Portfolio Theory with Matrix Algebra and No Short Sales

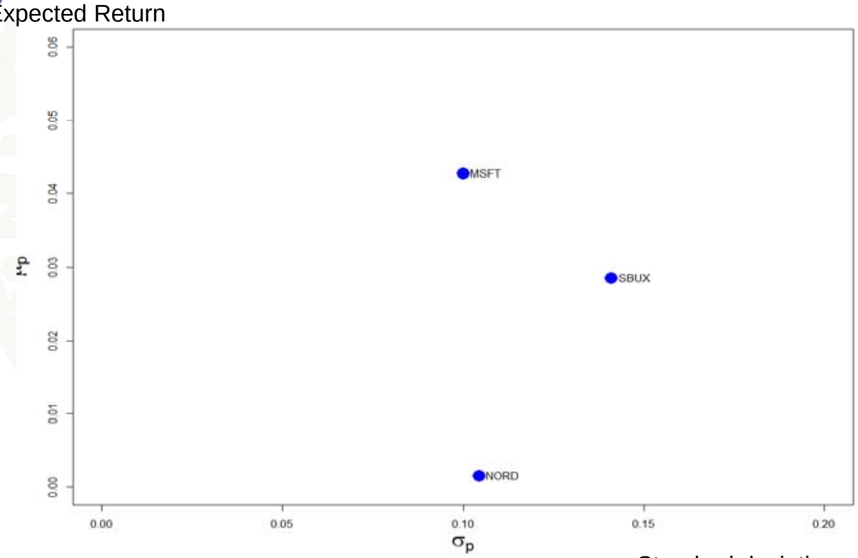
Amath 540/Econ 424
Eric Zivot
Summer 2012
Updated: August 7, 2012

3 Asset Example Data

```
> asset.names <- c("MSFT", "NORD", "SBUX")
> mu.vec = c(0.0427, 0.0015, 0.0285)
> names(mu.vec) = asset.names
> sigma.mat = matrix(c(0.0100, 0.0018, 0.0011,
+                 0.0018, 0.0109, 0.0026,
+                 0.0011, 0.0026, 0.0199),
+               nrow=3, ncol=3)
> dimnames(sigma.mat) = list(asset.names, asset.names)
> mu.vec
  MSFT   NORD   SBUX
0.0427 0.0015 0.0285

> sigma.mat
       MSFT   NORD   SBUX
MSFT 0.0100 0.0018 0.0011
NORD 0.0018 0.0109 0.0026
SBUX 0.0011 0.0026 0.0199
```

Risk-Return Characteristics

Expected Return



Standard deviation

Global Minimum Variance Portfolio with No Short Sales

```
# unconstrained solution
> gmin.port = globalMin.portfolio(mu.vec, sigma.mat)
> gmin.port
Call:
globalMin.portfolio(er = mu.vec, cov.mat = sigma.mat)

Portfolio expected return:     0.02489
Portfolio standard deviation:  0.07268
Portfolio weights:
  MSFT   NORD   SBUX
0.4411 0.3656 0.1933
```

Global Minimum Variance Portfolio with No Short Sales

```
# set restriction matrices
> D.mat = 2*sigma.mat
> d.vec = rep(0, 3)
> A.mat = cbind(rep(1,3), diag(3))
> b.vec = c(1, rep(0,3))
> D.mat
       MSFT   NORD   SBUX
MSFT 0.0200 0.0036 0.0022
NORD 0.0036 0.0218 0.0052
SBUX 0.0022 0.0052 0.0398

> d.vec
[1] 0 0 0

> A.mat
     [,1] [,2] [,3] [,4]
[1,]    1    1    0    0
[2,]    1    0    1    0
[3,]    1    0    0    1

> b.vec
[1] 1 0 0 0
```


Global Minimum Variance Portfolio with No Short Sales

```
# use solve.QP to minimize portfolio variance
> args(solve.QP)
function (Dmat, dvec, Amat, bvec, meq = 0, factorized =
FALSE)
> qp.out = solve.QP(Dmat=D.mat, dvec=d.vec,
+                   Amat=A.mat, bvec=b.vec, meq=1)
> class(qp.out)
[1] "list"

> names(qp.out)
[1] "solution"                "value"
[3] "unconstrained.solution"  "iterations"
[5] "Lagrangian"              "iact"

> qp.out$solution
[1] 0.4411 0.3656 0.1933    # portfolio weights

> sum(qp.out$solution)
[1] 1

> qp.out$value              # portfolio variance
[1] 0.005282
```

Number of equality constraints

Global Minimum Variance Portfolio with No Short Sales

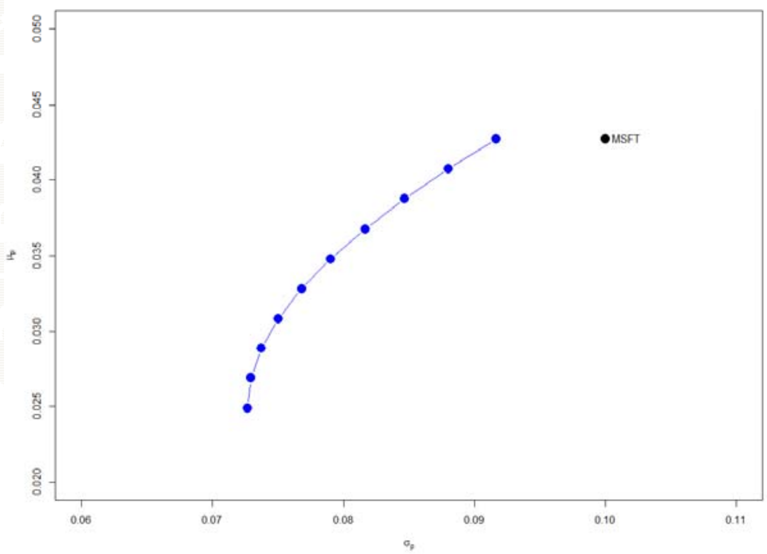
```
# compute mean, variance and sd
> w.gmin.ns = qp.out$solution
> names(w.gmin.ns) = names(mu.vec)
> w.gmin.ns
  MSFT   NORD   SBUX
0.4411 0.3656 0.1933

> er.gmin.ns = as.numeric(crossprod(w.gmin.ns, mu.vec))
> er.gmin.ns
[1] 0.02489

> var.gmin.ns = as.numeric(t(w.gmin.ns)%*%sigma.mat%*%w.gmin.ns)
> var.gmin.ns
[1] 0.005282

> sqrt(var.gmin.ns)
[1] 0.07268
```


Compute Efficient Frontier with Allowing Short Sales

```
# compute and plot efficient frontier with short-sales
> ef <- efficient.frontier(mu.vec, sigma.mat, alpha.min=0,
+                          alpha.max=1, nport=10)
> ef$weights
          MSFT     NORD   SBUX
port 1  0.8275 -0.09075 0.2633
port 2  0.7845 -0.04004 0.2555
port 3  0.7416  0.01067 0.2477
port 4  0.6987  0.06138 0.2399
port 5  0.6557  0.11209 0.2322
port 6  0.6128  0.16279 0.2244
port 7  0.5699  0.21350 0.2166
port 8  0.5270  0.26421 0.2088
port 9  0.4840  0.31492 0.2010
port 10 0.4411  0.36563 0.1933
```

Some efficient portfolios have short sales

Efficient Frontier with Allowing Short Sales

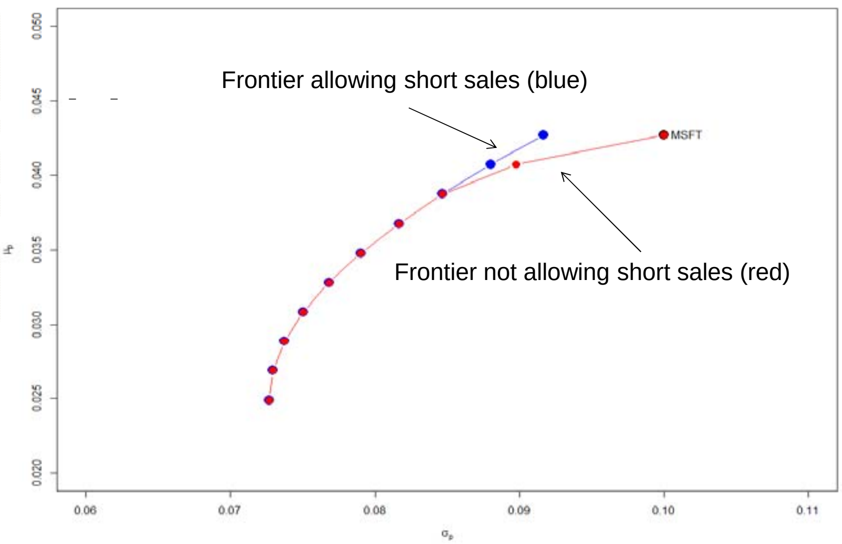


Compute Efficient Frontier without Short Sales

```
# compute efficient frontier with no-short sales
> mu.vals = seq(er.gmin.ns, max(mu.vec), length.out=10)
> w.mat = matrix(0, length(mu.vals), 3)
> sd.vec = rep(0, length(sd.vec))
> colnames(w.mat) = names(mu.vec)
> D.mat = 2*sigma.mat
> d.vec = rep(0, 3)
> A.mat = cbind(mu.vec, rep(1,3), diag(3))
> for (i in 1:length(mu.vals)) {
+   b.vec = c(mu.vals[i],1,rep(0,3))
+   qp.out = solve.QP(Dmat=D.mat, dvec=d.vec,
+                     Amat=A.mat, bvec=b.vec, meq=2)
+   w.mat[i, ] = qp.out$solution
+   sd.vec[i] = sqrt(qp.out$value)
+ }
```

2 equality constraints!

Compute Efficient Frontier without Short Sales



Example: Infeasible Frontier Portfolio

```
# illustrate infeasible portfolio
# set target return equal to 0.08
> b.vec = c(0.08,1,rep(0,3))
> qp.out = solve.QP(Dmat=D.mat, dvec=d.vec,
+                   Amat=A.mat, bvec=b.vec, meq=2)
Error in solve.QP(Dmat = D.mat, dvec = d.vec, Amat =
A.mat, bvec = b.vec,  :
  constraints are inconsistent, no solution!
```