Applied Financial Econometrics using Stata

4. Testing for Bubbles

Stan Hurn

Queensland University of Technology, and
National Center for Econometric Research
1. The Problem

2. Testing Framework

3. Recursive Procedures

4. Assessing Significance
Background

- During the 1990s, led by Dot-Com stocks and the internet sector, the United States stock market experienced a spectacular rise in all major indices, especially the NASDAQ index.
- The steep upward movement in the series continues until the late 1990s as investment in Dot-Com stocks grew in popularity.
- Early in the year 2000 the Index drops abruptly and then continues to fall to the mid-1990s level.
- What caused the unusual surge and fall in prices, whether there were bubbles, and whether the bubbles were rational or behavioural are among the most actively debated issues in macroeconomics and finance in recent years.
The Problem

NASDAQ Index Expressed in Real Terms
A recent series of papers places by Peter Phillips and Jun Yu and their co-authors, focuses on empirical tests for bubbles and rational exuberance. This is an interesting new development in the field of unit root testing.

- Alternative of stationarity requires using a one-sided test where the critical region is defined in the left-hand tail of the distribution of the unit root test statistic.
- Alternative of an explosive unit root requires using a one-sided test where the critical region is defined in the right-hand tail of the distribution.

The null and alternative hypotheses of interest are

\[ H_0 : \text{ (Variable is nonstationary, No price bubble)} \]
\[ H_1 : \text{ (Variable is explosive, Price bubble)} \]
The Problem

Simple Model

To motivate the presence of a price bubble, consider the following model

\[ P_t(1 + R) = \mathbb{E}_t [P_{t+1} + D_{t+1}], \]

where \( P_t \) is the price of an asset, \( R \) is the risk-free rate of interest assumed to be constant for simplicity, \( D_t \) is the dividend and \( \mathbb{E}_t [\cdot] \) is the conditional expectations operator. This equation highlights two types of investment strategies.

1. Investing in a risk-free asset at time \( t \) yielding a payoff of \( P_t(1 + R) \) in the next period.
2. Hold the asset and earn the capital gain plus a dividend payment.

In equilibrium there are no arbitrage opportunities so the two two types of investment are equal to each other.
Substituting Forward

Now write the equation as

\[ P_t = \beta \mathbb{E}_t [P_{t+1} + D_{t+1}] , \]

where \( \beta = (1 + R)^{-1} \) is the discount factor. Now

\[ P_{t+1} = \beta \mathbb{E}_t [P_{t+2} + D_{t+2}] , \]

which can be used to substitute out \( P_{t+1} \) to give

\[ P_t = \beta \mathbb{E}_t [\beta \mathbb{E}_t [P_{t+2} + D_{t+2}] + D_{t+1}] \]
\[ = \beta \mathbb{E}_t [D_{t+1}] + \beta^2 \mathbb{E}_t [D_{t+2}] + \mathbb{E}_t [P_{t+2}] . \]

Repeating this approach \( N \) times gives the price of the asset in terms of two components

\[ P_t = \sum_{j=1}^{N} \beta^j \mathbb{E}_t [D_{t+j}] + \beta^N \mathbb{E}_t [P_{t+N}] . \quad (1) \]
The first term on the right-hand side is the standard present value of an asset. The second term represents the price bubble

\[ B_t = \beta^N \mathbb{E}_t [P_{t+N}] , \]

Consider the conditional expectation of the bubble the next period discounted by \( \beta \) and using the property \( \mathbb{E}_t [\mathbb{E}_{t+1} [\cdot]] = \mathbb{E}_t [\cdot] \):

\[ \beta \mathbb{E}_t [B_{t+1}] = \beta \mathbb{E}_t \left[ \beta^N \mathbb{E}_{t+1} [P_{t+N+1}] \right] = \beta^{N+1} \mathbb{E}_t [P_{t+N+1}] \quad (2) \]

Compare expressions (2) and (1). It seems we can write

\[ B_t = \beta \mathbb{E}_t [B_{t+1}] \]

or, given that \( \beta = (1 + R)^{-1} \)

\[ \mathbb{E}_t [B_{t+1}] = (1 + R) B_t \]

which represents a random walk in \( B_t \) explosive parameter \( 1 + R \).
Stata provides a number of unit root tests

- dfuller
- pperron
- dfgls
- xtunitroot

A user written package by Baum will do the KPSS test. See

http://www.stata-journal.com/software/sj6-3

The discussion here will be based simply on the Dickey-Fuller framework.
Testing Framework

DF tests

There are three forms of the Dickey-Fuller tests:

Model 1: \[ \Delta y_t = \beta y_{t-1} + u_t \]
Model 2: \[ \Delta y_t = \alpha + \beta y_{t-1} + u_t \]
Model 3: \[ \Delta y_t = \alpha + \delta t_t + \beta y_{t-1} + u_t. \]

The associated forms of the Augmented Dickey-Fuller test are:

Model 1: \[ \Delta y_t = \beta y_{t-1} + \sum_{i=1}^{p} \phi_i \Delta y_{t-i} + u_t \]
Model 2: \[ \Delta y_t = \alpha + \beta y_{t-1} + \sum_{i=1}^{p} \phi_i \Delta y_{t-i} + u_t \]
Model 3: \[ \Delta y_t = \alpha + \delta t_t + \beta y_{t-1} + \sum_{i=1}^{p} \phi_i \Delta y_{t-i} + u_t, \]

in which the lag length \( p \) is chosen to ensure that \( u_t \) does not exhibit autocorrelation.
DF distributions

- For each of these models the form of the Dickey-Fuller test is still the same, namely the test of $\beta = 0$.

- The pertinent distribution in each case, however, is not the same because the distribution of the test statistic changes depending on whether a constant and or a time trend is included.

- The distributions of different versions of Dickey-Fuller tests must be obtained by simulation. The key point to note is that all three Dickey Fuller distributions are skewed to the left with respect to the standard normal distribution. In addition, the distribution becomes less negatively skewed as more deterministic components (constants and time trends) are included.
Simulating Nonstationary Data

. forvalues j=1(1)$N {
  2.      gen et`j´=rnormal(0,1)
  3.      gen y1t`j´=.
  4.      replace y1t`j´=et`j´ in 1
  5.      forvalues i=2(1)$T {
  6.        replace y1t`j´= y1t`j´[`i´-1]+et`j´[`i´] in `i´
  7.      }
  8.  }

Using the post Command

. // Initialise Dy and x to missing values
. gen Dy =.
(500 missing values generated)

. gen x =.
(500 missing values generated)

. tempname sim
. postfile `sim´ df1 df2 df3 using results, replace

. forvalues i = 1/$N {
  2.   replace Dy = D.y1t`i´
  3.   replace x = L.y1t`i´
  4.   reg Dy x, noconstant  // Model 1
  5.   scalar df1 = _b[x]/_se[x]
  6.   reg Dy x
  7.   scalar df2 = _b[x]/_se[x]
  8.   reg Dy x t
  9.   scalar df3 = _b[x]/_se[x]
 10.   post `sim´ (df1) (df2) (df3)
11. }

(499 real changes made)
(499 real changes made)
The record Feature

. // title edits
. gr_edit .title.text = {}
. gr_edit .title.text.Arrpush Distribution of the Dickey Fuller Tests
.
. // label[1] edits
. gr_edit .legend.plotregion1.label[1].text = {}
. gr_edit .legend.plotregion1.label[1].text.Arrpush no constant or trend
.
. // label[2] edits
. gr_edit .legend.plotregion1.label[2].text = {}
. gr_edit .legend.plotregion1.label[2].text.Arrpush constant but no trend
.
. // label[3] edits
. gr_edit .legend.plotregion1.label[3].text = {}
. gr_edit .legend.plotregion1.label[3].text.Arrpush constant and trend
.
. // label[4] edits
. gr_edit .legend.plotregion1.label[4].text = {}
. gr_edit .legend.plotregion1.label[4].text.Arrpush standard normal
Distribution of the Dickey Fuller Tests

- dotted line: no constant or trend
- dashed line: constant but no trend
- solid line: constant and trend
- orange line: standard normal

Hurn (NCER)
Applied Financial Econometrics using Stata
Log of Nasdaq

Logarithm of the NASDAQ Index

ln
Logarithm of the NASDAQ Index

Applied Financial Econometrics using Stata
ADF(1) test

. // traditional ADF test (lags restricted to be 1)
. dfuller ln, lags(1)
time variable not set, use -tsset varname ...-
    r(111);
end of do-file
r(111);
. do "/var/folders/yh/nld9zdgd00x_658c0v11_0xc0000gn/T//SD00266.000000"
. // setup
. gen t = _n // use this as the time series identifier
. tsset t // avoids problems with dates in the merge
          time variable:  t, 1 to 432
          delta:  1 unit

end of do-file
. do "/var/folders/yh/nld9zdgd00x_658c0v11_0xc0000gn/T//SD00266.000000"
. dfuller ln, lags(1)

Augmented Dickey-Fuller test for unit root
Number of obs = 430

Interpolated Dickey-Fuller

<table>
<thead>
<tr>
<th>Test Statistic</th>
<th>1% Critical Value</th>
<th>5% Critical Value</th>
<th>10% Critical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z(t)</td>
<td>-1.045</td>
<td>-3.446</td>
<td>-2.873</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MacKinnon approximate p-value for Z(t) = 0.7366
Detrended Nasdaq

OLS Detrended NASDAQ

Hurn (NCER) 
Applied Financial Econometrics using Stata
### OLS detrended ADF(1) test

```stata
. // detrend the data using OLS detrending
. qui reg ln t
. predict double u, residual
. lab var u "Detrended NASDAQ"
.
. // OLS detrended ADF test
. dfuller u, noconstant lags(1)
```

Augmented Dickey-Fuller test for unit root

<table>
<thead>
<tr>
<th>Test Statistic</th>
<th>1% Critical Value</th>
<th>5% Critical Value</th>
<th>10% Critical Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z(t)</td>
<td>-2.672</td>
<td>-2.580</td>
<td>-1.950</td>
</tr>
</tbody>
</table>

Number of obs = 430
Interestingly enough, if we were to follow the convention and apply the ADF test to the full sample (February 1973 to January 2009), the unit root test would not reject the null hypothesis $H_0 : \rho = 1$ in favour of the right-tailed alternative hypothesis $H_1 : \rho > 1$ at the 5 % level of significance. One would conclude that there is no significant evidence of exuberance in the behaviour of the NASDAQ index over the sample period. This result would sit comfortably with the consensus view that there is little empirical evidence to support the hypothesis of explosive behaviour in stock prices.
Evans (1991) argues that explosive behaviour is only temporary in the sense that economic eventually bubbles collapse and that therefore the observed trajectories of asset prices may appear rather more like an I(1) or even a stationary series than an explosive series, thereby confounding empirical evidence. Evans demonstrates by simulation that standard unit root tests have difficulties in detecting such periodically collapsing bubbles. In order for unit root test procedures to be powerful in detecting bubbles, the use of recursive unit root testing proves to an invaluable approach in the detection and dating of bubbles.
Using rolling

. // forward recursive adf test
. rolling fADF = r(Zt), window(40) r sa(./working/fadftests,replace) keep(date): ///
    dfuller ln, lags(1)
(running dfuller on estimation sample)

Rolling replications (393)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>100</td>
<td>150</td>
<td>200</td>
<td>250</td>
</tr>
<tr>
<td>300</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

file ./working/fadftests.dta saved
More Promising Line of Attack

Recursive Procedures

Recursive ADF test (1 lag)

Forward ADF

Backward ADF


Recursive ADF tests

1. The ADF statistic with 1 lag computed from forward recursive regressions (fixing the start of the sample period and progressively increasing the sample size observation by observation until the entire sample is being used) shows no evidence of rational exuberance until late in the 1990s.

2. The value of the recursive ADF test reported in the backward recursive procedure appears also to show fair variation.

3. First idea is then to compute the Supremum ADF statistic in a backward recursive sequence.
Recursions By Hand

. // do recursions by hand
. // backward recursive adf test
. tempname sim
. local k = 392
. postfile `sim´ index df using "./working/back", replace
.  forvalues i = `k´(-1)1 {
    2. qui dfuller ln in `i´/432, lags(1)
    3. post `sim´ (`i´) (`r(Zt)´)
    4. }
. postclose `sim´
Looks Identical
Forward Sup test of PWY
Recursive Procedures

Forward Sup test of PWY

![Graph showing stock price over time.](image-url)
Forward Sup test of PWY
Single Recursive SupADF test

. // forward recursive sup adf test
. tempname sim
. postfile `sim´ t df using "./working/forward", replace
. forvalues i = 40(1)432 {
    2. qui dfuller u in 1/`i´, noconstant lags(1)
    3. post `sim´ (`i´) ("r(Zt)´)
    4. }
. postclose `sim´
. use "./working/forward", clear
. qui summ df
. di "Recursive SupADF test is : " r(max)
Recursive SupADF test is : -.98873168
Double Recursive SupADF test
Recursive Procedures

Double Recursive SupADF test

![Graph showing stock price over time](image-url)
Double Recursive SupADF test
. // double recursive sup adf test
. tempname sim
. postfile `sim´ t df using "./working/forward", replace
. forvalues i = 1(1)392 {
  2. local k = `i´+39
  3. forvalues j = `k´(1)432 {
  4.      qui dfuller u in `i´/`j´, noconstant lags(1)
  5.      post `sim´ (`j´) (`r(Zt)´)
  6.  }
  7. }
. postclose `sim´
. use "./working/forward", clear
. qui summ df
. di "Double recursive SupADF test is : " r(max)
Double recursive SupADF test is : 2.8896754
Double Recursive SupADF test

![Graph of Double Recursive SupADF test]

- The graph illustrates the recursive process of the SupADF test over the years 1970m1 to 2010m1.
- The x-axis represents the date, with intervals of one year.
- The y-axis shows the r(max) values, ranging from -4 to 4.

Hurn (NCER)  |  Applied Financial Econometrics using Stata  |  36 / 42
Date stamping test
Date stamping test
Harris (1992) and Park (2003) Bootstrap

Want to bootstrap $t_\beta$ of the null hypothesis that $\beta = 0$ in

$$\Delta y_t = \beta y_{t-1} + \gamma \Delta y_{t-1} + u$$

(3)

1. Estimate the equation which has $\beta = 0$ imposed:

$$\Delta y_t = \gamma \Delta y_{t-1} + \nu_t$$

2. Recenter the residuals

$$\tilde{\nu}_t = \hat{\nu}_t - \frac{1}{T} \sum_{i=1}^{T} \hat{\nu}_i.$$ 

3. Set $y_1^* = y_1$ and $y_2^* = y_2$. Compute

$$y_t^* = y_{t-1}^* + \tilde{\gamma}(y_{t-1}^* - y_{t-2}^*) + \tilde{\nu}_t^*$$

4. Estimate the unrestricted equation (3) on the generated data, $y_t^*$, and compute the ADF statistic $t_\beta^*$. 

5. Repeat
// detrend the data using OLS detrending
qui reg ln t
predict double u, residual
lab var u "Detrended NASDAQ"

// estimate the model with HO imposed
qui reg D.u LD.u, noconstant
estimates store dfreg
global beta = _b[LD.u]
save "./working/tempdata",replace

// compute centered residuals and save
qui predict double v, residual
qui replace v = 0 if mi(v)
center v, inplace
keep v
save "./working/bootdata",replace

local nreps 500

qui simulate dfstat=r(Zt) pval=r(p), reps(`nreps') saving("./working/testbs`nreps'', replace): adf_bs
c<br>capt prog drop adf_bs  
prog adf_bs, rclass
version 12

// open data
use "./working/bootdata",clear

// resample and merge
ren v vstar
keep vstar
bsample
save "./working/bootdata2", replace

use "./working/tempdata", clear
merge 1:1 _n using "./working/bootdata2"
drop _merge

// generate bootstrap sample
tset t
gen dy = u in 1/2
replace dy = $beta * L.dy + vstar in 3/l
gen ystar = sum(dy) in 2/l

// run the ADF test on these data
dfuller ystar, lags(1)
return scalar Zt = r(Zt)
return scalar p = r(p)
end