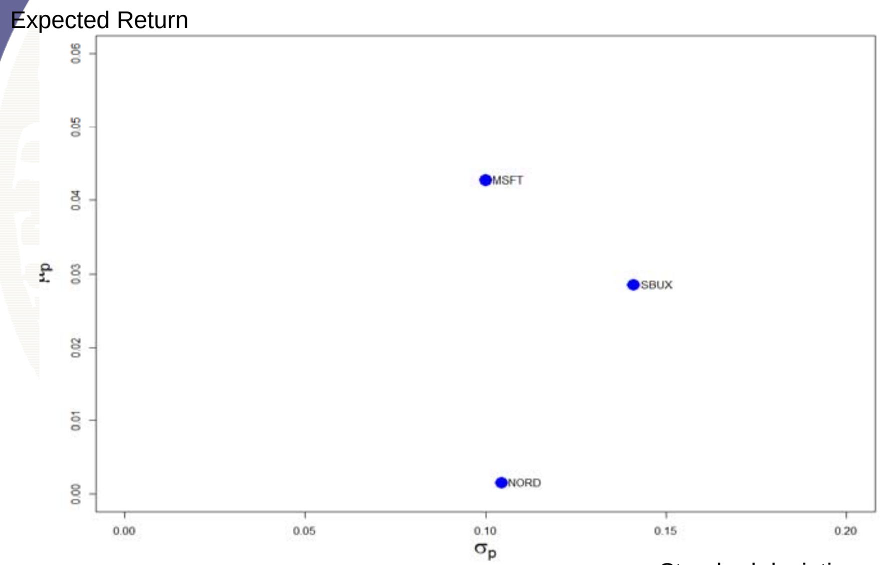
Portfolio Theory with Matrix Algebra

Amath 540/Econ 424
Eric Zivot
Summer 2012
Updated: August 2, 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



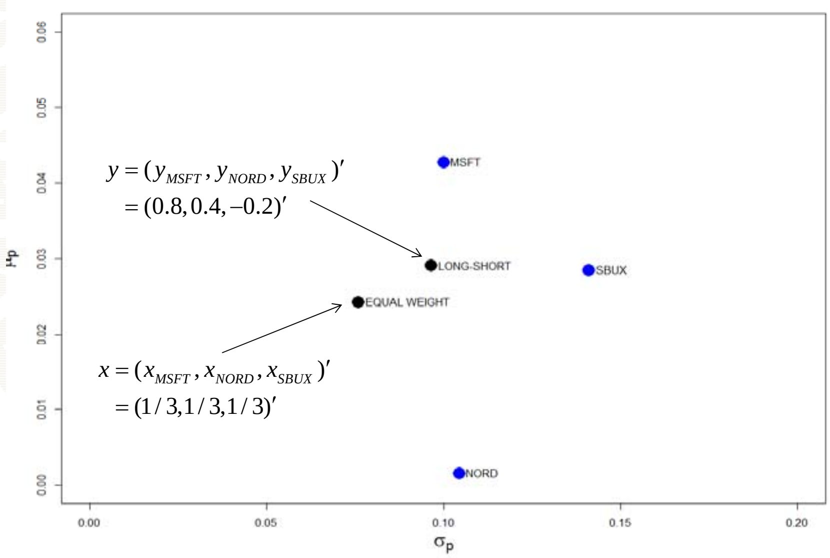
Example Portfolios: Equally Weighted

```
# Equally weighted portfolio
> x.vec = rep(1,3)/3
> names(x.vec) = asset.names
> sum(x.vec)
[1] 1

# Compute mean, variance and std deviation
> mu.p.x = crossprod(x.vec,mu.vec)
> sig2.p.x = t(x.vec)%*%sigma.mat%*%x.vec
> sig.p.x = sqrt(sig2.p.x)
> mu.p.x
        [,1]
[1,] 0.02423
> sig.p.x
        [,1]
[1,] 0.07587
```

Example Portfolios: Long-Short

```
# long-short portfolio
> y.vec = c(0.8, 0.4, -0.2)
> names(y.vec) = asset.names
> sum(y.vec)
[1] 1

# compute mean, variance and std deviation
> mu.p.y = crossprod(y.vec,mu.vec)
> sig2.p.y = t(y.vec)%*%sigma.mat%*%y.vec
> sig.p.y = sqrt(sig2.p.y)
> mu.p.y
        [,1]
[1,] 0.02906

> sig.p.y
        [,1]
[1,] 0.09656
```

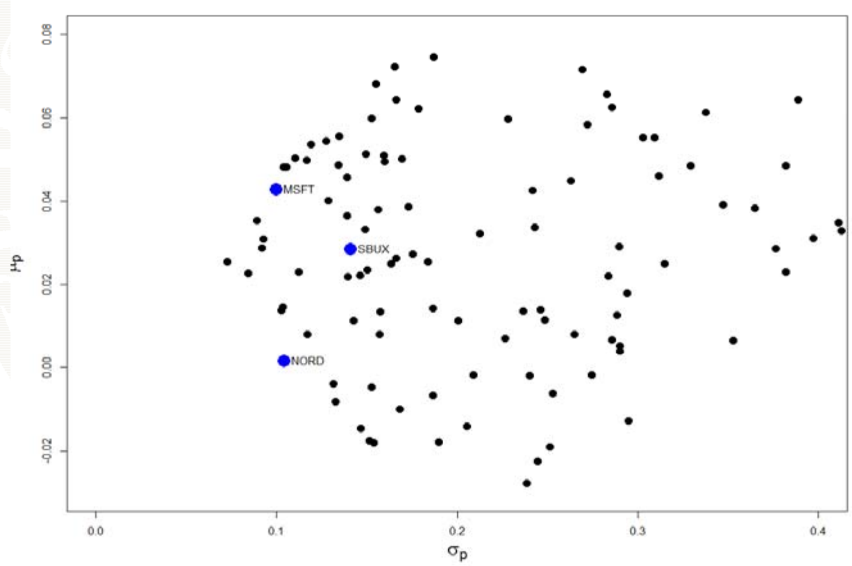

Risk-Return Characteristics: Example Portfolios



Covariance and Correlation of Portfolio Returns

```
# covariance and correlation between equally weighted
# and long-short portfolios
> sig.xy = t(x.vec)%*%sigma.mat%*%y.vec
> sig.xy
         [,1]
[1,] 0.005914

> rho.xy = sig.xy/(sig.p.x*sig.p.y)
> rho.xy
       [,1]
[1,] 0.8073
```


Risk-Return Characteristics: Random Portfolios



Compute Global Minimum Variance Portfolio

```
# method 1: use full system matrix algebra
> top.mat = cbind(2*sigma.mat, rep(1, 3))
> bot.vec = c(rep(1, 3), 0)
> Am.mat = rbind(top.mat, bot.vec)
> b.vec = c(rep(0, 3), 1)
> z.m.mat = solve(Am.mat)%*%b.vec
> m.vec = z.m.mat[1:3,1]
> m.vec
  MSFT   NORD   SBUX
0.4411 0.3656 0.1933
```

$$z_m = A_m^{-1} b,\ A_m = \begin{pmatrix} 2\Sigma & 1 \\ 1' & 0 \end{pmatrix},\ b = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

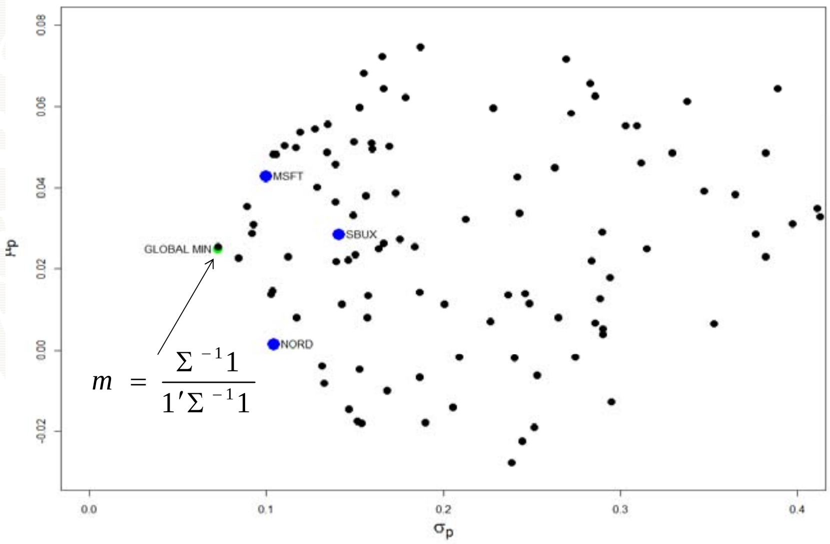
```
# Compute mean, variance and std deviation
> sig2.gmin = as.numeric(t(m.vec)%*%sigma.mat%*%m.vec)
> sig.gmin = sqrt(sig2.gmin)
> sig2.gmin
[1] 0.005282
> sig.gmin
[1] 0.07268
```


Compute Global Minimum Variance Portfolio

```
# method 2: direct calculation of m using matrix algebra
> one.vec = rep(1, 3)
> sigma.inv.mat = solve(sigma.mat)
> top.mat = sigma.inv.mat%*%one.vec
> bot.val = as.numeric((t(one.vec)%*%sigma.inv.mat%*%one.vec))
> m.mat = top.mat/bot.val
> m.mat[,1]
  MSFT   NORD   SBUX
0.4411 0.3656 0.1933
```

$$m = \frac{\Sigma^{-1} 1}{1' \Sigma^{-1} 1}$$

Global Minimum Variance Portfolio



$$m = \frac{\Sigma^{-1}1}{1'\Sigma^{-1}1}$$

Efficient Portfolio with Same Mean as MSFT

```
> top.mat = cbind(2*sigma.mat, mu.vec, rep(1, 3))
> mid.vec = c(mu.vec, 0, 0)
> bot.vec = c(rep(1, 3), 0, 0)
> Ax.mat = rbind(top.mat, mid.vec, bot.vec)
> bmsft.vec = c(rep(0, 3), mu.vec["MSFT"], 1)
> z.mat = solve(Ax.mat)%*%bmsft.vec
> x.vec = z.mat[1:3,]
> x.vec
    MSFT     NORD     SBUX
 0.82745 -0.09075  0.26329
# compute mean, variance and std deviation
> mu.px = as.numeric(crossprod(x.vec, mu.vec))
> mu.px
[1] 0.0427    # μMSFT = 0.0427

> sig2.px = as.numeric(t(x.vec)%*%sigma.mat%*%x.vec)
> sig.px = sqrt(sig2.px)
> sig.px
[1] 0.09166   # σMSFT = 0.10
```

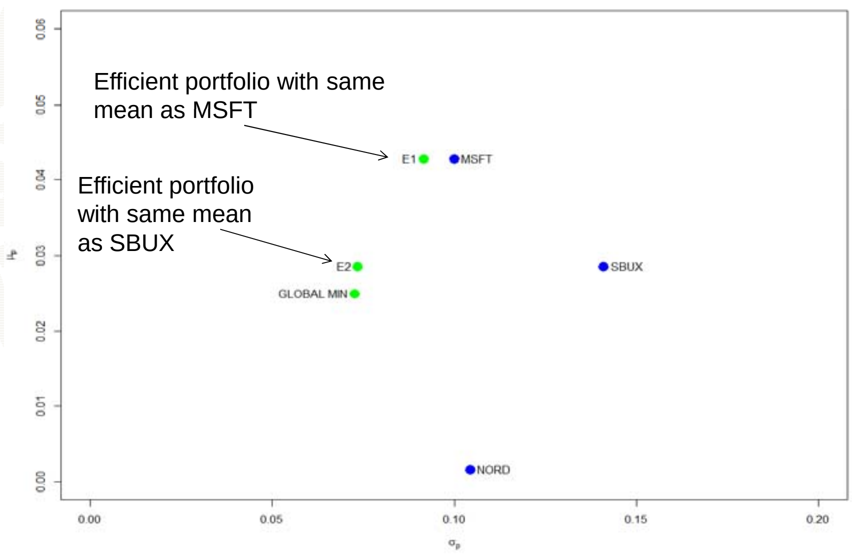
Efficient Portfolio with Same Mean as SBUX

```
> bsbux.vec = c(rep(0, 3), mu.vec["SBUX"], 1)
> z.mat = solve(Ax.mat)%*%bsbux.vec
> y.vec = z.mat[1:3,]
> y.vec
  MSFT   NORD   SBUX
0.5194 0.2732 0.2075

# compute mean, variance and std deviation
> mu.py = as.numeric(crossprod(y.vec, mu.vec))
> sig2.py = as.numeric(t(y.vec)%*%sigma.mat%*%y.vec)
> sig.py = sqrt(sig2.py)
> mu.py
[1] 0.0285     # μSBUX = 0.0285

> sig.py
[1] 0.07355    # σSBUX = 0.1411
```


Covariance between Efficient Portfolio Returns

```
# covariance and correlation between two portfolio returns
> sigma.xy = as.numeric(t(x.vec)%*%sigma.mat%*%y.vec)
> rho.xy = sigma.xy/(sig.px*sig.py)
> sigma.xy
[1] 0.005914

> rho.xy
[1] 0.8772
```

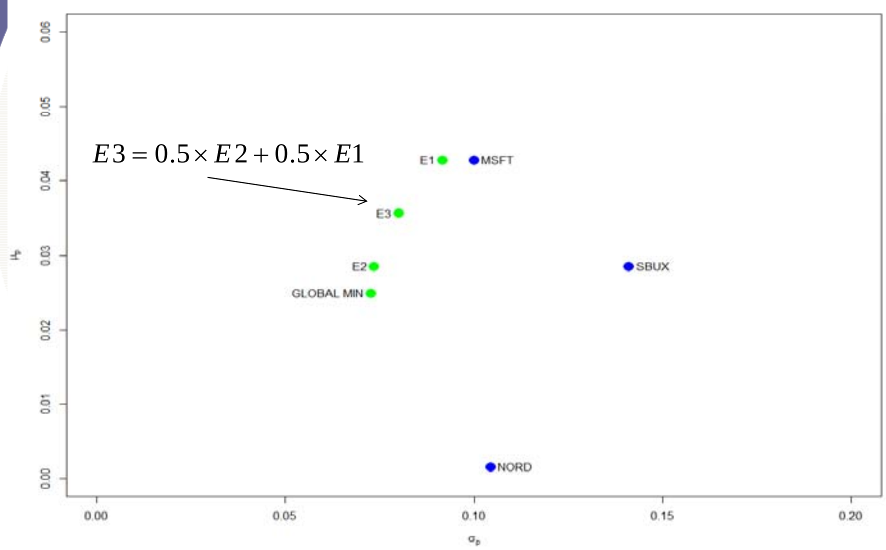
Efficient Portfolios



Find Efficient Portfolio from 2 Efficient Portfolios

```
# Efficient portfolio half way between efficient
# portfolio with same mean as SBUX and efficient
# portfolio with same mean as MSFT
> a = 0.5
> z.vec = a*x.vec + (1-a)*y.vec
> z.vec
   MSFT    NORD    SBUX
0.67342 0.09121 0.23537

# compute mean, variance and std deviation
> sigma.xy = as.numeric(t(x.vec)%*%sigma.mat%*%y.vec)
> mu.pz = as.numeric(crossprod(z.vec, mu.vec))
> sig2.pz = as.numeric(t(z.vec)%*%sigma.mat%*%z.vec)
> sig.pz = sqrt(sig2.pz)
> mu.pz
[1] 0.0356

> sig.pz
[1] 0.08006
```

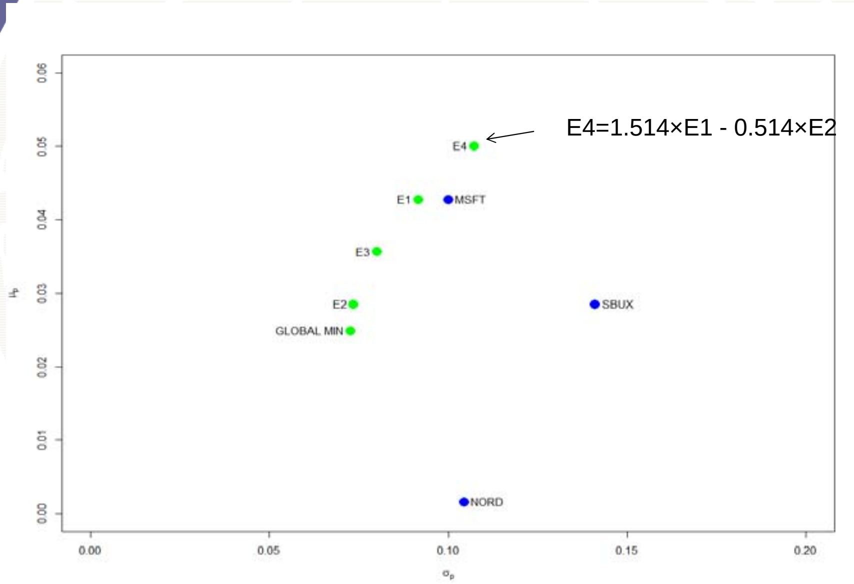
Find Efficient Portfolio from 2 Efficient Portfolios



Compute Efficient Portfolio with ER = 0.05

```
> a.05 = (0.05 - mu.py)/(mu.px - mu.py)
> a.05
[1] 1.514

> z.05 = a.05*x.vec + (1 - a.05)*y.vec
> z.05
   MSFT    NORD    SBUX
 0.9858 -0.2778  0.2920
# compute mean, var and sd
> mu.pz.05 = a.05*mu.px + (1-a.05)*mu.py
> sig2.pz.05 = a.05^2 * sig2.px + (1-a.05)^2 * sig2.py +
+             2*a.05*(1-a.05)*sigma.xy
> sig.pz.05 = sqrt(sig2.pz.05)
> mu.pz.05
[1] 0.05

> sig.pz.05
[1] 0.1072
```

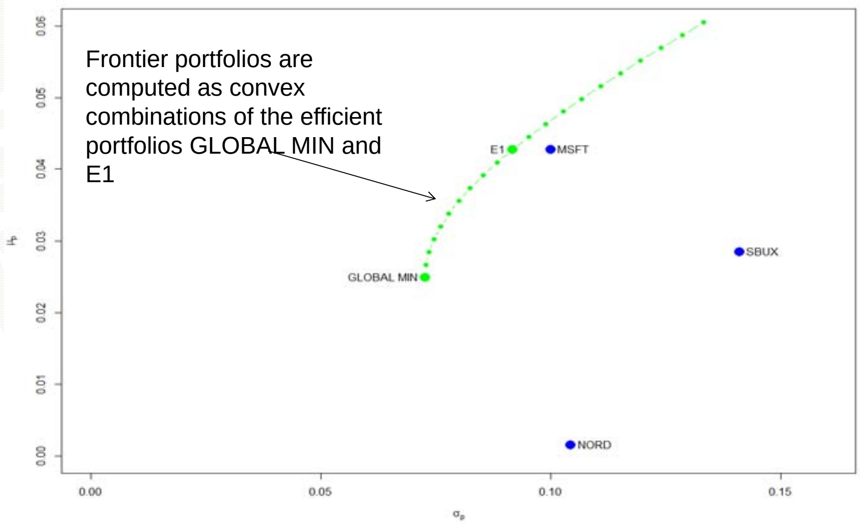
Find Efficient Portfolio with Expected Return 5% from 2 Efficient Portfolios



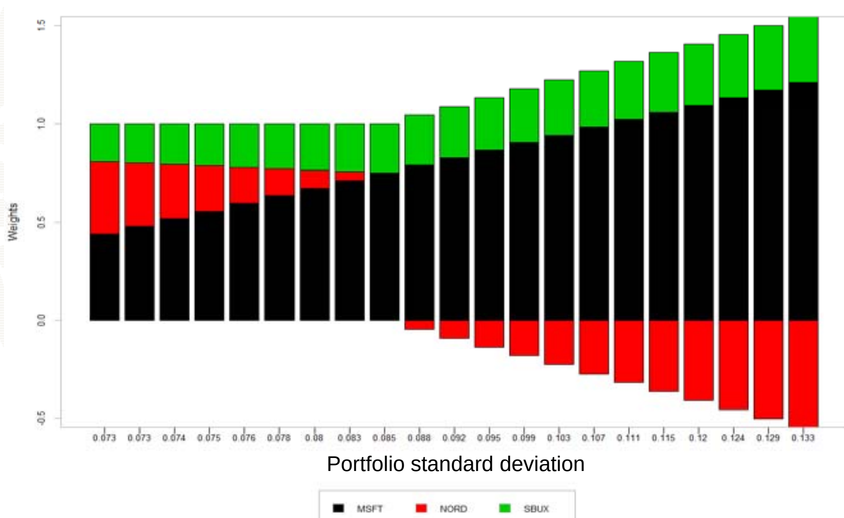
Compute Efficient Frontier

```
# Compute efficient portfolios as convex combinations
# of global min portfolio and efficient portfolio with
# same mean as MSFT
> a = seq(from=1, to=-1, by=-0.1)
> n.a = length(a)
> z.mat = matrix(0, n.a, 3)
> mu.z = rep(0, n.a)
> sig2.z = rep(0, n.a)
> sig.mx = t(m.vec)%*%sigma.mat%*%x.vec
> for (i in 1:n.a) {
+   z.mat[i, ] = a[i]*m.vec + (1-a[i])*x.vec
+   mu.z[i] = a[i]*mu.gmin + (1-a[i])*mu.px
+   sig2.z[i] = a[i]^2 * sig2.gmin + (1-a[i])^2 *
+             sig2.px + 2*a[i]*(1-a[i])*sig.mx
+ }
```

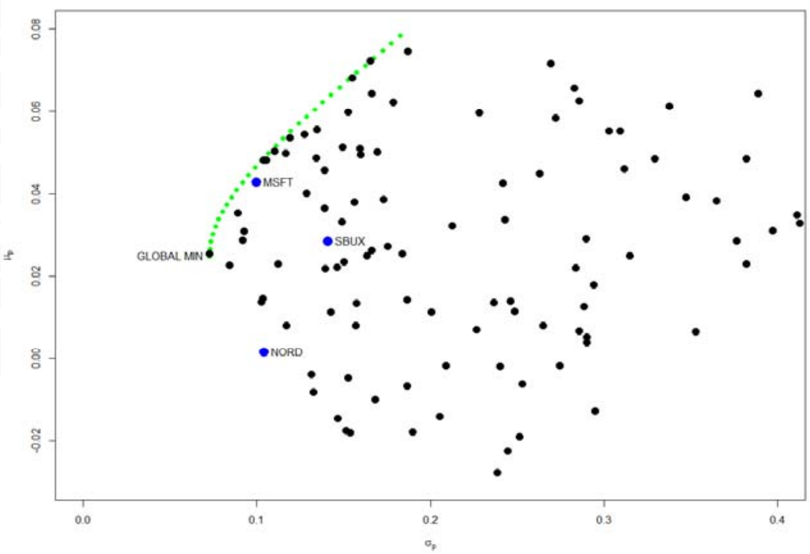
Efficient Frontier



Weights in Efficient Portfolios



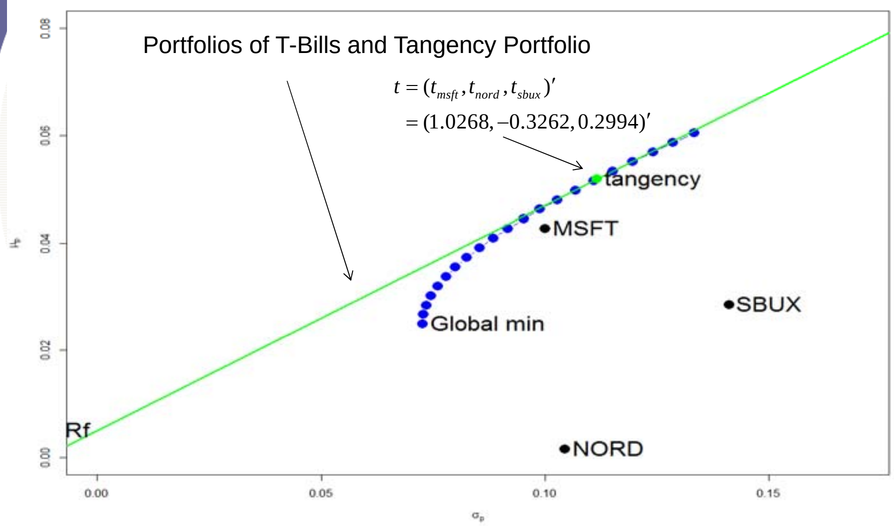
Efficient Frontier with Random Portfolios



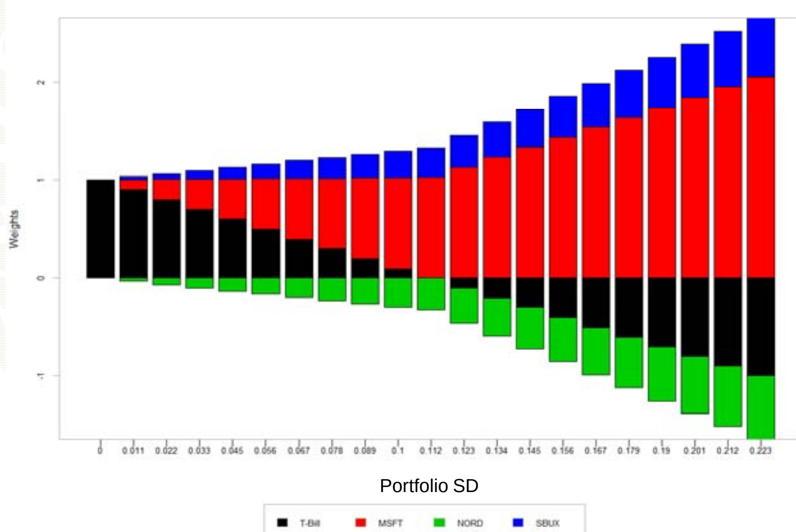
Compute Tangency Portfolio

```
> rf = 0.005
> sigma.inv.mat = solve(sigma.mat)
> one.vec = rep(1, 3)
> mu.minus.rf = mu.vec - rf*one.vec
> top.mat = sigma.inv.mat%*%mu.minus.rf
> bot.val = as.numeric(t(one.vec)%*%top.mat)
> t.vec = top.mat[,1]/bot.val
> t.vec
   MSFT    NORD    SBUX
 1.0268 -0.3263  0.2994

# compute mean, var and sd
> mu.t = as.numeric(crossprod(t.vec, mu.vec))
> sig2.t = as.numeric(t(t.vec)%*%sigma.mat%*%t.vec)
> sig.t = sqrt(sig2.t)
> mu.t
[1] 0.05189

> sig.t
[1] 0.1116
```

Efficient Portfolios with a Risk Free Asset



Efficient Portfolio Weights



Find Efficient Portfolio with Target Volatility

```
# find efficient portfolio with target volatility (risk)
# equal 0.02
> x.t.02 = 0.02/sig.t
> x.t.02
[1] 0.1792
> 1-x.t.02
[1] 0.8208

# shares in msft, nord and sbux
> x.t.02*t.vec
    MSFT      NORD     SBUX
 0.18405 -0.05848  0.05367

# comute mean, var and sd
> mu.t.02 = x.t.02*mu.t + (1-x.t.02)*rf
> sig.t.02 = x.t.02*sig.t
> mu.t.02
[1] 0.0134

> sig.t.02
[1] 0.02
```


Find Efficient Portfolio with Target Expected Return

```
# find efficient portfolio with target expected return
# equal to 0.07
> x.t.07 = (0.07 - rf)/(mu.t - rf)
> x.t.07
[1] 1.386
> 1-x.t.07
[1] -0.3862

# shares in msft, nord and sbux
> x.t.07*t.vec
   MSFT     NORD    SBUX
 1.4234 -0.4523  0.4151

# compute mean, var and sd
> mu.t.07 = x.t.07*mu.t + (1-x.t.07)*rf
> sig.t.07 = x.t.07*sig.t
> mu.t.07
[1] 0.07

> sig.t.07
[1] 0.1547
```

Efficient Portfolios