The Bootstrap

Eric Zivot

4/12/2021

Bootstrapping

Motivation

- Before modern computers, doing statistical analysis involved using mathematics and probability theory to derive statistical formulas for standard errors and confidence intervals. Often these formulas are approximations that rely on large samples
- With modern computers and statistical software, resampling methods (e.g.bootstrapping) can be used to produce standard errors and confidence intervals without the use of formulas that are often more reliable than statistical formulas
- Straightforward extension of Monte Carlo simulation

Advantages of Boostrapping

- Fewer assumptions
  - Resample from observed data *as if* resampling from the population (e.g. Monte Carlo simulation)
  - Do not need to assume a probability distribution for the data (e.g. normally distributed)
- Greater accuracy
  - Do not rely on very large sample sizes in contrast to Central Limit Theorem
- Generality
  - Same method applies to a wide variety of statistics

Bootstrapping in the GWN Model

GWN Model:

$$R_t = \mu + \epsilon_t \quad t = 1, \cdots, T$$
$$\epsilon_t \sim \text{iid } N(0, \sigma^2)$$

Observed sample: $\{R_1, \ldots, R_T\}$

Goal: Compute $\widehat{\mathrm{SE}}(\hat{\mu})$ and $\widehat{\mathrm{SE}}(\hat{\sigma})$ and 95% confidence intervals for μ and σ using the bootstrap. The analytic formulas are

$$\widehat{\mathrm{SE}}(\hat{\mu}) = \frac{\hat{\sigma}}{\sqrt{T}}, \widehat{\mathrm{SE}}(\hat{\sigma}) \approx \frac{\hat{\sigma}}{\sqrt{2T}}$$
$$\hat{\theta} \pm 2 \cdot \widehat{\mathrm{SE}}(\hat{\theta}),\ \hat{\theta} = \hat{\mu},\ \hat{\sigma}$$

Procedure for Nonparametric Bootstrapping

1. Resample - create B bootstrap samples by sampling *with replacement* from the original data. Each bootstrap sample has T observations (same as original sample)

$$
\begin{aligned}
\{R_{11}^*, \ldots, R_{1T}^*\} &= \text{1st bootstrap sample} \\
&\vdots \\
\{R_{B1}^*, \ldots, R_{BT}^*\} &= \text{Bth bootstrap sample}
\end{aligned}
$$

2. Calculate bootstrap distribution for statistic of interest $\hat{\theta}$ - for each bootstrap sample compute $\hat{\theta}^*$. There will be B values of $\hat{\theta}^* : \{\hat{\theta}_1^*, \ldots, \hat{\theta}_B^*\}$
3. Use the bootstrap distribution - the bootstrap distribution gives information about the shape, center and spread of the unknown pdf $f(\hat{\theta})$

Bootstrap Estimates

Bootstrap estimate of bias:

$$\mathrm{bias}_{boot}(\hat{\theta},\theta)=\frac{1}{B}\sum_{j=1}^{B}\hat{\theta}_j^{*}-\hat{\theta}$$

bootstrap mean - estimate

Bootstrap estimate of $\widehat{\mathrm{SE}}(\hat{\theta})$

$$\widehat{\mathrm{SE}}_{boot}(\hat{\theta})=\sqrt{\frac{1}{B-1}\sum_{j=1}^{B}\left(\hat{\theta}_j^{*}-\frac{1}{B}\sum_{j=1}^{B}\hat{\theta}_j^{*}\right)^{2}}$$

sample SD of bootstrap values $\{\hat{\theta}_1^{*},\ldots,\hat{\theta}_B^{*}\}$

Bootstrap 95% Confidence Intervals

- If bootstrap distribution is symmetric and looks normal use

$$\hat{\theta} \pm 2 \cdot \widehat{\mathrm{SE}}_{boot}(\hat{\theta})$$

- If bootstrap distribution is not symmetric and looks non-normal use

$$[q^*_{.025},\ q^*_{.975}]$$
$$q^*_{.025} = \text{ 2.5\% quantile from bootstrap distribution}$$
$$q^*_{.975} = \text{ 97.5\% quantile from bootstrap distribution}$$

Performing the Bootstrap in R

- Brute force using R function `sample()`
- R package **boot**

Brute force:

Perform the steps of the bootstrap one-by-one in R:

1. Sample with replacement from original data using R function `sample()` B times
2. Compute statistic of interest from B bootstrap samples using for loop or `apply()` function
3. Compute bootstrap bias and SE from bootstrap samples using R functions `mean()` and `sd()`
4. If necessary, compute bootstrap quantiles using R function `quantile()`

Example - R function `sample()`

Consider random sampling from the vector `R=c(0.1,0.05,-0.02,0.03,-0.04)`. Method 1: Use `sample()` to create a sampling vector.

```
r = c(0.1,0.05,-0.02,0.03,-0.04)
set.seed(123)
idx = sample(5, replace=TRUE)
idx
```

```
## [1] 3 3 2 2 3
```

Then apply the sampling vector to select a subset of the sample, drawn with replacement.

```
r[idx]
```

```
## [1] -0.02 -0.02  0.05  0.05 -0.02
```

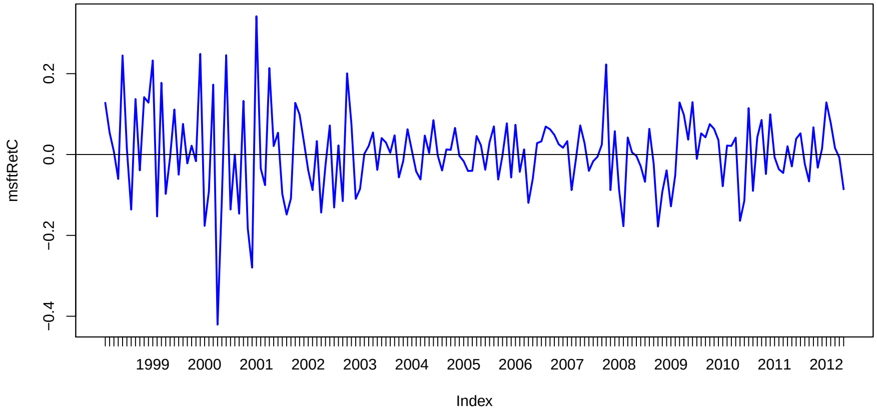
Example - R function `sample()`

Alternatively, use `sample()` directly to automate the previous two step process:

```r
set.seed(123)
sample(r, replace=TRUE)
```

```
## [1] -0.02 -0.02  0.05  0.05 -0.02
```

Example: GWN Model for Monthly cc Returns on MSFT



Put xts Data in Matrix

The R package **boot** requires data to be in a matrix object

```
returns.mat = as.matrix(gwnRetC)
MSFT = returns.mat[,"MSFT", drop=FALSE]
```

Estimate GWN Model Parameters for MSFT

Compute GWN model estimates for MSFT

```
n.obs = nrow(MSFT)
muhat.MSFT = mean(MSFT)
sigmahat.MSFT = sd(MSFT)
estimate = c(muhat.MSFT, sigmahat.MSFT)
```

Calculate analytic standard errors

```
se.muhat.MSFT = sigmahat.MSFT/sqrt(n.obs)
se.sigmahat.MSFT = sigmahat.MSFT/sqrt(2*n.obs)
se = c(se.muhat.MSFT, se.sigmahat.MSFT)
```

Estimate GWN Model Parameters for MSFT

Show estimates with analytic standard errors

```
ans = rbind(estimate, se)
colnames(ans) = c("Mu", "Sigma")
ans
```

```
##                 Mu  Sigma
## estimate 0.00413 0.1002
## se       0.00764 0.0054
```

Brute Force Bootstrap of $\hat{\mu}_{MSFT}$

Same idea as Monte Carlo simulation but instead of generating random data from an assumed distribution or model, you generate random data by sampling with replacement from the observed data using `sample()`. For each bootstrap sample, compute $\hat{\mu}_{MSFT}$

```
B = 999
muhat.boot = rep(0, B)
n.obs = nrow(MSFT)
set.seed(123)
for (i in 1:B) {
  boot.data = sample(MSFT, n.obs, replace=TRUE)
  muhat.boot[i] = mean(boot.data)
}
```

Bootstrap Estimates of Bias and SE

Bootstrap bias estimate (bootstrap mean - sample mean):

```
mean(muhat.boot) - muhat.MSFT
```

```
## [1] 0.000509
```

Bootstrap SE (SD of bootstrap samples)

```
sd(muhat.boot)
```

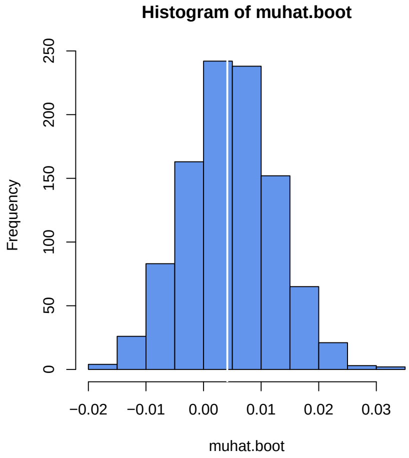
```
## [1] 0.00785
```

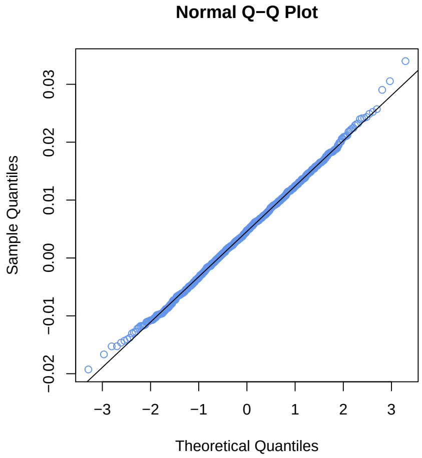
Compare with analytical SE

```
se.muhat.MSFT
```

```
## [1] 0.00764
```

Examine Bootstrap Distribution





Bootstrap distribution looks like normal distribution.

Bootstrap 95% Confidence Interval

Calculate bootstrap 95% confidence interval based on normal distribution

```
se.boot = sd(muhat.boot)
lower = muhat.MSFT - 2*se.boot
upper = muhat.MSFT + 2*se.boot
width = upper - lower
ans = c(lower, upper, width)
names(ans) = c("Lower", "Upper", "Width")
ans
```

```
##   Lower   Upper   Width
## -0.0116  0.0198  0.0314
```

Bootstrap 95% Confidence Interval

Calculate bootstrap 95% confidence interval based on empirical quantiles

```
qhat.boot = quantile(muhat.boot, probs=c(0.025, 0.975))
width = qhat.boot[2] - qhat.boot[1]
ans = c(qhat.boot, width)
names(ans) = c("Lower", "Upper", "Width")
ans
```

```
##   Lower   Upper   Width
## -0.0107  0.0201  0.0308
```

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 a user supplied function
  - `boot.ci()`: compute bootstrap confidence interval after call to `boot()`

Using the boot() Function

The function boot() requires a user-supplied function that take two arguments: *data* and a sampling *index*. The index is created internally by the boot() function and represents random resampling with replacement

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

Using the boot() Function

Bootstrap $\hat{\mu}$ using boot():

```
MSFT.muhat.boot = boot(MSFT, statistic = mean.boot, R=999)
class(MSFT.muhat.boot)
```

```
## [1] "boot"
```

```
names(MSFT.muhat.boot)
```

```
##  [1] "t0"        "t"         "R"         "data"      "seed'
##  [7] "sim"       "call"      "stype"     "strata"    "weigh
```

Using the `boot()` Function

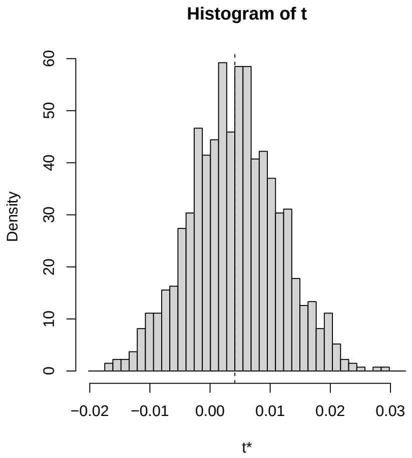
```
MSFT.muhat.boot
```

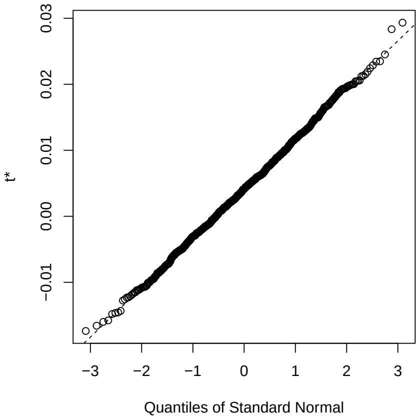
```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = MSFT, statistic = mean.boot, R = 999)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*  0.00413 2.01e-05     0.00748
```

Here, “original” is $\hat{\mu}$; “bias” is bootstrap bias; “std. error” is the bootstrap SE. Recall, the analytic SE is 0.008.

Plot Method for “boot” Objects

```
plot(MSFT.muhat.boot)
```





Bootstrap Confidence Intervals

Use boot.ci() on a “boot” object to compute bootstrap confidence intervals

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

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = MSFT.muhat.boot, conf = 0.95, type = c("
##     "perc"))
##
## Intervals :
## Level      Normal              Percentile
## 95%   (-0.0106,  0.0188 )   (-0.0107,  0.0194 )
## Calculations and Intervals on Original Scale
```

Example: Bootstrapping Value-at-Risk

In the GWN model for cc returns, 5% Value-at-Risk on an investment of $\$W_0$ is estimated using

$$\begin{aligned}\widehat{\mathrm{VaR}}_{.05} &= (e^{\hat{q}_{.05}} - 1) \times W_0 \\ \hat{q}_{.05} &= \hat{\mu} + \hat{\sigma} \cdot (-1.645) \\ -1.645 &= 5\% \text{ quantile from } N(0,1)\end{aligned}$$

Bootstrapping can be used to compute

$$\mathrm{bias}(\widehat{\mathrm{VaR}}_{.05}, \mathrm{VaR}_{.05}),\ \widehat{\mathrm{SE}}(\widehat{\mathrm{VaR}}_{.05})$$

as well as confidence intervals.

Bootstrapping Normal VaR for MSFT

Create function to compute normal VaR to be passed to boot()

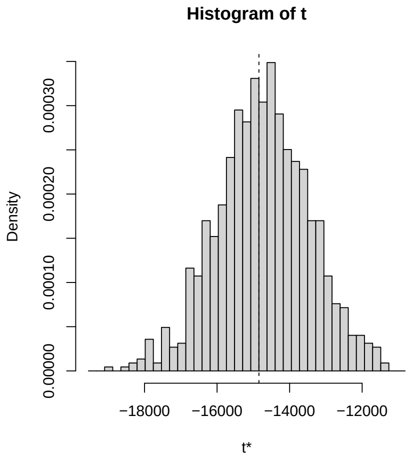
```
ValueAtRisk.boot = function(x, idx, p=0.05, w=100000) {
# x     data to be resampled
# idx       vector of scrambled indices created by boot()
# p       probability value for VaR calculation
# w       value of initial investment
# ans       Value-at-Risk computed using resampled data
    q = mean(x[idx]) + sd(x[idx])*qnorm(p)
    VaR = (exp(q) - 1)*w
    VaR
}
```

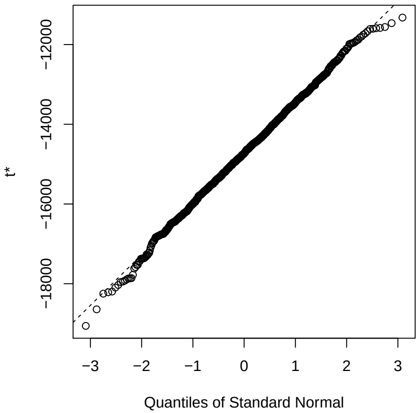
Bootstrapping Normal VaR

```
MSFT.VaR.boot = boot(MSFT, statistic = ValueAtRisk.boot, R=99
MSFT.VaR.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = MSFT, statistic = ValueAtRisk.boot, R = 999)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*   -14846     117        1271
```

Bootstrapping Normal VaR





Bootstrapping Normal VaR

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 999 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = MSFT.VaR.boot, conf = 0.95, type = c("no
##     "perc"))
##
## Intervals :
## Level      Normal              Percentile
## 95%   (-17454, -12472 )   (-17363, -12165 )
## Calculations and Intervals on Original Scale
```

95% confidence interval for 5% VaR is fairly wide.