An Eviews program to perform the fractional Dickey-Fuller test
Ahmed Bensalma

To cite this version:
Ahmed Bensalma. An Eviews program to perform the fractional Dickey-Fuller test. 2021. hal-03216194

HAL Id: hal-03216194
https://hal.science/hal-03216194
Preprint submitted on 8 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
An Eviews program to perform the fractional Dickey-Fuller test

Ahmed Bensalma
Department of Statistics,
Laboratoire de Modélisation de Phénomènes Stochastiques (LAMOPS)
Ecole Nationale Supérieure de Statistique et d’Economie Appliquée (ENSSEA)
Pôle universitaire de Koléa, Tipaza, Algeria
bensalma.ahmed@gmail.com

ABSTRACT
This paper demonstrates how sequential fractional Dickey-Fuller (FDF in short) test can be implemented in EViews. We first briefly introduce how to use the fracdiff an EViews add-in to compute the fractional difference of the Nile data. Next, we give the program that executes the sequential FDF testing on the Nile data series.

Key Words: ARFIMA; Dickey-Fuller test; Fractional Dickey-Fuller test; fractional integration parameter; type II fractional Brownian motion, Fracdiff, EViews add-in.

1 Introduction

Let consider an ARFIMA(0,d,0) process defined by

\[ y_t = (1 - L)^{-d}u_t^*, \quad t = 1, 2, \ldots, n, \tag{1.1} \]

with initial conditions \( y_t = 0 \), if \( t < 1 \) and where

\[ u_t^* = \begin{cases} u_t, & \text{if } t \geq 1 \\ 0, & \text{otherwise} \end{cases} \]

where \( u_t \) are independent and identically distributed (i.i.d) random variables and \( L \) is backward shift operator \( Ly_t = y_{t-1} \). The fractional integration operator \( (1 - L)^{-d} \) is defined by its Maclaurin series (by its binomial expansion, if \( d \) is an integer):

\[ (1 - L)^{-d} = \sum_{j=0}^{\infty} \frac{\Gamma(d+j)}{\Gamma(d+1)\Gamma(j+1)}L^j \]

where

\[ \Gamma(z) = \begin{cases} \int_0^\infty s^{z-1}e^{-s}ds & \text{If } z > 0 \\ \infty & \text{if } z = 0. \end{cases} \]

If \( z < 0 \), \( \Gamma(z) \) is defined by the recursion formula \( z\Gamma(z) = \Gamma(z+1) \).

In EViews 9, a general procedure to compute the fractional difference of a given series \( \{y_t, \quad t = 1, \ldots, n\} \) is to apply the formula

\[ x_t = (1 - L)^d y_t = \sum_{j=0}^{t-1} \frac{\Gamma(-d-j)}{\Gamma(-d)\Gamma(j+1)}y_{t-j}. \tag{1.2} \]

For example, to compute the fractional difference of the demeaned Nile data with \( d = 0.3 \), we can use the following command lines,
1. `series y=Nile-@mean(Nile)`

2. `y.fracdiff(d=0.3)`

The second command line, simply take the difference of order 0.3 and saves the output as `y_diff`.

![Image of two graphs showing the computation of fractional difference of the Nile series](image)

**Figure 1:** Compute fractional difference of the Nile series

For another naming output, we can use the third command line,

3. `rename y_diff x`

![Image of two graphs showing the computation of fractional difference of the Nile series](image)

**Figure 2:** Compute fractional difference of the Nile series

If one want to compute many fractional difference series of the Nile data for a sequence of different values of $d$, for example, $d_1 = 0.1$, $d_2 = 0.2$, $d_3 = 0.3$, $\ldots$, $d_{10} = 1$ we can use the following command lines

1. `series y=nile-@mean(nile)`
2. `for !i=1 to 10`
3. `!d=0.1*!i`
4. `y.fracdiff(d=!d)`
5. `rename y_diff x!i`
6. `next`

The output is $x_1$, $x_2$, $\ldots$, $x_{10}$, where

$$x_{i,t} = (1 - L)^{0.1*i}y_t$$
2 Fractional Dickey-Fuller test

If \( \{y_t, \ t = 1, \cdots , n\} \) is a sample of an ARFIMA\((0, d, 0)\) process with \( d \in (-0.5, +\infty) \) then we can use the process \( x_t = (1 - L)^{d_0-1}y_t \) to test the null hypothesis

\[
H_0 : d \geq d_0, \ \text{with} \ d_0 \in (-0.5, +\infty),
\]

by using the regression model

\[
(1 - L)^{d_0}y_t = \rho(1 - L)^{d_0-1}y_{t-1} + \varepsilon_t, \ \ t = 1, 2, \cdots , n.
\]

Bensalma (2018) show that the domains of limit probability density function of

\[
DF_n = n\hat{\rho} = \frac{\sum_{t=1}^{n} x_{t-1}\Delta x_t}{\sum_{t=1}^{n}(x_{t-1})^2} \quad \text{and} \quad DF_t = t\hat{\rho} = \frac{\sum_{t=1}^{n} x_{t-1}\Delta x_t}{(\sum_{t=1}^{n}(x_{t-1})^2)^{1/2}}
\]

are

\[
\begin{align*}
\mathbb{R}^- & \quad \text{if} \ d < d_0, \\
\mathbb{R} & \quad \text{if} \ d = d_0, \\
\mathbb{R}^+ & \quad \text{if} \ d > d_0,
\end{align*}
\]

\( x_t = (1 - L)^{d_0-1}y_t \) is an \( I(d - d_0 + 1) \) process and then we have the following three cases

\[
(d - d_0 + 1) \begin{cases} < 1 & \text{if} \ d < d_0, \\ = 1 & \text{if} \ d = d_0, \\ > 1 & \text{if} \ d > d_0. \end{cases}
\]

The limiting distribution of \( DF_n \) and \( DF_t \) is described in the following theorem.

Theorem 1: (Bensalma (2018)) Let be a sample of an ARFIMA\((0, d, 0)\) process. If a regression model (1.4) is fitted to a sample of size \( n \) then, as \( n \to \infty \), we have

1. \( DF_n \to -\infty \) and \( DF_t \to -\infty \) if \( d < d_0 \).
2. \( DF_n \to \frac{0.5[W^2(1)-1]}{\int_0^1 W^2(r)dr} \) and \( DF_t \to \frac{0.5[W^2(1)-1]}{\int_0^1 W^2(r)dr} \) if \( d = d_0 \).
3. \( DF_n \to \frac{1}{\int_0^1 W_{d-d_0+1}(r)dr} \) and \( DF_t \to +\infty \) if \( d > d_0 \).

where \( W(\cdot) \) is standard Brownian motion and \( W_d(\cdot) \) is a type 2 fractional Brownian motion defined by

\[
W_d(r) = \frac{1}{\Gamma(d)} \int_0^r (r-s)^{d-1}dW(s), \ \ r \in [0,1]
\]
In practice, when the null is composite, for a given critical value of size $\alpha$ (i.e. $cv(\alpha)$), the probability of type I error is controlled by imposing the following constraint

$$P[type\ I\ error] = \sup_{d \geq d_0} P[DF_t < cv(\alpha)] \leq \alpha.$$  

Given that the limit of probability density of $DF_t$ have a remarkable arrangement (1.5) it is easy to show that

$$\sup_{d \geq d_0} P[DF_t < cv(\alpha)] = P_{d=d_0} [DF_t < cv(\alpha)]$$

This later result combined with the second result of theorem above, which show that $DF_t$ have the Dickey-Fuller limit distribution when $d = d_0$, demonstrate that to perform the fractional Dickey-Fuller test we can use the usual tabulated value of the standard Dickey-Fuller test. Our testing procedure will not enable us to apprehend the case of $H_0 : d \geq 0.5$, but also the general case of $H_0 : d \geq d_0$, with $d_0 \in (-0.5, +\infty)$. Moreover, if we use upward or downward testing sequence for a set of values $d_0^1 < d_0^2 < \cdots < d_0^l$, it is possible to determine an overlap domain of the parameter $d$.

3 Empirical application with Eviews

In this section, we consider the well-known series of annual minima of the Nile, as studied by Hurst (1951) and reproduced in Beran (1994). The sample size is $n = 633$ annual observations ($622 - 1284 AD$)

![Nile series](image)

Figure 4: Nile series

In the following, we apply the sequential F-DF test to the demeaned Nile series, namely

$$y_t = \text{Nile}_t - \bar{\text{Nile}}$$

where $\bar{\text{Nile}} = \frac{\sum_{t=1}^{633} \text{Nile}_t}{633}$. We use upward testing sequence for a set of values $d_0^i \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$.

For a particular value $d_0^i$ from this later set, we test the hypothesis

$$H_{0i} : d \geq d_0^i$$
by using the \( t_{\hat{\rho}_i} = DF_t \) calculated via the estimation of the following autoregression model

\[
(1 - L)^{d_0}y_t = (1 - L)^{d_0-1}y_{t-1} + \varepsilon_t.
\]

The Table 1 summarize the sequential upward testing procedure of the standard FDF test on the \( y_t \) series.

<table>
<thead>
<tr>
<th>( d_0 )</th>
<th>FDF test on ( t_{\hat{\rho}_i} = DF_t )</th>
<th>Dickey-Fuller ( cv(5%) )</th>
<th>Reject or Accept ( H_{0i} : d \geq d_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \Delta^{-1}y_t )</td>
<td>-0.1986</td>
<td>-1.94</td>
</tr>
<tr>
<td>0.1</td>
<td>( \Delta^{-0.9}y_t )</td>
<td>-0.2746</td>
<td>-1.94</td>
</tr>
<tr>
<td>0.2</td>
<td>( \Delta^{-0.8}y_t )</td>
<td>-0.4201</td>
<td>-1.94</td>
</tr>
<tr>
<td>0.3</td>
<td>( \Delta^{-0.7}y_t )</td>
<td>-0.7148</td>
<td>-1.94</td>
</tr>
<tr>
<td><strong>0.4</strong></td>
<td>( \Delta^{-0.6}y_t )</td>
<td><strong>-1.2174</strong></td>
<td><strong>-1.94</strong></td>
</tr>
<tr>
<td><strong>0.5</strong></td>
<td>( \Delta^{-0.5}y_t )</td>
<td><strong>-2.0029</strong></td>
<td><strong>-1.94</strong></td>
</tr>
<tr>
<td>0.6</td>
<td>( \Delta^{-0.4}y_t )</td>
<td>-3.2481</td>
<td>-1.94</td>
</tr>
<tr>
<td>0.7</td>
<td>( \Delta^{-0.3}y_t )</td>
<td>-4.9185</td>
<td>-1.94</td>
</tr>
<tr>
<td>0.8</td>
<td>( \Delta^{-0.2}y_t )</td>
<td>-7.1955</td>
<td>-1.94</td>
</tr>
<tr>
<td>0.9</td>
<td>( \Delta^{-0.1}y_t )</td>
<td>-10.0671</td>
<td>-1.94</td>
</tr>
<tr>
<td>1</td>
<td>( x_t )</td>
<td>-13.3303</td>
<td>-1.94</td>
</tr>
</tbody>
</table>

**Conclusion**

\( 0.4 \leq d < 0.5 \)

The table 1 show that we can apply a downward testing sequence, in this case we take the largest value, (the maximum value of \( d_0 \), i.e. \( d \geq 1 \)), under consideration as the first maintained hypothesis and then decrease the order of differenced each time the current null is rejected. The table 1 show, also, that an upward testing sequence can be applied. In this case, we take the smallest value of \( d_0 \), (i.e. \( d \geq 0 \)) under consideration as the first maintained hypothesis and then increase the order of differenced each time the current alternative is accepted. Table 1 show that the lower and upper bound of fractional integration order of the demeaned Nile data are 0.4 and 0.5 respectively.

The program that executes this sequential testing procedure is shown in the following. The first 30 lines constitute the main program. The rest of the program consists of an understandable formatting of the results in a table. The figure 5, show how the results are displayed in EViews after the execution of the EViews sequential FDF program.
series y=nile-@mean(nile)
vector (10) Accept_Reject_H0

' Set of sequential values of $d_0$
for !i=1 to 10
!d0=0.1*!i
!d=-1+0.1*!i

'Compute $x(t) = (1 - L)^{d_0} y(t)$
y.fracdiff(d=!d, )
rename y.diff x!i

'Testing the null $H_0: d \geq d_0$ by means the t-stat of $c(1)$
'coefficient in the model $(1 - L)^{d_0} y_t = c(1) * (1 - L)^{d_0-1} y_{t-1} + \varepsilon_t$
equation eq!i.ls d(x!i) x!i(-1)
if eq!i.@tstat(1)>-1.94 then Accept_Reject_H0(!i)=1 else Accept_Reject_H0(!i)=0
endif
delete x!i
next

'Find the lower and upper bound of d
for !i= 1 to 9
if Accept_Reject_H0(!i)=1 and Accept_Reject_H0(!i+1)=0 then
scalar Lower_bound_of_d=0.1*!i
scalar Upper_bound_of_d=0.1*(!i+1)
endif
next

' display results in table
table tab1
setcolwidth(tab1,1,20)
tab1(1,1)="Table 1"
tab1(2,1)="Sequential testing procedure,"
tab1(3,1)="of the standard Dickey-Fuller test"
tab1(4,1)="applied to the nile series: $H_0: d \geq d_0,$"
tab1(4,2)=""
tab1(4,3)=""
tab1(4,4)=""
tab1(4,5)=""
setline(tab1,5)
<table>
<thead>
<tr>
<th>d0,i</th>
<th>d0,i-1</th>
<th>DFt</th>
<th>DF cv(5%)</th>
<th>Accept/Reject H0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Accept</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0.1</th>
<th>0.9</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0.2</th>
<th>0.8</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0.3</th>
<th>0.7</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0.4</th>
<th>0.6</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0.5</th>
<th>0.5</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0.6</th>
<th>0.4</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0.7</th>
<th>0.3</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>0.8</th>
<th>0.2</th>
<th>1.0</th>
<th>-1.94</th>
<th>Accept</th>
</tr>
</thead>
</table>
tab1(22,2)="0.2"
tab1(22,3)=eq8.@tstat(1)
tab1(22,4)="-1.94"
tab1(22,5)=Accept_Reject_H0(8)
setline(tab1,23)
tab1(24,1)="0.9"
tab1(24,2)="0.1"
tab1(24,3)=eq9.@tstat(1)
tab1(24,4)="-1.94"
tab1(24,5)=Accept_Reject_H0(9)
setline(tab1,25)
tab1(26,1)="1"
tab1(26,2)="0"
tab1(26,3)=eq10.@tstat(1)
tab1(26,4)="-1.94"
tab1(26,5)=Accept_Reject_H0(10)
setline(tab1,27)
tab1(28,1)="Lower bound of d= + @str(Lower_bound_of_d)"
tab1(29,1)="Upper bound of d= + @str(Upper_bound_of_d)"
tab1(30,1)=@str(Lower_bound_of_d) + "< d < " + @str(Upper_bound_of_d)
show tab1

Figure 5: Output of the execution of the Eviews sequential FDF program
Table 1
Sequential F-DF test, on the nile series:
H0: d = d0,i; i = 1 to 10
Lower bound of d = 0.4
Upper bound of d = 0.5
0.4 < d < 0.5

<table>
<thead>
<tr>
<th>d0,i</th>
<th>d0,i-1</th>
<th>DFt = eqli@tsat(1)</th>
<th>DF distribution cv(5%)</th>
<th>Accept or Reject H0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>-0.293125</td>
<td>-1.94</td>
<td>1.000000</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8</td>
<td>-0.453430</td>
<td>-1.94</td>
<td>1.000000</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7</td>
<td>-0.751589</td>
<td>-1.94</td>
<td>1.000000</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td>-1.241860</td>
<td>-1.94</td>
<td>1.000000</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>-2.000791</td>
<td>-1.94</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>-3.138691</td>
<td>-1.94</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3</td>
<td>-4.784756</td>
<td>-1.94</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2</td>
<td>-7.035215</td>
<td>-1.94</td>
<td>0.000000</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
<td>-9.866600</td>
<td>-1.94</td>
<td>0.000000</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-13.07665</td>
<td>-1.94</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Table 1: Output of the execution of the Eviews sequential FDF program

References


