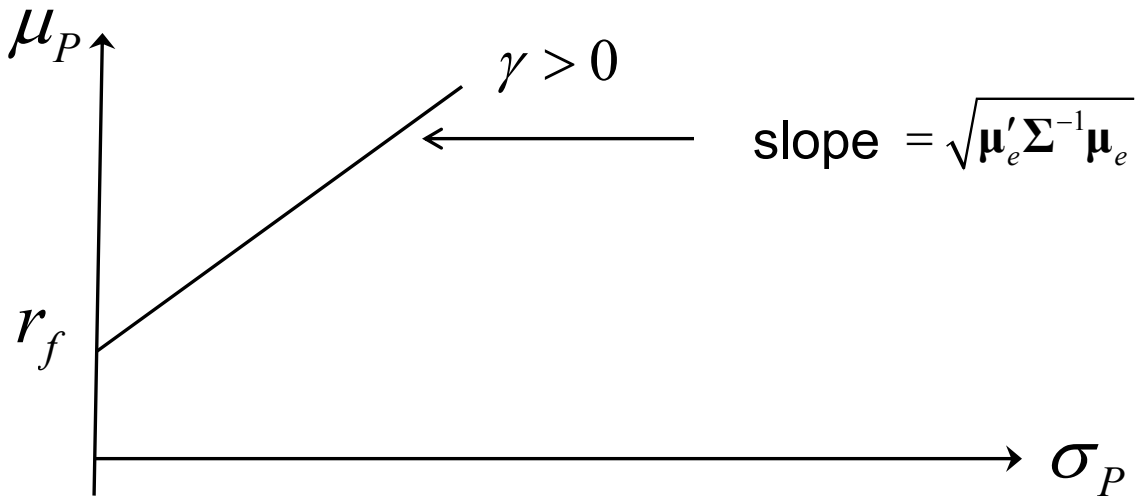
$$= \frac{1}{\boldsymbol{\mu}'_e\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e} \cdot \mu^2_{P,e}$$

Since $\mu_{P,e} > 0$ we have the linear relationship:

$$\mu_{P,e} = \sqrt{\boldsymbol{\mu}'_e\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e} \cdot \sigma_{opt}$$

THEOREM: Quadratic utility optimal portfolios with cash have the following mean versus volatility linear efficient frontier relationship:

$$\mu_P = r_f + \sqrt{\boldsymbol{\mu}_e' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_e} \cdot \sigma_P$$



Is Full Risky-Assets Investment Possible?

You might wonder whether or not there is a point on the preceding linear efficient frontier that corresponds to full-investment in risky assets. This is easy to check. Full-investment in risky asset requires that:

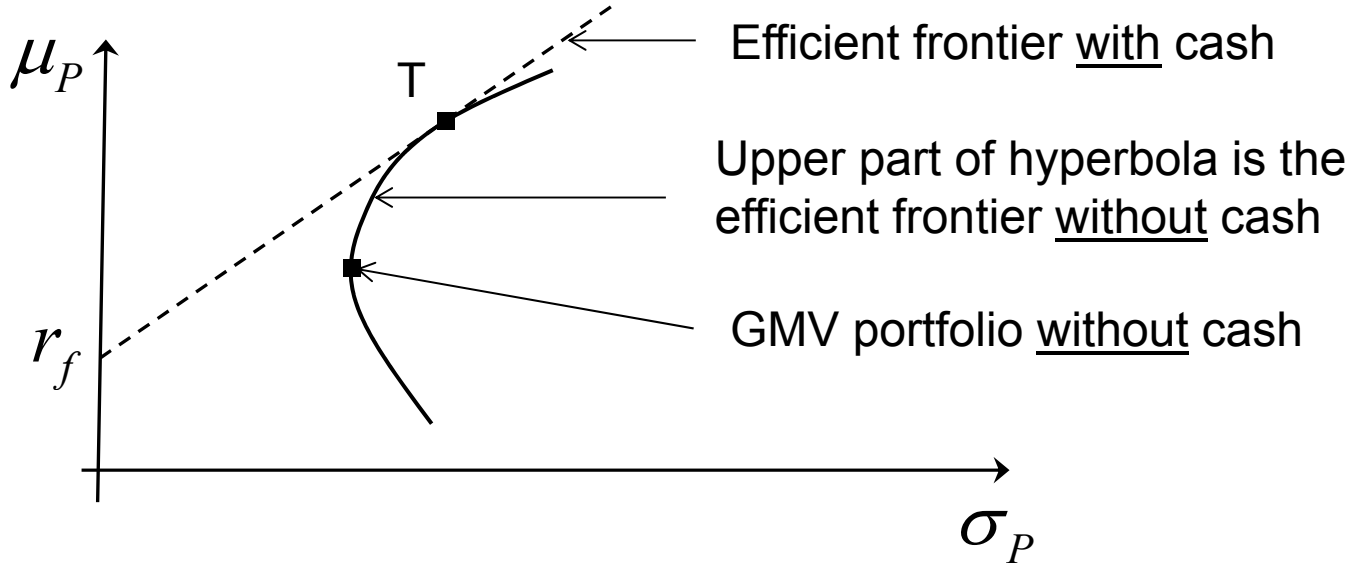
$$\mathbf{1}'\mathbf{w}_{opt} = \lambda^{-1}\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e = 1 \quad \Rightarrow \quad \lambda = \mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e$$

But since $\lambda > 0$, this is only possible if $\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e > 0$. Surprisingly, this condition has a simple interpretation. Recalling the formula for the mean return μ_{GMV} of the GMV portfolio, the excess mean return of the GMV portfolio is:

$$\mu_{GMV,e} = \frac{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}} - r_f = \frac{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_e}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}$$

Result: Full-investment is possible if and only if the excess return of the GMV portfolio is positive.

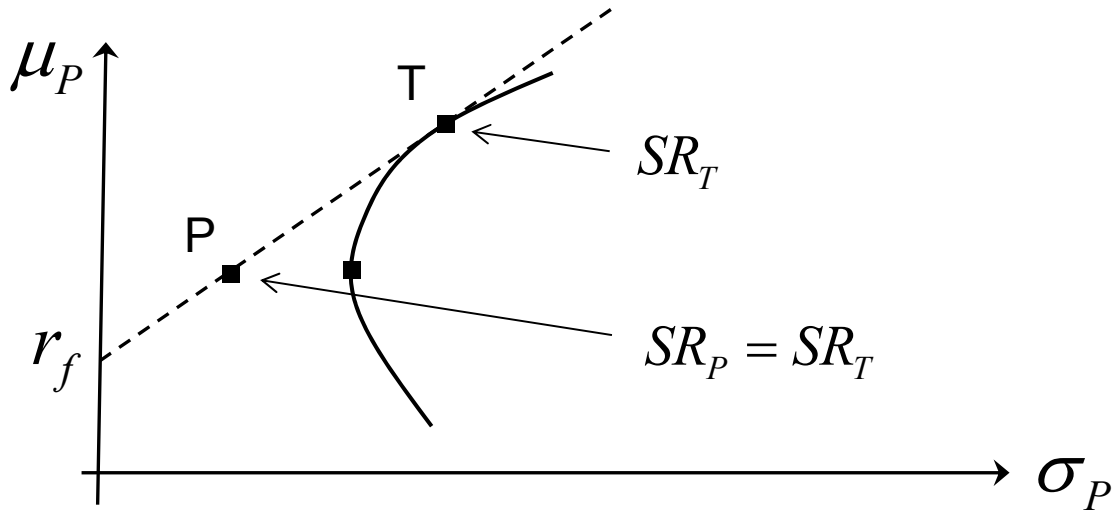
The Two Efficient Frontiers Revisited



THEOREM: If the excess mean return of the GMV portfolio is positive, then the linear efficient frontier with cash and risky assets is tangent to the no-cash efficient frontier at the fully-invested portfolio T, and the weights for T do not depend on investors risk preferences.

Proof: Exercise, including getting the tangent portfolio weights, mean return and volatility.

Sharpe Ratio on Linear Efficient Frontier



The formula for the Sharpe ratio along the linear efficient follows for the mean versus volatility relationship previous derived:

$$\mu_P = r_f + \sqrt{\boldsymbol{\mu}_e' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_e} \cdot \sigma_P \quad \Rightarrow \quad SR_P = \frac{1}{\sqrt{\boldsymbol{\mu}_e' \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_e}}$$

Optimal Risk Aversion Parameter

You can easily get the general form for the risk aversion parameter for a maximum quadratic utility portfolio with cash from the stationary equation that defines the maximum quadratic utility weights:

$$\boldsymbol{\mu}_e = \lambda \boldsymbol{\Sigma} \mathbf{w}_{opt}$$

This gives:

$$\mathbf{w}'_{opt}\boldsymbol{\mu}_e = \lambda \mathbf{w}'_{opt}\boldsymbol{\Sigma}\mathbf{w}_{opt} \quad \Rightarrow \quad \mu_{opt,e} = \lambda \sigma^2_{opt} \quad \Rightarrow \quad \lambda = \frac{1}{\sigma_{opt}} \cdot SR_{opt}$$

Furthermore, under the conditions of the previous Theorem we have that $SR_{opt} = SR_T$ along the linear efficient frontier where T is the tangency portfolio. Thus in that case we have:

$$\lambda = \frac{1}{\sigma_{opt}} \cdot SR_T$$

Mean Returns Sensitivity of Portfolio

Consider the quadratic utility solution optimal weights $\mathbf{w} = \lambda^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ for the case of two assets with mean returns, variances and covariance are given by $\mu_1, \mu_2, \sigma_{11}, \sigma_{22}, \sigma_{12} = \sigma_{21}$ respectively. If you derive the expression for the two weights w_1, w_2 , you can easily compute the derivatives of each of the weights with respect to the two mean returns. So for example:

$$\frac{dw_1}{d\mu_1} = \lambda^{-1} \frac{1}{\sigma_{11} \cdot (1 - \rho^2)}$$

This reveals a very important sensitivity issue for the weights of a MVO portfolio:

The larger the correlation between the assets the greater the sensitivity of the weights to changes in the asset mean returns.

Maximum Sharpe Ratio Portfolio

It is straightforward to find the approximate weights, mean return and volatility of the tangency portfolio, i.e., the maximum Sharpe ratio portfolio of risk assets, once you have computed the efficient frontier at a discrete set of mean returns: just compute the ratio of mean excess return to volatility for each such portfolio and choose the one that has the maximum ratio. In addition to this being only an approximation, it requires computation of a lot of efficient frontier points and for large portfolios this may take a lot of time. This raises the question of whether or not there is a direct way to find the weights, and hence mean return and volatility of the portfolio that maximizes the Sharpe ratio. The answer to this question is positive, and the next slides show how to do this.

We want to solve:

$$\max_{\mathbf{w}} \frac{\mathbf{w}'\boldsymbol{\mu} - r_f}{\sqrt{\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}}} \quad , \qquad \mathbf{w} \in \mathbf{C} \text{ (constraint set)}$$

where $\mathbf{C}$ contains the full-investment constraint $\mathbf{w}'\mathbf{1} = 1$.

Let $$\mathbf{w}'\boldsymbol{\mu} - r_f = \mathbf{w}'\boldsymbol{\mu} - \mathbf{w}'\mathbf{1}r_f = \mathbf{w}'\boldsymbol{\mu}_e$$

and assume there exists some $\tilde{\mathbf{w}}$ such that $\tilde{\mathbf{w}}'\boldsymbol{\mu}_e > 0$, for otherwise there is no point in investing in risky assets. Then for any such weight vector the above problem is equivalent to:

$$\max_{\mathbf{w}} \frac{1}{\sqrt{\dfrac{\mathbf{w}'}{\mathbf{w}'\boldsymbol{\mu}_e}\boldsymbol{\Sigma}\dfrac{\mathbf{w}'}{\mathbf{w}'\boldsymbol{\mu}_e}}} \quad , \qquad \mathbf{w} \in \mathbf{C} \text{ (constraint set)}$$

With $$\mathbf{y} = \frac{\mathbf{w}}{\mathbf{w}'\boldsymbol{\mu}_e}$$

the problem is

$$\max_{\mathbf{y}} \frac{1}{\sqrt{\mathbf{y}'\boldsymbol{\Sigma}\mathbf{y}}} \quad , \qquad \mathbf{y} \in \mathbf{C}_y \text{ (induced } \mathbf{y} \text{ constraints)}$$

Note that for any feasible $\mathbf{w} \in \mathbf{C}$, we have $\mathbf{y}'\boldsymbol{\mu}_e = \frac{\mathbf{w}'\boldsymbol{\mu}_e}{\mathbf{w}'\boldsymbol{\mu}_e} = 1$.

Thus the maximum Sharpe ratio is obtained by:

$$\mathbf{y}_0 = \arg\min_{\mathbf{y}} \mathbf{y}'\boldsymbol{\Sigma}\mathbf{y} \quad \rightarrow \quad \mathbf{w}_{SR} = \frac{\mathbf{y}_0}{\mathbf{y}_0'\mathbf{1}}$$

1.2 ESTIMATES & ESTIMATION ERROR

Mean and Covariance Matrix Estimates

You need estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ of the unknown mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The following estimates are the classical normal distribution maximum likelihood estimates (MLE's).

Row vectors of rturns at time t for n assets

$$\mathbf{r}_t = \left(r_{t1},...,r_{tn}\right)', \quad t = 1,...,T$$

Sample mean returns estimates (assumes stationarity in the mean)

$$\hat{\boldsymbol{\mu}} = \bar{\mathbf{r}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{r}_t' = \left(\bar{r}_1,...,\bar{r}_n\right)', \qquad \bar{r}_i = \frac{1}{T}\sum_{t=1}^{T} r_{ti}$$

Returns covariance matrix estimate (assumes covariance stationarity)

$$\hat{\mathbf{\Sigma}} = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{r_t} - \bar{\mathbf{r}})(\mathbf{r_t} - \bar{\mathbf{r}})'$$

$$\hat{\Sigma}_{ij} = \frac{1}{T}\sum_{t=1}^{T}\left(r_{ti} - \bar{r}_i\right)\left(r_{tj} - \bar{r}_j\right)$$

Exercise 1.1: Check that the covariance matrix estimate $\hat{\mathbf{\Sigma}}$ has as elements the $\hat{\Sigma}_{ij}$ given above.

NOTE: Useful "robust" alternatives to the above normal distribution MLE's are often useful and we will discuss these later on.

Alternative representation of the sample covariance matrix

Returns matrix: $$\mathbf{R} = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{T1} & \cdots & r_{Tn} \end{pmatrix}$$

Sample means matrix: $$\mathbf{1}\bar{\mathbf{r}}' = \begin{pmatrix} \bar{r}_1 & \cdots & \bar{r}_n \\ \vdots & \ddots & \vdots \\ \bar{r}_1 & \cdots & \bar{r}_n \end{pmatrix} \quad \mathbf{1} = (1, 1, \cdots\cdots, 1)' \quad n \times 1$$

Sample covariance matrix: $$\hat{\boldsymbol{\Sigma}} = \frac{1}{T}\left(\mathbf{R} - \mathbf{1}\bar{\mathbf{r}}'\right)'\left(\mathbf{R} - \mathbf{1}\bar{\mathbf{r}}'\right)$$

Regression Version of MVO

$$\min_{\mathbf{w}} \ \mathbf{w}'\hat{\Sigma}\mathbf{w}, \quad \mathbf{w} \in \mathbf{C} \ \text{(linear "inequality" constraints)}$$

Substituting $\hat{\Sigma} = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{r_t} - \bar{\mathbf{r}})(\mathbf{r_t} - \bar{\mathbf{r}})'$ into the above gives:

$$\begin{aligned} T \cdot \mathbf{w}'\hat{\Sigma}\mathbf{w} &= \sum_{t=1}^{T} \mathbf{w}'(\mathbf{r_t} - \bar{\mathbf{r}})(\mathbf{r_t} - \bar{\mathbf{r}})'\mathbf{w} \\ &= \sum_{t=1}^{T} (\mathbf{w}' \cdot (\mathbf{r_t} - \bar{\mathbf{r}}))^2 \\ &= \sum_{t=1}^{T} (\mathbf{w}'\mathbf{r_t} - \bar{\mathbf{r}}_\mathbf{p})^2 \end{aligned}$$

Thus the problem $\min_{\mathbf{w}} \mathbf{w}' \hat{\sum} \mathbf{w}$ is equivalent to:

$$\min_{\mathbf{w}} \sum_{t=1}^{T} (\mathbf{w}'\mathbf{r_t} - \overline{\mathbf{r}}_{\mathbf{p}})^2$$

But it is easy to see that this problem is equivalent to:

$$\min_{\gamma, \mathbf{w}} \sum_{t=1}^{T} (\gamma + \mathbf{w}'\mathbf{r_t})^2$$

For minimization first with respect to γ for fixed $\mathbf{w}$ results in:

$$\sum_{t=1}^{T} (\gamma + \mathbf{w}'\mathbf{r_t}) = 0 \quad \Rightarrow \quad \gamma = -\sum_{t=1}^{T} \mathbf{w}'\mathbf{r_t} = \overline{r}_P$$

Consider the linear regression model:

$$y_t = \gamma + \mathbf{r}_t'\mathbf{w} + \varepsilon_t, \quad t = 1, 2, \cdots, T$$

Where the $\mathbf{r}_t$ are the known n x 1 vectors of asset returns, the portfolio weight vector $\mathbf{w}$ are the unknown regression coefficients and the y_t are response variables. The ordinary least squares (OLS) estimate of the weight vector is the solution of the "normal" equations:

$$\min_{\gamma, \mathbf{w}} \sum_{t=1}^{T} (y_t - \gamma - \mathbf{w}'\mathbf{r}_t)^2$$

Thus we see that the MVO portfolio weights are the OLS solution to a linear regression problem with response variables that are identically zero, $y_t = 0, \quad t = 1, 2, \cdots, T$ and subject to the constraints $\mathbf{w} \in \mathbf{C}$. This allows one to use a wide range of constrained linear regression methods to get MVO weights, e.g., robust regression and penalized regression methods.

Example: Robust Portfolio Optimization

For robustness toward outliers influence:

$$\min_{\gamma,\mathbf{w}} \sum_{t=1}^{T} \rho\left(\frac{\gamma + \mathbf{w}'\mathbf{r_t}}{\hat{s}}\right)$$

Possible choices ρ:

LAD $\rho(t) = |t|$

Huber $\rho_{H,c}(t)$

Optimal Bias Robust $\rho_{YZ,c}(t)$ (Yohai & Zamar,1997)

See Maronna, Martin & Yohai (2006), *Robust Statistics: Theory & Methods*

Portfolio Optimizers are Error Maximizers

"The error maximization effect: the investor over-weights the assets whose mean return is over-estimated, leading to a positive mean return forecast bias. This also leads to a bias toward a higher-risk portfolio: the manager has an exaggerated perception in the marginal increase in expected return per unit of risk and so chooses a portfolio that is riskier than optimal." Connor et. al. (2010).

See Section 2.2 of Connor et. al. for some empirical details.

Bootstrap Efficient Frontiers

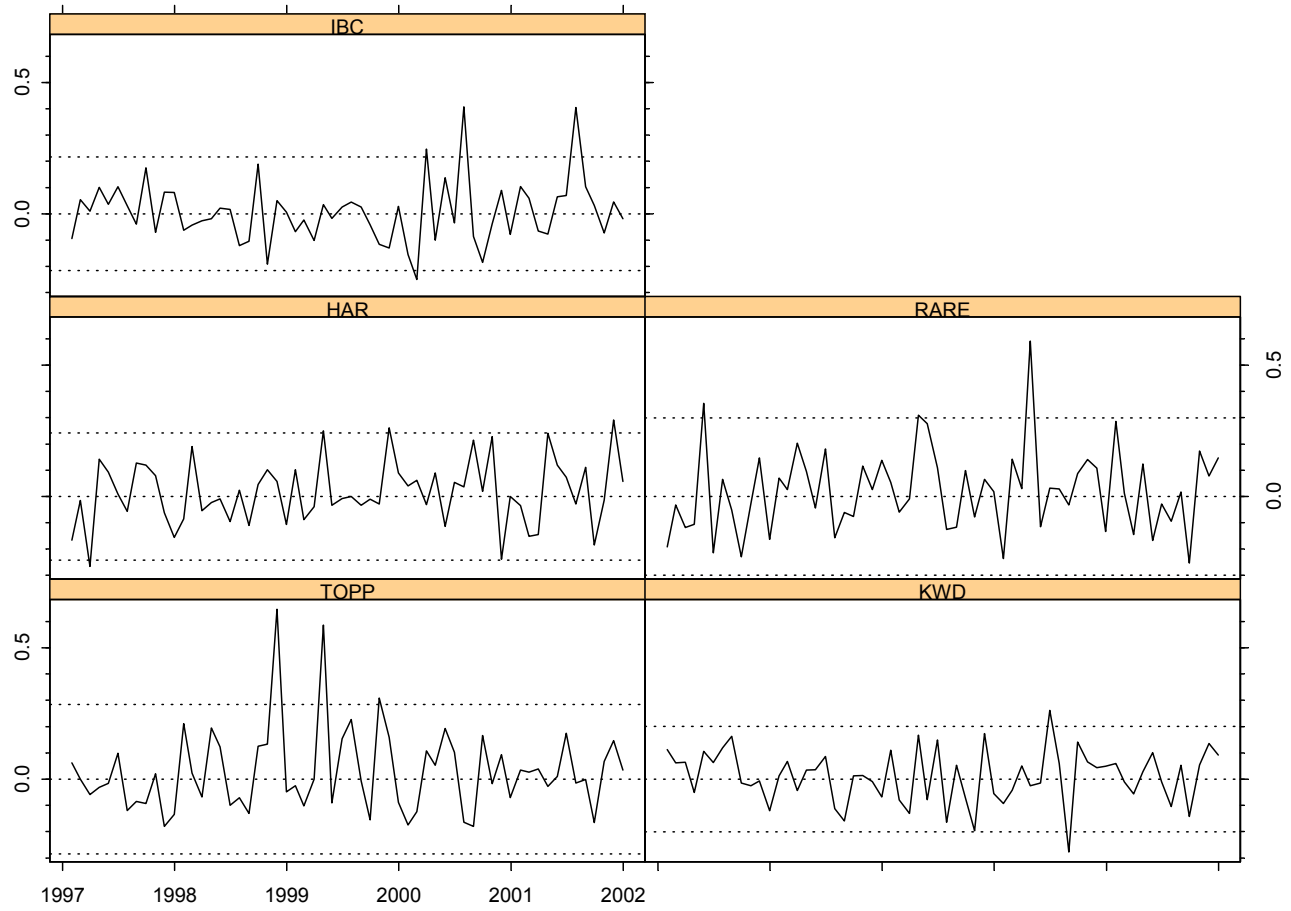
Non-Parametric Bootstrap

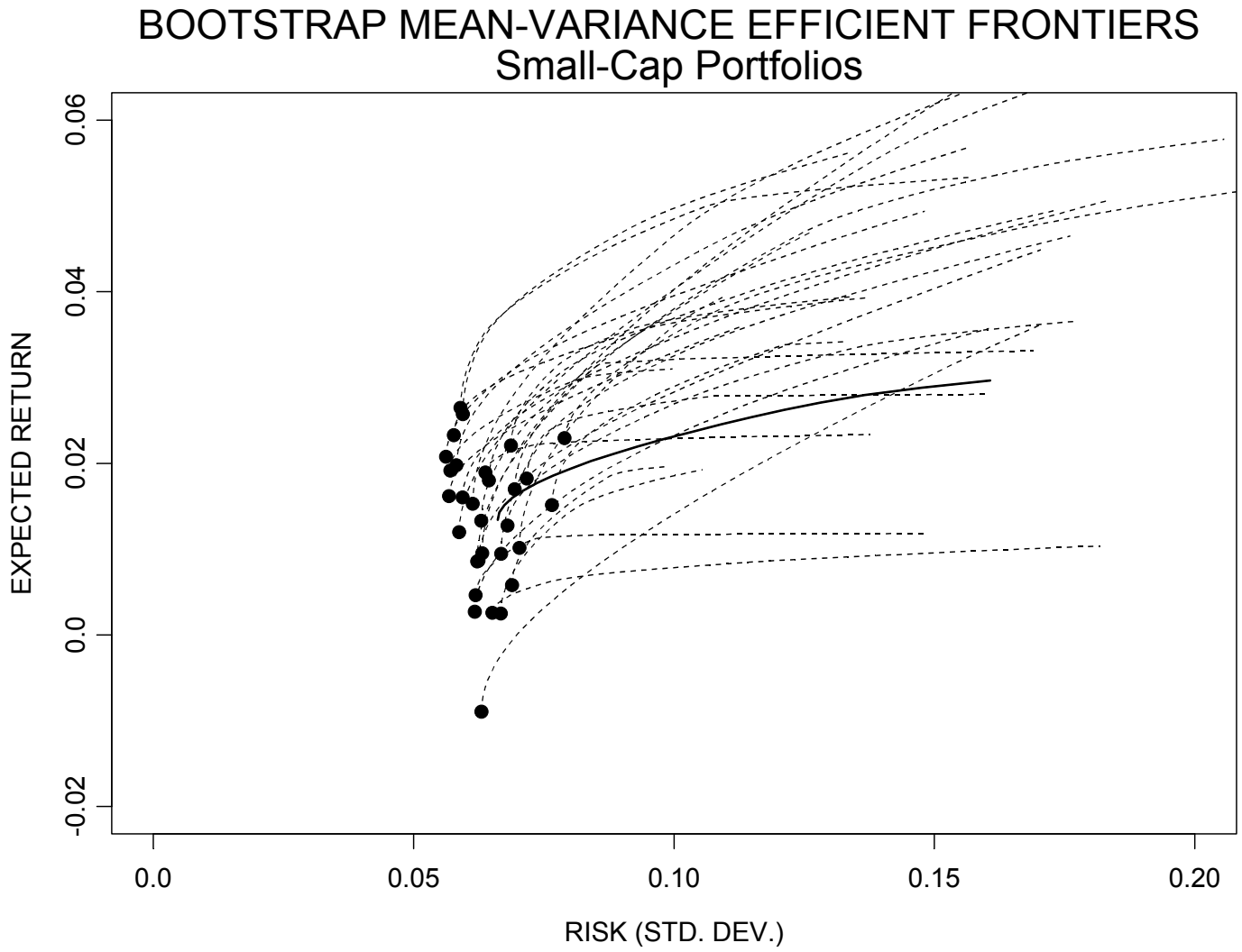
1. Let $\hat{F}_n$ be the *n*-dimensional multivariate empirical distribution of the *T* x *n* table of returns.

2. Do a Monte Carlo simulation of *T* returns from $\hat{F}_n$ to produce a single bootstrap table sample. This is equivalent to taking *T* samples with replacement from the original table. For each such sample compute an efficient frontier.

3. Compute *B* bootstrapped efficient frontiers by repeating the 2nd step *B* times.

Small-Cap Portfolio Example





Things to Notice

1. For the GMV portfolios that don't depend directly on mean return estimates the portfolio volatility variability is relatively much smaller than the portfolio mean return volatility. You don't have much confidence in the latter (it is even negative in one bootstrap sample)

2. As you move up the efficient frontier, where the sample mean return estimate is used, the variability in portfolio mean returns get larger.

NOTE: It will be interesting to study the bootstrap variability of the maximum Sharpe ratio portfolio.

1.3 CONSTRAINED GMV PORTFOLIOS

You will be using the R function **`solve.QP`** in the R package *quadprog* to solve portfolio mean-variance optimization with constraints. **`solve.QP`** is a basic quadratic programming (QP) optimizer that solves quadratic programming problems with constraints. In addition you will use the function **`Rglpk_solve_LP`** from the package *Rglpk*.

NOTE: The R code in this section will be posted to class web site.

Portfolio weights $\mathbf{w} = (w_1, w_2, \cdots, w_n)'$

The weights represent fractions of wealth invested in risky assets.

Asset Mean Return and Covariance Estimates

In the sequel whenever we use the notation $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ we assume that these are data based estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ unless otherwise noted.

Quadratic Programming with `solve.QP`

`solve.QP` solves quadratic programming problems of the following general type where `"'"` denotes the transpose of a vector or matrix.

$$\underset{\mathbf{x}}{\text{minimize}}\left(\frac{1}{2}\mathbf{x}'\mathbf{D}\mathbf{x} - \mathbf{d}'\mathbf{x}\right)$$

$\mathbf{d}$ n x 1 vector

$\mathbf{D}$ n x n matrix

$\mathbf{x}$ n x 1 vector

$$\text{subject to} \qquad \mathbf{A}'\mathbf{x} \geq \mathbf{b}$$

$\mathbf{A}'$ k x n matrix

$\mathbf{b}$ k x 1 vector

```
> library(quadprog)
> args(solve.QP)
function (Dmat,dvec,Amat,bvec,meq = 0,factorized = FALSE)
```

Dmat: the matrix $\mathbf{D}$ in the previous slide
dvec: the vector $\mathbf{d}$ in the previous slide
Amat: the matrix $\mathbf{A}$ in the previous slide
bvec: the vector $\mathbf{b}_{lo}$ in the previous slide
meq: the first meq constraints are treated as equality constraints, and the remainder are inequality constraints
Factorized: factorized logical flag: if TRUE, then we are passing $\mathbf{R}^{-1}$, where $\mathbf{D} = \mathbf{R}'\mathbf{R}$ instead of $\mathbf{D}$ in the argument Dmat

NOTE: Maybe the authors of solve.QP mean $\mathbf{R}'$ instead of $\mathbf{R}^{-1}$? This needs to be checked.

Notation Mapping

$\mathbf{d} \rightarrow \mathbf{0} = n \times 1$ vector of zero's (no linear part)

$\mathbf{D} \rightarrow \hat{\mathbf{\Sigma}} = n \times n$ returns covariance matrix estimate

$\mathbf{x} \rightarrow \mathbf{w} = n \times 1$ vector of portfolios weights

$$\underset{\mathbf{w}}{\text{minimize}} \ \mathbf{w}'\hat{\mathbf{\Sigma}}\mathbf{w}$$

$$\text{subject to} \quad \mathbf{A}'\mathbf{w} \geq \mathbf{b}$$

NOTE 1: The matrix $\mathbf{A}$ will often contain the vector of mean returns estimates $\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, \hat{\mu}_2, \cdots, \hat{\mu}_n)'$. I

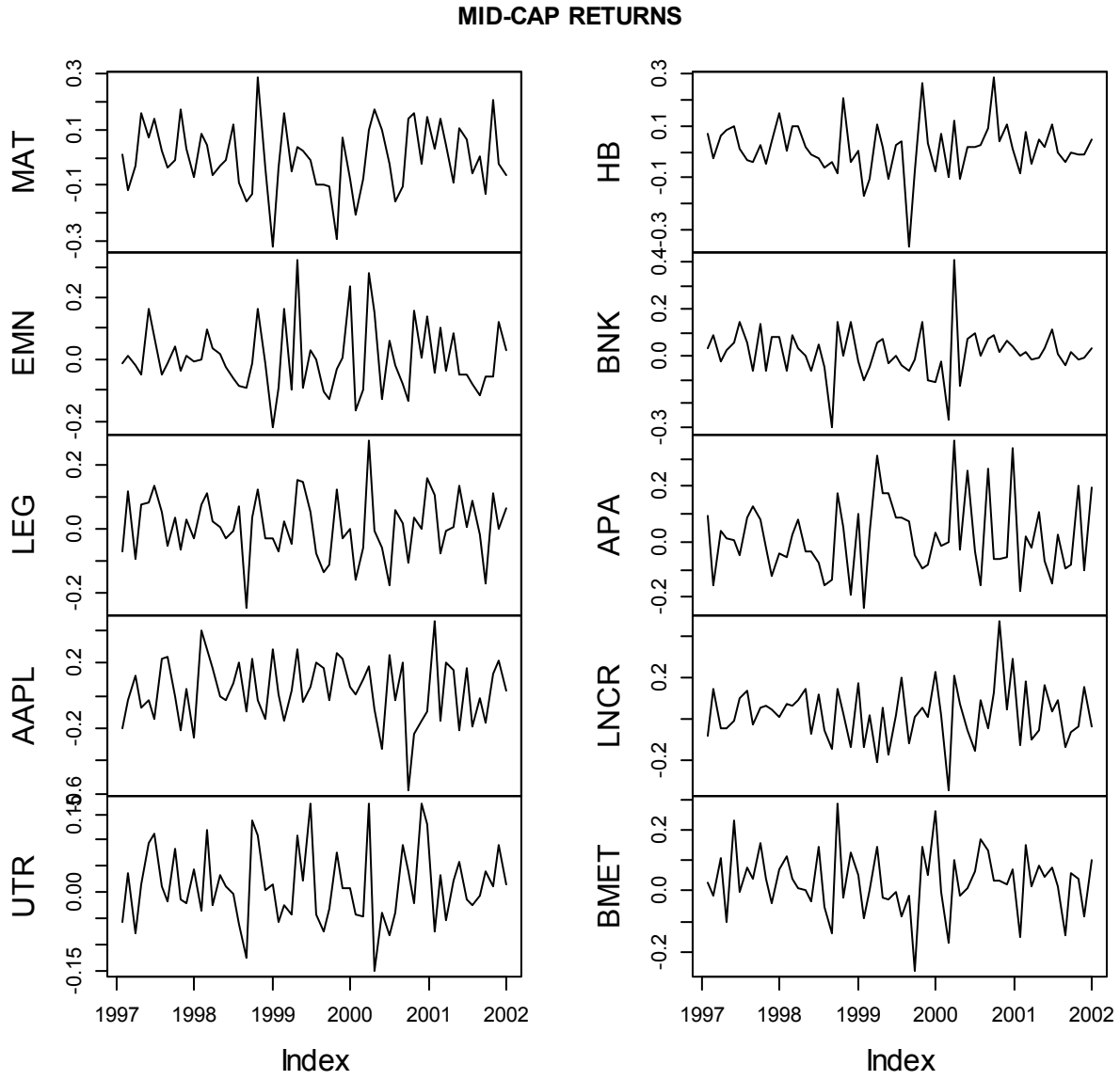
NOTE 2: Often the "hats" for estimates will be omitted for simplicity of notation, and we will still mean estimates unless otherwise noted.

Mid-Cap Returns for Following Examples

The following code loads the libraries and CRSP mid-cap returns data you need for this and makes the plot on the next slide.

```
library(quadprog)
library(xts)
load("C:/Doug/AMath Courses/AMATH
543/Data/crsp.short.Rdata")
returns.ts = midcap.ts[,1:10]
plot.zoo(returns.ts,plot.type = "multiple",main =
              "MID-CAP RETURNS")
returns = coredata(midcap.ts[,1:10])
```

Mid-Cap Returns Data for Following Examples



General Constrained GMV Code

Constraint Builder Function

It will be convenient to have the following simple constraint builder function to use as input to a general global minimum variance (GMV) optimization function

```
constraints = function(A,b,meq)
{
 list(A = A, b = b, meq = meq)
}
```

Example: Fully-invested but no other constraints

```
p = ncol(returns)
A = matrix(rep(1,p),ncol=1)
cset.nc = constraints(A,1,1)
```

General GMV Function

```
# If cset = NULL, then unconstrained gmv
# For back-test use default wts.only = T
# For efficient frontier use wts.only = Fgm
# For  printout of wts, mu and sd use digits = 3 or 4

gmv = function(returns,cset=NULL,wts.only=T,digits = NULL)
{
    V = var(returns)
    mu = apply(returns,2,mean)
    p = ncol(returns)
    d = rep(0,p)    # No linear term in objective
    if(is.null(cset))
    {A = matrix(rep(1,p),ncol =1)
    b = 1
    meq = 1}
    else
    {A = cset$A
    b = cset$b
    meq = cset$meq}
```

```
    port.gmv = solve.QP(V,d,A,b,meq)
    wts = port.gmv$solution      # Get optimal weights
    mu = sum(wts*mu)
    wts = as.matrix(wts)
    sd = as.numeric(sqrt((t(wts)%*%V%*%wts)))
    wts = as.numeric(wts)
    if(!is.null(digits))
    {names(wts)= dimnames(returns)[[2]]
    out = list(WTS = wts,MU.PORT = mu,SD.PORT = sd)
    lapply(out,round,digits)}
    else
    {if(wts.only) wts else mu}
}
```

Ex1.1 Unconstrained GMV Portfolio

```
> gmv(returns,digits = 3)

$WTS
  MAT    EMN   LEG   AAPL    UTR    HB    BNK    APA   LNCR   BMET
 .264 -.059 -.085  .151   0.407 .166 -.039  .016   .034   .145

$MU.PORT
[1] 0.017

$SD.PORT
[1] 0.052
```

Exercise: Confirm the above results using the analytic weights formula.

Ex1.2 Long-Only GMV Portfolio

Require full-investment and no short selling. The constraint matrix $\mathbf{A}$ and lower bound vector $\mathbf{b}$ needed to achieve this are:

$$\mathbf{A}' = \left(\begin{array}{cc|ccc} 1 & 1 & \cdots & 1 & 1 \\ \hline 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{array}\right) \qquad (n+1)\times n$$

$$\mathbf{b} = (1,0,0,\cdots,0)' \qquad (n+1)\times 1$$

And you again use the choice **`meq = 1`**.

The GMV Long-Only Constraint Object:

```
p = ncol(returns)
A = cbind(rep(1,p), diag(rep(1,p))) # Constraint matrix
b = c(1,rep(0,p))                   # Constraint bound
cset.lo = constraints(A,b,1)
cset.lo
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
 [1,]    1    1    0    0    0    0    0    0    0     0     0
 [2,]    1    0    1    0    0    0    0    0    0     0     0
 [3,]    1    0    0    1    0    0    0    0    0     0     0
 [4,]    1    0    0    0    1    0    0    0    0     0     0
 [5,]    1    0    0    0    0    1    0    0    0     0     0
 [6,]    1    0    0    0    0    0    1    0    0     0     0
 [7,]    1    0    0    0    0    0    0    1    0     0     0
 [8,]    1    0    0    0    0    0    0    0    1     0     0
 [9,]    1    0    0    0    0    0    0    0    0     1     0
[10,]    1    0    0    0    0    0    0    0    0     0     1
```

```
 [1] 1 0 0 0 0 0 0 0 0 0 0
```

(solve.QP treats this as a column vector)

```
gmv(returns,cset,F,3)

$WTS
  MAT   EMN    LEG   AAPL   UTR   HB    BNK    APA   LNCR   BMET
 .224  .000   .000  .132   .322  .164  .000   .011   .011  0 136

$MU.PORT
[1] 0.0155

$SD.PORT
[1] 0.0526
```

Note that the long-only constraints results in zero weights, i.e., the constraints are binding, for the stocks EMN, LEG and BNK that had short positions in Ex1.1. Note also that imposing the long-only constraint has resulted in a somewhat smaller portfolio mean return and slightly larger portfolio risk than in the Ex1.1.

GMV with Box Constraints

Investors often prefer additional weights constraints beyond the full-investment and long-only constraints. The simplest such constraints are “box” constraints, consisting of upper and lower bounds on weights. Box constraints are used to help insure diversification and avoid over-concentration.

When using solve.QP you need to cast the upper and lower bounds in terms of lower bounds only. This is quite easy to do by reversing the signs in the A matrix and b vector for those constraints that are upper bounds, as shown on the next slide.

$$\mathbf{A}' = \left(\begin{array}{cc:ccc} 1 & 1 & \cdots & 1 & 1 \\ \hdashline 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ \hdashline -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 0 \\ 0 & 0 & \cdots & 0 & -1 \end{array}\right) \qquad (2n+1)\times n \qquad \mathbf{b} = \left(\begin{array}{c} 1 \\ \hdashline \text{blo}_1 \\ \text{blo}_2 \\ \vdots \\ \vdots \\ \text{blo}_n \\ \hdashline -\text{bup}_1 \\ -\text{bup}_2 \\ \vdots \\ \vdots \\ -\text{bup}_n \end{array}\right) \qquad (2n+1)\times 1$$

Use meq = 1 in solve.QP for the full-investment equality constraint

Ex1.3 GMV with Box Constraints

Let's use a lower bound of 5% and an upper bound of 15%.

```
p = ncol(returns)
A = cbind(rep(1,p), diag(rep(1,p)),diag(rep(-1,p)))
b = c(1,rep(.03,p),rep(-.25,p))
cset.box = constraints(A,b,1)
gmv.constrained(returns,cset.box,F,3)

$WTS
  MAT  EMN  LEG  AAPL  UTR  HB   BNK  APA  LNCR  BMET
 .200 .030 .030 .124  .250 .161 .030 .030  .030  .115
$MU.PORT
[1] 0.0153
$SD.PORT
[1] 0.0537
```

The mean return is just slightly lower and risk is slightly higher than in Ex1.2. The upper constraint is binding for just 1 of the stocks but the lower constraint is binding for 5 of the stocks, 3 of which were binding in Ex1.1.

Group Constraints plus Long-Only

Investors often prefer group constraints in addition the long-only constraints, and possibly group constraints combined with box constraints. For examples you might want market capitalization concentration constraints such as the following.

Market Cap Constraints:

At least 10% and no more than 25% in micro-caps
At least 15% and no more than 35% in small-caps
No more than 35% in mid-caps
No more than 45% in large-caps

You might at the same time want the following box constraints, but we will ignore that for the moment:

Weights Constraints: No stock less than 3% or more than 25%

Constraints Matrix for Groups & Long-Only

$$\mathbf{A}' = \begin{pmatrix} \mathbf{1}_n' \\ \hline \mathbf{I}_n \\ \hline \mathbf{A}_g' \\ \hline -\mathbf{A}_g' \end{pmatrix}$$

$(1+n+2k)\times n$

$\mathbf{1}_n'$ $1\times n$ vector of 1's

$\mathbf{I}_n$ $n\times n$ identity matrix

$\mathbf{A}_g'$ k$\times n$ lower bounds group constraints matrix

$-\mathbf{A}_g'$ k$\times n$ negative upper bounds group constraints matrix

$$\mathbf{b} = \begin{pmatrix} 1 \\ \hline \mathbf{0}_n \\ \hline \mathbf{b}_{g,lo} \\ \hline -\mathbf{b}_{g,up} \end{pmatrix}$$

$(1+n+2k)\times 1$

1 1×1 the numeric value "one"

$\mathbf{0}_n$ $n\times 1$ vector of zero's

$\mathbf{b}_{g,lo}$ $k\times 1$ vector of lower bounds

$-\mathbf{b}_{g,up}$ $k\times 1$ negative vector of upper bounds

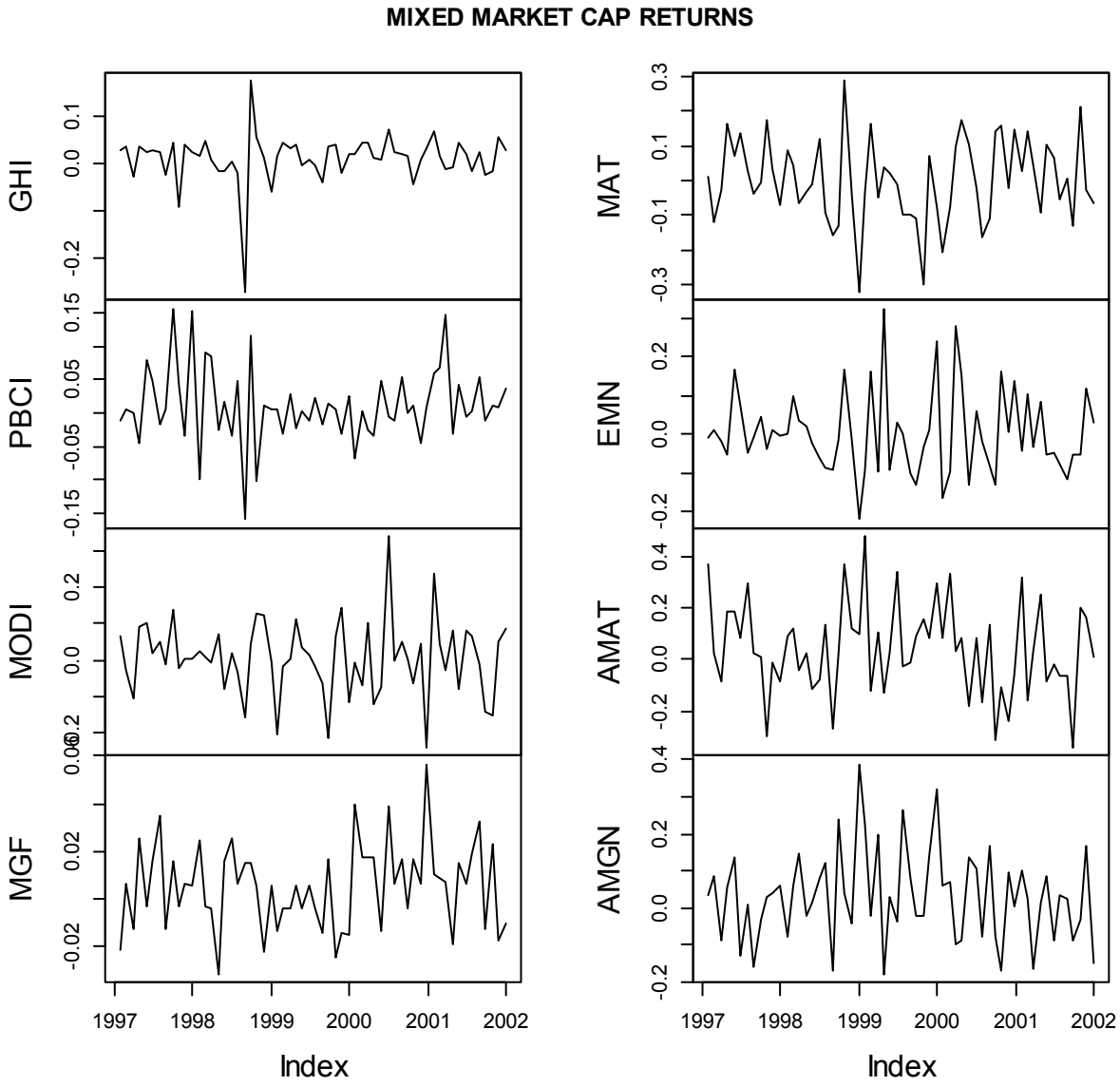
Use a meq = 1 argument in solve.QP for the full-investment constraint

Ex1.4 GMV Group and Long-Only Constraints

By way of an example let's use a portfolio of 8 stocks with 2 from each of the four CRSP data market cap groups, created as follows:

```
> returns8 = cbind(microcap.ts[,1:2],smallcap.ts[,1:2],
+                  midcap.ts[,1:2],largecap.ts[,1:2])

> names(returns8)
[1] "GHI"  "PBCI" "MODI" "MGF"  "MAT"  "EMN"  "AMAT" "AMGN"

> plot.zoo(crsp.port8,plot.type = "multiple",main = "MIXED
MARKET CAP RETURNS")
```

Micro-caps:	GHI	PBCI
Small-caps:	MODI	MGF
Mid-caps:	MAT	EMN
Large-caps:	AMAT	AMGN


MIXED MARKET CAP RETURNS
GHI
PBCI
MODI
MGF
MAT
EMN
AMAT
AMGN
1997
1998
1999
2000
2001
2002
Index

Example Constraints Matrix and Vector

At least 10% and no more than 25% in micro-caps
At least 15% and no more than 35% in small-caps
No more than 35% in mid-caps
No more than 45% in large-caps

$$\mathbf{A}_g' = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{b}_{g,lo}' = (.1,\ .15,\ 0,\ 0)$$

$$-\mathbf{b}_{g,up} = (-.25,\ -.35,\ -.35,\ -.45)$$

```
p = ncol(returns)
Ag = rbind(c(1,1,0,0,0,0,0,0),c(0,0,1,1,0,0,0,0),
           c(0,0,0,0,1,1,0,0),c(0,0,0,0,0,0,1,1))
Atp = rbind(rep(1,p), diag(rep(1,p)),Ag,-Ag)
A = t(Atp)
bglo = c(.1,.15,0,0)
bgup = c(.25,.35,.35,.45)
b = c(1,rep(0,p),bglo,-bgup)
cset.groups = constraints(A,b,1)
gmv(returns,cset.groups,F,4)

$WTS
   GHI    PBCI    MODI     MGF     MAT     EMN    AMAT    AMGN
0.0157 0.2343 0.0000 0.3500 0.1084 0.1309 0.0208 0.1399

$MU.PORT
[1] 0.0101

$SD.PORT
[1] 0.0306
```

If in addition you may want to see the total allocation to each of the four market cap groups you can easily do it with the following code:

```
wts = gmv(returns,cset)
wts.groups = wts[c(1,3,5,7)]+wts[c(2,4,6,8)]
names(wts.groups) = c("Micro","Small","Mid","Large")
round(wts.groups,3)

Micro  Small   Mid   Large
0.250  0.350  0.239  0.161
```

Note that the upper bound constraint is binding for the micro-cap and small-cap groups, but not for mid-cap and large-cap groups.

The Constraints Objects and Returns

It will be convenient for the remainder of the computations in this lecture to create the three constraint objects in an R script named **`constraint.sets.r`** that can be sourced, and this script will be posted at the web site. Note that the constraint sets are portfolio specific to a lessor or greater degree. In the case of long-only and box constraints the only aspect of the portfolio that is needed is the number of returns. But the groups constraint object requires the group structure of the assets as well as the number of assets. Thus the **`constraint.sets.r`** imports CRSP data and creates the data objects returns and returns8 as well as creating the constraint objects.

1.4 CONSTRAINED MAX MEAN RETURN

This section is a preparation for computing efficient frontiers with constraints. In order to compute an efficient frontier with constraints you need to know what the range of target mean returns is feasible. The mean return of a constrained GMV, which we now know how to compute, provides a natural minimum mean return to use in computing an efficient frontier. But the maximum mean return under constraints needs to be computed numerically except in the special case of a long-only portfolio. In this section we show how to do that. But we first note in passing on the next slide how easy it is to get the maximum mean return of a long-only portfolio.

Maximum Mean Return for Long-Only Portfolios

If the only constraint is a long-only constraint then you can easily see that the largest achievable return is the mean return of the asset having the highest mean return. For the mid-cap returns of the previous section the output below reveals this to be .0324:

```
round(mu,3)
 MAT   EMN   LEG   AAPL   UTR   HB    BNK    APA   LNCR   BMET
.000  .003  .011  .032   .013  .014  .015   .016   .026   .031
round(max(mu),4)
[1] 0.0324
```

However, it is convenient to deal with this special case in a general constraints framework as in the following slides.

Constrained Maximum Mean Return

The problem is now the following pure linear programming special case of the constrained quadratic programming problem, where $\hat{\boldsymbol{\mu}}$ is the vector of estimated mean asset returns:

$$\underset{\mathbf{w}}{\text{maximize}}\ \hat{\boldsymbol{\mu}}'\mathbf{w} = \underset{\mathbf{w}}{\text{minimize}}\left(\frac{1}{2}\mathbf{w}'\mathbf{0}\mathbf{w} - \hat{\boldsymbol{\mu}}'\mathbf{w}\right)$$

$$\text{subject to} \qquad \mathbf{A}'\mathbf{x} \geq \mathbf{b}$$

It would be nice if we could continue to use solve.QP with $\mathbf{D} = \mathbf{0}$, but unfortunately this does not work. So instead we use the pure linear programming package Rlpgk.

Rglpk_solve_LP Formulation

The Rglpk_solve_LP function in the R package Rglpk solves the general linear programming problem

$$\underset{\mathbf{x}}{\text{maximize}} \ \mathbf{d}'\mathbf{x}$$

$$\text{subject to} \qquad \mathbf{A}\mathbf{x} \sim \mathbf{b}$$

Where the symbol " $\sim$ " indicates an arbitrary mix of inequality or equality constraints on a constraint by constrain basis (see next slide)

Note that the constraint matrix is now $\mathbf{A}$ not $\mathbf{A}'$! For our constrained maximum mean portfolio return problem the above becomes:

$$\underset{\mathbf{w}}{\text{maximize}} \ \hat{\boldsymbol{\mu}}'\mathbf{w}$$

$$\text{subject to} \qquad \mathbf{A}\mathbf{w} \sim \mathbf{b}$$

```
> library(Rglpk)
> args(Rglpk_solve_LP)
function (obj, mat, dir, rhs, types = NULL, max = FALSE,
          bounds = NULL, verbose = FALSE)
```

obj: vector of objective coefficients
mat: constraints matrix
dir: gt, gte, lt, lte, eq for each constraint (a character string of these)
rhs: vector of constraints bounds
types: type of variables (continuous, integer, binary)
max: TRUE results in maximization, FALSE in minimization
bounds: box constraints on variables
verbose: FALSE gives minimal output, TRUE gives maximum output

In terms of the previous matrix and vector notation:

obj = $\mathbf{d}$
mat = $\mathbf{A}$
rhs = $\mathbf{b}$

Constrained Max Mean Return Code

We note that the constraint specification requirements of **`Rglpk_solve_LP`** are somewhat different than those of **`solve.QP`**. Rather than create a separate constraint builder object for the former, one can simply automatically convert the **`solve.QP`** constraint object into the needed **`Rglpk_solve_LP`** constraint object inside the general maximum mean return code on the next slide. The lines of the code that do the needed conversion based on providing the **`solve.QP`** cset object as an input argument are as follows:

```
A = cset$A
k = nrow(A)-1
dir = c("==",rep(">=",k))
b = cset$b
dir = cset$dir
```

General Max Mean Return Function

```
# Default is to return only the constrained max mean return
# For weights, mean & std. dev. choose digits = 3 or 4
# mu.only = FALSE results in returning only the weights

maxmu = function(returns,cset, mu.only=TRUE, digits=NULL,
                 verbose = FALSE)
{
    V = var(returns) # Only need for risk calculation
    mu = apply(returns,2,mean)
    d = mu
    A = t(cset$A)  # Because solve.QP uses transpose of A
    k = nrow(A)-1
    dir = c("==",rep(">=",k))
    b = cset$b
    port.maxmu = Rglpk_solve_LP(d,A,dir,b,max = TRUE,
                                verbose = verbose)
    wts = port.maxmu$solution      # Get optimal weights
```

```
mu = sum(wts*mu)
wts = as.matrix(wts)
sd = as.numeric(sqrt((t(wts)%*%V%*%wts)))
wts = as.numeric(wts)
if(!is.null(digits))
{names(wts)= dimnames(returns)[[2]]
out = list(WTS = wts,MU.PORT = mu,SD.PORT = sd)
lapply(out,round,digits)}
else
{if(mu.only) mu else wts}
```

Ex1.5 Max Mean Return Long-Only

```
library(Rglpk)
source("mvo.constrained.r") # loads maxmu function
source("constraint.sets.r") # also loads returns, returns8

# Compute long-only constrained maximum mean return
maxmu(returns,cset.lo)
[1] 0.03236464    # As expected, see slide 68

# Return all output
maxmu(returns,cset.lo,digits = 3)
$WTS
 MAT  EMN  LEG AAPL  UTR   HB  BNK  APA LNCR BMET
   0    0    0    1    0    0    0    0    0    0

$MU.PORT
[1] 0.032

$SD.PORT
[1] 0.193
```

Max Mean Return with Box Constraints in Rglpk

A natural way to use Rglpk_solve_LP for this is as below.

$$\mathbf{A} = \left(\begin{array}{cc:ccc} 1 & 1 & \cdots & 1 & 1 \\ \hdashline 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ \hdashline 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{array}\right) \quad (2n+1)\times n$$

$$\text{rhs: } \mathbf{b}_0 = \left(\begin{array}{c} 1 \\ \hdashline \text{blo}_1 \\ \text{blo}_2 \\ \vdots \\ \vdots \\ \text{blo}_n \\ \hdashline \text{bup}_1 \\ \text{bup}_2 \\ \vdots \\ \vdots \\ \text{bup}_n \end{array}\right) \quad (2n+1)\times 1$$

and the Rglpk_solve_LP argument $\text{dir} = \text{c}(\text{"}==\text{"}, rep(\text{"}>=\text{"}, n), rep(\text{"}<=\text{"}, n))$

However, doing it as below results in being able to use the transpose of the constraint matrix used in **`solve.QP`**.

$$\mathbf{A} = \left(\begin{array}{cc|ccc} 1 & 1 & \cdots & 1 & 1 \\ \hline 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ \hline -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 0 \\ 0 & 0 & \cdots & 0 & -1 \end{array}\right) \quad (2n+1) \times n \qquad \mathbf{b} = \left(\begin{array}{c} 1 \\ \hline \text{blo}_1 \\ \text{blo}_2 \\ \vdots \\ \vdots \\ \text{blo}_n \\ \hline -\text{bup}_1 \\ -\text{bup}_2 \\ \vdots \\ \vdots \\ -\text{bup}_n \end{array}\right) \quad (2n+1) \times 1$$

and the Rglpk_solve_LP argument dir: $\text{c}(\text{"}==\text{"}, rep(\text{"}>=\text{"}, 2*n))$

Ex1.6 Max Mean Return Box Constraints

```
maxmu(returns,cset.box)
[1] 0.02520268


# For all output:
> maxmu(returns,cset.gmvbox,digits = 3)
$WTS
 MAT  EMN  LEG AAPL  UTR   HB  BNK  APA LNCR BMET
0.03 0.03 0.03 0.25 0.03 0.03 0.03 0.07 0.25 0.25

$MU.PORT
[1] 0.025

$SD.PORT
[1] 0.074
```

Note that all constraints are binding.

Ex1.7 Max Mean Return Group Constraints

```
maxmu(returns8,cset.groups)
[1] 0.02311682


# For all output:
maxmu(returns8,cset.groups,F,3)
$WTS
 GHI PBCI MODI  MGF  MAT  EMN AMAT AMGN
0.25 0.00 0.30 0.00 0.00 0.00 0.45 0.00

$MU.PORT
[1] 0.023

$SD.PORT
[1] 0.098
```

1.5 MEAN RETURN CONSTRAINED MVO

Now that we know how to compute the maximum and minimum mean returns achievable under the constraints imposed, we can compute the minimum variance portfolio for any specified mean return in between these two limits. We illustrate the general method for the case of a long-only portfolio below where we use the choice **`meq = 2`** in **`solve.QP`**.

$$\mathbf{A}' = \left(\begin{array}{ccc|ccc} \hat{\mu}_1 & \hat{\mu}_2 & \cdots & \hat{\mu}_{n-1} & \hat{\mu}_n \\ 1 & 1 & \cdots & 1 & 1 \\ \hline 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) \quad (n+2)\times n$$

Important: This is the way to do it in general, i.e., append the vector of estimated mean returns in the first row.

$$\mathbf{b}_{lo} = (\mu_0, 1, 0, 0, \cdots, 0)' \qquad (n+2)\times 1$$

General Constrained MVO Function

```
# If cset = NULL, then unconstrained mvo
# For back-test use default wts.only = T
# For efficient frontier use wts.only = F
# For  printout of wts, mu and sd use digits = 3 or 4

mvo = function(returns,mu0,cset=NULL,wts.only=T,digits =
NULL)
{
    V = var(returns)
    mu = apply(returns,2,mean)
    p = ncol(returns)
    d = rep(0,p)     # No linear term in objective
    if(is.null(cset))
    {A = cbind(mu,rep(1,p))
    b = c(mu0,1)
    meq = 2}
    else
```

```
    {A = cset$A
    A = cbind(mu,A)
    b = cset$b
    b = c(mu0,b)
    meq = cset$meq + 1}
    port.gmv = solve.QP(V,d,A,b,meq)
    wts = port.gmv$solution      # Get optimal weights
    mu = sum(wts*mu)
    wts = as.matrix(wts)
    sd = as.numeric(sqrt((t(wts)%*%V%*%wts)))
    wts = as.numeric(wts)
    if(!is.null(digits))
    {names(wts)= dimnames(returns)[[2]]
    out = list(WTS = wts,MU.PORT = mu,SD.PORT = sd)
    lapply(out,round,digits)}
    else
    {if(wts.only) wts else c(mu,sd,wts)}
}
```

Ex1.8 Long-Only Portfolio with 2.5% Mean Return

Recall from Ex1.2 that the mean return of the GMV long-only portfolio was .015 and the maximum long-only mean return was .032. So let's use .025 as the mean return constraint.

```
library(Rglpk)
source("mvo.constrained.r") # loads mvo function
source("constraint.sets.r") # also loads returns, returns8

# Compute long-only constrained maximum mean return
mvo(returns,.025,cset.lo, digits = 3)

$WTS
  MAT   EMN   LEG  AAPL  UTR  HB   BNK   APA  LNCR  BMET
 .065 .000  .000 .182  .088 .114 .000  .006  .142  .405

$MU.PORT
[1] 0.025

$SD.PORT
[1] 0.067  # Higher than GMV and lower than maxmu portfolio
```

Ex1.9 Box Constraints and 2% Mean Return

Recall from Ex1.3 that with box constraints the mean return of the GMV portfolio was .0153 and the maximum mean return was .025. So let's use .02 as the mean return constraint.

```
# Compute long-only constrained maximum mean return
mvo(returns,.02,cset.box, digits = 3)

$WTS
  MAT   EMN   LEG   AAPL   UTR   HB    BNK   APA   LNCR   BMET
 .113  .030  .030  .152   .146  .138  .030  .030   .081   .250

$MU.PORT
[1] 0.02

$SD.PORT
[1] 0.058
```

←— Slightly higher than the GMV risk of .56 and lower than the maximum mean return portfolio risk of .065.

Ex1.10 Group Constraints and 1.5% Mean Return

Recall from Ex1.4 that with group constraints the mean return of the GMV portfolio was .0101 and from Ex1.7 the maximum mean return was .023. So let's use .015 as the mean return constraint.

```
# Compute long-only constrained maximum mean return
mvo(returns,.015,cset.groups, digits = 3)

$WTS
  GHI   PBCI   MODI    MGF    MAT    EMN   AMAT   AMGN
0.043 0.207 0.000 0.350 0.052 0.049 0.101 0.197

$MU.PORT
[1] 0.015

$SD.PORT
[1] 0.039
```

1.6 MAXIMIZING QUADRATIC UTILITY

The preceding solutions in this section can be obtained by maximizing quadratic utility

$$\mathbf{w}'\boldsymbol{\mu} - \frac{\lambda}{2} \cdot \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}$$

under the various constraints. When using **`solve.QP`** to solve

$$\underset{\mathbf{x}}{\text{minimize}} \left(\frac{1}{2}\mathbf{x}'\mathbf{D}\mathbf{x} - \mathbf{d}'\mathbf{x} \right)$$

$$\text{subject to} \qquad \mathbf{A}'\mathbf{x} \geq \mathbf{b}_{lo}$$

$\mathbf{d}$ n x 1 vector

$\mathbf{D}$ n x n matrix

$\mathbf{x}$ n x 1 vector

Just set $\mathbf{D} = \lambda \cdot \boldsymbol{\Sigma}$, $\mathbf{d} = \boldsymbol{\mu}$ and $\mathbf{x} = \mathbf{w}$.

Equivalently set $\mathbf{D} = \boldsymbol{\Sigma}$ and $\mathbf{d} = \lambda^{-1} \cdot \boldsymbol{\mu}$.

QU Maximization R Code

```
# For back-test use default wts.only = T
# For efficient frontier use wts.only = F
# For  printout of wts, mu and sd use digits = 3 or 4
qu = function(returns,cset=NULL,lambda,wts.only=T,
              digits = NULL)
{
    require(quadprog)
    V = var(returns)
    mu = apply(returns,2,mean)
    p = ncol(returns)
    d = mu/lambda
    if(is.null(cset))
    {A = cbind(rep(1,p))
    b = 1
    meq = 1}
    else
    {A = cset$A
    b = cset$b
    meq = cset$meq}
```

```
port.mvo = solve.QP(V,d,A,b,meq)
    wts = port.mvo$solution      # Get optimal weights
    mu = sum(wts*mu)
    wts = as.matrix(wts)
    sd = as.numeric(sqrt((t(wts)%*%V%*%wts)))
    wts = as.numeric(wts)
    if(!is.null(digits))
    {names(wts)= dimnames(returns)[[2]]
    out = list(WTS = wts,MU.PORT = mu,SD.PORT = sd)
    lapply(out,round,digits)}
    else
    {if(wts.only) wts else c(mu,sd,wts)}
}
```

Ex1.11 Approximate GMV and Max Mu

Compare the following results with Ex1.2 and Ex1.5.

```
qu(returns,cset.lo,20,digits = 4)
$WTS
  MAT    EMN   LEG    AAPL  UTR     HB    BNK    APA    LNCR  BMET
0.222 0.000 0.000 0.133 0.320 0.163 0.000 0.011 0.012 0.139

$MU.PORT
[1] 0.016

$SD.PORT
[1] 0.053

qu(returns,cset.lo,.01,digits = 4)
$WTS
 MAT   EMN   LEG AAPL   UTR    HB   BNK   APA LNCR BMET
   0     0     0     1     0     0     0     0     0     0

$MU.PORT
[1] 0.032

$SD.PORT
[1] 0.193
```

1.7 R CODE FOR EXAMPLES

We now have a basic set of tools to carry out the computations in Section 1.3 through 1.6 as follows:

1. A **`constraints`** function and a set of constraint objects
2. A **`gmv`** function to compute the global minimum variance portfolio
3. A **`minmu`** function to compute mean return of the GMV portfolio
4. A **`maxmu`** function to compute the maximum mean return
5. An **`mvo`** function to minimize variance at a specified mean return

The above functions are all provided in **`mvo.constrained.r`**, which you should look at, note optional argument usage comments, and source for carrying out constrained MVO.

Some additional scripts and functions are useful in carrying out the computations in Sections 1.3 through 1.6, and for computing efficient frontiers, as described in the next slides.

constraints.set.r

```
library(xts)
load("C:/Doug/AMATH Courses/AMATH 543/Data/
                                    crsp.short.Rdata")
returns.ts = midcap.ts[,1:10]
returns = coredata(midcap.ts[,1:10])
returns8 = cbind(microcap.ts[,1:2],smallcap.ts[,1:2],
                         midcap.ts[,1:2],largecap.ts[,1:2])
returns8 = coredata(returns8)

# Long-Only Constraints
p = ncol(returns)
A = cbind(rep(1,p), diag(rep(1,p))) # Constraint matrix
b = c(1,rep(0,p))                   # Constraint bound
cset.lo = constraints(A,b,1)

# Box Constraints
p = ncol(returns)
A = cbind(rep(1,p), diag(rep(1,p)),diag(rep(-1,p)))
b = c(1,rep(.03,p),rep(-.25,p))
cset.box = constraints(A,b,1)
```

```
# Group Constraints
p = ncol(returns8)
Ag = rbind(c(1,1,0,0,0,0,0,0),c(0,0,1,1,0,0,0,0),
+           c(0,0,0,0,1,1,0,0),c(0,0,0,0,0,0,1,1))
Atp = rbind(rep(1,p), diag(rep(1,p)),Ag,-Ag)
A = t(Atp)
bglo = c(.1,.15,0,0)
bgup = c(.25,.35,.35,.45)
b = c(1,rep(0,p),bglo,-bgup)
cset.groups = constraints(A,b,1)
```

NOTE 1: Sourcing **constraint.set.r** results in loading the data needed for the computations as well as creating the constraint objects.

NOTE 2: The scripts **minmu.constrained.r** **maxmu.constrained.r** and **mean.constrained.r** on the next slides create Ex1.1 through Ex1.10 in the previous sections.

minmu.examples.r

```
# Ex1.1 GMV No Constraints
gmv(returns,digits = 3)
minmu(returns)

# Ex1.2 GMV long-only
gmv(returns,cset.lo,F,3)
minmu(returns,cset.lo)

#Ex1.3 GMV box constraints
gmv(returns,cset.box,F,4)
minmu(returns,cset.box)

# Ex1.4 GMV Group Constraints and Long-Only
gmv(returns8,cset.groups,F,4)
minmu(returns8,cset.groups)

# Group weights
wts = gmv(returns8,cset.groups)
wts.groups = wts[c(1,3,5,7)]+wts[c(2,4,6,8)]
names(wts.groups) = c("Micro","Small","Mid","Large")
round(wts.groups,3)
```

maxmu.examples.r

```
library(Rglpk)
library(xts)
source("mvo.constrained.r")
source("constraint.sets.r")

# Ex1.5 Maximum mean long-only
maxmu(returns,cset.lo)
maxmu(returns,cset.lo,digits =3)

#Ex1.6 Max Mean Return box constraints
maxmu(returns,cset.box)
maxmu(returns,cset.box,digits = 3)

# Ex1.7 GMV Group Constraints and Long-Only
maxmu(returns8,cset.groups)
maxmu(returns8,cset.groups,digits = 3)
```

mean.constrained.examples.r

```
library(Rglpk)
library(xts)
source("mvo.constrained.r")
source("constraint.sets.r")

# Ex1.8 MVO Long-Only and 2.5% Mean Return
mvo(returns,.025,cset.lo,digits = 3)

# Ex1.9 MVO With Box Constraints and 2% Mean Return
mvo(returns,.02,cset.box,digits = 3)

# Ex1.10 MVO With Group Constraints and 1.85% Mean Return
mvo(returns8,.015,cset.groups,digits = 3)
```

1.8 EFFICIENT FRONTIERS

With the code of the previous section in hand we will be able to compute and plot efficient frontiers with constraints by creating the following:

1. A wrapper function to compute a set of constrained MVO values along an efficient frontier
2. A flexible plotting function with various options

The function is as follows, with code on next slide:

```
efrontMV(returns, cset = NULL, npoints = 10)
```

The second function is:

```
efrontPlot(returns,c set = NULL, minmu = NULL,
          maxmu = NULL, rf = NULL, npoints = 10,
       wts.plot = T, printout = F, bar.ylim = c(0,2))
```

efrontPlot is long and is posted at the class web site. Note that it uses a special **barplot.wts** function to handle negative weights.

```
efrontMV = function(returns,cset = NULL,npoints = 10)
{
    if(is.null(cset))
    {mu = apply(returns,2,mean)
    minmu = min(mu)
    maxmu = max(mu)}
    else
    {minmu = minmu(returns,cset)
    maxmu = .999*maxmu(returns,cset)}
    p = ncol(returns)
    efront = matrix(rep(0,npoints*(p+2)),ncol = p+2)
    muvals = seq(minmu,maxmu,length.out = npoints)
    for(i in 1:npoints)
    {efront[i,] = mvo(returns,muvals[i],cset,wts.only = F)}
    dimnames(efront)[[2]]=c("MU","SD",dimnames(returns)[[2]])
    efront
}
```

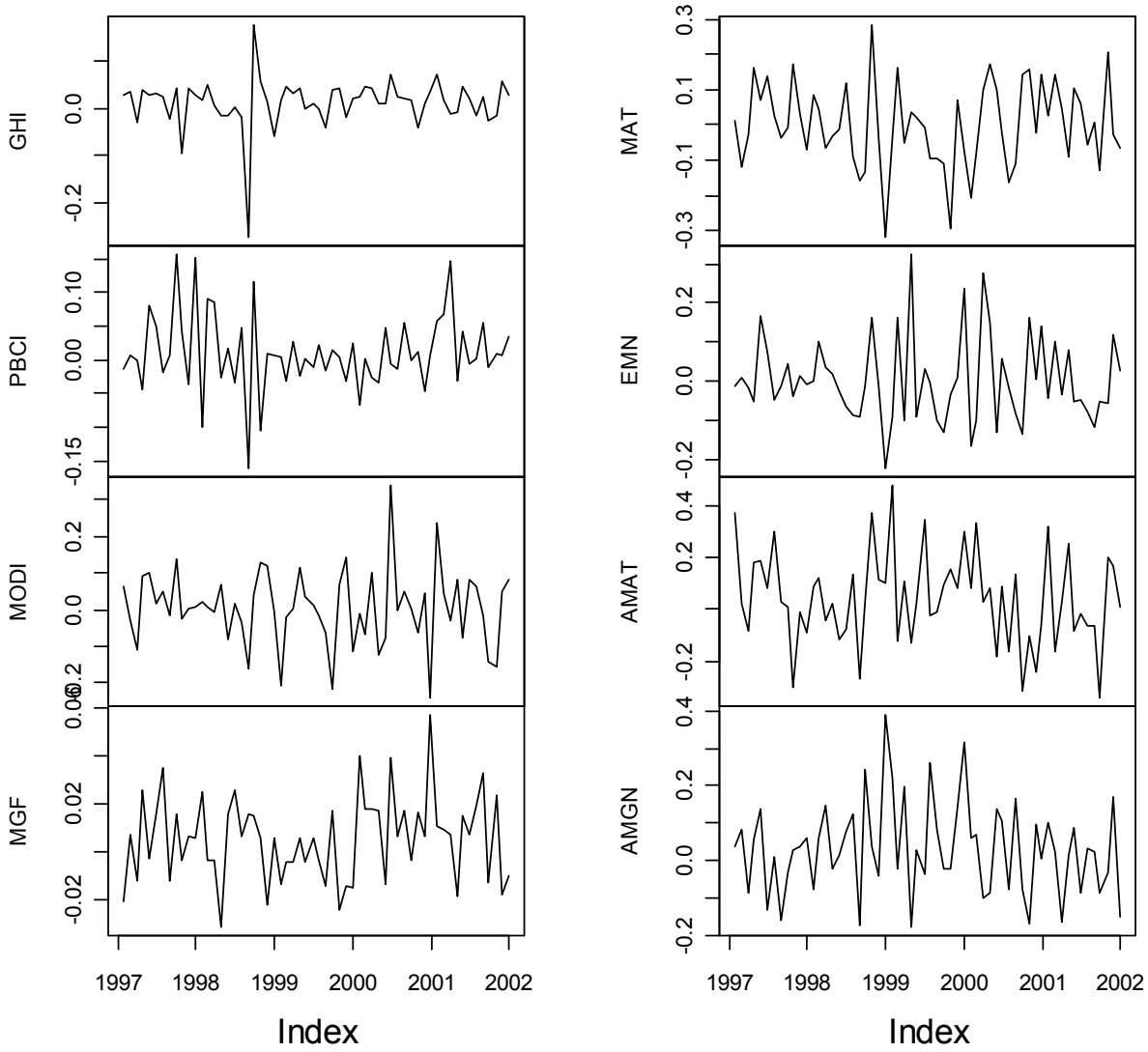
Special Barplot Function for Negative Weights

```
barplot.wts = function(x,legend.text = NULL,col = NULL,ylab =
                       NULL ,xlab = NULL, bar.ylim = NULL)
{
    n = ncol(x); p = nrow(x)
    xpos = (abs(x)+x)/2
    xneg = (x-abs(x))/2
    if(is.null(ylim))
    {ymax <- max(colSums(xpos,na.rm=T))
    ymin <- min(colSums(xneg,na.rm=T))
    ylim = c(ymin,ymax)}
    else {ylim = ylim}
    barplot(xpos,legend.text = legend.text,col = col,ylab =
            ylab,xlab = xlab, ylim = bar.ylim)
    barplot(xneg,add = T,col = col)
    abline(h=0)
}
x = cbind(c(.1,.3,.8,-.2),c(.2,.3,.2,.3))
barplot.wts(x,legend.text = T, col = topo.colors(4),ylab =
              "WEIGHTS", xlab = "VOL",bar.ylim = c(0,2))
```

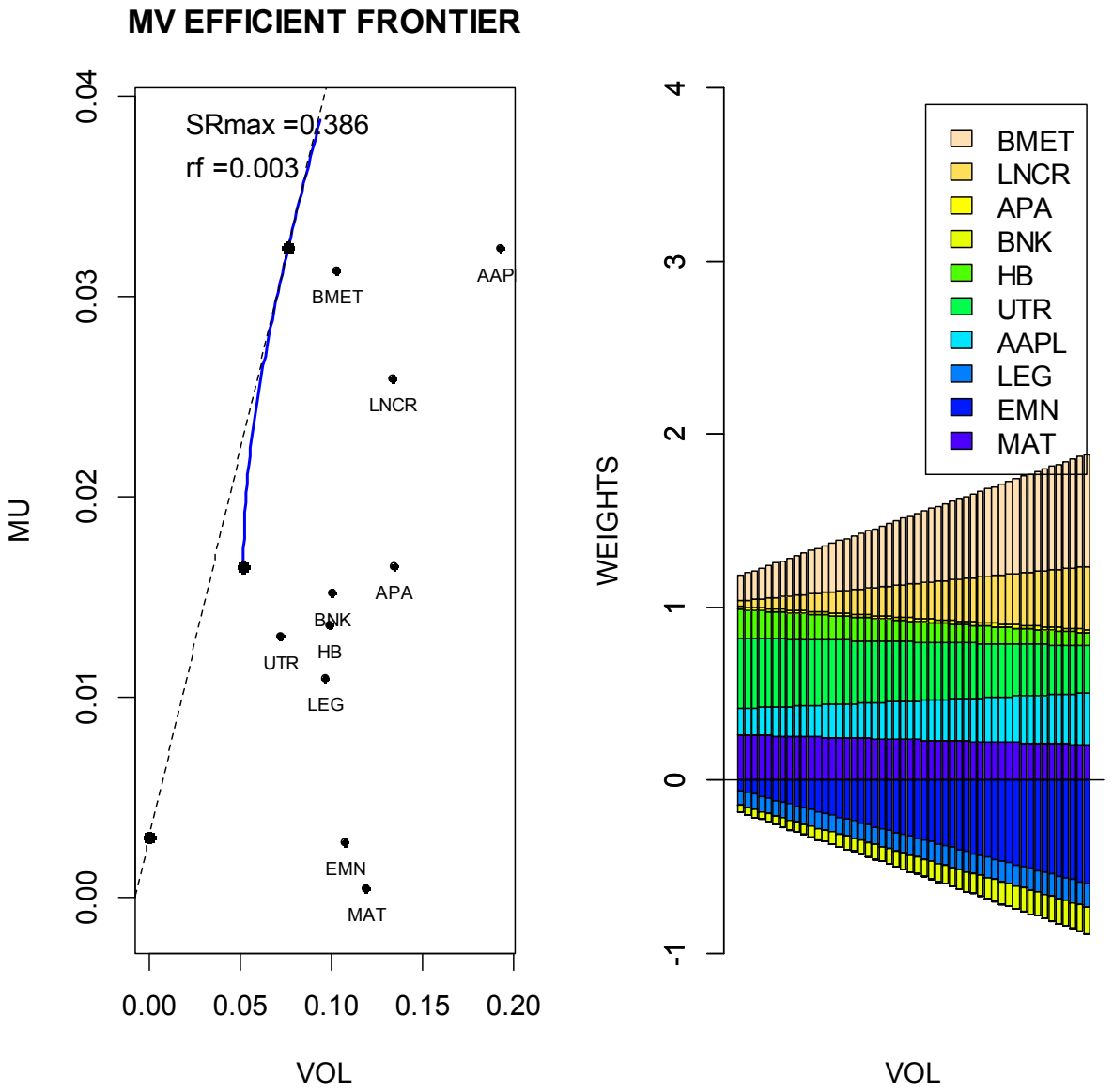
efront.examples.r

```
library(xts)
source("constraint.sets.r")
source("mvo.constrained.r")
source("efront.constrained.r")

# Ex1.12 Unconstrained Efficient Frontier
efrontPlot(returns, rf = .003, npoints = 50,wts.plot =
           T,bar.ylim = c(-1,4))

# Ex1.13 Long-Only Efficient Frontier
efrontPlot(returns, cset.lo, rf = .003, npoints = 50,
          wts.plot = T)

# Ex1.14 Box Constraints Efficient Frontier
efrontPlot(returns, cset.box, rf = .003, npoints =
          50,wts.plot = T)

# Ex1.15 Groups Constraints Efficient Frontier
efrontPlot(returns8, cset.groups, rf = .003, npoints =
           50,wts.plot = T)
```

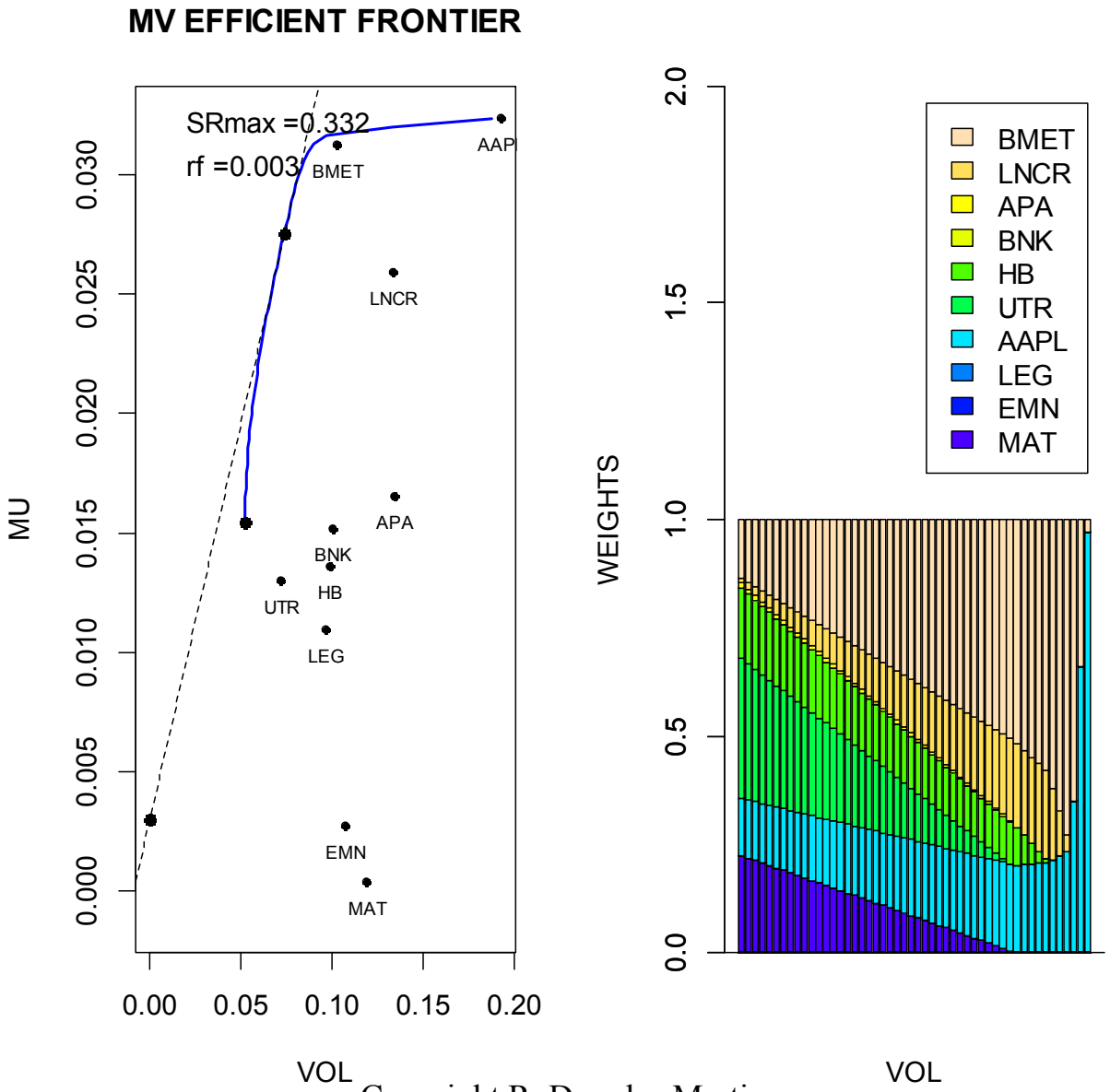
```
> plot.zoo(crsp.port8)
```



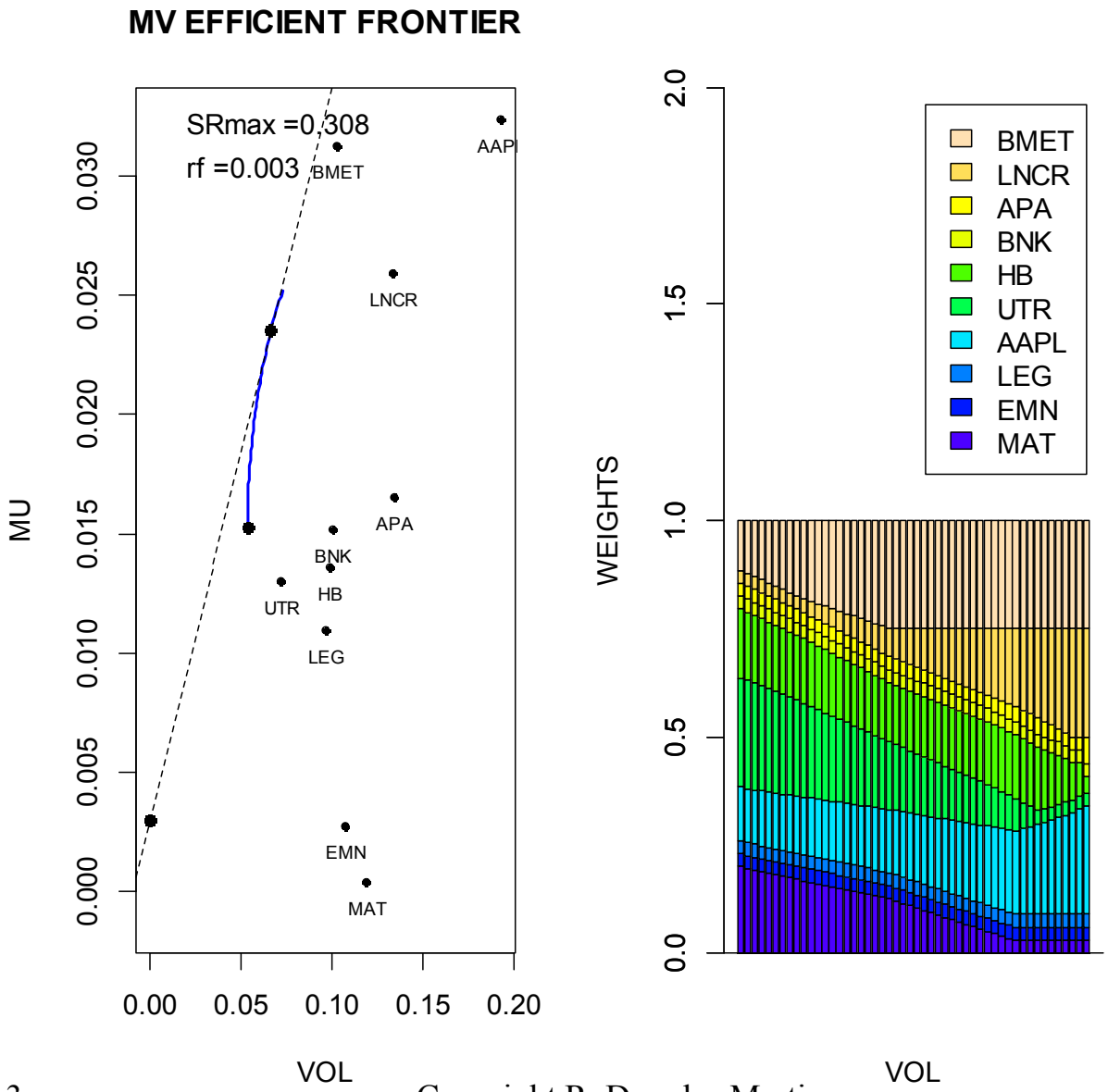
Ex1.12 Fully-Invested Portfolios with Unlimited Shorting



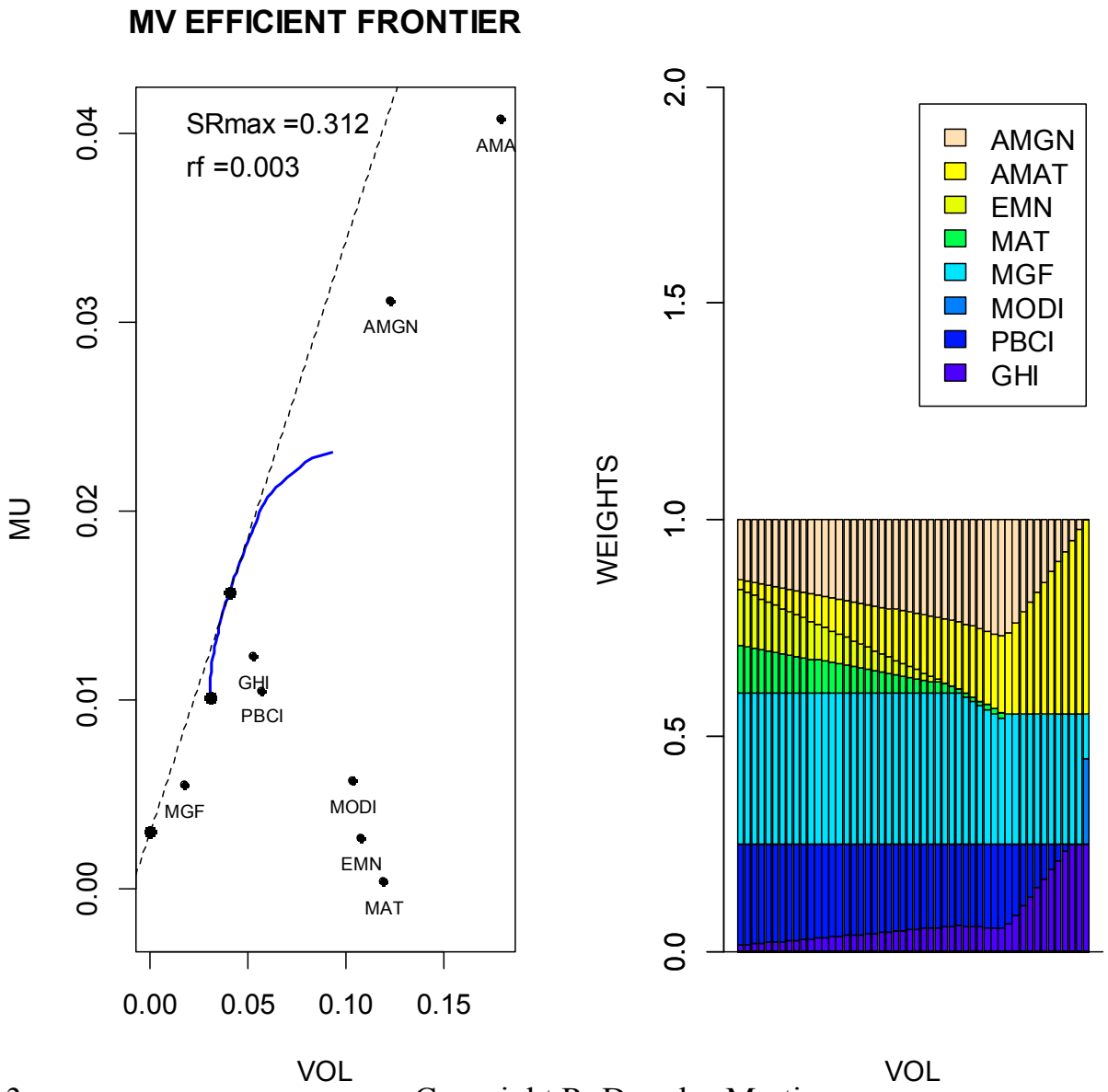
Ex1.13 Long-Only Portfolios



Ex1.14 Box Constraints [3%,25%] Portfolios



Ex1.15 Group Constraints Portfolios

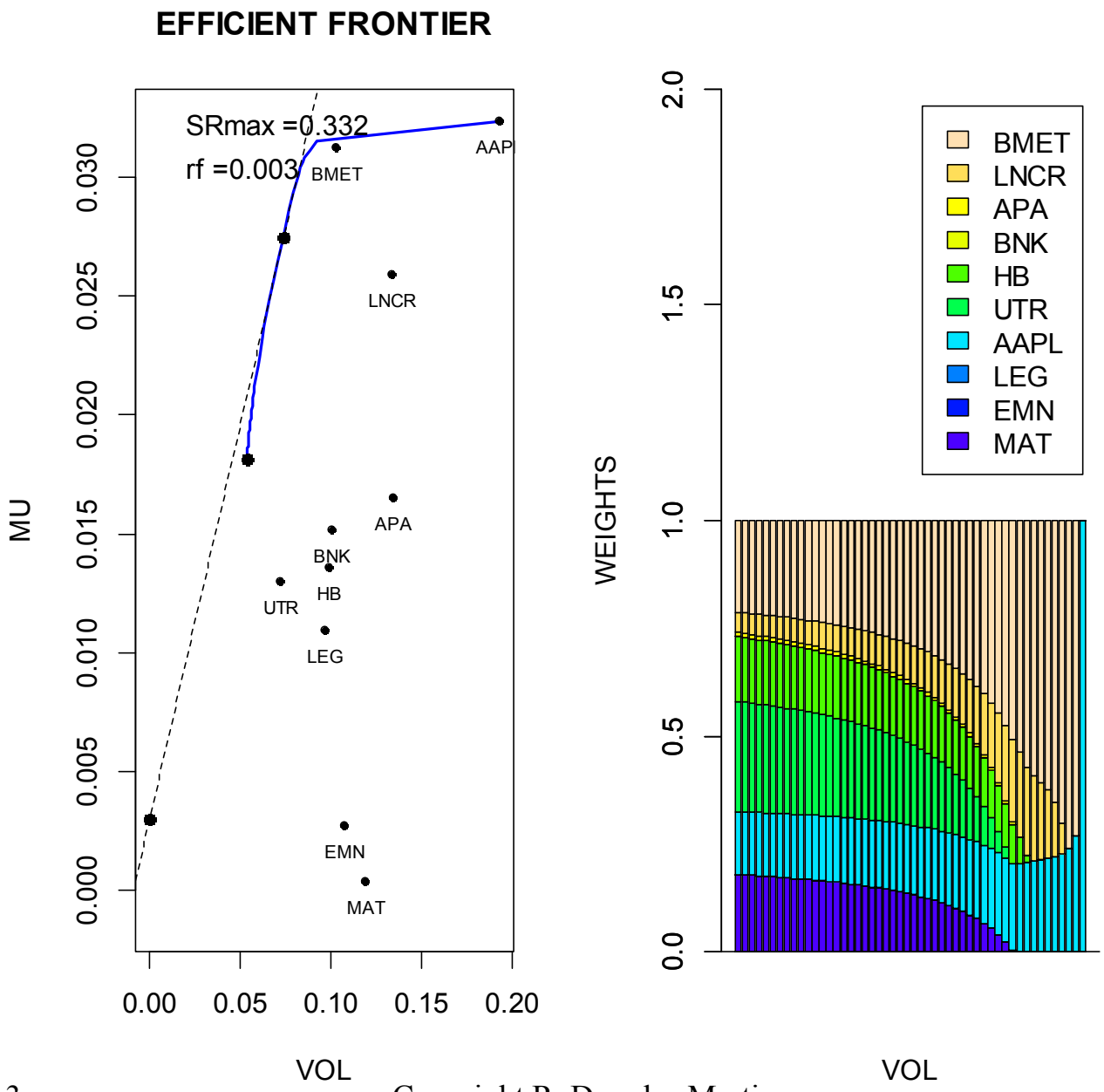


Quadratic Utility Efficient Frontiers

Use functions **efront.qu** and **efrontQuPlot** in the following script **efront.qu.r**.

```
library(quadprog); library(xts)
source("barplot.wts.r");source("qu.r")
source("constraint.sets.r")
efrontQU = function(returns,cset = NULL,lambda = c(5,1,.1))
{
    p = ncol(returns)
    npoints = length(lambda)
    efront = matrix(rep(0,npoints*(p+2)),ncol = p+2)
    for(i in 1:npoints)
    if(is.null(cset))
    {efront[i,] = qu(returns,lambda[i],wts.only = F)}
    else
    {efront[i,] = qu(returns,cset,lambda[i],wts.only = F)}
    dimnames(efront)[[2]]=c("MU","SD",dimnames(returns)[[2]])
    efront
}
```

```
efrontQuPlot = function(returns,cset = NULL,lambda =
                        c(10,5,.1), rf = NULL, wts.plot = T,
                        printout = F,bar.ylim = c(0,2))
{
    if(is.null(cset))
    {efront = efrontQU(returns,cset = NULL, lambda)}
    else
    {efront = efrontQU(returns,cset,lambda)}
    if(printout) print(efront)
    # Set smart x and y plotting region limits
. . . .

#  The rest of this function is identical to efrontPlot.
#  The following commands produce the plots on next slide,
#  which should be compared with Ex1.13.

lambda = seq(20,.01,length.out = 50)
efrontQuPlot(returns, cset.lo, lambda, rf = .003,
             wts.plot = T,bar.ylim = c(0,2))
```


EFFICIENT FRONTIER
SRmax =0.332
rf =0.003
AAP
BMET
LNCR
APA
BNK
HB
UTR
LEG
EMN
MAT
MU
VOL
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.00
0.05
0.10
0.15
0.20
WEIGHTS
VOL
0.0
0.5
1.0
1.5
2.0
BMET
LNCR
APA
BNK
HB
UTR
AAPL
LEG
EMN
MAT

1A.1 NO-CASH MVO PORTFOLIOS

Two equivalent approaches, both allowing short-selling:

1) Maximize quadratic utility
2) Minimize risk with a mean return constraint

Key Result 1: Two-fund theorem

Key Result 2: Hyperbolic portfolio mean return versus volatility relationship

No-Cash Quadratic Utility Optimality

Maximize $$U(\mathbf{w}) = \mu_P(\mathbf{w}) - \frac{1}{2}\lambda \cdot \sigma_P^2(\mathbf{w}) \qquad \lambda > 0$$

$$= \mathbf{w}'\boldsymbol{\mu} - \frac{1}{2}\lambda \cdot \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}$$

With no weight constraints the optimal weights are

$$\mathbf{w} = \lambda^{-1} \cdot \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$$

N.B. Full-investment implies that $\mathbf{1}'\mathbf{w} = 1$. But this in turn implies that $\lambda = \mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$. But if $\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} < 0$ this is not possible. You can only have full-investment with quadratic utility optimality if $\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} > 0$.

So try optimizing with the full-investment constraint:

Maximize $$U(\mathbf{w}) = \mathbf{w}'\boldsymbol{\mu} - \frac{1}{2}\lambda \cdot \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}$$

subject to $$\mathbf{1}'\mathbf{w} = 1 \qquad \text{(full investment)}$$

Lagrangian: $$L(\mathbf{w}) = \mathbf{w}'\boldsymbol{\mu} - \frac{1}{2}\lambda \cdot \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w} - \gamma \cdot (\mathbf{1}'\mathbf{w} - 1)$$

$$\frac{dL(\mathbf{w})}{d\mathbf{w}} = \boldsymbol{\mu} - \lambda \cdot \boldsymbol{\Sigma}\mathbf{w} - \gamma \cdot \mathbf{1} = \mathbf{0}$$

$$\Rightarrow \qquad \mathbf{w} = \lambda^{-1} \cdot \boldsymbol{\Sigma}^{-1} \cdot \left(\boldsymbol{\mu} - \gamma \cdot \mathbf{1}\right)$$

Solving for the Lagrange multiplier γ and rearranging gives the:

Two-Fund Separation Theorem

$$\mathbf{w} = \left(\lambda^{-1}\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\underbrace{\frac{\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}}}_{\mathbf{w}_G} + \left(1-\lambda^{-1}\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right)\cdot\underbrace{\frac{\boldsymbol{\Sigma}^{-1}\mathbf{1}}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}}_{\mathbf{w}_{GMV}}$$

Alternate form:

$$\mathbf{w} = \frac{\boldsymbol{\Sigma}^{-1}\mathbf{1}}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}} + \frac{1}{\lambda}\frac{(\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1})\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - (\mathbf{1}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}\mathbf{1}}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1}}$$

This form is given in QHS 2.2.2 with $\boldsymbol{\mu} \to \mathbf{f}$ the mean returns forecast.

No-Cash MVO with Mean Return Constraint

$$\min_{\mathbf{w}} \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}$$

subject to

$$\mathbf{w}'\boldsymbol{\mu} = \mu_P \quad \text{(specified by investor)}$$

$$\mathbf{w}'\mathbf{1} = 1 \quad \text{(full investment)}$$

$$L(\mathbf{w}) = \frac{1}{2}\mathbf{w}'\boldsymbol{\Sigma}\mathbf{w} + \lambda_1 \underbrace{\left(\mu_P - \mathbf{w}'\boldsymbol{\mu}\right)}_{\text{Constraint 1}} + \lambda_2 \underbrace{\left(1 - \mathbf{w}'\mathbf{1}\right)}_{\text{Constraint 2}}$$

$$\frac{d}{d\mathbf{w}} L(\mathbf{w}) = 0 \quad \Rightarrow \quad \mathbf{w} = \lambda_1 \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \lambda_2 \boldsymbol{\Sigma}^{-1}\mathbf{1}$$

Now apply the two constraints $\mathbf{w}'\boldsymbol{\mu} = \mu_P$ and $\mathbf{w}'\mathbf{1} = 1$ to the optimal weight vector

$$\mathbf{w} = \lambda_1 \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \lambda_2 \boldsymbol{\Sigma}^{-1}\mathbf{1}$$

and solve for the two Lagrange multipliers:

$$\lambda_1 = \frac{c \cdot \mu_P - a}{d} \qquad \lambda_2 = \frac{b - a \cdot \mu_P}{d}$$

where

$$a = \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{1} \qquad b = \boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$$

$$c = \mathbf{1}'\boldsymbol{\Sigma}^{-1}\mathbf{1} \qquad d = bc - a^2$$

Proposition 1: The optimal weight vector may written as

$$\mathbf{w}_{opt} = \mathbf{g}_1 + \mathbf{g}_2 \mu_P$$

where

$$\mathbf{g}_1 = \frac{1}{d}\left(b \cdot \boldsymbol{\Sigma}^{-1}\mathbf{1} - a \cdot \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right) \qquad \mathbf{g}_2 = \frac{1}{d}\left(c \cdot \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - a \cdot \boldsymbol{\Sigma}^{-1}\mathbf{1}\right)$$

Proof: Exercise.

Note that as you vary μ_P arbitrarily you will typically get some $w_i < 0$ that correspond to short selling.

Setting $\mathbf{z}_1 = \boldsymbol{\Sigma}^{-1}\mathbf{1}$ and $\mathbf{z}_2 = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$ you can write the above as:

$$\mathbf{g}_1 = \frac{1}{d}\left(b \cdot \mathbf{z}_1 - a \cdot \mathbf{z}_2\right) \qquad \mathbf{g}_2 = \frac{1}{d}\left(c \cdot \mathbf{z}_2 - a \cdot \mathbf{z}_1\right)$$

Proposition 2: Let P_1 , P_2 be any two efficient frontier portfolios.

Then

$$\operatorname{cov}\left(r_{P_1}, r_{P_2}\right) = \mathbf{w}'_{P_1}\boldsymbol{\Sigma}\mathbf{w}_{P_2}$$

$$= \frac{1}{c} + \frac{c}{d}\left(\mu_{P_1} - \frac{a}{c}\right)\left(\mu_{P_2} - \frac{a}{c}\right)$$

Proof: Exercise (Hint: use Proposition 1)

The special case of the above with $P_1 = P_2$ gives the portfolio frontier formula:

$$\sigma_P^2 = \operatorname{cov}\left(r_P, r_P\right) = \frac{1}{c} + \frac{c}{d}\left(\mu_P - \frac{a}{c}\right)^2$$

The Portfolio Frontier is a Hyperbola

Re-arranging gives

$$\frac{\sigma_P^2}{1/c} - \frac{\left(\mu_P - \frac{a}{c}\right)^2}{d/c^2} = 1$$

Recall that the equation of a hyperbola centered at $(0, 0)$ is:

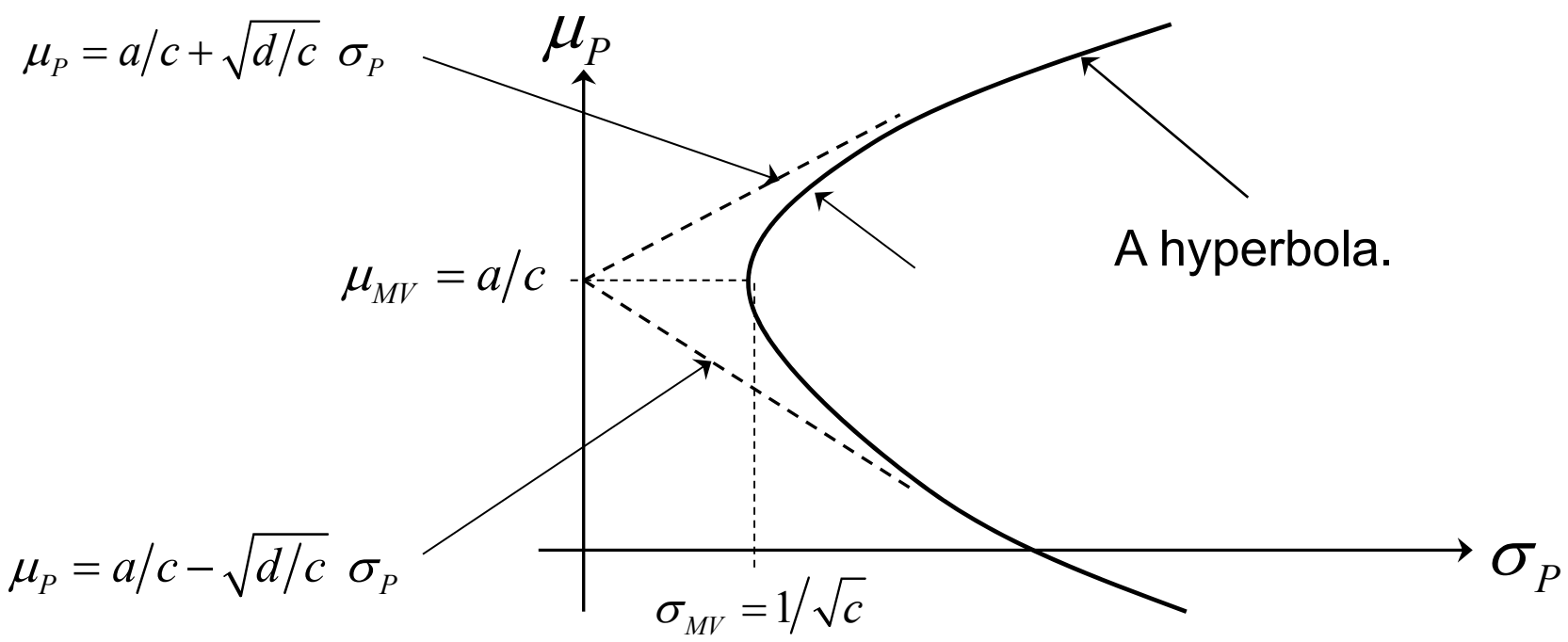
$$\frac{x^2}{A^2} - \frac{y^2}{B^2} = \left(\frac{x}{A} - \frac{y}{B}\right)\left(\frac{x}{A} + \frac{y}{B}\right) = 1, \text{ with asymptotes } y = \pm\frac{B}{A}x$$

Thus μ_P versus σ_P is a hyperbola centered at $(0, a/c)$, with asymptotes

$$\mu_P = \frac{a}{c} \pm \sqrt{\frac{d}{c}}\sigma_P$$

In Summary:

$$\mu_P = \frac{a}{c} \pm \sqrt{\frac{d}{c}\sigma_P^2 - \frac{d}{c^2}} \quad \text{for } \sigma_P^2 \geq 1/c$$



1A.2 Optimal Risk Aversion Parameter

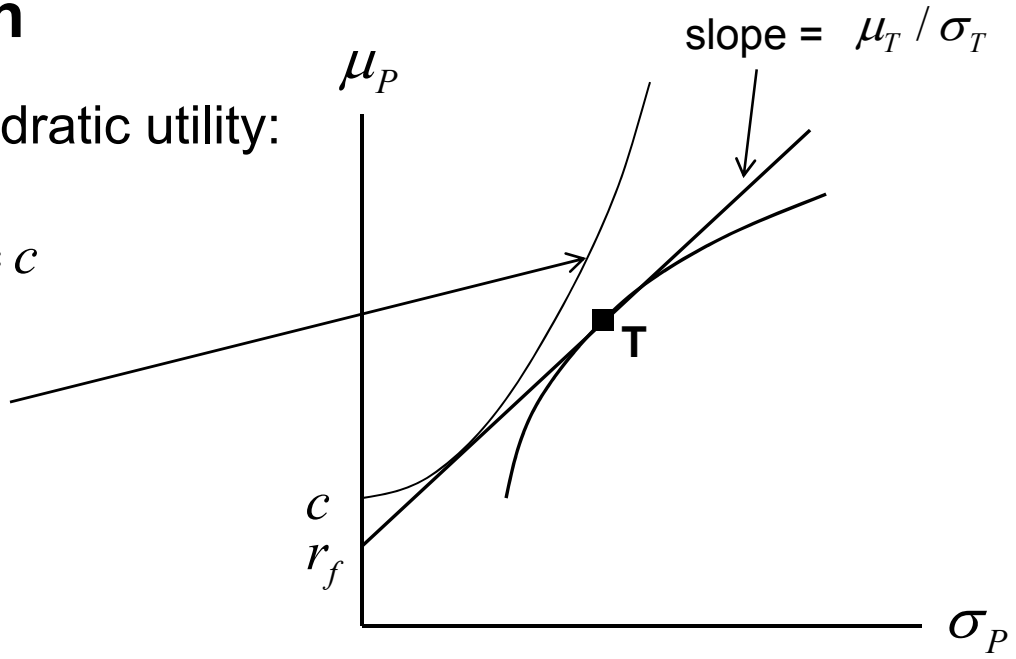
Alternate Derivation

Contours of constant quadratic utility:

$$U(\mu_P, \sigma_P) = \mu_P - \frac{\lambda}{2}\sigma_P^2 = c$$

$$\Rightarrow \qquad \mu_P = c + \frac{\lambda}{2}\sigma_P^2$$

$$\Rightarrow \qquad \frac{d\mu_P}{d\sigma_P} = \lambda \cdot \sigma_P$$



Slope of constant expected utility contour line at all tangency points is equal to the maximum Sharpe ratio (SR):

$$\frac{d\mu_P}{d\sigma_P} = \lambda \cdot \sigma_P = \frac{\mu_P}{\sigma_P} = \frac{\mu_T}{\sigma_T} = \mathrm{SR}_T \quad \Rightarrow \quad \boxed{\lambda = \frac{\mu_P}{\sigma_P^2} = \frac{1}{\sigma_P} SR_T}$$

A way to back out risk aversion for any linear efficient frontier portfolio !