



# Time Series

(ver. 1.5)

*Oscar Torres-Reyna*

*Data Consultant*

*otorres@princeton.edu*



	date1	date2	date3	date4
1	1-Jan-95	1/1/1995	19950101	199511
2	2-Jan-95	1/2/1995	19950102	199512
3	3-Jan-95	1/3/1995	19950103	199513
4	4-Jan-95	1/4/1995	19950104	199514
5	5-Jan-95	1/5/1995	19950105	199515
6	6-Jan-95	1/6/1995	19950106	199516
7	7-Jan-95	1/7/1995	19950107	199517
8	8-Jan-95	1/8/1995	19950108	199518
9	9-Jan-95	1/9/1995	19950109	199519
10	10-Jan-95	1/10/1995	19950110	1995110
11	11-Jan-95	1/11/1995	19950111	1995111
12	12-Jan-95	1/12/1995	19950112	1995112
13	13-Jan-95	1/13/1995	19950113	1995113
14	14-Jan-95	1/14/1995	19950114	1995114
15	15-Jan-95	1/15/1995	19950115	1995115
16	16-Jan-95	1/16/1995	19950116	1995116
17	17-Jan-95	1/17/1995	19950117	1995117
18	18-Jan-95	1/18/1995	19950118	1995118
19	19-Jan-95	1/19/1995	19950119	1995119
20	20-Jan-95	1/20/1995	19950120	1995120
21	21-Jan-95	1/21/1995	19950121	1995121
22	22-Jan-95	1/22/1995	19950122	1995122
23	23-Jan-95	1/23/1995	19950123	1995123
24	24-Jan-95	1/24/1995	19950124	1995124
25	25-Jan-95	1/25/1995	19950125	1995125
26	26-Jan-95	1/26/1995	19950126	1995126
27	27-Jan-95	1/27/1995	19950127	1995127
28	28-Jan-95	1/28/1995	19950128	1995128
29	29-Jan-95	1/29/1995	19950129	1995129
30	30-Jan-95	1/30/1995	19950130	1995130

For **'date1'** type:

```
gen datevar = date(date1,"DMY", 2099)
format datevar %td /*For daily data*/
```

For **'date2'** type:

```
gen datevar = date(date2,"MDY", 2099)
format datevar %td /*For daily data*/
```

For **'date3'** type:

```
tostring date3, gen(date3a)
gen datevar=date(date3a,"YMD")
format datevar %td /*For daily data*/
```

For **'date4'** see next page

# Date in integers and unbalanced

```
use "http://www.princeton.edu/~otorres/Stata/date.dta", clear
drop date1 date2
rename date3 date4
```

	date4	date4a	len	year	month	day	datevar
1	199511	199511	6	1995	1	1	01jan1995
2	199512	199512	6	1995	1	2	02jan1995
3	199513	199513	6	1995	1	3	03jan1995
4	199514	199514	6	1995	1	4	04jan1995
5	199515	199515	6	1995	1	5	05jan1995
6	199516	199516	6	1995	1	6	06jan1995
7	199517	199517	6	1995	1	7	07jan1995
8	199518	199518	6	1995	1	8	08jan1995
9	199519	199519	6	1995	1	9	09jan1995
10	1995110	1995110	7	1995	.	.	10jan1995
11	1995111	1995111	7	1995	.	.	11jan1995
12	1995112	1995112	7	1995	.	.	12jan1995
13	1995113	1995113	7	1995	.	.	13jan1995

## **\*Year, month, day**

```
tostring date4, gen(date4a)
```

```
gen len = length(date4a)
```

```
gen year = substr(date4a,1,4)
```

## **\*When len=6, month is in 5th position and day in 6th**

```
gen month = substr(date4a,5,1) if len == 6
```

```
gen day = substr(date4a,6,1) if len == 6
```

## **\*When len=7 is hard to distinguish month/day, we skip**

## **\*When len=8, month is in 5th/6th position and day in 7th/8th**

```
replace month = substr(date4a,5,2) if len == 8
```

```
replace day = substr(date4a,7,2) if len == 8
```

```
destring month day year, replace
```

## **\*Creating datevar**

```
gen datevar = mdy(month,day,year)
```

```
format datevar %td
```

## **\*Filling in the missing dates**

```
replace datevar = datevar[_n-1] + 1 if datevar == .
```

If the original date variable is **string** (i.e. color red):

```
gen week= weekly(stringvar,"wy")
gen month= monthly(stringvar,"my")
gen quarter= quarterly(stringvar,"qy")
gen half = halfyearly(stringvar,"hy")
gen year= yearly(stringvar,"y")
```

**NOTE:** Remember to format the date variable accordingly. After creating it type:

```
format datevar %t? /*Change 'datevar' with your date variable*/
```

Change “?” with the correct format: w (week), m (monthly), q (quarterly), h (half), y (yearly).

If the components of the original date are in different **numeric** variables (i.e. color black):

```
gen daily = mdy(month,day,year)
gen week = yw(year, week)
gen month = ym(year,month)
gen quarter = yq(year,quarter)
gen half = yh(year,half-year)
```

**NOTE:** Remember to format the date variable accordingly. After creating it type:

```
format datevar %t? /*Change 'datevar' with your date variable*/
```

Change “?” with the correct format: w (week), m (monthly), q (quarterly), h (half), y (yearly).

To extract days of the week (Monday, Tuesday, etc.) use the function `dow()`

```
gen dayofweek= dow(date)
```

Replace “date” with the date variable in your dataset. This will create the variable ‘dayofweek’ where 0 is ‘Sunday’, 1 is ‘Monday’, etc. (type `help dow` for more details)

To specify a range of dates (or integers in general) you can use the `tin()` and `twithin()` functions. `tin()` includes the first and last date, `twithin()` does not. Use the format of the date variable in your dataset.

```
/* Make sure to set your data as time series before using tin/twithin */
```

```
tsset date
```

```
regress y x1 x2 if tin(01jan1995,01jun1995)
regress y x1 x2 if twwithin(01jan2000,01jan2001)
```

## Date variable (example)

Time series data is data collected over time for a single or a group of variables. For this kind of data the first thing to do is to check the variable that contains the time or date range and make sure is the one you need: yearly, monthly, quarterly, daily, etc.

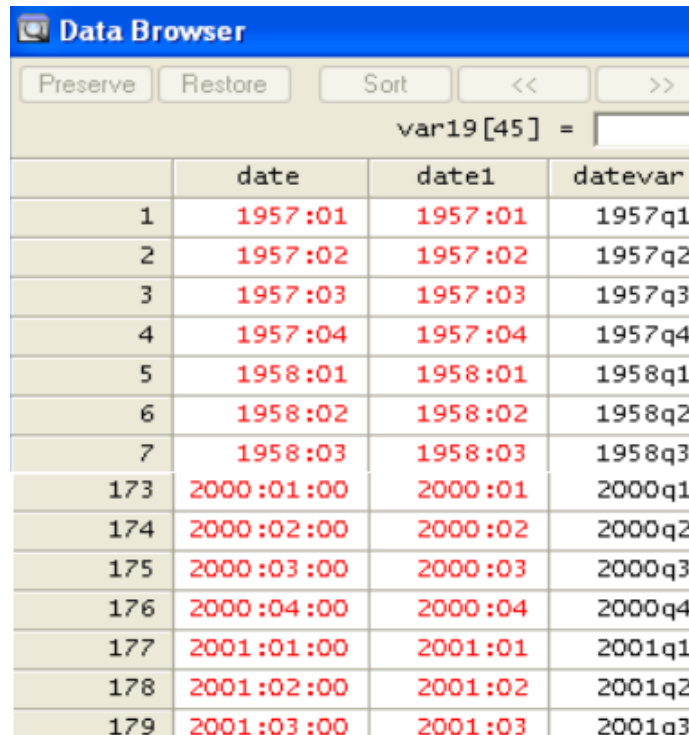
The next step is to verify it is in the correct format. In the example below the time variable is stored in “date” but it is a string variable not a date variable. In Stata you need to convert this string variable to a date variable.\*

A closer inspection of the variable, for the years 2000 the format changes, we need to create a new variable with a uniform format. Type the following:

```
use http://dss.princeton.edu/training/tsdata.dta
gen date1=substr(date,1,7)
gen datevar=quarterly(date1,"yq")
format datevar %tq
browse date date1 datevar
```

For more details type

```
help date
```



	date	date1	datevar
1	1957:01	1957:01	1957q1
2	1957:02	1957:02	1957q2
3	1957:03	1957:03	1957q3
4	1957:04	1957:04	1957q4
5	1958:01	1958:01	1958q1
6	1958:02	1958:02	1958q2
7	1958:03	1958:03	1958q3
173	2000:01:00	2000:01	2000q1
174	2000:02:00	2000:02	2000q2
175	2000:03:00	2000:03	2000q3
176	2000:04:00	2000:04	2000q4
177	2001:01:00	2001:01	2001q1
178	2001:02:00	2001:02	2001q2
179	2001:03:00	2001:03	2001q3

\*Data source: Stock & Watson's companion materials

# From daily/monthly date variable to quarterly

```
use "http://dss.princeton.edu/training/date.dta", clear
```

## \*Quarterly date from daily date

```
gen datevar=date(date2,"MDY", 2099) /*Date2 is a string date variable*/  
format datevar %td  
gen quarterly = qofd(datevar)  
format quarterly %tq
```

## \*Quarterly date from monthly date

```
gen month = month(datevar)  
gen day=day(datevar)  
gen year=year(datevar)  
gen monthly = ym(year,month)  
format monthly %tm  
  
gen quarterly1 = qofd(dofm(monthly))  
format quarterly1 %tq
```

```
browse date2 datevar quarterly monthly quarterly1
```

	date2	datevar	quarterly	monthly	quarterly1
1	1/1/1995	01jan1995	1995q1	1995m1	1995q1
2	1/2/1995	02jan1995	1995q1	1995m1	1995q1
3	1/3/1995	03jan1995	1995q1	1995m1	1995q1
4	1/4/1995	04jan1995	1995q1	1995m1	1995q1
5	1/5/1995	05jan1995	1995q1	1995m1	1995q1
6	1/6/1995	06jan1995	1995q1	1995m1	1995q1
7	1/7/1995	07jan1995	1995q1	1995m1	1995q1
8	1/8/1995	08jan1995	1995q1	1995m1	1995q1
9	1/9/1995	09jan1995	1995q1	1995m1	1995q1
10	1/10/1995	10jan1995	1995q1	1995m1	1995q1
11	1/11/1995	11jan1995	1995q1	1995m1	1995q1
12	1/12/1995	12jan1995	1995q1	1995m1	1995q1
13	1/13/1995	13jan1995	1995q1	1995m1	1995q1

# From daily to weekly and getting yearly

```
use "http://dss.princeton.edu/training/date.dta", clear
gen datevar = date(date2, "MDY", 2099)
format datevar %td
gen year= year(datevar)
gen w = week(datevar)
gen weekly = yw(year,w)
format weekly %tw

browse
```

	date2	datevar	year	w	weekly
1	1/1/1995	01jan1995	1995	1	1995w1
2	1/2/1995	02jan1995	1995	1	1995w1
3	1/3/1995	03jan1995	1995	1	1995w1
4	1/4/1995	04jan1995	1995	1	1995w1
5	1/5/1995	05jan1995	1995	1	1995w1
6	1/6/1995	06jan1995	1995	1	1995w1
7	1/7/1995	07jan1995	1995	1	1995w1
8	1/8/1995	08jan1995	1995	2	1995w2
9	1/9/1995	09jan1995	1995	2	1995w2
10	1/10/1995	10jan1995	1995	2	1995w2
11	1/11/1995	11jan1995	1995	2	1995w2
12	1/12/1995	12jan1995	1995	2	1995w2
13	1/13/1995	13jan1995	1995	2	1995w2

\*\*\*\*\*

```
* From daily to yearly
gen year1 = year(datevar)
* From quarterly to yearly
gen year2 = yofd(dofq(quarterly))
* From weekly to yearly
gen year3 = yofd(dofw(weekly))
```

	datevar	quarterly	weekly	year1	year2	year3
1	01jan1995	1995q1	1995w1	1995	1995	1995
2	02jan1995	1995q1	1995w1	1995	1995	1995
3	03jan1995	1995q1	1995w1	1995	1995	1995
4	04jan1995	1995q1	1995w1	1995	1995	1995
5	05jan1995	1995q1	1995w1	1995	1995	1995
6	06jan1995	1995q1	1995w1	1995	1995	1995
7	07jan1995	1995q1	1995w1	1995	1995	1995
8	08jan1995	1995q1	1995w2	1995	1995	1995
9	09jan1995	1995q1	1995w2	1995	1995	1995
10	10jan1995	1995q1	1995w2	1995	1995	1995
11	11jan1995	1995q1	1995w2	1995	1995	1995
12	12jan1995	1995q1	1995w2	1995	1995	1995
13	13jan1995	1995q1	1995w2	1995	1995	1995

Once you have the date variable in a 'date format' you need to declare your data as time series in order to use the time series operators. In Stata type:

```
tsset datevar
```

```
. tsset datevar  
      time variable:  datevar, 1957q1 to 2005q1  
      delta:         1 quarter
```

If you have **gaps** in your time series, for example there may not be data available for weekends. This complicates the analysis using lags for those missing dates. In this case you may want to create a continuous time trend as follows:

```
gen time = _n
```

Then use it to set the time series:

```
tsset time
```

In the case of ***cross-sectional time series*** type:

```
sort panel date  
by panel: gen time = _n  
xtset panel time
```



# Filling gaps in time variables

Use the command `tsfill` to fill in the gap in the time series. You need to `tset`, `tsset` or `xtset` the data before using `tsfill`. In the example below:

```
tset quarters
```

```
tsfill
```

	quarters	unemp	cpi	interest	gdp
1	2000q1	4	170	5.85	.55
2	2000q3	4	173	6.52	.134
3	2000q4	3.9	174	6.4	.631
4	2001q1	4.2	176	5.31	.284
5	2001q4	5.5	177	1.82	-.959
6	2002q1	5.7	178	1.73	-.0585
7	2002q2	5.8	180	1.75	.0205
8	2002q3	5.7	180	1.75	.651
9	2003q1	5.8	183	1.25	-.816
10	2003q3	6.1	184	1.01	.312
11	2003q4	5.9	185	.98	.628



	quarters	unemp	cpi	interest	gdp
1	2000q1	4	170	5.85	.55
2	2000q2	.	.	.	.
3	2000q3	4	173	6.52	.134
4	2000q4	3.9	174	6.4	.631
5	2001q1	4.2	176	5.31	.284
6	2001q2	.	.	.	.
7	2001q3	.	.	.	.
8	2001q4	5.5	177	1.82	-.959
9	2002q1	5.7	178	1.73	-.0585
10	2002q2	5.8	180	1.75	.0205
11	2002q3	5.7	180	1.75	.651
12	2002q4	.	.	.	.
13	2003q1	5.8	183	1.25	-.816
14	2003q2	.	.	.	.
15	2003q3	6.1	184	1.01	.312
16	2003q4	5.9	185	.98	.628

Type `help tsfill` for more details.

With `tsset` (time series set) you can use two time series commands: `tin` ('times in', from *a* to *b*) and `twwithin` ('times within', between *a* and *b*, it excludes *a* and *b*). If you have yearly data just include the years.

```
. list datevar unemp if tin(2000q1,2000q4)
```

	<b>datevar</b>	<b>unemp</b>
173.	<b>2000q1</b>	<b>4.033333</b>
174.	<b>2000q2</b>	<b>3.933333</b>
175.	<b>2000q3</b>	<b>4</b>
176.	<b>2000q4</b>	<b>3.9</b>

```
. list datevar unemp if twwithin(2000q1,2000q4)
```

	<b>datevar</b>	<b>unemp</b>
174.	<b>2000q2</b>	<b>3.933333</b>
175.	<b>2000q3</b>	<b>4</b>

```
/* Make sure to set your data as time series before using tin/twithin */
```

```
tsset date
```

```
regress y x1 x2 if tin(01jan1995,01jun1995)
```

```
regress y x1 x2 if twwithin(01jan2000,01jan2001)
```

See

<http://dss.princeton.edu/training/Merge101.pdf>

Another set of time series commands are the lags, leads, differences and seasonal operators. It is common to analyzed the impact of previous values on current ones.

To generate values with past values use the “L” operator

```
generate unempL1=L1.unemp
generate unempL2=L2.unemp
list datevar unemp unempL1 unempL2 in 1/5
```

```
. generate unempL1=L1.unemp
(1 missing value generated)
```

```
. generate unempL2=L2.unemp
(2 missing values generated)
```

```
. list datevar unemp unempL1 unempL2 in 1/5
```

	datevar	unemp	unempL1	unempL2
1.	1957q1	3. 933333	.	.
2.	1957q2	4. 1	3. 933333	.
3.	1957q3	4. 233333	4. 1	3. 933333
4.	1957q4	4. 933333	4. 233333	4. 1
5.	1958q1	6. 3	4. 933333	4. 233333

In a regression you could type:

```
regress y x L1.x L2.x
```

or

```
regress y x L(1/5).x
```

To generate forward or lead values use the “F” operator

```
generate unempF1=F1.unemp
generate unempF2=F2.unemp
list datevar unemp unempF1 unempF2 in 1/5
```

- . generate unempF1=F1.unemp  
(1 missing value generated)
- . generate unempF2=F2.unemp  
(2 missing values generated)
- . list datevar unemp unempF1 unempF2 in 1/5

	<b>datevar</b>	<b>unemp</b>	<b>unempF1</b>	<b>unempF2</b>
1.	<b>1957q1</b>	<b>3. 933333</b>	<b>4. 1</b>	<b>4. 233333</b>
2.	<b>1957q2</b>	<b>4. 1</b>	<b>4. 233333</b>	<b>4. 933333</b>
3.	<b>1957q3</b>	<b>4. 233333</b>	<b>4. 933333</b>	<b>6. 3</b>
4.	<b>1957q4</b>	<b>4. 933333</b>	<b>6. 3</b>	<b>7. 366667</b>
5.	<b>1958q1</b>	<b>6. 3</b>	<b>7. 366667</b>	<b>7. 333333</b>

In a regression you could type:

```
regress y x F1.x F2.x
or
regress y x F(1/5).x
```

## Lag operators (difference)

To generate the difference between current and previous values use the “D” operator

```
generate unempD1=D1.unemp /* D1 =  $y_t - y_{t-1}$  */  
generate unempD2=D2.unemp /* D2 =  $(y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$  */  
list datevar unemp unempD1 unempD2 in 1/5
```

```
. generate unempD1=D1.unemp  
(1 missing value generated)
```

```
. generate unempD2=D2.unemp  
(2 missing values generated)
```

```
. list datevar unemp unempD1 unempD2 in 1/5
```

	<b>datevar</b>	<b>unemp</b>	<b>unempD1</b>	<b>unempD2</b>
1.	1957q1	3.933333	.	.
2.	1957q2	4.1	.1666665	.
3.	1957q3	4.233333	.1333332	-.0333333
4.	1957q4	4.933333	.7000003	.5666671
5.	1958q1	6.3	1.3666667	.6666665

In a regression you could type:

```
regress y x D1.x
```

To generate seasonal differences use the “S” operator

```
generate unempS1=S1.unemp /* S1 =  $y_t - y_{t-1}$  */
generate unempS2=S2.unemp /* S2 =  $(y_t - y_{t-2})$  */
list datevar unemp unempS1 unempS2 in 1/5
```

- . generate unempS1=S1.unemp  
(1 missing value generated)
- . generate unempS2=S2.unemp  
(2 missing values generated)
- . list datevar unemp unempS1 unempS2 in 1/5

	<b>datevar</b>	<b>unemp</b>	<b>unempS1</b>	<b>unempS2</b>
1.	1957q1	3.933333	.	.
2.	1957q2	4.1	.1666665	.
3.	1957q3	4.233333	.1333332	.2999997
4.	1957q4	4.933333	.7000003	.8333335
5.	1958q1	6.3	1.366667	2.066667

In a regression you could type:

```
regress y x S1.x
```

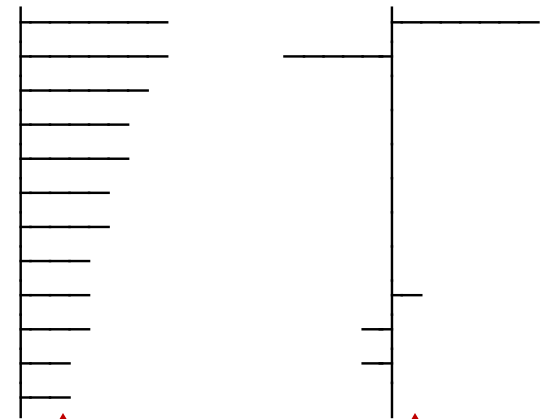
# Correlograms: autocorrelation

To explore autocorrelation, which is the correlation between a variable and its previous values, use the command `corrgram`. The number of lags depend on theory, AIC/BIC process or experience. The output includes autocorrelation coefficient and partial correlations coefficients used to specify an ARIMA model.

```
corrgram unemp, lags(12)
```

```
. corrgram unemp, lags(12)
```

LAG	AC	PAC	Q	Prob>Q	-1	0	1	-1	0	1
					[Autocorrelation]			[Partial Autocor]		
1	0.9641	0.9650	182.2	0.0000						
2	0.8921	-0.6305	339.02	0.0000						
3	0.8045	0.1091	467.21	0.0000						
4	0.7184	0.0424	569.99	0.0000						
5	0.6473	0.0836	653.86	0.0000						
6	0.5892	-0.0989	723.72	0.0000						
7	0.5356	-0.0384	781.77	0.0000						
8	0.4827	0.0744	829.17	0.0000						
9	0.4385	0.1879	868.5	0.0000						
10	0.3984	-0.1832	901.14	0.0000						
11	0.3594	-0.1396	927.85	0.0000						
12	0.3219	0.0745	949.4	0.0000						



AC shows that the correlation between the current value of `unemp` and its value three quarters ago is 0.8045. AC can be used to define the  $q$  in  $MA(q)$  only in stationary series

PAC shows that the correlation between the current value of `unemp` and its value three quarters ago is 0.1091 without the effect of the two previous lags. PAC can be used to define the  $p$  in  $AR(p)$  only in stationary series

Box-Pierce' Q statistic tests the null hypothesis that all correlation up to lag  $k$  are equal to 0. This series shows significant autocorrelation as shown in the Prob>Q value which at any  $k$  are less than 0.05, therefore rejecting the null that all lags are not autocorrelated.

Graphic view of AC which shows a slow decay in the trend, suggesting non-stationarity. See also the `ac` command.

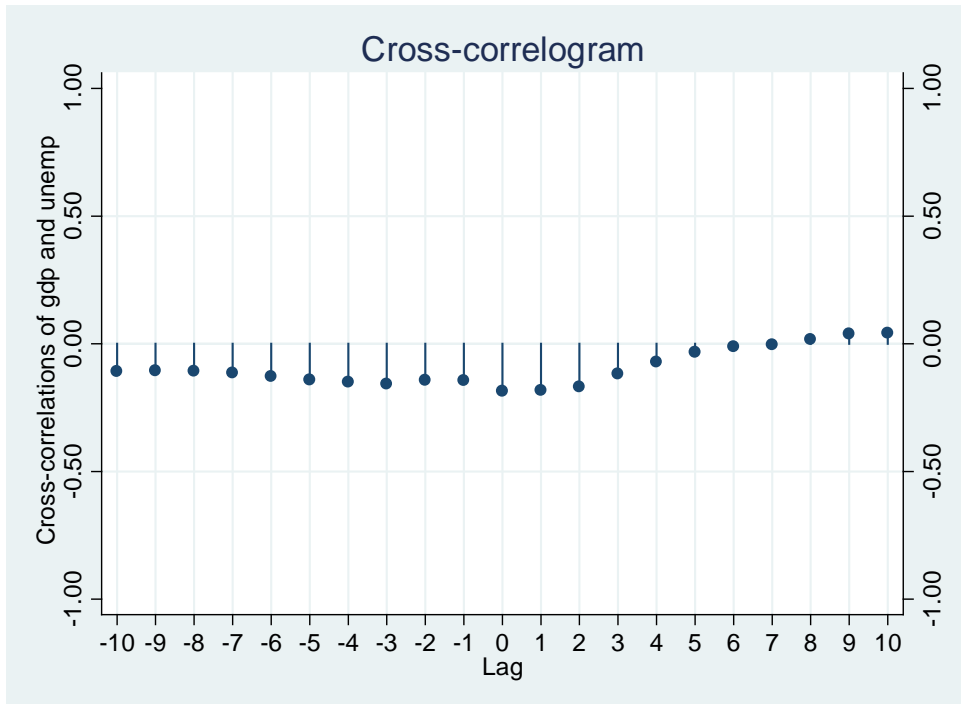
Graphic view of PAC which does not show spikes after the second lag which suggests that all other lags are mirrors of the second lag. See the `pac` command.



# Correlograms: cross correlation

To explore the relationship between two time series use the command `xcorr`. The graph below shows the correlation between GDP quarterly growth rate and unemployment. When using `xcorr` list the independent variable first and the dependent variable second. type

```
xcorr gdp unemp, lags(10) xlabel(-10(1)10,grid)
```



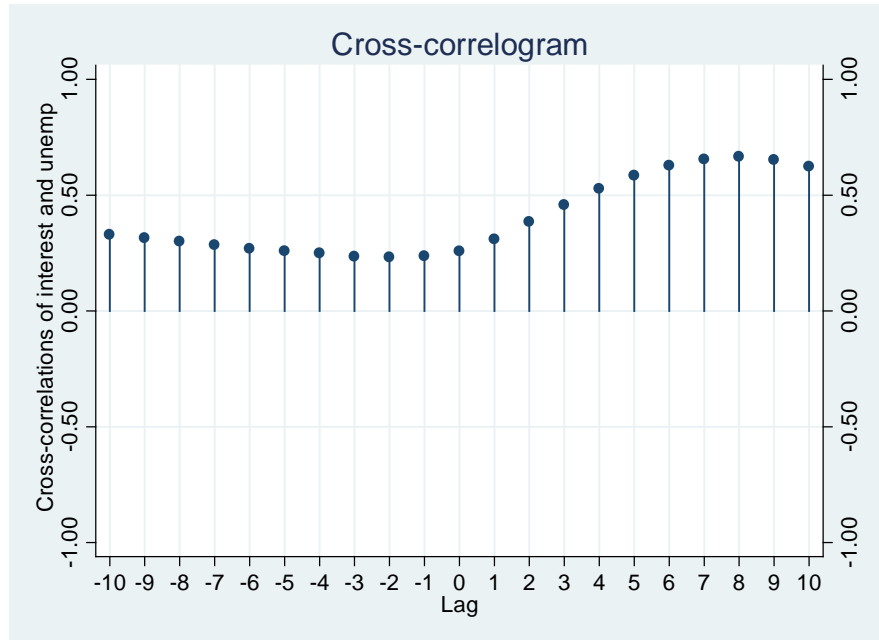
```
. xcorr gdp unemp, lags(10) table
```

LAG	CORR	-1	0	1
[Cross-correlation]				
-10	-0.1080			
-9	-0.1052			
-8	-0.1075			
-7	-0.1144			
-6	-0.1283			
-5	-0.1412			
-4	-0.1501			
-3	-0.1578			
-2	-0.1425			
-1	-0.1437			
0	-0.1853			
1	-0.1828			
2	-0.1685			
3	-0.1177			
4	-0.0716			
5	-0.0325			
6	-0.0111			
7	-0.0038			
8	0.0168			
9	0.0393			
10	0.0419			

At lag 0 there is a negative immediate correlation between GDP growth rate and unemployment. This means that a drop in GDP causes an immediate increase in unemployment.

# Correlograms: cross correlation

```
xcorr interest unemp, lags(10) xlabel(-10(1)10,grid)
```



Interest rates have a positive effect on future level of unemployment, reaching the highest point at lag 8 (four quarters or two years). In this case, interest rates are positive correlated with unemployment rates eight quarters later.

```
. xcorr interest unemp, lags(10) table
```

LAG	CORR	-1	0	1
		[Cross-correlation]		
-10	0.3297			
-9	0.3150			
-8	0.2997			
-7	0.2846			
-6	0.2685			
-5	0.2585			
-4	0.2496			
-3	0.2349			
-2	0.2323			
-1	0.2373			
0	0.2575			
1	0.3095			
2	0.3845			
3	0.4576			
4	0.5273			
5	0.5850			
6	0.6278			
7	0.6548			
8	0.6663			
9	0.6522			
10	0.6237			

Too many lags could increase the error in the forecasts, too few could leave out relevant information\*. Experience, knowledge and theory are usually the best way to determine the number of lags needed. There are, however, information criterion procedures to help come up with a proper number. Three commonly used are: Schwarz's Bayesian information criterion (SBIC), the Akaike's information criterion (AIC), and the Hannan and Quinn information criterion (HQIC). All these are reported by the command 'varsoc' in Stata.

```
. varsoc gdp cpi, maxlag(10)
```

Selecti on-order criteria  
Sample: 1959q4 - 2005q1

Number of obs = 182

lag	LL	LR	df	p	FPE	AIC	HQIC	SBIC
0	-1294.75				5293.32	14.25	14.2642	14.2852
1	-467.289	1654.9	4	0.000	.622031	5.20098	5.2438	5.30661
2	-401.381	131.82	4	0.000	.315041	4.52067	4.59204	4.69672*
3	-396.232	10.299	4	0.036	.311102	4.50804	4.60796	4.75451
4	-385.514	21.435*	4	0.000	.288988*	4.43422*	4.56268*	4.7511
5	-383.92	3.1886	4	0.527	.296769	4.46066	4.61766	4.84796
6	-381.135	5.5701	4	0.234	.300816	4.47401	4.65956	4.93173
7	-379.062	4.1456	4	0.387	.307335	4.49519	4.70929	5.02332
8	-375.483	7.1585	4	0.128	.308865	4.49981	4.74246	5.09836
9	-370.817	9.3311	4	0.053	.306748	4.4925	4.76369	5.16147
10	-370.585	.46392	4	0.977	.319888	4.53391	4.83364	5.27329

Endogenous: gdp cpi  
Exogenous: \_cons

When all three agree, the selection is clear, but what happens when getting conflicting results? A paper from the CEPR suggests, in the context of VAR models, that AIC tends to be more accurate with monthly data, HQIC works better for quarterly data on samples over 120 and SBIC works fine with any sample size for quarterly data (on VEC models)\*\*. In our example above we have quarterly data with 182 observations, HQIC suggest a lag of 4 (which is also suggested by AIC).

\* See Stock & Watson for more details and on how to estimate BIC and SIC

\*\* Ivanov, V. and Kilian, L. 2001. 'A Practitioner's Guide to Lag-Order Selection for Vector Autoregressions'. CEPR Discussion Paper no. 2685. London, Centre for Economic Policy Research. <http://www.cepr.org/pubs/dps/DP2685.asp>.

Having a unit root in a series mean that there is more than one trend in the series.

```
. regress unemp gdp if tin(1965q1, 1981q4)
```

Source	SS	df	MS			
Model	36.1635247	1	36.1635247	Number of obs =	68	
Residual	124.728158	66	1.88982058	F( 1, 66) =	19.14	
Total	160.891683	67	2.4013684	Prob > F =	0.0000	
				R-squared =	0.2248	
				Adj R-squared =	0.2130	
				Root MSE =	1.3747	

unemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
gdp	-.4435909	.1014046	-4.37	0.000	-.6460517	-.2411302
_cons	7.087789	.3672397	19.30	0.000	6.354572	7.821007

```
. regress unemp gdp if tin(1982q1, 2000q4)
```

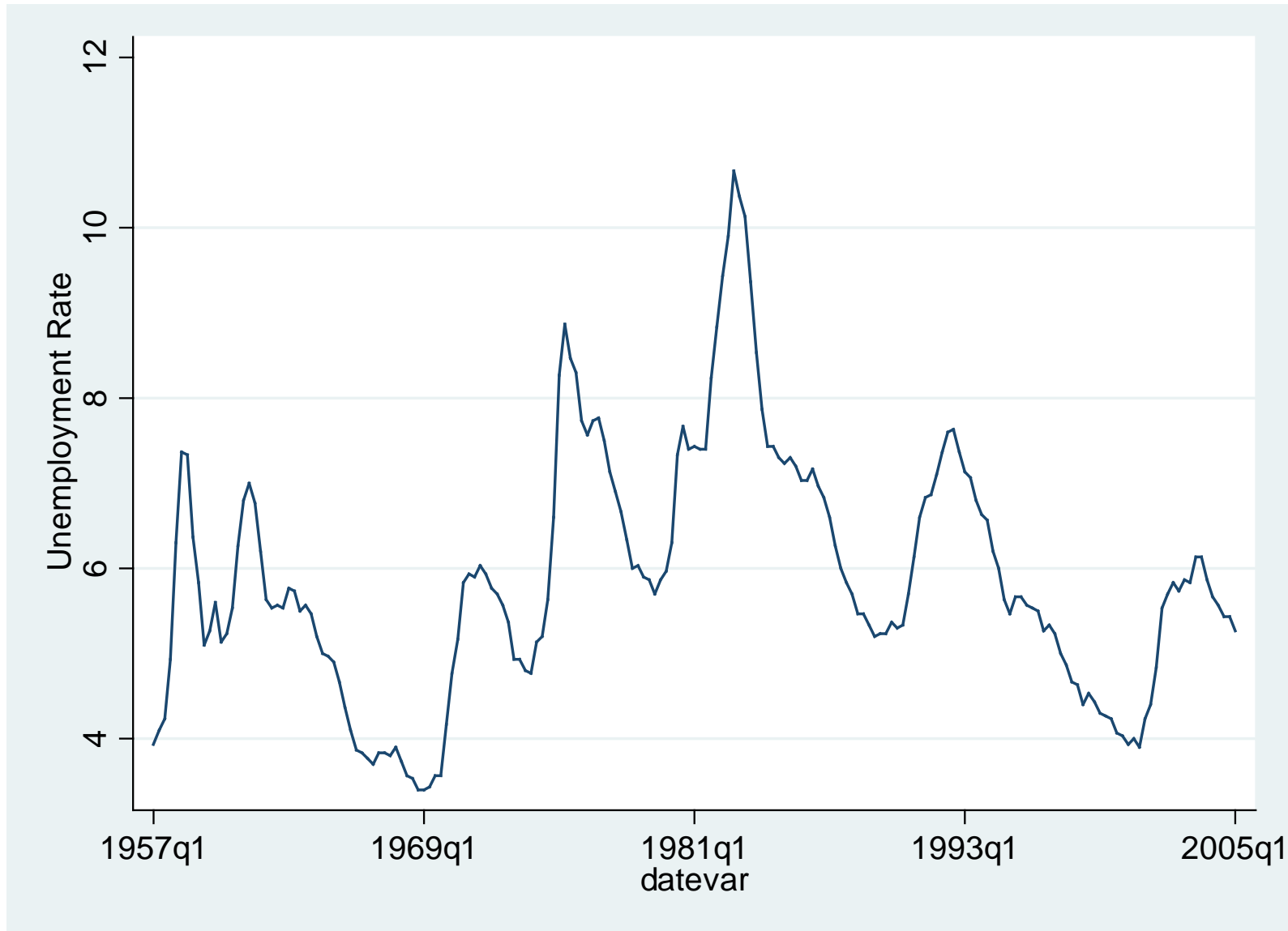
Source	SS	df	MS			
Model	8.83437339	1	8.83437339	Number of obs =	76	
Residual	180.395848	74	2.43778172	F( 1, 74) =	3.62	
Total	189.230221	75	2.52306961	Prob > F =	0.0608	
				R-squared =	0.0467	
				Adj R-squared =	0.0338	
				Root MSE =	1.5613	

unemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
gdp	.3306551	.173694	1.90	0.061	-.0154377	.6767479
_cons	5.997169	.2363599	25.37	0.000	5.526211	6.468126

Unemployment rate.

line unemp datevar



The Dickey-Fuller test is one of the most commonly use tests for stationarity. The null hypothesis is that the series has a unit root. The test statistic shows that the unemployment series have a unit root, it lies within the acceptance region.

One way to deal with stochastic trends (unit root) is by taking the first difference of the variable (second test below).

```
. dfuller unemp, lag(5)
```

Augmented Dickey-Fuller test for unit root                      Number of obs =                      187

**Unit root**

	Test Statistic	1% Critical Value	5% Critical Value	10% Critical Value
Z(t)	-2.597	-3.481	-2.884	-2.574

MacKinnon approximate p-value for Z(t) = 0.0936

```
. dfuller unempD1, lag(5)
```

Augmented Dickey-Fuller test for unit root                      Number of obs =                      186

**No unit root**

	Test Statistic	1% Critical Value	5% Critical Value	10% Critical Value
Z(t)	-5.303	-3.481	-2.884	-2.574

MacKinnon approximate p-value for Z(t) = 0.0000

# Testing for cointegration

Cointegration refers to the fact that two or more series share an stochastic trend (Stock & Watson). Engle and Granger (1987) suggested a two step process to test for cointegration (an OLS regression and a unit root test), the EG-ADF test.

```
regress unemp gdp
```

Run an OLS regression

```
predict e, resid
```

Get the residuals

```
dfuller e, lags(10)
```

Run a unit root test on the residuals.

Augmented Dickey-Fuller test for unit root

Number of obs = 181

Unit root*	Test Statistic	Interpolated Dickey-Fuller		
		1% Critical Value	5% Critical Value	10% Critical Value
Z(t)	-2.535	-3.483	-2.885	-2.575

MacKinnon approximate p-value for Z(t) = 0.1071

Both variables are not cointegrated

See Stock & Watson for a table of critical values for the unit root test and the theory behind.

# Granger causality: using OLS

If you regress 'y' on lagged values of 'y' and 'x' and the coefficients of the lag of 'x' are statistically significantly different from 0, then you can argue that 'x' Granger-cause 'y', this is, 'x' can be used to predict 'y' (see Stock & Watson -2007-, Green -2008).

1 . regress unemp L(1/4).unemp L(1/4).gdp

Source	SS	df	MS			
Model	373.501653	8	46.6877066	Number of obs =	188	
Residual	12.5037411	179	.069853302	F( 8, 179) =	668.37	
Total	386.005394	187	2.06419997	Prob > F =	0.0000	
				R-squared =	0.9676	
				Adj R-squared =	0.9662	
				Root MSE =	.2643	

unemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
unemp						
L1.	1.625708	.0763035	21.31	0.000	1.475138	1.776279
L2.	-.7695503	.1445769	-5.32	0.000	-1.054845	-.484256
L3.	.0868131	.1417562	0.61	0.541	-.1929152	.3665415
L4.	.0217041	.0726137	0.30	0.765	-.1215849	.1649931
gdp						
L1.	.0060996	.0136043	0.45	0.654	-.0207458	.0329451
L2.	-.0189398	.0128618	-1.47	0.143	-.0443201	.0064405
L3.	.0247494	.0130617	1.89	0.060	-.0010253	.0505241
L4.	.003637	.0129079	0.28	0.778	-.0218343	.0291083
_cons	.1702419	.096857	1.76	0.081	-.0208865	.3613704

2 . test L1.gdp L2.gdp L3.gdp L4.gdp

- ( 1) L.gdp = 0
- ( 2) L2.gdp = 0
- ( 3) L3.gdp = 0
- ( 4) L4.gdp = 0

F( 4, 179) = 1.67  
 Prob > F = 0.1601

You cannot reject the null hypothesis that all coefficients of lag of 'x' are equal to 0. Therefore 'gdp' does not Granger-cause 'unemp'.



# Granger causality: using VAR

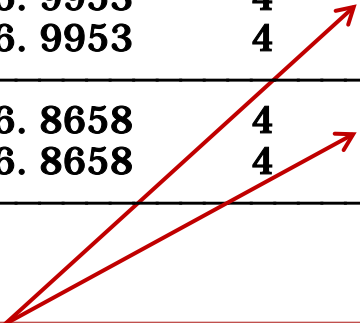
The following procedure uses VAR models to estimate Granger causality using the command 'vargranger'

1 . quietly var unemp gdp, lags(1/4)

2 . vargranger

## Granger causality Wald tests

Equation	Excluded	chi 2	df	Prob > chi 2
unemp	gdp	6.9953	4	0.136
unemp	ALL	6.9953	4	0.136
gdp	unemp	6.8658	4	0.143
gdp	ALL	6.8658	4	0.143



The null hypothesis is 'var1 does not Granger-cause var2'. In both cases, we cannot reject the null that each variable does not Granger-cause the other

# Chow test (testing for known breaks)

The Chow test allows to test whether a particular date causes a break in the regression coefficients. It is named after Gregory Chow (1960)\*.

**Step 1.** Create a dummy variable where 1 if date > break date and 0 <= break date. Below we'll test whether the first quarter of 1982 causes a break in the regression coefficients.

```
tset datevar  
gen break = (datevar>tq(1981q4))
```

Change "tq" with the correct date format: tw (week), tm (monthly), tq (quarterly), th (half), ty (yearly) and the corresponding date format in the parenthesis

**Step 2.** Create interaction terms between the lags of the independent variables and the lag of the dependent variables. We will assume lag 1 for this example (the number of lags depends on your theory/data)

```
generate break_unemp = break*l1.unemp  
generate break_gdp = break*l1.gdp
```

**Step 3.** Run a regression between the outcome variables (in this case 'unemp') and the independent along with the interactions and the dummy for the break.

```
reg unemp l1.unemp l1.gdp break break_unemp break_gdp
```

**Step 4.** Run an F-test on the coefficients for the interactions and the dummy for the break

```
test break break_unemp break_gdp
```

```
. test break break_unemp break_gdp
```

```
( 1) break = 0  
( 2) break_unemp = 0  
( 3) break_gdp = 0
```

```
F( 3, 185) = 1.14  
Prob > F = 0.3351
```

The null hypothesis is no break. If the p-value is < 0.05 reject the null in favor of the alternative that there is a break. In this example, we fail to reject the null and conclude that the first quarter of 1982 does not cause a break in the regression coefficients.

\* See Stock & Watson for more details

# Testing for unknown breaks

The Quandt likelihood ratio (QLR test –Quandt,1960) or sup-Wald statistic is a modified version of the Chow test used to identify break dates. The following is a modified procedure taken from Stock & Watson's companion materials to their book *Introduction to Econometrics*, I strongly advise to read the corresponding chapter to better understand the procedure and to check the critical values for the QLR statistic. Below we will check for breaks in a GDP per-capita series (quarterly).

```
/* Replace the words in bold with your own variables, do not change anything else*/
/* The log file 'qlrtest.log' will have the list for QLR statistics (use Word to read it)*/
/* See next page for a graph*/
/* STEP 1. Copy-and-paste-run the code below to a do-file, double-check the quotes (re-type them if necessary)*/

log using qlrtest.log
tset datevar
sum datevar
local time=r(max)-r(min)+1
local i = round(`time'*.15)
local f = round(`time'*.85)
local var = "gdp"
gen diff`var' = d.`var'
gen chow`var' = .
gen qlr`var' = .
set more off
while `i'<=(`f') {
    gen di = (_n > `i')
    cap gen d_`var'1 = di*11.`var'
    cap gen d_`var'2 = di*12.`var'
    cap gen d_`var'3 = di*13.`var'
    cap gen d_`var'4 = di*14.`var'
    qui reg diff`var' L(1/4).diff`var' di, r
    qui test di
    sca chow = r(F)
    cap replace chow`var' = r(F) in `i'
    qui reg diff`var' L(1/4).diff`var' di d_`var'1 d_`var'2 d_`var'3 d_`var'4, r
    qui test di d_`var'1 d_`var'2 d_`var'3 d_`var'4
    sca qlr = r(F)
    cap replace qlr`var' = r(F) in `i'
    dis "`i' " %tq datevar["`i'"] " " %8.3f chow " " %8.3f qlr
    drop di d_`var'1 d_`var'2 d_`var'3 d_`var'4
    local i = `i' + 1
}
```

**NOTE:** change %tq if your date variable is other than quarterly. Use %td for daily, %tm for monthly, %ty for yearly

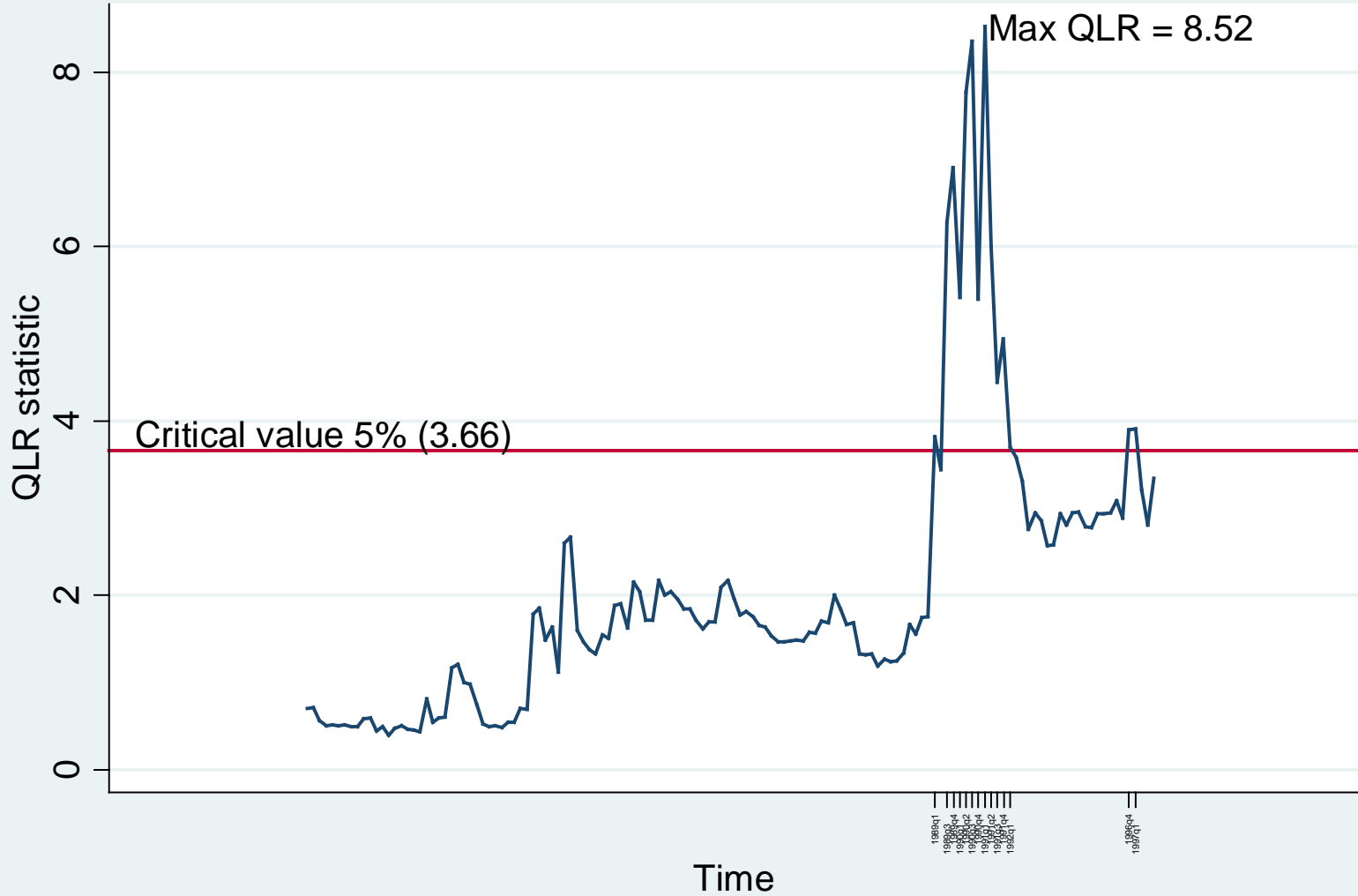
# Testing for unknown breaks: graph

```
/* Replace the words in bold with your own variables, do not change anything else*/
/* The code will produce the graph shown in the next page*/
/* The critical value 3.66 is for q=5 (constant and four lags) and 5% significance*/
/* STEP 2. Copy-and-paste-run the code below to a do-file, double-check the quotes (re-type them if necessary)*/

sum qlr`var'
local maxvalue=r(max)
gen maxdate=datevar if qlr`var'==`maxvalue'
local maxvalue1=round(`maxvalue',0.01)
local critical=3.66 /*Replace with the appropriate critical value (see Stock & Watson)*/
sum datevar
local mindate=r(min)
sum maxdate
local maxdate=r(max)
gen break=datevar if qlr`var'>=`critical' & qlr`var'!=.
dis "Below are the break dates..."
list datevar qlr`var' if break!=.
levelsof break, local(break1)
tway tsline qlr`var', title(Testing for breaks in GDP per-capita (1957-2005)) ///
    xlabel(`break1', angle(90) labsize(0.9) alternate) ///
    yline(`critical') ytitle(QLR statistic) xtitle(Time) ///
    ttext(`critical' `mindate' "Critical value 5% (`critical')", placement(ne)) ///
    ttext(`maxvalue' `maxdate' "Max QLR = `maxvalue1'", placement(e))
```

	<b>datevar</b>	<b>ql rgdp</b>
<b>129.</b>	<b>1989q1</b>	<b>3. 823702</b>
<b>131.</b>	<b>1989q3</b>	<b>6. 285852</b>
<b>132.</b>	<b>1989q4</b>	<b>6. 902882</b>
<b>133.</b>	<b>1990q1</b>	<b>5. 416068</b>
<b>134.</b>	<b>1990q2</b>	<b>7. 769114</b>
<b>135.</b>	<b>1990q3</b>	<b>8. 354294</b>
<b>136.</b>	<b>1990q4</b>	<b>5. 399252</b>
<b>137.</b>	<b>1991q1</b>	<b>8. 524492</b>
<b>138.</b>	<b>1991q2</b>	<b>6. 007093</b>
<b>139.</b>	<b>1991q3</b>	<b>4. 44151</b>
<b>140.</b>	<b>1991q4</b>	<b>4. 946689</b>
<b>141.</b>	<b>1992q1</b>	<b>3. 699911</b>
<b>160.</b>	<b>1996q4</b>	<b>3. 899656</b>
<b>161.</b>	<b>1997q1</b>	<b>3. 906271</b>

### Testing for breaks in GDP per-capita (1957-2005)



White noise refers to the fact that a variable does not have autocorrelation. In Stata use the `wntestq` (white noise Q test) to check for autocorrelation. The null is that there is no serial correlation (type `help wntestq` for more details):

```
. wntestq unemp
```

```
Portmanteau test for white noise
```

```
-----  
Portmanteau (Q) statistic = 1044.6341  
Prob > chi2(40)          = 0.0000
```

```
. wntestq unemp, lags(10)
```

```
Portmanteau test for white noise
```

```
-----  
Portmanteau (Q) statistic = 901.1399  
Prob > chi2(10)          = 0.0000
```

Serial correlation



If your variable is not white noise then see the page on correlograms to see the order of the autocorrelation.

# Time Series: Testing for serial correlation

Breusch-Godfrey and Durbin-Watson are used to test for serial correlation. The null in both tests is that there is no serial correlation (type `help estat dwatson`, `help estat dubinalt` and `help estat bgodfrey` for more details).

```
. regress unempd gdp
```

Source	SS	df	MS			
Model	.205043471	1	.205043471	Number of obs =	192	
Residual	24.0656991	190	.126661574	F( 1, 190) =	1.62	
Total	24.2707425	191	.12707195	Prob > F =	0.2048	
				R-squared =	0.0084	
				Adj R-squared =	0.0032	
				Root MSE =	.3559	

unempd1	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
gdp	-.015947	.0125337	-1.27	0.205	-.0406701	.0087761
_cons	.0401123	.036596	1.10	0.274	-.0320743	.1122989

```
. estat dwatson
```

Durbin-Watson d-statistic( 2, 192) = .7562744

```
. estat durbinalt
```

Durbin's alternative test for autocorrelation

lags(p)	chi 2	df	Prob > chi 2
1	118.790	1	0.0000

H0: no serial correlation

```
. estat bgodfrey
```

Breusch-Godfrey LM test for autocorrelation

lags(p)	chi 2	df	Prob > chi 2
1	74.102	1	0.0000

H0: no serial correlation

Serial correlation

# Time Series: Correcting for serial correlation

Run a Cochrane-Orcutt regression using the `prais` command (type `help prais` for more details)

```
. prais unemp gdp, corc
```

```
Iteration 0: rho = 0.0000
Iteration 1: rho = 0.9556
Iteration 2: rho = 0.9660
Iteration 3: rho = 0.9661
Iteration 4: rho = 0.9661
```

Cochrane-Orcutt AR(1) regression -- iterated estimates

Source	SS	df	MS	Number of obs =	191
Model	.308369041	1	.308369041	F( 1, 189) =	2.48
Residual	23.4694088	189	.124176766	Prob > F =	0.1167
Total	23.7777778	190	.125146199	R-squared =	0.0130
				Adj R-squared =	0.0077
				Root MSE =	.35239

unemp	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
gdp	-.020264	.0128591	-1.58	0.117	-.0456298 .0051018
_cons	6.105931	.7526023	8.11	0.000	4.621351 7.590511
rho	.966115				

```
Durbin-Watson statistic (original) 0.087210
Durbin-Watson statistic (transformed) 0.758116
```



## Useful links / Recommended books

- DSS Online Training Section <http://dss.princeton.edu/training/>
- UCLA Resources to learn and use STATA <http://www.ats.ucla.edu/stat/stata/>
- DSS help-sheets for STATA [http://dss/online\\_help/stats\\_packages/stata/stata.htm](http://dss/online_help/stats_packages/stata/stata.htm)
- *Introduction to Stata* (PDF), Christopher F. Baum, Boston College, USA. “A 67-page description of Stata, its key features and benefits, and other useful information.” <http://fmwww.bc.edu/GStat/docs/StataIntro.pdf>
- STATA FAQ website <http://stata.com/support/faqs/>
- Princeton DSS Libguides <http://libguides.princeton.edu/dss>

### Books

- *Introduction to econometrics* / James H. Stock, Mark W. Watson. 2nd ed., Boston: Pearson Addison Wesley, 2007.
- *Data analysis using regression and multilevel/hierarchical models* / Andrew Gelman, Jennifer Hill. Cambridge ; New York : Cambridge University Press, 2007.
- *Econometric analysis* / William H. Greene. 6th ed., Upper Saddle River, N.J. : Prentice Hall, 2008.
- *Designing Social Inquiry: Scientific Inference in Qualitative Research* / Gary King, Robert O. Keohane, Sidney Verba, Princeton University Press, 1994.
- *Unifying Political Methodology: The Likelihood Theory of Statistical Inference* / Gary King, Cambridge University Press, 1989
- *Statistical Analysis: an interdisciplinary introduction to univariate & multivariate methods* / Sam Kachigan, New York : Radius Press, c1986
- *Statistics with Stata (updated for version 9)* / Lawrence Hamilton, Thomson Books/Cole, 2006