

# The Bootstrap

The Econometrics of Predictability

*This version: April 29, 2014*

April 30, 2014

- Part I: Many Predictions
  - ▶ The Bootstrap
  - ▶ Technical Trading Rules
  - ▶ Formalized Data Snooping: Reality Check and the Test of Superior Predictive Ability
  - ▶ False Discovery, Stepwise Testing and the Model Confidence Set
- Part II: Many Predictors
  - ▶ Dynamic Factor Models
    - ▷ The Kalman Filter
    - ▷ Expectations Maximization Algorithm
  - ▶ Partial Least Squares and The 3 Pass Regression Filter
  - ▶ Regularized Reduced Rank Regression
  - ▶ LASSO

- 2 Assignments
  1. Group Work
    - ▷ Group of 2
    - ▷ 40% of course
    - ▷ If odd number of students, 1 group of 3 allowed
    - ▷ Empirical
    - ▷ Due **Friday Week 9, 12:00 at SBS**
  2. Individual Work
    - ▷ Formal Assignment
    - ▷ 60% of course
    - ▷ Empirical
    - ▷ Due **Friday Week 9, 12:00 (Informal)**
- Both assignment will make extensive use of MATLAB
- Presentation and content of results counts – code is not important
- Weekly problems to work on will be distributed – a subset of these will compromise the assigned material

## Definition (The Bootstrap)

The bootstrap is a statistical procedure where data is resampled, and the resampled data is used to estimate quantities of interest.

- Bootstraps come in many forms
  - Structure
    - ▷ Parametric
    - ▷ Nonparametric
  - Dependence Type
    - ▷ IID
    - ▷ Wild
    - ▷ Block and other for dependent data
- All share common structure of using simulated random numbers in combination with original data to compute quantities of interest
- Applications
  - Confidence Intervals
  - Refinements
  - Bias estimation

- Compute standard deviation for an estimator
- For example, in case of mean  $\bar{x}$  for i.i.d. data, we know

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

is usually a reasonable estimator of the standard deviation of the data

- The standard error of the mean is then

$$V[\bar{x}] = \frac{s^2}{n}$$

which can be used to form confidence intervals or conduct hypothesis tests (in conjunction with CLT)

- How could you estimate the standard error for the median of  $x_1, \dots, x_n$ ?
- What about inference about a quantile, for example that 5<sup>th</sup> percentile of  $x_1, \dots, x_n$ ?
- Bootstrap is a computational method to construct standard error estimates of confidence interval for a wide range of estimators.

- Assume  $n$  i.i.d. random (possibly vector valued) variables  $\mathbf{x}_1, \dots, \mathbf{x}_n$
- Estimator of a parameter of interest  $\hat{\theta}$ 
  - For example, the mean

## Definition (Empirical Distribution Function)

The empirical distribution function assigns probability  $1/n$  to each observation value. For a scalar random variable  $x_i, i = 1, \dots, n$ , the EDF is defined

$$\hat{F}(X) = \frac{1}{n} \sum_{i=1}^n I_{[x_i < X]}.$$

- Also known as the empirical CDF
- CDF of  $X$  should have information about precision of  $\hat{\theta}$ , so ECDF might also

## Algorithm (IID Bootstrap)

1. Simulate a set of  $n$  i.i.d. uniform random integers  $u_i$ ,  $i = 1, \dots, n$  from the range  $1, \dots, n$  (with replacement)
2. Construct a bootstrap sample  $x_b^* = \{x_{u_1}, x_{u_2}, \dots, x_{u_n}\}$
3. Compute the mean

$$\hat{\theta}_b^* = \frac{1}{n} \sum_{i=1}^n x_{b,i}^*$$

4. Repeat steps 1–3  $B$  times
5. Estimate the standard of  $\hat{\theta}$  using

$$\frac{1}{B} \sum_{i=1}^B (\theta_b^* - \hat{\theta})^2$$

```
n = 100; x = randn(n,1);
% Mean of x
mu = mean(x);
B = 1000;
% Initialize muStar
muStar = zeros(B,1);
% Loop over B bootstraps
for b=1:B
    % Uniform random numbers over 1...n
    u = ceil(n*rand(n,1));
    % x-star sample simulation
    xStar = x(u);
    % Mean of x-star
    muStar(b) = mean(xStar);
end
s2 = 1/(n-1)*sum((x-mu).^2);
stdErr = s2/n
bootstrapStdErr = mean((muStar-mu).^2)
```

- $B$  is used for the number of bootstrap replications
- Bootstrap theory assumes  $B \rightarrow \infty$  quickly
- This ensures that the bootstrap distribution is identical to the case where all unique bootstraps were computed
  - There are a lot of unique bootstraps
  - $n^n$  in the i.i.d. case
- Using finite  $B$  adds some extra variation since two bootstraps with the same data won't produce identical estimates
- **Note:** Often useful to set the state of your random number generator so that results are reproducible

```
% A non-negative integer
```

```
seed = 26031974
```

```
rng(seed)
```

- Should choose  $B$  large enough that the *Monte Carlo error* is negligible
- In practice little reason to use less than 1,000 replications

- Balanced resampling
  - In standard i.i.d. bootstrap, some values will inevitably appear more than others
  - Balanced resampling ensures that all values appear the same number of times
  - In practice simple to implement

## Algorithm (IID Bootstrap with Balanced Resampling)

1. Replicate the data so that there are  $B$  copies of each  $x_i$ . The data set should have  $Bn$  observations
2. Construct a random random permutation of the numbers  $1, \dots, Bn$  as  $u_1, \dots, u_{Bn}$
3. Construct the bootstrap sample  $x_b^* = \{x_{u_{n(b-1)+1}}, x_{u_{n(b-1)+2}}, \dots, x_{u_{n(b-1)+n}}\}$

- This algorithm samples *without replacement* from the replicated dataset of  $Bn$  observations
- Each data point will appear exactly  $B$  times in the  $B$  bootstrap samples

```
n = 100; x = randn(n,1);  
% Replicate the data  
xRepl = repmat(x,B,1);  
B = 1000;  
% Random permutaiton of 1,...,B*n  
u = randperm(n*B);  
% Loop over B bootstraps  
for b=1:B  
    % Uniform random numbers over 1...n  
    ind = n*(b-1)+(1:n);  
    xb = xRepl(u(ind));  
end
```

- Antithetic Random Variables
- If samples are *negatively* correlated, variance of statistics can be reduced
  - Basic idea is to order data so that if one sample has too many large values of  $x$ , then the next will have too many small
  - This can induce negative correlation while not corrupting bootstrap

## Algorithm (IID Bootstrap with Antithetic Resampling)

1. Order the data so that  $x_1 \leq x_2 \dots \leq x_n$ . Treat these indices as the original data.
2. Simulate a set of  $n$  i.i.d. uniform random integers  $u_i$ ,  $i = 1, \dots, n$  from the range  $1, \dots, n$  (with replacement)
3. Construct the bootstrap sample  $x_b^* = \{x_{u_1}, x_{u_2}, \dots, x_{u_n}\}$
4. Construct  $\tilde{u}_i = n - u_i + 1$
5. Construct the antithetic bootstrap sample  $x_{b+1}^* = \{x_{\tilde{u}_1}, x_{\tilde{u}_2}, \dots, x_{\tilde{u}_n}\}$
6. Repeat for  $b = 1, 3, \dots, B - 1$

- Using antithetic random variables is a general principle applicable to virtually all simulation estimators



```
n = 100; x = randn(n,1);
% Mean of x
mu = mean(x);
B = 1000;
% Initialize muStar
muStar = zeros(B,1);
% Sort x
x = sort(x);
% Loop over B bootstraps
for b=1:2:B
    % Uniform random numbers over 1...n
    u = ceil(n*rand(n,1)); xStar = x(u);
    % Mean of x-star
    muStar(b) = mean(xStar);
    % Uniform random numbers over 1...n
    u = n-u+1; xStar = x(u);
    % Mean of x-star
    muStar(b+1) = mean(xStar);
end
corr(muStar(1:2:B),muStar(2:2:B))
```

- Many statistics have a *finite sample bias*
- This is equivalent to saying that  $\hat{\theta} - \theta \approx c/n$  for some  $c \neq 0$ 
  - Many estimators have  $c = 0$ , for example the sample mean
  - These estimators are unbiased
- Biased estimators usually arise when the estimator is a non-linear function of the data
- Bootstrap can be used to estimate the bias, and the estimate can be used to debias the original estimate
- Recall the definition of bias

## Definition (Bias)

The bias of an estimator is

$$E [\hat{\theta} - \theta]$$

## Algorithm

1. *Estimate the parameter of interest  $\hat{\theta}$*
2. *Generate a bootstrap sample  $x_b$  and estimate the parameter on the bootstrap sample. Denote this estimate as  $\hat{\theta}_b^*$*
3. *Repeat 2 a total of  $B$  times*
4. *Estimate the bias as*

$$\text{Bias} = B^{-1} \sum_{i=1}^B \hat{\theta}_b^* - \hat{\theta}$$

- Example of bootstrap bias adjustment will be given later once more results for time-series have been established

## Algorithm

1. Estimate the parameter of interest  $\hat{\theta}$
2. Generate a bootstrap sample  $x_b$  and estimate the parameter on the bootstrap sample. Denote this estimate as  $\hat{\theta}_b^*$
3. Repeat 2 a total of  $B$  times
4. Estimate the standard error as

$$\text{Std. Err} = \sqrt{B^{-1} \sum_{i=1}^B (\hat{\theta}_b^* - \hat{\theta})^2}$$

- Other estimators are also common

$$\text{Std. Err} = \sqrt{(B-1)^{-1} \sum_{i=1}^B (\hat{\theta}_b^* - \overline{\hat{\theta}_b^*})^2}$$

- $B$  should be sufficiently large that  $B$  or  $B-1$  should not matter

- Bootstraps can also be used to construct confidence intervals
- Two methods:
  1. Estimate the standard error of the estimator and use a CLT
  2. Estimate the confidence interval directly using the bootstrap estimators  $\{\hat{\theta}_b^*\}$
- The first method is simple and have previously been explained
- The second is also very simple, and is known as the *percentile method*

## Algorithm (Percentile Method)

A confidence interval  $[C_{\alpha_L}, C_{\alpha_H}]$  with coverage  $\alpha_H - \alpha_L$  can be constructed:

1. Construct a bootstrap sample  $x_b$
2. Compute the bootstrap estimate  $\hat{\theta}_b^*$
3. Repeat steps 1–2
4. The confidence interval is constructed using the empirical  $\alpha_L$  quantile and the empirical  $\alpha_H$  quantile of  $\{\hat{\theta}_b^*\}$

- If the bootstrap estimates are ordered from smallest to largest, and  $B\alpha_L$  and  $B\alpha_H$  are integers, then the confidence interval is

$$[\hat{\theta}_{B\alpha_L}^*, \hat{\theta}_{B\alpha_H}^*]$$

- This method may not work well in all situations
  - $n$  small
  - Highly asymmetric distribution

```
n = 100; x = randn(n,1);
% Mean of x
mu = mean(x);
B = 1000;
% Initialize muStar
muStar = zeros(B,1);
% Loop over B bootstraps
for b=1:B
    % Uniform random numbers over 1...n
    u = ceil(n*rand(n,1));
    % x-star sample simulation
    xStar = x(u);
    % Mean of x-star
    muStar(b) = mean(xStar);
end
alphaL = .05;alphaH=.95;
muStar = sort(muStar);
CI = [muStar(alphaL*B) muStar(round(alphaH*B))]
CI - mu
```

- Bootstraps can be used in more complex scenarios
- One simple extension is to regressions
- Using a model, rather than estimating a simple statistic, allows for a richer set of bootstrap options
  - Parametric
  - Non-parametric
- Basic idea, however, remains the same:
  - Simulate random data from the same DGP
  - Now requires data for both the regressor  $y$  and the regressand  $x$

- Parametric bootstraps are based on a model
- They exploit the structure of the model to re-sample residuals rather than the actual data
- Suppose

$$y_i = \mathbf{x}_i \boldsymbol{\beta} + \epsilon_i$$

where  $\epsilon_i$  is homoskedastic

- The parametric bootstrap would estimate the model and the residuals as

$$\hat{\epsilon}_i = y_i - \mathbf{x}_i \hat{\boldsymbol{\beta}}$$

- The bootstrap would then construct the re-sampled “data” by sampling  $\hat{\epsilon}_i$  separately from  $\mathbf{x}_i$ 
  - In other words, use two separate sets of i.i.d. uniform indices
- Construct  $y_{b,i}^* = \mathbf{x}_{u_{1i}} \hat{\boldsymbol{\beta}} + \hat{\epsilon}_{u_{2i}}$
- Compute statistics using these values

- Many examples use `bsxfun`

## BSXFUN Binary Singleton Expansion Function

`C = BSXFUN(FUNC,A,B)` applies the element-by-element binary operation specified by the function handle `FUNC` to arrays `A` and `B`, with singleton expansion enabled. `FUNC` must be able to accept as input either two column vectors of the same size, or one column vector and one scalar, and return as output a column vector of the same size as the input(s). `FUNC` can either be a function handle for an arbitrary function satisfying the above conditions or one of the following built-in:

- Allows  $k$  by  $n$  matrix to be added/subtracted from  $k$  by 1 vector or 1 by  $n$  vector

```
x = randn(1000,10);  
mu = mean(x);  
err = bsxfun(@minus,x,mu);
```



```

n = 100; x = randn(n,2); e = randn(n,1); y = x*ones(2,1) + e;
% Bhat
Bhat = x\y; ehat = y - x*Bhat;
B = 1000;
% Initialize BStar
BStar = zeros(B,2);
% Loop over B bootstraps
for b=1:B
    % Uniform random numbers over 1...n
    uX = ceil(n*rand(n,1)); uE = ceil(n*rand(n,1));
    % x-star sample simulation
    xStar = x(uX,:); eStar = e(uE);
    yStar = xStar*Bhat + eStar;
    % Mean of x-star
    BStar(b,:) = (xStar\yStar)';
end
Berr=bsxfun(@minus, BStar , Bhat ');
bootstrapVCV = Berr '* Berr/B
trueVCV = eye(2)/100
OLSVCV = (e'*e)/n * inv(x'*x)
    
```

- Non-parametric bootstrap is simpler
- It does not use the structure of the model to construct artificial data
- The vector  $[y_i, \mathbf{x}_i]$  is instead directly re-sampled
- The parameters are constructed from the pairs

## Algorithm (Non-parametric Bootstrap for i.i.d. Regression Data)

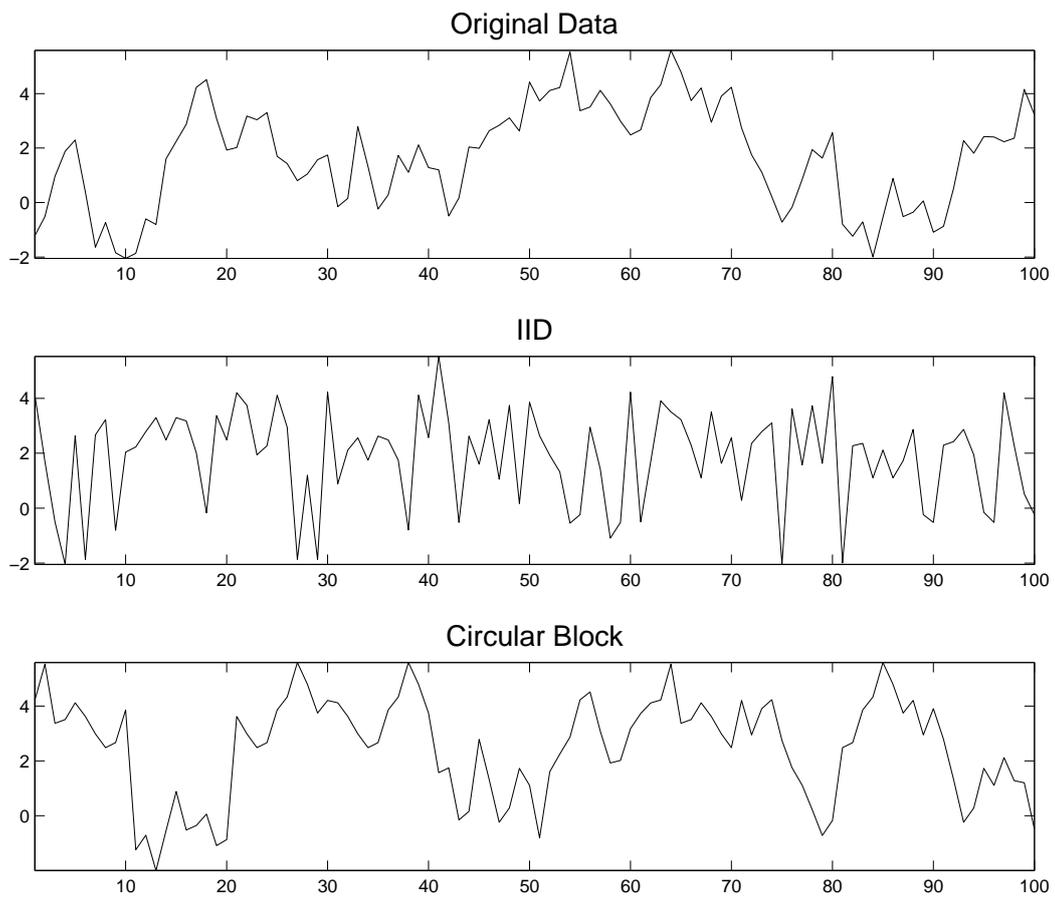
1. *Simulate a set of  $n$  i.i.d. uniform random integers  $u_i, i = 1, \dots, n$  from the range  $1, \dots, n$  (with replacement)*
2. *Construct the bootstrap sample  $\mathbf{z}_b = \{y_{u_i}, \mathbf{x}_{u_i}\}$*
3. *Estimate the bootstrap  $\boldsymbol{\beta}$  by fitting the model*

$$y_{u_i} = \mathbf{x}_{u_i} \hat{\boldsymbol{\beta}}_b^* + \epsilon_{b,i}^*$$

```
n = 100; x = randn(n,2); e = randn(n,1); y = x*ones(2,1) + e;
% Bhat
Bhat = x\y; ehat = y - x*Bhat;
B = 1000;
% Initialize BStar
BStar = zeros(B,2);
% Loop over B bootstraps
for b=1:B
    % Uniform random numbers over 1...n
    u = ceil(n*rand(n,1));
    % x-star sample simulation
    yStar = y(u);
    xStar = x(u,:);
    % Mean of x-star
    BStar(b,:) = (xStar\yStar)';
end
Berr=bsxfun(@minus, BStar , Bhat ');
bootstrapVCV = Berr '* Berr/B
trueVCV = eye(2)/100
OLSVCV = (e'*e)/n * inv(x'*x)
```

- i.i.d. bootstrap is only appropriate for i.i.d. data
  - **Note:** Usually OK for data that is not serially correlated
- Two strategies for bootstrapping time-series data
  - Parametric & i.i.d. bootstrap: If the model postulates that the residuals are i.i.d. or at least white noise, then a residual-based i.i.d. bootstrap may be appropriate
    - Examples: AR models, GARCH models using appropriately standardized residuals
  - Nonparametric *block* bootstrap: Weak assumptions, basically that blocks can be sampled so that they (**blocks**) are approximately i.i.d.
    - Similar to the notion of ergodicity which is related to asymptotic independence
    - **Important:** Like Newey-West covariance estimator, *block length* must grow with sample size
    - Fundamentally same reason

# The problem with the IID Bootstrap





```
% Number of time periods
T = 100;
% Random errors
e = randn(T,1);
y = zeros(T,1);
% Y is an AR(1), phi1 = 0.5
y(1) = e(1)*sqrt(1/(1-.5^2));
for t=2:T
    y(t)=0.5*y(t-1)+e(t);
end
% 10,000 replications
B = 10000;
% Initial place for mu-star
muStar = zeros(B,1);
```

- Samples blocks of  $m$  consecutive observations
- Uses blocks which start at indices  $1, \dots, T - m + 1$

## Algorithm (Moving Block Bootstrap)

1. Initialize  $i = 1$
2. Draw a uniform integer  $v_i$  on  $1, \dots, T - m + 1$
3. Assign  $u_{(i-1)+j} = v_i + j - 1$  for  $j = 1, \dots, m$
4. Increment  $i$  and repeat 2-3 until  $i \geq \lceil T/m \rceil$
5. Trim  $u$  so that only the first  $T$  remain if  $T/m$  is not an integer



```
% Block size
m = 10;
% Loop over B bootstraps
for b=1:B
    % ceil(T/m) Uniform random numbers over 1...T-m+1
    u = ceil((T-m+1)*rand(ceil(T/m),1));
    u = bsxfun(@plus,u,0:m-1)';
    % Transform to col vector, and remove excess
    u = u(:); u = u(1:T);
    % y-star sample simulation
    yStar = y(u);
    % Mean of y-star
    muStar(b) = mean(yStar);
end
```

- Simple extension of MBB which assumes the data live on a circle so that  $y_{T+1} = y_1, y_{T+2} = y_2$ , etc.
- Has better finite sample properties since all data points get sampled with equal probability
- Only step 2 changes in a very small way

## Algorithm (Circular Block Bootstrap)

1. Initialize  $i = 1$
2. Draw a uniform integer  $v_i$  on  $1, \dots, T$
3. Assign  $u_{(i-1)+j} = v_i + j - 1$  for  $j = 1, \dots, m$
4. Increment  $i$  and repeat 2–3 until  $i \geq \lceil T/m \rceil$
5. Trim  $u$  so that only the first  $T$  remain if  $T/m$  is not an integer



```
% Block size
m = 10;
% Loop over B bootstraps
yRepl = [y;y];
for b=1:B
    % ceil(T/m) Uniform random numbers over 1...T-m+1
    u = ceil(T*rand(ceil(T/m),1));
    u = bsxfun(@plus,u,0:m-1)';
    % Transform to col vector, and remove excess
    u = u(:); u = u(1:T);
    % y-star sample simulation
    yStar = yRepl(u);
    % Mean of y-star
    muStar(b) = mean(yStar);
end
```

- Differs from MBB and CBB in that the block size is no longer fixed
- Chooses an average block size of  $m$  rather than an exact block size
- Randomness in block size is worse when  $m$  is known, but helps if  $m$  may be suboptimal
- Block size is *exponentially distributed* with mean  $m$

## Algorithm (Stationary Bootstrap)

1. Draw  $u_1$  uniform on  $1, \dots, T$
2. For  $i = 2, \dots, t$ 
  - a. Draw a uniform  $v$  on  $(0, 1)$
  - b. If  $v \geq 1/m$   $u_i = u_{i-1} + 1$ 
    - i. If  $u_i > T$ ,  $u_i = u_i - T$
  - c. If  $v < 1/m$ , draw  $u_i$  uniform on  $1, \dots, T$



```
% Average block size
m = 10;
% Loop over B bootstraps
yRepl = [y;y];
u = zeros(T,1);
for b=1:B
    u(1) = ceil(T*rand);
    for t=2:T
        if rand<1/m
            u(t) = ceil(T*rand);
        else
            u(t) = u(t-1) + 1;
        end
    end
    % y-star sample simulation
    yStar = yRepl(u);
    % Mean of y-star
    muStar(b) = mean(yStar);
end
```

- MBB was the first
- CBB has simpler theoretical properties and usually requires fewer corrections to address “end effects”
- SB is theoretically worse than MBB and CBB, but is the most common choice in time-series econometrics
  - Theoretical optimality assumes that the the “optimal” block size is used
- Popularity of SB stems from difficulty in determining optimal  $m$ 
  - More on this in a minute
- Random block size brings some robustness at the cost of extra variability

- The stationary AR(P) model can be parametrically bootstraps
- Assume

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_P y_{t-P} + \epsilon_t$$

- Usual assumptions, including stationarity
- Can use a parametric bootstrap by estimating the residuals

$$\hat{\epsilon}_t = y_t - \hat{\phi}_1 y_{t-1} + \dots + \hat{\phi}_P y_{t-P}$$

## Algorithm (Stationary Autoregressive Bootstrap)

1. Estimate the AR(P) and the residuals for  $t = P + 1, \dots, T$
2. Recenter the residuals so that they have mean 0

$$\tilde{\epsilon}_t = \hat{\epsilon}_t - \bar{\hat{\epsilon}}$$

3. Draw  $u$  uniform from  $1, \dots, T - P + 1$  and set  $y_1^* = y_u$ ,  
 $y_2^* = y_{u+1}, \dots, y_P^* = y_{u+P+1}$
4. Recursively simulate  $y_{P+1}^* \dots y_T^*$  using  $\tilde{\epsilon}$  drawn using an i.i.d. bootstrap

```
phi = y(1:T-1)\y(2:T);
ehat = y(2:T)-phi*y(1:T-1);
etilde = ehat-mean(ehat);
yStar = zeros(T,1);
for i=1:B
    % Initialize to one of the original values
    yStar(1) = y(ceil(T*rand));
    % Indices for errors
    u = ceil((T-1)*rand(T,1));
    % Recursion to simulate AR
    for t=2:T
        yStar(t) = phi*yStar(t-1) + ehat(u(t));
    end
end
```



- Block size selection is crucial for good performance of block bootstraps
- Small block sizes are too close to i.i.d. while large block sizes are overly noisy
- Politis and White (2004) provide a data dependent lag length selection procedure
  - See also Patton, Politis, and White (2007) correction
- Code is available by searching the internet for “opt\_block\_length\_REV\_dec07”

- Politis and White (2004) show for stationary bootstrap

$$B_{opt,SB} = \left( \frac{2G^2}{D_{SB}} \right) N^{1/3}$$

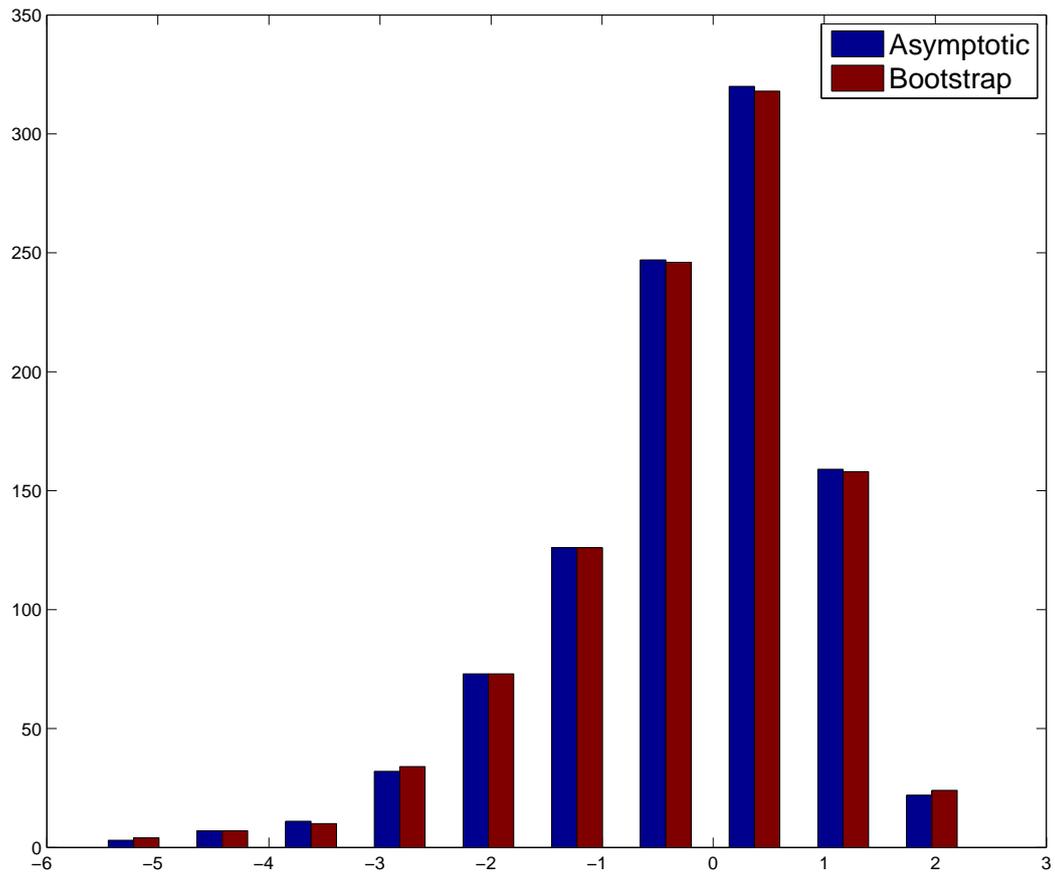
- ▶  $G = \sum_{k=-\infty}^{\infty} |k| \gamma_k$  where  $\gamma_k$  is the autocovariance
- ▶  $D_{SB} = 2g(0)^2$  where  $g(w) = \sum_{s=-\infty}^{\infty} \gamma_s \cos(ws)$  is the spectral density function
- Need to estimate  $\hat{G}$  and  $\hat{D}_{SB}$  to estimate  $\hat{B}_{opt,SB}$ 
  - ▶  $\hat{G} = \sum_{k=-M}^M \lambda(k/M) |k| \hat{\gamma}_k$ ,

$$\lambda(s) = \begin{cases} 1 & \text{if } |s| \in [0, 1/2] \\ 2(1 - |s|) & \text{if } |s| \in [1/2, 1] \\ 0 & \text{otherwise} \end{cases}$$

- ▶  $\hat{D}_{SB} = 2\hat{g}(0)$ ,  $\hat{g}(w) = \sum_{k=-M}^M \lambda(k/M) \hat{\gamma}_k \cos(wk)$
- ▶  $M$  is set to  $2\hat{m}$
- ▶  $\hat{m}$  is the smallest integer where if  $\hat{\rho}_j > 2\sqrt{\log T/T}$ ,  $j = m + 1, \dots, K_T$  where  $K_T = 2 \max(5, \sqrt{\log_{10}(T)})$

- $y_i \stackrel{\text{i.i.d.}}{\sim} LN(0, 1)$
- $n = 100$
- $B = 1000$  using i.i.d. bootstrap
- This is a check that the bootstrap works
- Also shows that bootstrap will not work miracles
- Performance of bootstrap is virtually identical to that of asymptotic theory
  - Gains to bootstrap are more difficult to achieve
  - Most useful property is in estimating standard error in hard to compute cases

# Example 1: Mean Estimation for Log-Normal



## Example 2: Bias in AR(1)

- Assume  $y_t = \phi y_{t-1} + \epsilon_t$  where  $\epsilon_t \stackrel{\text{i.i.d.}}{\sim} N(0, 1)$
- $\phi = 0.9, T = 50$
- Use parametric bootstrap
- Estimate bias using the different between bootstrap estimates and the actual estimate

	Direct	Debiased
$\hat{\phi}$	0.8711	0.8810
Var	0.0052	0.0044

- Reduced the bias by about 1/3
- Reduced variance (**rare**)