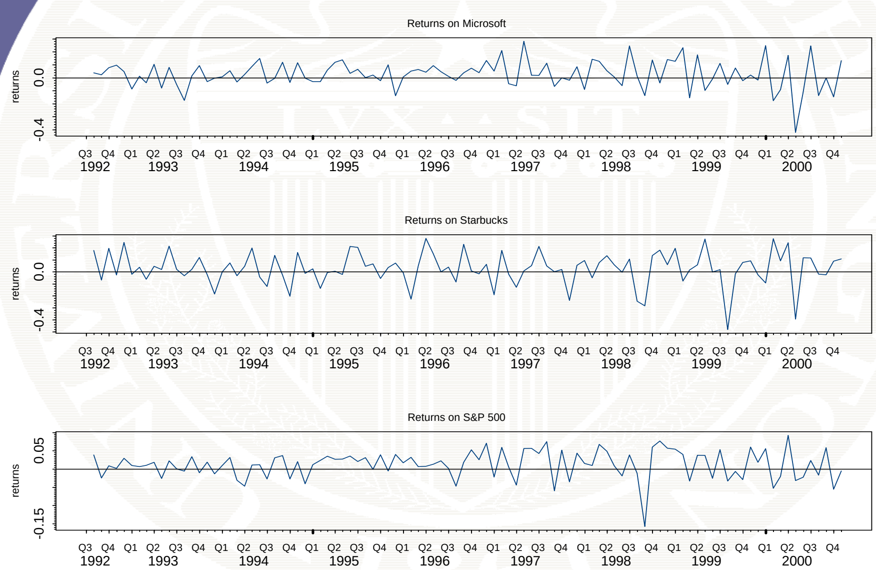
# Bootstrapping Estimates of the CER Model

Econ 424/Amath 540
Eric Zivot
Summer 2012
Updated: July 17, 2012



## Monthly Returns on MSFT, SBUX and S&P 500



## Estimated Standard Errors

```
> se.muhat = sigmahat.vals/sqrt(nobs)
> rbind(muhat.vals,se.muhat)
              sbux      msft      sp500
muhat.vals    0.0277    0.0275    0.01253
  se.muhat    0.0135    0.0106    0.00378
```

```
> se.sigma2hat = sigma2hat.vals/sqrt(nobs/2)
> rbind(sigma2hat.vals,se.sigma2hat)
                   sbux       msft       sp500
sigma2hat.vals     0.01845    0.01141    0.00143
  se.sigma2hat     0.00261    0.00161    0.00020
```

```
> se.sigmahat = sigmahat.vals/sqrt(2*nobs)
> rbind(sigmahat.vals,se.sigmahat)
                 sbux       msft      sp500
sigmahat.vals    0.1358     0.1068    0.0378
  se.sigmahat    0.0096     0.0075    0.0026
```



## R function **sample()**

```
# random permutations of the index vector 1:5
> sample(5)
[1] 1 3 2 5 4

> sample(5)
[1] 4 2 3 5 1

# random sample of size 5 from MSFT return with replacement
> sample(MSFT, 5, replace=TRUE)
[1] -0.02904  0.12130 -0.01890 -0.15332 -0.14627
```

## Brute Force Bootstrap

Same idea as Monte Carlo Simulation but instead of generating random data from an assumed distribution, you generate random data by sampling with replacement from the observed data

```
> B = 999 # why use 999?
> muhat.boot = rep(0, B)
> nobs = length(MSFT)
> for (i in 1:B) {
+  boot.data = sample(MSFT, nobs, replace=TRUE)
+  muhat.boot[i] = mean(boot.data)
}
```

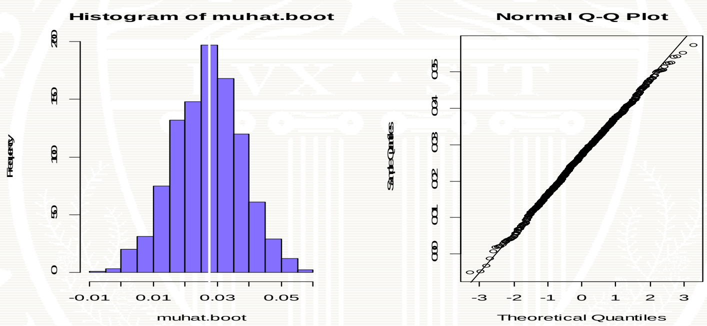


## Brute Force Bootstrap

```
# bootstrap bias
> mean(muhat.boot) - muhat.MSFT
[1] -0.0005643

# bootstrap SE
> sd(muhat.boot)
[1] 0.01045

# analytic SE
> sigmahat.MSFT/sqrt(length(MSFT))
[1] 0.01068
```

Bootstrap SE is very close to analytic SE

# Brute Force Bootstrap




```
par(mfrow=c(1,2))
  hist(muhat.boot, col="slateblue1")
  abline(v=muhat.MSFT, col="white", lwd=2)
  qqnorm(muhat.boot)
  qqline(muhat.boot)
par(mfrow=c(1,1))
```



# R Package boot

- Implements a variety of bootstrapping functions
- Background material is book by Davidson and Hinkley, *Bootstrap Methods and Their Application*, Cambridge University Press, 1997.
- Main functions are:
  - `boot()` bootstrap user supplied function
  - `boot.ci()` compute bootstrap confidence interval

## Example: Bootstrapping sample mean

```
# function for bootstrapping sample mean
mean.boot = function(x, idx) {
# arguments:
# x        data to be resampled
# idx      vector of scrambled indices created
#          by boot() function
# value:
# ans      mean value computed using resampled
#          data
     ans = mean(x[idx])
     ans
}
```



## Example: Bootstrapping sample mean

```
> MSFT.mean.boot = boot(MSFT, statistic = mean.boot, R=999)
> class(MSFT.mean.boot)
[1] "boot“
```

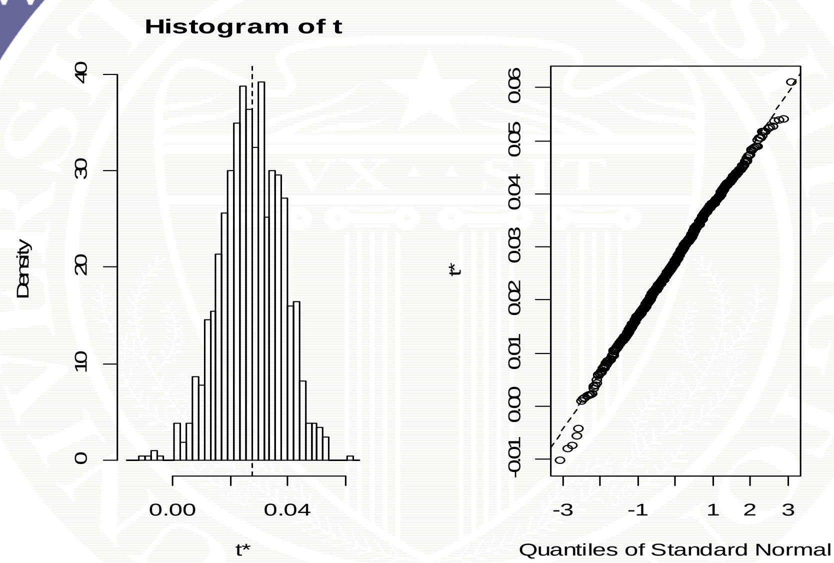
Number of bootstrap samples

```
> MSFT.mean.boot

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = MSFT, statistic = mean.boot, R = 999)


Bootstrap Statistics :
      original        bias     std. error
t1*   0.02756      -0.00013   0.01052
```

Sample mean

Bootstrap estimate of bias

Bootstrap estimate of SE




```
> plot(MSFT.mean.boot)
```



## Compare Bootstrap Statistics with Analytic Formulas

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = MSFT, statistic = mean.boot, R = 999)


Bootstrap Statistics :
      original        bias     std. error
t1*   0.02756      -0.00013    0.01052

# compare boot SE with analytic SE
> se.muhat.MSFT = sigmahat.MSFT/sqrt(length(MSFT))
> se.muhat.MSFT

[1] 0.01068
```

# Bootstrap Confidence Intervals

```
> boot.ci(MSFT.mean.boot, conf = 0.95, type =
+         c("norm","perc"))
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = MSFT.mean.boot, conf = 0.95, type =
        c("norm", "perc"))

Intervals :
Level      Normal              Percentile
95%   ( 0.0071,  0.0483 )   ( 0.0065,  0.0471 )
Calculations and Intervals on Original Scale
```



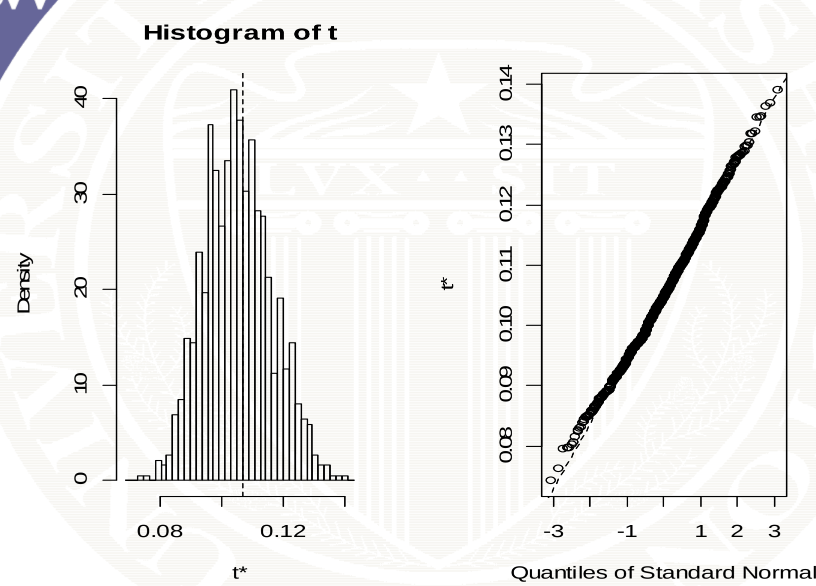
# Example: Bootstrapping Sample SD

```
# function for bootstrapping sample standard deviation
sd.boot = function(x, idx) {
# arguments:
# x          data to be resampled
# idx        vector of scrambled indices created by
#            boot() function
# value:
# ans        sd value computed using resampled data
     ans = sd(x[idx])
     ans
}
```

# Example: Bootstrapping Sample SD

```
> MSFT.sd.boot = boot(MSFT, statistic = sd.boot, R=999)
> MSFT.sd.boot

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = MSFT, statistic = sd.boot, R = 999)


Bootstrap Statistics :
     original     bias      std. error
t1*  0.1068     -0.00145   0.01078

# compare boot SE with analytic SE based on CLT
> se.sigmahat.MSFT = sigmahat.MSFT/sqrt(2*length(MSFT))
> se.sigmahat.MSFT

[1] 0.00755
```






```
> plot(MSFT.sd.boot)
```

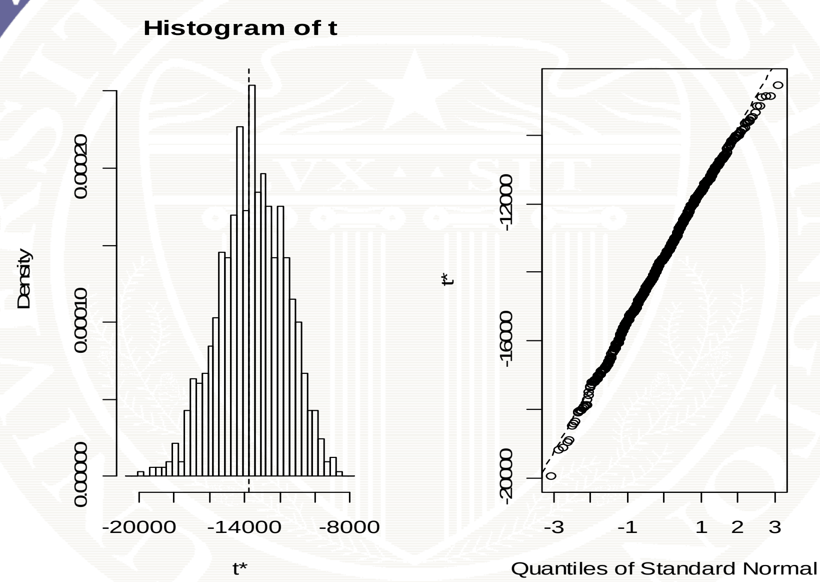
## Example: Boostrapping Normal VaR

```
ValueAtRisk.boot = function(x, idx, p=0.05, w=100000) {
# x.mat     data to be resampled
# idx       vector of scrambled indices created by
#           boot() function
# p         probability value for VaR calculation
# w         value of initial investment
# value:
# ans       Value-at-Risk computed using resampled data

      q = mean(x[idx]) + sd(x[idx])*qnorm(p)
      VaR = (exp(q) - 1)*w
      VaR
}
```



## Example: Boostrapping Normal VaR

```
> MSFT.VaR.boot

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = MSFT, statistic = ValueAtRisk.boot, R = 999)


Bootstrap Statistics :
     original    bias    std. error
t1* -13769.40 210.2801    1886.953
```




```
> plot(MSFT.VaR.boot)
```



# Example: Boostrapping Normal VaR

```
> boot.ci(MSFT.VaR.boot, conf=0.95, type=c("norm", "perc"))

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = MSFT.VaR.boot, conf = 0.95, type =
c("norm", "perc"))

Intervals :
Level      Normal              Percentile
95%   (-17678, -10281 )   (-17212, -10009 )
```

$\hat{\theta} \pm 2 \times SE_{boot}(\hat{\theta})$  $[q^*_{.025},\ q^*_{.975}]$