Bootstrapping time-dependent stationary processes

Christopher F Baum

UK Stata Conference, September 2026

This presentation is based on a forthcoming *Stata Journal* article with the same title, coauthored by Jesús Otero of the Universidad del Rosario.

The concept of the bootstrap, introduced by Efron (1979) and further developed by Efron and Tibshirani (1993), is a computationally intensive method used to estimate the sampling distribution of a statistic based on the observed data.

In Stata, the `bootstrap` command performs nonparametric bootstrap estimation of specified statistics (or expressions) under the assumption that the underlying observations are independent and identically distributed (i.i.d.).

The concept of the bootstrap, introduced by Efron (1979) and further developed by Efron and Tibshirani (1993), is a computationally intensive method used to estimate the sampling distribution of a statistic based on the observed data.

In Stata, the `bootstrap` command performs nonparametric bootstrap estimation of specified statistics (or expressions) under the assumption that the underlying observations are independent and identically distributed (i.i.d.).

Bootstrapping commands operate by resampling individual data points with replacement, making them well suited to cross-sectional data in which observations are independent of one another. In such cases, the empirical distribution of the data provides a good approximation to the population distribution, so resampling effectively mimics repeated sampling from the population.

However, when observations are dependent, the standard i.i.d. bootstrap fails because drawing individual observations at random destroys the relevant dependence structure.

In time series data, successive observations are naturally correlated over time, and block bootstrapping – resampling blocks of consecutive observations – preserves local dependence within each block while introducing variation across blocks

Bootstrapping commands operate by resampling individual data points with replacement, making them well suited to cross-sectional data in which observations are independent of one another. In such cases, the empirical distribution of the data provides a good approximation to the population distribution, so resampling effectively mimics repeated sampling from the population.

However, when observations are dependent, the standard i.i.d. bootstrap fails because drawing individual observations at random destroys the relevant dependence structure.

In time series data, successive observations are naturally correlated over time, and block bootstrapping – resampling blocks of consecutive observations – preserves local dependence within each block while introducing variation across blocks

Bootstrapping commands operate by resampling individual data points with replacement, making them well suited to cross-sectional data in which observations are independent of one another. In such cases, the empirical distribution of the data provides a good approximation to the population distribution, so resampling effectively mimics repeated sampling from the population.

However, when observations are dependent, the standard i.i.d. bootstrap fails because drawing individual observations at random destroys the relevant dependence structure.

In time series data, successive observations are naturally correlated over time, and block bootstrapping – resampling blocks of consecutive observations – preserves local dependence within each block while introducing variation across blocks

We present the community-contributed `blockboot` command to bootstrap time-dependent stationary processes using four schemes that preserve the processes' dependence structure by resampling blocks of observations.

These schemes include the nonoverlapping block bootstrap of Carlstein, the moving block bootstrap of Künsch and Liu and Singh, the circular block bootstrap of Politis and Romano and the stationary block bootstrap of Politis and Romano.

We present the community-contributed `blockboot` command to bootstrap time-dependent stationary processes using four schemes that preserve the processes' dependence structure by resampling blocks of observations.

These schemes include the nonoverlapping block bootstrap of Carlstein, the moving block bootstrap of Künsch and Liu and Singh, the circular block bootstrap of Politis and Romano and the stationary block bootstrap of Politis and Romano.

The non-overlapping block bootstrap (NBB) is described in Carlstein (1986). It splits original series into non-overlapping blocks and then resamples the blocks multiple times to form a new series. Because NBB uses only the predefined blocks, the resulting bootstrap series may be shorter than the original if the sample size is not a multiple of the specified block length.

The moving block bootstrap (MBB) uses a method described in Künsch (1989) and Liu and Sing (1992) which splits original series into overlapping blocks and then resamples the blocks multiple times to form a new series. This produces greater flexibility, as any interior segment of the series can appear as a valid block.

The non-overlapping block bootstrap (NBB) is described in Carlstein (1986). It splits original series into non-overlapping blocks and then resamples the blocks multiple times to form a new series. Because NBB uses only the predefined blocks, the resulting bootstrap series may be shorter than the original if the sample size is not a multiple of the specified block length.

The moving block bootstrap (MBB) uses a method described in Künsch (1989) and Liu and Sing (1992) which splits original series into overlapping blocks and then resamples the blocks multiple times to form a new series. This produces greater flexibility, as any interior segment of the series can appear as a valid block.

The circular block bootstrap (CBB) uses a method described in Politis and Romano (1992). This improves on the moving block bootstrap scheme by making provisions for observations at the tail end of the original series that could have been cut off from resampling simply because the number of remaining elements does not equal a specified block length. If so, the remaining elements are completed using the first element(s) of the original series to form a circle.

The stationary block bootstrap, (SBB), described in Politis and Romano (1994), instead resamples blocks of random, geometrically distributed length to form a new series. Starting points are selected at random, and blocks of random length are drawn and concatenated to form a new series. Because block lengths differ, transitions between blocks can occur at any point, producing a bootstrap series that better reproduces the stationary dependence structure of the original data.

The circular block bootstrap (CBB) uses a method described in Politis and Romano (1992). This improves on the moving block bootstrap scheme by making provisions for observations at the tail end of the original series that could have been cut off from resampling simply because the number of remaining elements does not equal a specified block length. If so, the remaining elements are completed using the first element(s) of the original series to form a circle.

The stationary block bootstrap, (SBB), described in Politis and Romano (1994), instead resamples blocks of random, geometrically distributed length to form a new series. Starting points are selected at random, and blocks of random length are drawn and concatenated to form a new series. Because block lengths differ, transitions between blocks can occur at any point, producing a bootstrap series that better reproduces the stationary dependence structure of the original data.

We present an illustration of the `blockboot` command on an artificial x series for each scheme. The commands used:

```
. set obs 30
. gen x = _n
. tsset x
. blockboot x, type(nbb) prefix(n) lblock(8) seed(123)
. blockboot x, type(mbb) prefix(m) lblock(8) seed(123)
. blockboot x, type(cbb) prefix(c) lblock(8) seed(123)
. blockboot x, type(sbb) prefix(s) lblock(8) seed(123)
```

x_t	x_t^*			
	nbb	mbb	cbb	sbb
1	9	11	11	13
2	10	12	12	14
3	11	13	13	15
4	12	14	14	7
5	13	15	15	7
6	14	16	16	8
7	15	17	17	9
8	16	18	18	16
9	17	18	18	17
10	18	19	19	18
11	19	20	20	19
12	20	21	21	20
13	21	22	22	21
14	22	23	23	28
15	23	24	24	29
16	24	25	25	30
17	17	7	24	1
18	18	8	25	2
19	19	9	26	3
20	20	10	27	6
21	21	11	28	7
22	22	12	29	8
23	23	13	30	9
24	24	14	1	10
25		7	7	11
26		8	8	12
27		9	9	13
28		10	10	14
29		11	11	15
30		12	12	30

The community-contributed `blockboot` command, available from the SSC Archive, has the syntax:

blockboot *varlist* [if] [in] , type(*string*) prefix(*string*) lblock(*string*) [seed(*integer*)]

Multiple time-series variables can be specified in the *varlist*. As shown earlier, the `type()` specifies which of the four bootstrap schemes are to be used. The command produces new variables for each element of the *varlist*, with their names preceded by `prefix()`. These variables must not exist in the current dataset.

The `lblock()` option specifies the block length. In the case of SBB, it is the average length of the blocks. Setting it to 1 yields the simple bootstrap.

The `lblock()` option can also be specfied as `auto`, whcih triggers the block length selection mechanism devised by Politis and White (2004), Patton et al. (2009) and Politis and Romano (1995).

While all of these options are required, there is also a `seed()` option that will provide the same sequence of pseudo-random draws.

The `lblock()` option specifies the block length. In the case of SBB, it is the average length of the blocks. Setting it to 1 yields the simple bootstrap.

The `lblock()` option can also be specfied as `auto`, whcih triggers the block length selection mechanism devised by Politis and White (2004), Patton et al. (2009) and Politis and Romano (1995).

While all of these options are required, there is also a `seed()` option that will provide the same sequence of pseudo-random draws.

The `lblock()` option specifies the block length. In the case of SBB, it is the average length of the blocks. Setting it to 1 yields the simple bootstrap.

The `lblock()` option can also be specfied as `auto`, whcih triggers the block length selection mechanism devised by Politis and White (2004), Patton et al. (2009) and Politis and Romano (1995).

While all of these options are required, there is also a `seed()` option that will provide the same sequence of pseudo-random draws.

An illustration of these four block bootstrap schemes for time-series data in the context of computing the size of unit-root tests extends and updates the findings of Schwert, "Tests for Unit Roots: A Monte Carlo Investigation" (*JBES*, 1989).

Schwert's extensive Monte Carlo experiments showed that the unit root tests developed by Fuller (1976) and Dickey and Fuller (1979, 1981) are highly sensitive to the assumption of a purely autoregressive data-generating process. When a moving-average component is present, the empirical distribution of the test statistics can differ markedly from those reported in the original Dickey–Fuller studies.

An illustration of these four block bootstrap schemes for time-series data in the context of computing the size of unit-root tests extends and updates the findings of Schwert, "Tests for Unit Roots: A Monte Carlo Investigation" (*JBES*, 1989).

Schwert's extensive Monte Carlo experiments showed that the unit root tests developed by Fuller (1976) and Dickey and Fuller (1979, 1981) are highly sensitive to the assumption of a purely autoregressive data-generating process. When a moving-average component is present, the empirical distribution of the test statistics can differ markedly from those reported in the original Dickey–Fuller studies.

Schwert further found that the procedures proposed by Said and Dickey (1984), Phillips (1987) and Philiips and Perron (1988), intended to correct for such misspecification, perform poorly when the moving-average parameter is large. In particular, the Phillips–Perron tests fail to approximate their asymptotic distributions even for samples as large as 10,000 observations.

The most reliable procedure across cases was the high-order autoregressive t-test proposed by Said and Dickey (1984), which maintained size close to its nominal level for all moving-average parameters.

Schwert further found that the procedures proposed by Said and Dickey (1984), Phillips (1987) and Philiips and Perron (1988), intended to correct for such misspecification, perform poorly when the moving-average parameter is large. In particular, the Phillips–Perron tests fail to approximate their asymptotic distributions even for samples as large as 10,000 observations.

The most reliable procedure across cases was the high-order autoregressive t-test proposed by Said and Dickey (1984), which maintained size close to its nominal level for all moving-average parameters.

We replicate Schwert's study in a broad sense. That is, rather than reproducing the exact tabulated results, we revisit the experimental design with modern computational tools and resampling methods.

Specifically, we employ the four bootstrap schemes implemented in the command to reassess the size properties of unit-root tests in the presence of data dependence. We consider two specifications imposing a unit root: one difference-stationary around an intercept, and the other difference-stationary around an intercept and trend. We consider autocorrelation coefficients of $\theta \in \{0.8, 0.5, 0, -0.5, -0.8\}$.

We use Monte Carlo simulation with 10,000 repetitions and 199 replications of each of the four block bootstrap schemes, with sample sizes $T = 25$, 50, 100, 250, 500, 1000. The bootstrap schemes are implemented on the stationary first-differenced series.

We replicate Schwert's study in a broad sense. That is, rather than reproducing the exact tabulated results, we revisit the experimental design with modern computational tools and resampling methods.

Specifically, we employ the four bootstrap schemes implemented in the command to reassess the size properties of unit-root tests in the presence of data dependence. We consider two specifications imposing a unit root: one difference-stationary around an intercept, and the other difference-stationary around an intercept and trend. We consider autocorrelation coefficients of $\theta \in \{0.8, 0.5, 0, -0.5, -0.8\}$.

We use Monte Carlo simulation with 10,000 repetitions and 199 replications of each of the four block bootstrap schemes, with sample sizes $T = 25$, 50, 100, 250, 500, 1000. The bootstrap schemes are implemented on the stationary first-differenced series.

We replicate Schwert's study in a broad sense. That is, rather than reproducing the exact tabulated results, we revisit the experimental design with modern computational tools and resampling methods.

Specifically, we employ the four bootstrap schemes implemented in the command to reassess the size properties of unit-root tests in the presence of data dependence. We consider two specifications imposing a unit root: one difference-stationary around an intercept, and the other difference-stationary around an intercept and trend. We consider autocorrelation coefficients of $\theta \in \{0.8, 0.5, 0, -0.5, -0.8\}$.

We use Monte Carlo simulation with 10,000 repetitions and 199 replications of each of the four block bootstrap schemes, with sample sizes $T = 25$, 50, 100, 250, 500, 1000. The bootstrap schemes are implemented on the stationary first-differenced series.

As reported in the forthcoming article, we confirm Schwert's (1989) main insight that unit root test size is highly sensitive to the data?s dependence structure. Under strong negative dependence, short block lengths lead to severe over-rejection, while longer or data-driven block lengths substantially reduce these distortions. With weaker dependence, size distortions are modest and disappear entirely under independence.

Differences among the four bootstrap schemes are minor relative to the influence of block length. Overall, the results highlight the importance of adequately capturing dependence when bootstrapping time-series data.

As reported in the forthcoming article, we confirm Schwert's (1989) main insight that unit root test size is highly sensitive to the data?s dependence structure. Under strong negative dependence, short block lengths lead to severe over-rejection, while longer or data-driven block lengths substantially reduce these distortions. With weaker dependence, size distortions are modest and disappear entirely under independence.

Differences among the four bootstrap schemes are minor relative to the influence of block length. Overall, the results highlight the importance of adequately capturing dependence when bootstrapping time-series data.

Careful selection of block length is therefore essential. Very short blocks can produce substantial size distortions when serial dependence is strong, particularly under negative correlation. Data-driven selection procedures are preferable whenever available, as they adapt to the dependence structure of the data. Reporting the selected block length and documenting sensitivity to alternative values can further enhance transparency and reproducibility.