Rolling Analysis of Portfolios

Eric Zivot

3/28/2021

The GWN Model

Let R_{it} denote the return (cc or simple) on asset i in month t and assume that R_{it} follows GWN model:

$$\begin{aligned} R_{it} &\sim iid\ N(\mu_i, \sigma_i^2), \\ & \quad i = 1, \ldots, N \text{ (assets)} \\ & \quad t = 1, \ldots, T \text{ (months)} \\ cov(R_{it}, R_{jt}) &= \sigma_{ij} \end{aligned}$$

We estimate the GWN model parameters using T months of data using sample statistics giving

$$\hat{\mu}_i, \hat{\sigma}_i^2, \hat{\sigma}_{ij}, \hat{\rho}_{ij}$$

Key assumption: GWN model parameters are constant over time

Diagnostics for Constant Parameters

In the language of hypothesis testing, consider the hypotheses:

$$
\begin{aligned}
H_0 &: \mu_i \text{ is constant over time vs. } H_1 : \mu_i \text{ changes over time} \\
H_0 &: \sigma_i \text{ is constant over time vs. } H_1 : \sigma_i \text{changes over time} \\
H_0 &: \rho_{ij} \text{ is constant over time vs. } H_1 : \rho_{ij} \text{changes over time}
\end{aligned}
$$

- H_0 is the null (maintained) hypothesis and H_1 is the alternative hypothesis
- Formal hypothesis tests use test statistics computed from data to test H_0 against H_1

Diagnostics for Constant Parameters

Remarks:

- Formal test statistics are available but require advanced statistics beyond the level of this course.
  - e.g., see the R package **strucchange**
- Easy to compute informal graphical diagnostics: estimates of μ_i, σ_i and ρ_{ij} over rolling windows of fixed length (rolling estimates)
  - We will use the **zoo** function `rollapply()`

zoo function rollapply()

rollapply() compute functions of data over rolling windows:

```
args(zoo:::rollapply.zoo)
```

```
## function (data, width, FUN, ..., by = 1, by.column = TRUE,
##     na.pad = FALSE, partial = FALSE, align = c("center", "l
##         "right"), coredata = TRUE)
## NULL
```

- data: "zoo" or "xts" time series
- width: integer window width
- FUN: function to be applied over the rolling windows
- by: integer increment to move windows by
- by.column: logical, if TRUE apply FUN to each column of data, otherwise apply FUN> to all columns of data
- align: character, what part of the window gets the time stamp

Example Data

Microsoft, Nordstrom, Starbucks and S&P 500 monthly returns over the 20 year period January 1995 to December 2014.

```
# get data and create monthly returns
data(msftDailyPrices, jwnDailyPrices, sbuxDailyPrices,
     sp500DailyPrices)
gwnDailyPrices = merge(msftDailyPrices, jwnDailyPrices,
                       sbuxDailyPrices, sp500DailyPrices)
gwnMonthlyPrices = to.monthly(gwnDailyPrices, OHLC = FALSE)
gwnReturns = CalculateReturns(gwnMonthlyPrices, method="simple
gwnReturns = gwnReturns["1995::2014"]
colnames(gwnReturns)[2] = "NORD"
```

Rolling Means

Idea: compute estimate of μ_i over rolling windows of length $n < T$:

$$
\begin{aligned}
\hat{\mu}_{it}(n) &= \frac{1}{n}\sum_{j=0}^{n-1} R_{it-j} \\
&= \frac{1}{n}(R_{it} + R_{it-1} + \cdots + R_{it-n+1})
\end{aligned}
$$

- If $H_0 : \mu_i$ is constant is true, then $\hat{\mu}_{it}(n)$ should stay fairly constant over different windows.
- If $H_0 : \mu_i$ is constant is false, then $\hat{\mu}_{it}(n)$ should fluctuate across different windows

Compute 24-month Rolling Means for SBUX

```
roll.muhat = rollapply(gwnReturns[,"SBUX"], width=24,
                       FUN=mean, align="right")
class(roll.muhat)
```

```
## [1] "xts" "zoo"
```

First 24 months are missing (why?)

```
t(roll.muhat[1:5])
```

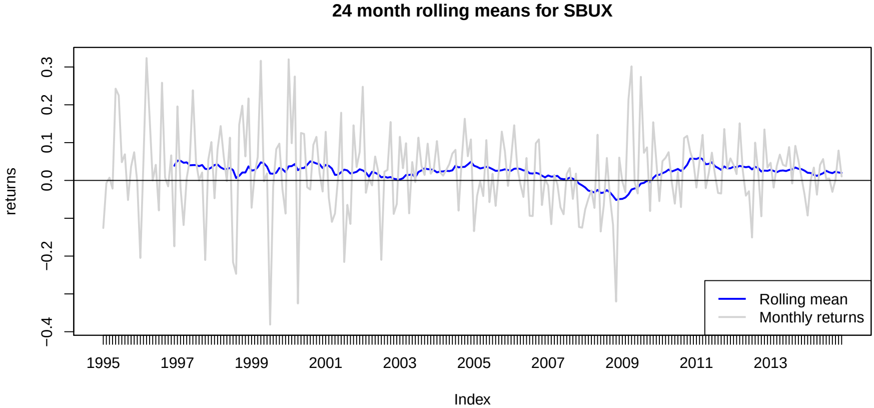
```
##      Jan 1995 Feb 1995 Mar 1995 Apr 1995 May 1995
## SBUX       NA       NA       NA       NA       NA
```

First rolling mean starts Dec 1996 (after first 24 months)

```
t(na.omit(roll.muhat)[1:5])
```

```
##      Dec 1996 Jan 1997 Feb 1997 Mar 1997 Apr 1997
## SBUX   0.0389   0.0523   0.0519   0.0467   0.0478
```

Plot Rolling Means for SBUX



- Rolling means become negative during the financial crisis.
- Rolling means are not constant over the sample $\Longrightarrow$ SBUX returns are not stationary

Rolling means for GWN

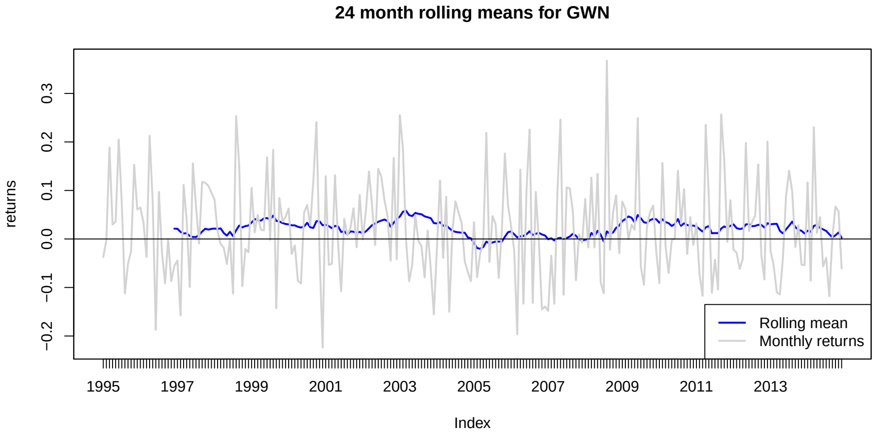
Create GWN with same mean and volatility as SBUX - simulated data has constant mean and volatility

```
set.seed(123)
GWN = rnorm(nrow(gwnReturns[,"SBUX"]),
            mean=mean(gwnReturns[,"SBUX"]),
            sd=sd(gwnReturns[,"SBUX"]))
GWN = xts(GWN, index(gwnReturns))
```

Compute 24-month rolling means:

```
roll.muhat.GWN = rollapply(GWN, width=24,
                           FUN=mean, align="right")
```

Rolling means for GWN



- 24-month rolling means are noisy so it looks like they are changing over time!

Rolling Variances and Standard Deviations

Idea: Compute estimates of σ_i^2 and σ_i over rolling windows of length $n < T$

$$\hat{\sigma}_{it}^2(n) = \frac{1}{n-1}\sum_{j=0}^{n-1}(R_{it-j} - \hat{\mu}_{it}(n))^2$$

$$\hat{\sigma}_{it}(n) = \sqrt{\hat{\sigma}_{it}^2(n)}$$

If $H_0 : \sigma_i$ is constant is true, then $\hat{\sigma}_{it}(n)$ should stay fairly constant over different windows.

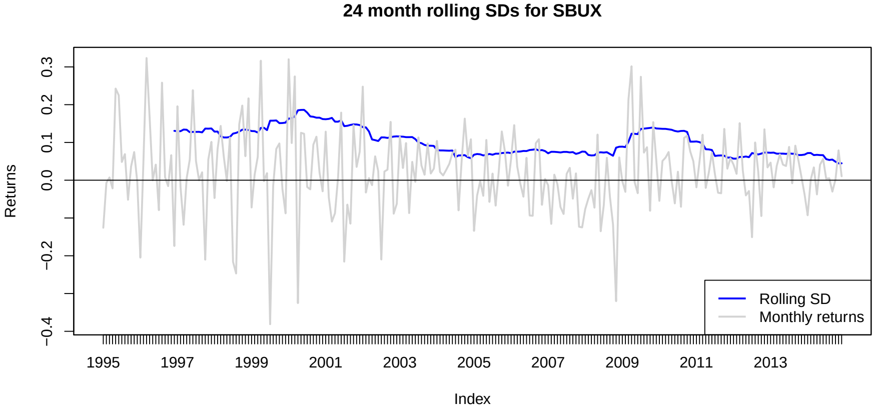
If $H_0 : \sigma_i$ is constant is false, then $\hat{\sigma}_{it}(n)$ should fluctuate across different windows

Compute 24-month Rolling Standard Deviations for SBUX

```
roll.sigmahat = rollapply(gwnReturns[,"SBUX"],width=24,
                          FUN=sd, align="right")
t(na.omit(roll.sigmahat)[1:5])
```

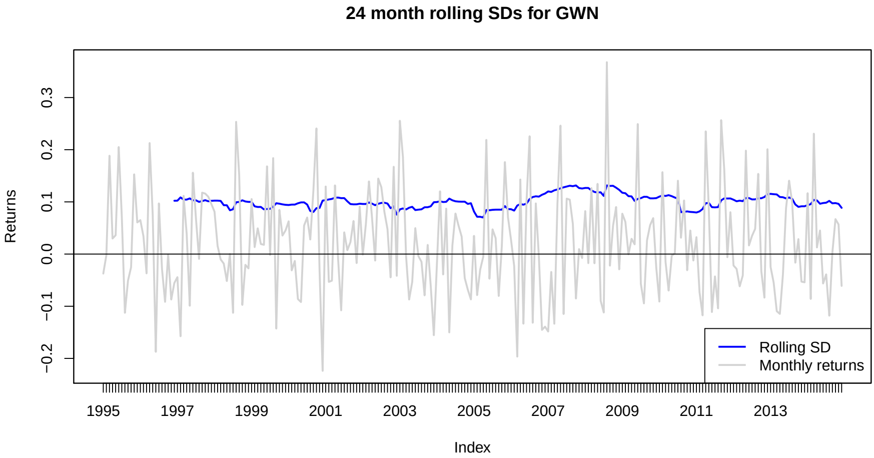
```
##      Dec 1996 Jan 1997 Feb 1997 Mar 1997 Apr 1997
## SBUX    0.131     0.13     0.13    0.134    0.134
```

Plot Rolling Standard Deviations for SBUX



- Rolling SDs fluctuate between 0.1 and 0.2 and increase during crisis periods
- Volatility of SBUX returns does not look constant over time

Rolling Standard Deviations for GWN



- Rolling volatilities fluctuate less than rolling means

24-Month Rolling Estimates with SE bands

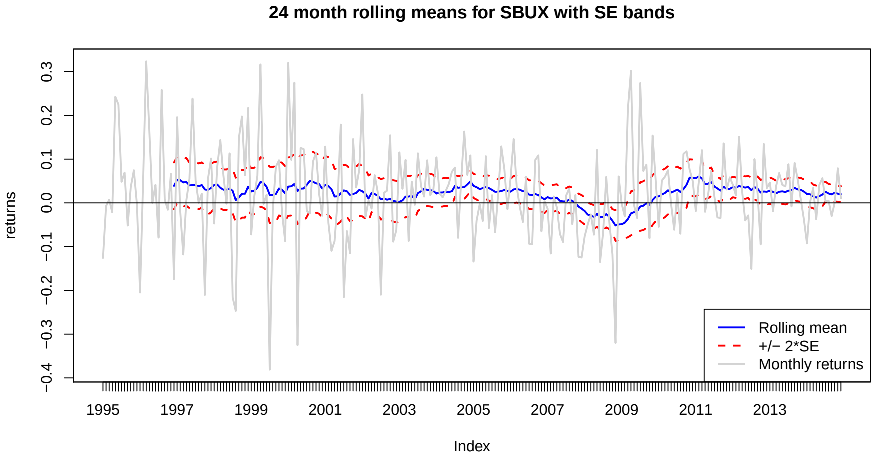
- The rolling estimates appear to show substantial time variation over the sample
- However, one must always keep in mind that estimates have estimation error and part of the observed time variation is due to random estimation error.
- To account for estimation error, rolling estimates are often displayed with estimated standard error bands (i.e., 95% confidence intervals)

24-Month Rolling Estimates with SE bands

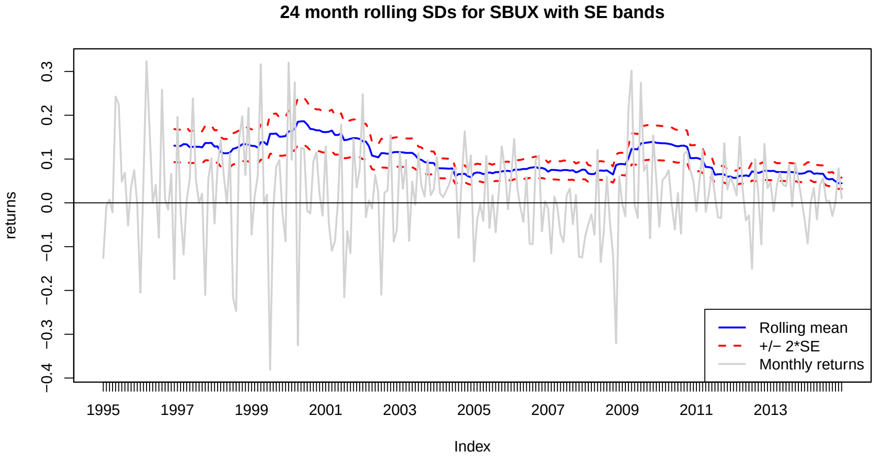
Compute 95% confidence bands for rolling estimates

```
se.muhat.SBUX = roll.sigmahat/sqrt(24)
se.sigmahat.SBUX = roll.sigmahat/sqrt(2*24)
lower.muhat.SBUX = roll.muhat - 2*se.muhat.SBUX
upper.muhat.SBUX = roll.muhat + 2*se.muhat.SBUX
lower.sigmahat.SBUX = roll.sigmahat - 2*se.sigmahat.SBUX
upper.sigmahat.SBUX = roll.sigmahat + 2*se.sigmahat.SBUX
```

24-Month Rolling Means with SE bands



24-Month Rolling Volatilities with SE bands



Rolling Covariances and Correlations

Idea: Compute estimates of σ_{jk} and ρ_{jk} over rolling windows of length $n < T$

$$\hat{\sigma}_{jk,t}(n) = \frac{1}{n-1}\sum_{i=0}^{n-1}(r_{jt-i} - \hat{\mu}_j(n))(r_{kt-i} - \hat{\mu}_k(n))$$

$$\hat{\rho}_{jk,t}(n) = \frac{\hat{\sigma}_{jk,t}(n)}{\hat{\sigma}_{jt}(n)\hat{\sigma}_{kt}(n)}$$

If $H_0 : \rho_{jk}$ is constant is true, then $\hat{\rho}_{jk,t}(n)$ should stay fairly constant over different windows.

If $H_0 : \rho_{jk}$ is constant is false, then $\hat{\rho}_{jk,t}(n)$ should fluctuate across different windows

24-Month rolling correlations between SP500 and SBUX

First, compute function to compute pairwise correlation between two series:

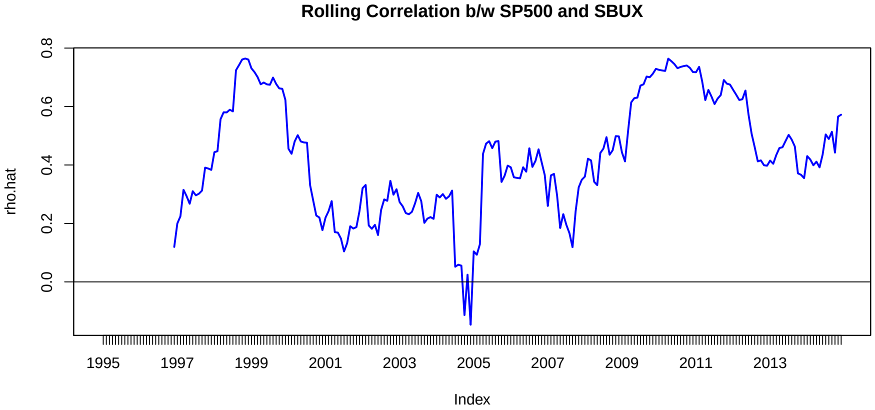
```
rhohat = function(x) {
  cor(x)[1,2]
}
```

Next, call rollapply() with the user-written function rhohat()

```
roll.rhohat = rollapply(gwnReturns[,c("SP500","SBUX")],
                        width=24,FUN=rhohat, by.column=FALSE,
                        align="right")
t(na.omit(roll.rhohat)[1:5])
```

```
##   Dec 1996 Jan 1997 Feb 1997 Mar 1997 Apr 1997
## x     0.12      0.2    0.224    0.315    0.293
```

Rolling correlations between SP500 and SBUX



- Notice how correlations increase during crisis periods (dot-com bust, financial crisis)

Are Mean-Variance Optimized Portfolios Constant Over Time?

We have seen evidence that the parameters of the GWN model for various assets are not constant over time:

- Rolling estimates of μ, σ, and ρ_{ij} show variation over time
- Can show that rolling estimates of Σ show variation over time

Implication: Since estimates of μ, σ, and $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$ are inputs to efficient portfolio calculations, then time variation in $\hat{\mu}$. $\hat{\sigma}$, and $\hat{\sigma}_{ij}$ imply time variation in efficient portfolio weights, expected returns and volatilities.

Rolling Efficient Portfolios

Idea: Using rolling estimates of μ and Σ compute rolling efficient portfolios

- global minimum variance portfolio
- efficient portfolio for target return
- tangency portfolio
- efficient frontier

Look at time variation in resulting portfolio weights, expected returns and volatilities

- Time variation has implications for portfolio rebalancing

Rolling Estimates of μ and Σ

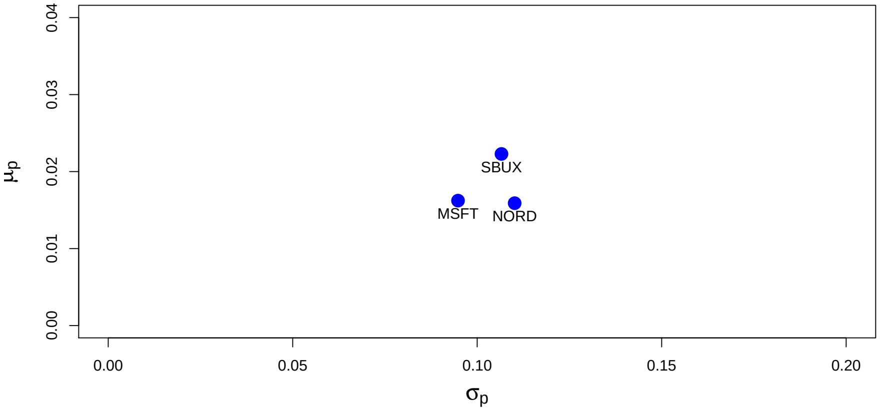
Let $\mathbf{R}_t$ denote the $k \times 1$ vector of asset returns in month t. Compute estimate of μ and Σ over rolling windows of length $n < T$:

$$
\begin{aligned}
\hat{\mu}_t(n) &= \frac{1}{n}\sum_{j=0}^{n-1}\mathbf{R}_{t-j} \\
&= \frac{1}{n}(\mathbf{R}_t + \mathbf{R}_{t-1} + \cdots + \mathbf{R}_{t-n+1}) \\
\hat{\Sigma}_t(n) &= \frac{1}{n}\sum_{j=0}^{n-1}\left(\mathbf{R}_{t-j} - \hat{\mu}_t(n)\right)\left(\mathbf{R}_{t-j} - \hat{\mu}_t(n)\right)'
\end{aligned}
$$

Estimated inputs to portfolio theory: full sample

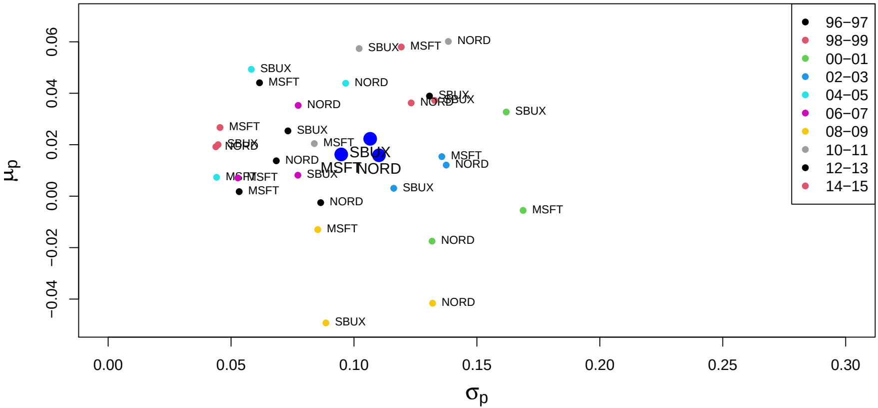
```
nobs = nrow(gwnReturns)
stockNames = colnames(gwnReturns)[-4]
muhat.vals = colMeans(gwnReturns[, stockNames])
sigmahat.vals = apply(gwnReturns[, stockNames],2,sd)
cov.mat = var(gwnReturns[, stockNames])
cor.mat = cor(gwnReturns[, stockNames])
```

Risk-return Tradeoff: Full 12 Year Sample



- Starbucks is best, followed by Microsoft and Nordstrom

Risk-return Tradeoff: Every 2-Years



Rolling Global Minimum Variance Portfolio

Idea: compute estimates of portfolio weights $\mathbf{m}$ over rolling windows of length $n < T$:

$$\min_{\mathbf{m}(n)} \; \mathbf{m}_t(n)'\hat{\Sigma}_t(n)\mathbf{m}_t(n) \;\; \text{s.t. } \mathbf{m}_t(n)'\mathbf{1} = 1$$

$$t = n, \ldots, T$$

$$\hat{\Sigma}_t(n) = \text{rolling estimate of } \Sigma \text{ in month } t$$

If $\hat{\Sigma}_n(n) \approx \hat{\Sigma}_{n+1}(n) \approx \cdots \approx \hat{\Sigma}_T(n)$, then

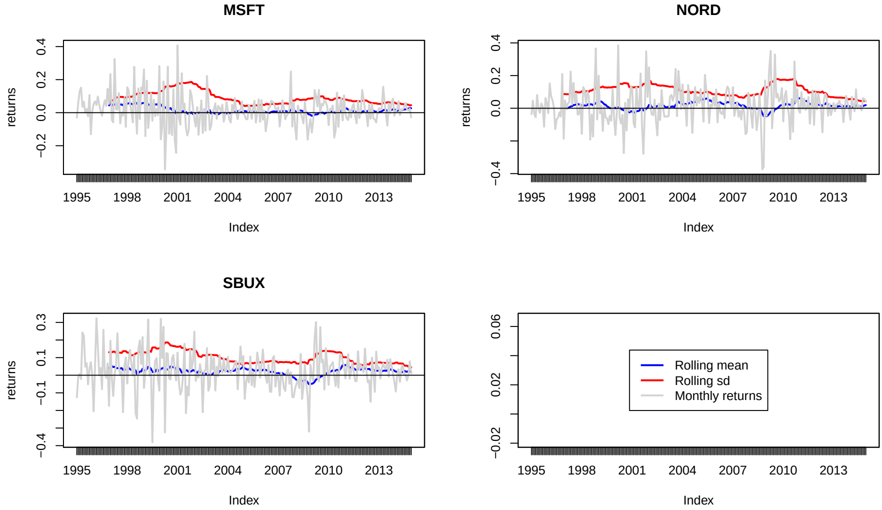
$$\mathbf{m}_n(n) \approx \mathbf{m}_{n+1}(n) \approx \cdots \approx \mathbf{m}_T(n)$$

24-month Rolling Means and Volatilities

- First, survey the evidence for time variation in means and volatilities of the three assets
- Use **zoo** function `rollapply()` to compute 24-month rolling means and volatilities for MSFT, NORD and SBUX.

```
roll.muhat = rollapply(gwnReturns[,stockNames], width=24,
                       FUN=mean, align="right")
roll.sigmahat = rollapply(gwnReturns[,stockNames],width=24,
                          FUN=sd, align="right")
```

24-month Rolling Means and Volatilities



24-month Rolling Correlations

- Next, survey the evidence for time varying correlations. Here there are three pair-wise correlations: msft-nord, msft-sbux, and nord-sbux.
- The following function extracts these pair-wise correlations from the estimated correlation matrix.

```
roll.cor = function(x) {
  cor.hat = cor(x)
  cor.vals = cor.hat[lower.tri(cor.hat)]
  names(cor.vals) = c("msft.nord","msft.sbux","nord.sbux")
  return(cor.vals)
}
```

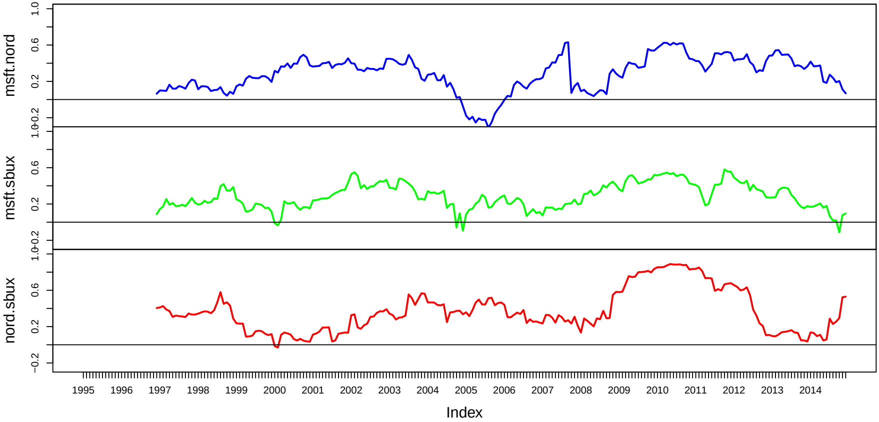
24-month Rolling Correlations

Compute all pair-wise 24-month rolling correlations using `rollapply()`

```
roll.cor.vals = rollapply(gwnReturns[,stockNames],
                          width=24,
                          by.column=FALSE,
                          FUN=roll.cor,
                          align="right")
t(na.omit(roll.cor.vals)[1:3, ])
```

```
##             Dec 1996 Jan 1997 Feb 1997
## msft.nord     0.0646    0.100   0.0981
## msft.sbux     0.0880    0.144   0.1716
## nord.sbux     0.4044    0.412   0.4259
```

24-month Rolling Correlations



Global Minimum Variance Portfolio: Full Sample

```
gmin.full = globalMin.portfolio(er=muhat.vals,cov.mat=cov.mat)
gmin.full
```

```
## Call:
## globalMin.portfolio(er = muhat.vals, cov.mat = cov.mat)
##
## Portfolio expected return:     0.0179
## Portfolio standard deviation:  0.0756
## Portfolio weights:
##  MSFT  NORD  SBUX
## 0.459 0.250 0.291
```

- Full sample global minimum variance portfolio is close to equally weighted

24-month Rolling Global Min Var Portfolio

Use the **IntroCompfinR** function `globalMin.portfolio()` to return the global minimum variance portfolio weights, means and volatilities for each rolling window.

```
rollGmin = function(x) {
    mu.hat = colMeans(x)
    cov.hat = var(x)
    gmin = globalMin.portfolio(er=mu.hat,cov.mat=cov.hat)
    ans = c(gmin$er,gmin$sd,gmin$weights)
    names(ans)[1:2] = c("er","sd")
    return(ans)
}
```

24-month Rolling Global Min Var Portfolio

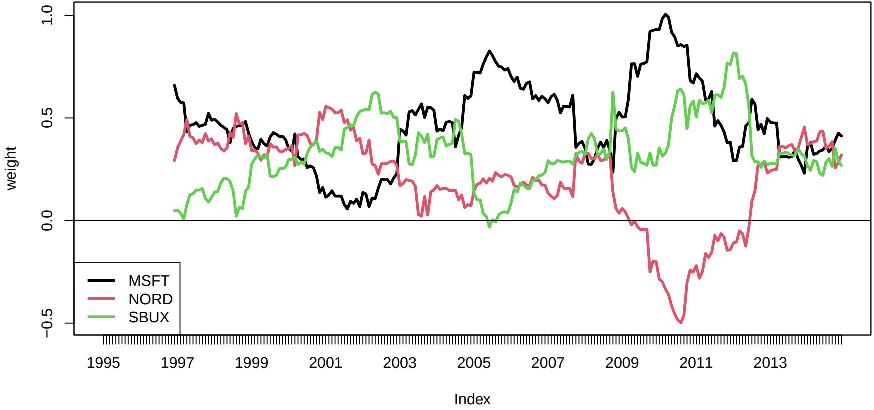
Now use `rollapply()` with `rollGmin()` to do the calculations.

```
roll.gmin = rollapply(gwnReturns[, stockNames], width=24,
                      by.column=FALSE,align="right",
                      FUN=rollGmin)

na.omit(roll.gmin)[1:3, ]
```

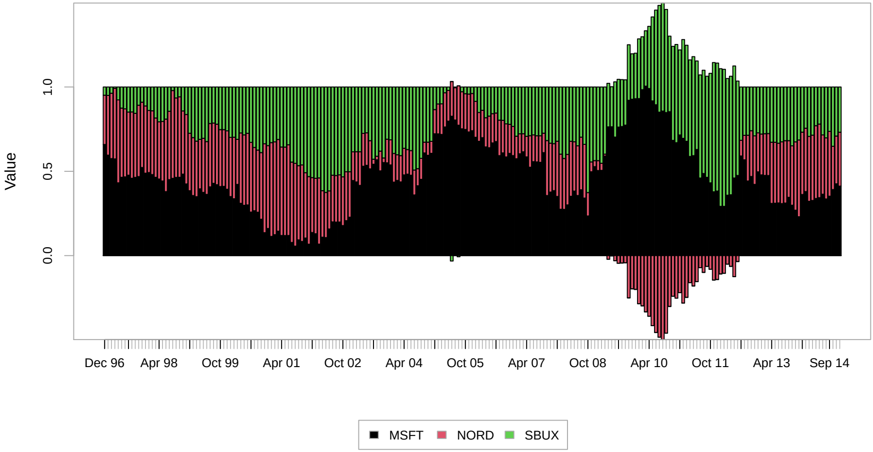
```
##              er     sd  MSFT  NORD   SBUX
## Dec 1996 0.0302 0.0513 0.659 0.292 0.0486
## Jan 1997 0.0358 0.0571 0.596 0.354 0.0496
## Feb 1997 0.0306 0.0584 0.575 0.388 0.0370
```

24-month Rolling Global Min Var Portfolio Weights

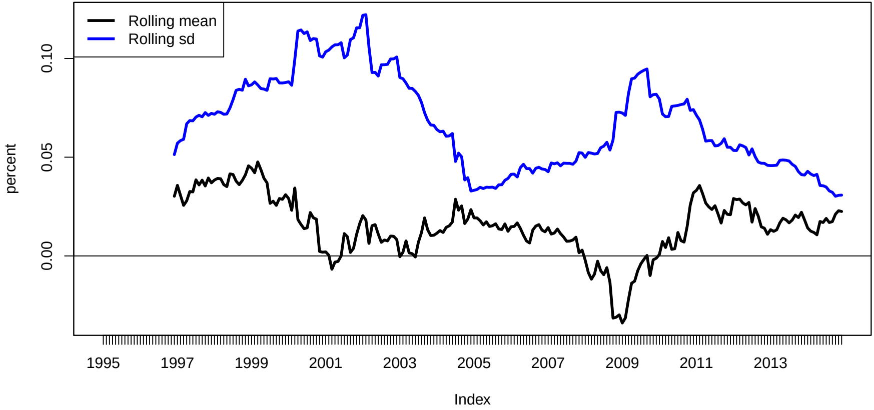


- Rolling portfolio weights change a lot over time!

24-month Rolling Global Min Var Portfolio Weights



Rolling Global Min Var Port means and vols



- Notice how mean and volatility move in opposite directions: when mean goes down vol goes up!

Rolling Efficient Portfolios

Idea: compute estimates of portfolio weights $\mathbf{x}$ over rolling windows of length $n < T$ for $t = n, \ldots, T$:

$$\min_{\mathbf{x}(n)} \ \mathbf{x}_t(n)'\hat{\Sigma}_t(n)\mathbf{x}_t(n)$$

$$\text{s.t. } \mathbf{x}_t(n)'\mathbf{1} = 1, \mathbf{x}_t(n)'\hat{\mu}_t(n) = \mu_p^{\text{target}}$$

$$\hat{\mu}_t(n) = \text{ rolling estimate of } \mu \text{ in month } t$$

$$\hat{\Sigma}_t(n) = \text{rolling estimate of } \Sigma \text{ in month } t$$

Rolling Efficient Portfolios

If

$$\hat{\mu}_n(n) \approx \hat{\mu}_{n+1}(n) \approx \cdots \approx \hat{\mu}_T(n)$$
$$\hat{\Sigma}_n(n) \approx \hat{\Sigma}_{n+1}(n) \approx \cdots \approx \hat{\Sigma}_T(n)$$

then

$$\mathbf{x}_n(n) \approx \mathbf{x}_{n+1}(n) \approx \cdots \approx \mathbf{x}_T(n)$$

Efficient Portfolio with Target Return 2%: Full Sample

```
eport.01 = efficient.portfolio(er=muhat.vals,cov.mat=cov.mat,
                               target.return=0.02)
eport.01
```

```
## Call:
## efficient.portfolio(er = muhat.vals, cov.mat = cov.mat, tar
##
## Portfolio expected return:     0.02
## Portfolio standard deviation:  0.0833
## Portfolio weights:
##   MSFT   NORD   SBUX
## 0.3044 0.0693 0.6262
```

Full sample portfolio is a long only with most weight in Microsoft and Starbucks.

24-month Rolling Efficient Portfolio with Target Return 2%

Use **IntroCompfinR** function `efficient.portfolio()` to return the efficient portfolio weights, means and volatilities for each rolling window:

```
rollefficient = function(x,target=0.02) {
    mu.hat = colMeans(x)
    cov.hat = var(x)
    eport = efficient.portfolio(er=mu.hat,
                                cov.mat=cov.hat,
                                  target.return=target)
    ans = c(eport$er,eport$sd,eport$weights)
    names(ans)[1:2] = c("er","sd")
    return(ans)
}
```

Rolling Efficient Portfolio with Target Return 2%

Now use `rollapply()` with `rollefficient()` to do the calculations.

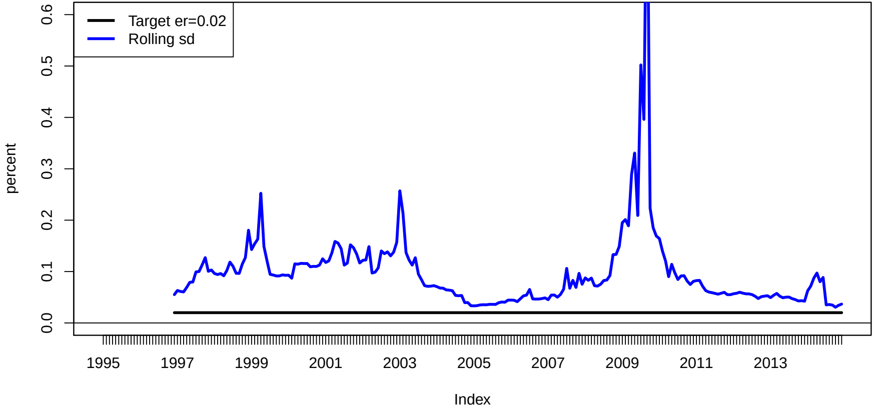
```
roll.eport = rollapply(gwnReturns[,stockNames], width=24,
                       by.column=F,align="right",
                       FUN=rollefficient)
na.omit(roll.eport)[1:3, ]
```

```
##            er     sd  MSFT  NORD    SBUX
## Dec 1996 0.02 0.0551 0.507 0.520 -0.0262
## Jan 1997 0.02 0.0631 0.412 0.651 -0.0631
## Feb 1997 0.02 0.0613 0.459 0.590 -0.0494
```

Rolling Efficient Portfolio with Target Return 2%

- Efficient portfolio weights change a lot over time and become unstable during crisis period

Rolling Efficient Portfolio means and volatilities



- Very unstable volatilities!