COMPUTATIONAL FINANCE & RISK MANAGEMENT
UNIVERSITY of WASHINGTON
Department of Applied Mathematics

Markowitz Mean-Variance Optimization

CFRM 425 (016)

R Programming for Quantitative Finance

Lecture References/Package Downloads

- Ang, Ch 7: §§7.3 – 7.4 (the latter involves short selling)
- Solving Quadratic Progams with R's quadprog package (blog article):

https://rwalk.xyz/solving-quadratic-progams-with-rs-quadprog-package/

- Ch 30, www.rmetrics.org/downloads/9783906041025-basicr.pdf

- Topics:
  - The Optimization Problem
  - Using the quadprog package

The Optimization Problem

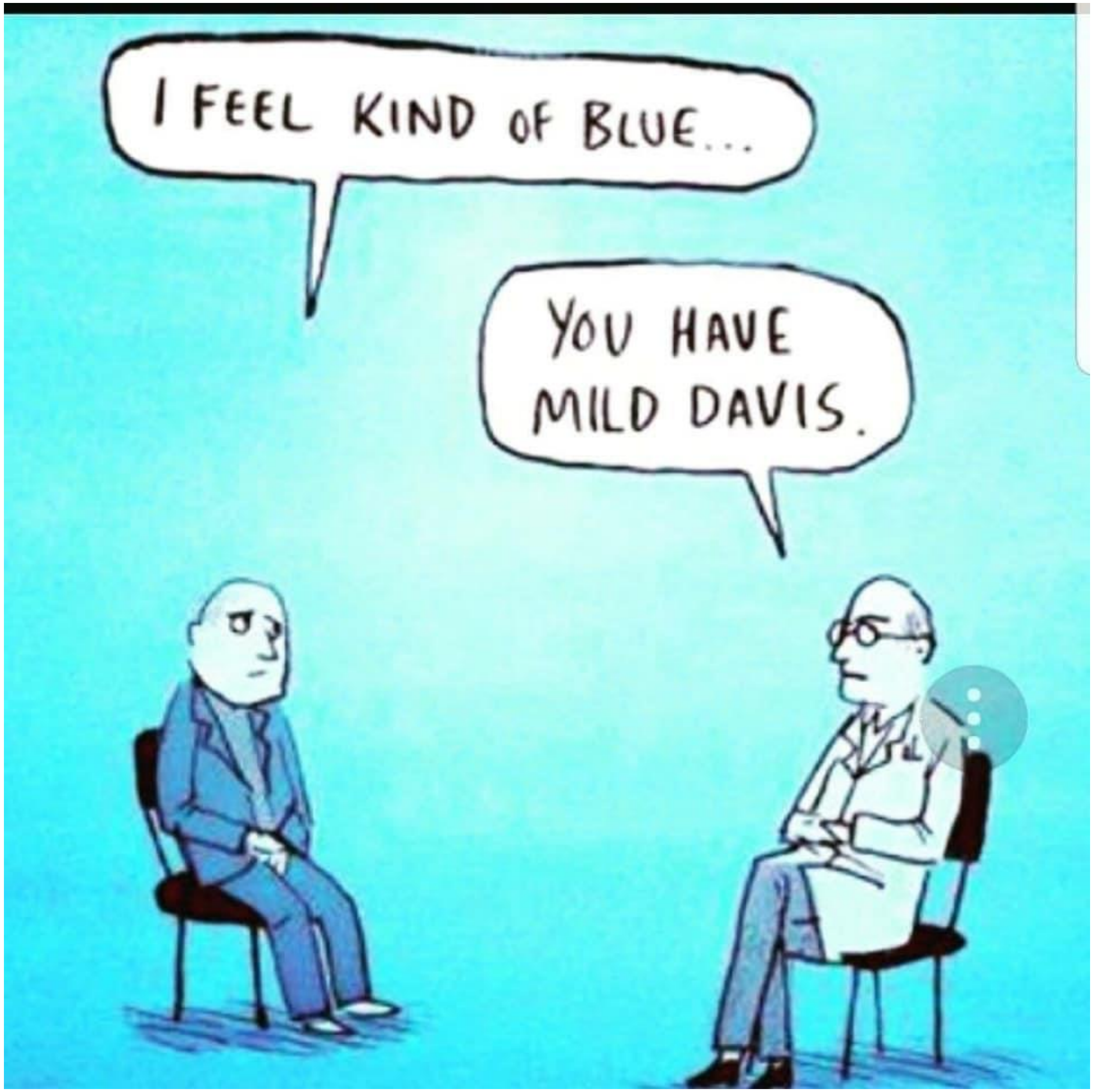
Portfolio Optimization Example

- The classic Markowitz application is a quadratic optimization problem, to find the optimal weights among several assets that would minimize the risk (variance) subject to a target return (based on the means of historical returns) and correlations between the (risky) assets
- The definition of a *risky asset* here is that the variance of each is greater than zero
- We will see later what happens when a risk-free asset enters into the model
- The optimization problem:
- Where:
  - $\{\omega_i\}$ is the set of fund weights,
  - $\boldsymbol{\mu}$ is the empirical fund mean vector,
  - and $\boldsymbol{\Sigma}$ is the covariance matrix

mean-variance portfolio optimization

$$
\begin{aligned}
min_b \quad & \omega^T \boldsymbol{\Sigma} \omega \\
s.t. \quad & \omega^T \boldsymbol{\mu} = \mu_p \\
& \omega^T \mathbf{1} = 1 \\
& \omega_{min} \leqq \omega_i \leqq \omega_{max}
\end{aligned}
$$

Using the quadprog package



The quadprog package

- The function to use for the quadratic programming problem is **`solve.QP(.)`**
- More specifically, it implements the dual method of Goldfarb and Idnani (1982, 1983) for solving quadratic programming problems of the form

$$\min_{b} -\boldsymbol{d}^T \boldsymbol{b} + \frac{1}{2} \boldsymbol{b}^T \boldsymbol{D} \boldsymbol{b}$$

$$ST\ \boldsymbol{A}^T \boldsymbol{b} \ \geq \ \boldsymbol{b}_0,$$

$$\boldsymbol{D} \in \mathbb{R}^{n \times n}\ symmetric$$

$$\boldsymbol{b}, \boldsymbol{b}_0 \in \mathbb{R}^n, \boldsymbol{A} \in \mathbb{R}^{n \times p}$$

- $\boldsymbol{p}$ = number of constraints
- For our case, $\boldsymbol{d} = \boldsymbol{0}$, and $\boldsymbol{D} = 2\boldsymbol{\Sigma}$
- The factor of 2, however, is not necessary since the first term in the objective function disappears, so just put $\boldsymbol{D} = \boldsymbol{\Sigma}$

The quadprog package

- For a first example, let's just enforce the constraints

$$\omega^T \mathbf{1} = 1$$

$$\omega^T \mu = \mu_p$$

- While the weights must sum to 1, they can be positive or negative
- This means short sales are allowed
- The first two rows of $\boldsymbol{A}^T$:
    - 1 1 … 1
    - μ_1 μ_2 … μ_n

The quadprog package

- In code, given an xts set of asset returns **`rtns`**, our setup goes as follows:
- Note the use of **`coredata(.)`** to extract the data matrix from the xts object

```
meansVec <- colMeans(coredata(rtns))

covMtx <- cov(coredata(rtns))

# Our D matrix is
D <- covMtx
```

- The sum of the weights is the same as taking the dot product with a unit vector, so the first row of A is a row of ones

```
A <- matrix(data = 1, nrow = 1, ncol = nrow(D))
```

- The 2nd constraint says the weighted average of means must sum to the target return, so assign the historical means of each asset to get the dot product

```
# Append the expected returns:
A <- rbind(A, meansVec)
```

- Our constraint vector $\boldsymbol{b}_0$ is formed by putting

```
b0 <- c(1, minRtn)
```

- The **`solve.QP(.)`** function looks something like the following example:

```
sol <- solve.QP(Dmat=D, dvec = rep(0,nrow(D)), Amat=t(A),
                bvec=b0, meq=1)
```

- **D** is the matrix 2 x Covariance matrix
- **`rep(0, nrow(D))`** is a vector of zeros with the same number of elements as the number of rows of the matrix **D** (not used in Markowitz optimization)
- **A** is our matrix of coefficients of the constraints
- **`b0`** is the vector of real values on the right hand side of the constraints (same as $\boldsymbol{b}_0$ in the mathematical description)
- The **`meq`** parameter means the number of constraints that must attain equality; otherwise, the constraint is LHS $\geq$ RHS

The quadprog package

- Our matrix **A** in the code is actually $\boldsymbol{p} \times \boldsymbol{n}$, so we need to transpose it to $\boldsymbol{n} \times \boldsymbol{p}$ to satisfy the conditions of the algorithm
- We also require both constraints to be binding (=); setting **`meq=2`** in the call to **`solve.QP(.)`** enforces this

- Then:

```
sol <- solve.QP(Dmat=D, dvec = rep(0,nrow(D)), Amat=t(A),
                bvec=b0, meq=2)
```

- And we get our optimal portfolio weights:

```
sol$solution
[1]  0.07851751  0.10173572  0.06404380  0.06180230 -0.10318638
[6]  0.23184826  0.18270103  0.08272886  0.34224685 -0.04243794
```

- Now, what if we want to prevent short sales
- Append an identity matrix (size = number of assets) to the bottom of our **A** matrix in the code, and the same number of zeroes to the **d0** constraint vector:

```
A_no_short <- rbind(A, diag(nrow(D)))
b0_no_short <- c(b0, rep(0, nrow(D)))
```

- Run the optimization again:

```
sol_no_short <- solve.QP(Dmat=D, dvec = rep(0,nrow(D)),
                         Amat=t(A_no_short), bvec=b0_no_short, meq=2)

sol_no_short$solution
```

- Result:

```
[1]  1.277730e-01  5.290414e-17  4.817832e-02  8.949107e-02 -4.164283e-17
[6]  2.279359e-01  2.162308e-01  1.458849e-02  2.758025e-01 -1.749745e-17
```

- The boxed results can be interpreted as zeroes (no allocations in the portfolio)