MATLAB for Finance

15
Optimization Toolbox

Portfolio Theory

Two Risky Assets

- 12-month Time Horizon

```
mese = 0:12
```

- Settlement Prices of Asset A

```
prezziA = [25 24.12 23.37 24.75 26.62 26.5
  28 28.88 29.75 31.38 36.25 37.13 36.88]'
```

- Settlement Prices of Asset B

```
prezziB= [45 44.85 46.88 45.25 50.87 53.25
  53.25 62.75 65.5 66.87 78.5 78 68.23]'
```

Yields

$$\frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \approx \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln P_t - \ln P_{t-1}$$

```
rA = log(prezziA(2:end)) - log(prezziA(1:end-1))
rB = log(prezziB(2:end)) - log(prezziB(1:end-1))
```

Yields Hypothesis

- Hypothesis:
  - The last 12 month yields represent future yields distribution

```
mA = mean(rA)
mB = mean(rB)
varA = var(rA,1)
varB = var(rB,1)
volA = std(rA,1)
volB = std(rB,1)
```

Variance of sample

$$\sigma_x = \frac{1}{N}\sum_i (x_i - \mu_x)^2$$

- "[...] note that the N–1 should be changed to N if you are ever in the situation of measuring the variance of a distribution whose mean μ is known a priori rather than being estimated from data", *Numerical Recipes*

Covariance

- *Measure of how much two random variables change together*

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

```
covAB = ((rA - mean(rA))' * (rB - mean(rB))) /numel(rA)
```

- or

```
cov([rA rB],1)
```

Efficient Portfolio

Optimization Quadratic Problem

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}'\Sigma\mathbf{x}$$

$$s.t.$$

linear constraints (equalities)

$$\mathbf{x}'\mathbf{m} = \mu$$

At a given return

$$\mathbf{x}'\mathbf{1} = 1$$

$$\mathbf{x} \geq \mathbf{0}$$

Short-sell is not allowed

Efficient Portfolio

- The method of Lagrange multipliers provides a strategy for finding the local maxima/minima of a function subject to equality constraints

Optimization *unconstrained* Quadratic Problem

$$\min_{\mathbf{x}} L(\mathbf{x}, \lambda_1, \lambda_2) = \frac{1}{2}\mathbf{x}'\Sigma\mathbf{x} - \lambda_1\left(\mathbf{x}'\mathbf{1} - 1\right) - \lambda_2\left(\mathbf{x}'\mathbf{m} - \mu\right)$$

stationary point (*partial derivatives of L are zero*)

$$\frac{\partial L}{\partial \mathbf{x}} = \Sigma\mathbf{x} - \lambda_1\mathbf{1} - \lambda_2\mathbf{m} \stackrel{!}{=} 0 \rightarrow \mathbf{x} = \Sigma^{-1}\left(\lambda_1\mathbf{1} + \lambda_2\mathbf{m}\right)$$

$$\frac{\partial L}{\partial \lambda_1} = \mathbf{x}'\mathbf{1} - 1 \stackrel{!}{=} 0 \rightarrow \mathbf{x}'\mathbf{1} = 1$$

$$\frac{\partial L}{\partial \lambda_2} = \mathbf{x}'\mathbf{m} - \mu \stackrel{!}{=} 0 \rightarrow \mathbf{x}'\mathbf{m} = \mu$$

Efficient Portfolio

$$\left[\Sigma^{-1}\left(\lambda_1 \mathbf{1} + \lambda_2 \mathbf{m}\right)\right]' \mathbf{1} = 1$$

$$\rightarrow \lambda_1 \left(\mathbf{1}'\Sigma^{-1}\mathbf{1}\right) + \lambda_2 \left(\mathbf{m}'\Sigma^{-1}\mathbf{1}\right) = 1$$

$$\rightarrow \lambda_1 C + \lambda_2 B = 1$$

$$B = \mathbf{m}'\Sigma^{-1}\mathbf{1}$$

$$C = \mathbf{1}'\Sigma^{-1}\mathbf{1}$$

Efficient Portfolio

$$\left[\Sigma^{-1}\left(\lambda_1 \mathbf{1} + \lambda_2 \mathbf{m}\right)\right]' \mathbf{m} = \mu$$

$$\rightarrow \lambda_1\left(\mathbf{1}'\Sigma^{-1}\mathbf{m}\right) + \lambda_2\left(\mathbf{m}'\Sigma^{-1}\mathbf{m}\right) = \mu$$

$$\rightarrow \lambda_1 B + \lambda_2 A = \mu$$

$$A = \mathbf{m}'\Sigma^{-1}\mathbf{m}$$

$$B = \mathbf{1}'\Sigma^{-1}\mathbf{m}$$

$$C = \mathbf{1}'\Sigma^{-1}\mathbf{1}$$

Efficient Portfolio

$$\begin{cases} \lambda_1 C + \lambda_2 B = 1 \\ \lambda_1 B + \lambda_2 A = \mu \end{cases} \rightarrow$$

$$\lambda_1 = \frac{A - \mu B}{CA - B^2}$$

$$\lambda_2 = \frac{\mu C - B}{CA - B^2}$$

Efficient Portfolio

$$\mathbf{x} = \Sigma^{-1}\left(\lambda_1 \mathbf{1} + \lambda_2 \mathbf{m}\right)$$

$$\rightarrow \mathbf{x} = \frac{\left(C\Sigma^{-1}\mathbf{m} - B\Sigma^{-1}\mathbf{1}\right)\mu + A\Sigma^{-1}\mathbf{1} - B\Sigma^{-1}\mathbf{m}}{AC - B^2}$$

```
S = cov([rA rB],1)
A = [mA mB] * S^-1 * [mA mB]'
B = ones(1,2) * S^-1 * [mA mB]'
C = ones(1,2) * S^-1 * ones(2,1)
mu = .033
x = ((C*S^-1 * [mA;mB] - B * S^-1 * ones(2,1)) * mu ...
+ A * S^-1 * ones(2,1) - B * S^-1 * [mA;mB]) / (A*C-B^2)
```

to be continue on the next line

Efficient Portfolio

- Varying μ:

```
mu = [.033 0.12]
x = ((C*S^-1 * [mA;mB] - B * S^-1 * ones(2,1)) * mu ...
     + (A * S^-1 * ones(2,1) - B * S^-1 * ...
       [mA;mB])*ones(1,numel(mu))) / (A*C-B^2)
```

$$\sigma_P = \sqrt{\frac{\mu^2 C - 2\mu B + A}{AC - B^2}} = \sqrt{\mathbf{x}'\Sigma\mathbf{x}}$$

```
volP = sqrt((mu.^2*C-2*mu*B+A)/(A*C-B^2))
```

or

```
diag(sqrt(x' * S * x))'
```

Optimization Problems

Optimization

- Unconstrained minimization (**fminunc**):

$$\min_x \ f(x)$$

- Linear Programming (**linprog**):

$$\min_x z = c'x \text{ subject to } Ax \le b, x \ge 0$$

- Quadratic Programming (**quadprog**):

$$\min_x z = \tfrac{1}{2} x'Hx \text{ subject to } Ax \le b, x \ge 0$$

- Non-linear Programming (**fmincon**):

$$\min_x z = f(x) \text{ subject to } g(x) \le b, x \ge 0$$

Unconstrained Minimization

- **fminunc**
- Example: $\min_x f(x) = e^{x_1}\left(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1\right)$
- funzioneobiettivo.m:

```
function f = funzioneobiettivo(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^2
  +4*x(1)*x(2)+2*x(2)+1);
end
```

Calling The Solver

- Script esegui.m or command window:

```
% initial value of x
x0 = [0 0];
% medium-scale problem
opzioni = optimset('LargeScale', 'off');
x = fminunc('funzioneobiettivo', x0,
  opzioni);
```

Non-Linear Programming

- *constrained programming*
- **fmincon**
- Example: $\min_x f(x) = e^{x_1}\left(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1\right)$

subject to

$$x_1x_2 - x_1 - x_2 \le -1.5$$

$$x_1x_2 \ge -10$$

$$x_1x_2 - x_1 - x_2 + 1.5 \le 0$$

$$-x_1x_2 - 10 \le 0$$

- funzioneobiettivo.m
- vincoliNL.m

vincoliNL.m

```
function [cdis, cug] = vincoliNL(x)
cdis = [x(1)*x(2)-x(1)-x(2)+1.5;
        -x(1)*x(2)-10];
cug = [];
```

Calling The Solver

- Script eseguiNL.m or command window:

```
% intial values of x
x0 = [0 0];
% medium-scale problem
opzioni = optimset('LargeScale', 'off');
x = fmincon('funzioneobiettivo', x0 ,[] ,[]
  ,[] ,[] ,[] ,[], 'vincoliNL', opzioni);
```

fmincon()

$$\min_{x} f(x)$$

$$s.t.$$

$$c(x) \le 0$$

$$ceq(x) = 0$$

$$A \cdot x \le b$$

$$Aeq \cdot x = beq$$

$$lb \le x \le ub$$

```
x = fmincon(
fun,  % objective function name
x0,  % initial values
A,  % linear inequalities coefficients
b,  % linear inequalities terms
Aeq,  % linear equalities coefficients
beq,  % linear equalities terms
lb,ub,  % lower-upper bounds
nonlcon,  % NLconstraints function
options)  % options
```

exitflag

- [x,fval,exitflag] = fmincon(...)
- returns a value exitflag that describes the exit condition of fmincon.

1	First order optimality conditions were satisfied to the specified tolerance.
2	Change in x was less than the specified tolerance.
3	Change in the objective function value was less than the specified tolerance.
4	Magnitude of the search direction was less than the specified tolerance and constraint violation was less than options.TolCon.
5	Magnitude of directional derivative was less than the specified tolerance and constraint violation was less than options.TolCon.
0	Number of iterations exceeded options.MaxIter or number of function evaluations exceeded options.FunEvals
-1	Algorithm was terminated by the output function.
-2	No feasible point was found.

Linear Programming

$$\min_x f^T x$$

$$s.t.$$

$$A \cdot x \leq b$$

$$Aeq \cdot x = beq$$

$$lb \leq x \leq ub$$

```
x = linprog(f,A,b)
x = linprog(f,A,b,Aeq,beq)
x = linprog(f,A,b,Aeq,beq,lb,ub)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0)
x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)
```

```
[x,fval] = linprog(...)
[x,lambda,exitflag] = linprog(...)
[x,lambda,exitflag,output] = linprog(...)
[x,fval,exitflag,output,lambda] = linprog(...)
```

Linear Programming

- **linprog**
- Example:

$$\min_x \; f(x) = -5x_1 - 4x_2 - 6x_3$$

subject to

$$x_1 - x_2 + x_3 \le 20$$

$$3x_1 + 2x_2 + 4x_3 \le 42$$

$$3x_1 + 2x_2 \le 30$$

$$x_i \ge 0 \;\; \forall i = 1,2,3$$

Quadratic Programming

- **quadprog**

(compare `doc quadprog`)

- Example:

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}'\Sigma\mathbf{x}$$

$$s.t.$$

$$\mathbf{x}'\mathbf{m} = \mu$$

$$\mathbf{x}'\mathbf{1} = 1$$

$$\mathbf{x} \geq \mathbf{0}$$

Quadratic Programming

```
[x,fval,exitflag] =
   quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)

x0 = [0 0];
opzioni = optimset('LargeScale', 'off');

   x = quadprog(S,
                [],[],[],
                [mA mB; ones(1,2)],
                [mu 1],
                0,
                [],
                x0,
                opzioni)
```

covariance matrix

equalities matrix

rhs vector

lower bound ($x \geq 0$)

upper bound (inf)

starting point

options (middle-scale)