Title

stata.com

arch — Autoregressive conditional heteroskedasticity (ARCH) family of estimators

Description

`arch` fits regression models in which the volatility of a series varies through time. Usually, periods of high and low volatility are grouped together. ARCH models estimate future volatility as a function of prior volatility. To accomplish this, `arch` fits models of autoregressive conditional heteroskedasticity (ARCH) by using conditional maximum likelihood. In addition to ARCH terms, models may include multiplicative heteroskedasticity. Gaussian (normal), Student's t, and generalized error distributions are supported.

Concerning the regression equation itself, models may also contain ARCH-in-mean and ARMA terms.

Quick start

ARCH model of `y` with first- and second-order ARCH components and regressor `x` using `tsset` data

```
arch y x, arch(1,2)
```

Add a second-order GARCH component

```
arch y x, arch(1,2) garch(2)
```

Add an autoregressive component of order 2 and a moving-average component of order 3

```
arch y x, arch(1,2) garch(2) ar(2) ma(3)
```

As above, but with the conditional variance included in the mean equation

```
arch y x, arch(1,2) garch(2) ar(2) ma(3) archm
```

EGARCH model of order 2 for `y` with an autoregressive component of order 1

```
arch y, earch(2) egarch(2) ar(1)
```

Menu

ARCH/GARCH

Statistics > Time series > ARCH/GARCH > ARCH and GARCH models

EARCH/EGARCH

Statistics > Time series > ARCH/GARCH > Nelson's EGARCH model

ABARCH/ATARCH/SDGARCH

Statistics > Time series > ARCH/GARCH > Threshold ARCH model

ARCH/TARCH/GARCH

Statistics > Time series > ARCH/GARCH > GJR form of threshold ARCH model

ARCH/SAARCH/GARCH

Statistics > Time series > ARCH/GARCH > Simple asymmetric ARCH model

PARCH/PGARCH

Statistics > Time series > ARCH/GARCH > Power ARCH model

NARCH/GARCH

Statistics > Time series > ARCH/GARCH > Nonlinear ARCH model

NARCHK/GARCH

Statistics > Time series > ARCH/GARCH > Nonlinear ARCH model with one shift

APARCH/PGARCH

Statistics > Time series > ARCH/GARCH > Asymmetric power ARCH model

NPARCH/PGARCH

Statistics > Time series > ARCH/GARCH > Nonlinear power ARCH model

Syntax

```
arch depvar [indepvars] [if] [in] [weight] [, options]
```

options	Description
Model	
noconstant	suppress constant term
arch(*numlist*)	ARCH terms
garch(*numlist*)	GARCH terms
saarch(*numlist*)	simple asymmetric ARCH terms
tarch(*numlist*)	threshold ARCH terms
aarch(*numlist*)	asymmetric ARCH terms
narch(*numlist*)	nonlinear ARCH terms
narchk(*numlist*)	nonlinear ARCH terms with single shift
abarch(*numlist*)	absolute value ARCH terms
atarch(*numlist*)	absolute threshold ARCH terms
sdgarch(*numlist*)	lags of σ_t
earch(*numlist*)	news terms in Nelson's (1991) EGARCH model
egarch(*numlist*)	lags of $\ln(\sigma_t^2)$
parch(*numlist*)	power ARCH terms
tparch(*numlist*)	threshold power ARCH terms
aparch(*numlist*)	asymmetric power ARCH terms
nparch(*numlist*)	nonlinear power ARCH terms
nparchk(*numlist*)	nonlinear power ARCH terms with single shift
pgarch(*numlist*)	power GARCH terms
constraints(*constraints*)	apply specified linear constraints
Model 2	
archm	include ARCH-in-mean term in the mean-equation specification
archmlags(*numlist*)	include specified lags of conditional variance in mean equation
archmexp(*exp*)	apply transformation in *exp* to any ARCH-in-mean terms
arima(#$_p$,#$_d$,#$_q$)	specify ARIMA(p, d, q) model for dependent variable
ar(*numlist*)	autoregressive terms of the structural model disturbance
ma(*numlist*)	moving-average terms of the structural model disturbances
Model 3	
distribution(*dist* [#])	use *dist* distribution for errors (may be gaussian, normal, t, or ged; default is gaussian)
het(*varlist*)	include *varlist* in the specification of the conditional variance
savespace	conserve memory during estimation

Priming	
`arch0(xb)`	compute priming values on the basis of the expected unconditional variance; the default
`arch0(xb0)`	compute priming values on the basis of the estimated variance of the residuals from OLS
`arch0(xbwt)`	compute priming values on the basis of the weighted sum of squares from OLS residuals
`arch0(xb0wt)`	compute priming values on the basis of the weighted sum of squares from OLS residuals, with more weight at earlier times
`arch0(zero)`	set priming values of ARCH terms to zero
`arch0(#)`	set priming values of ARCH terms to #
`arma0(zero)`	set all priming values of ARMA terms to zero; the default
`arma0(p)`	begin estimation after observation p, where p is the maximum AR lag in model
`arma0(q)`	begin estimation after observation q, where q is the maximum MA lag in model
`arma0(pq)`	begin estimation after observation $(p+q)$
`arma0(#)`	set priming values of ARMA terms to #
`condobs(#)`	set conditioning observations at the start of the sample to #
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `opg`, `robust`, or `oim`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`detail`	report list of gaps in time series
`nocnsreport`	do not display constraints
display_options	control columns and column formats, row spacing, and line width
Maximization	
maximize_options	control the maximization process; seldom used
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

You must `tsset` your data before using `arch`; see [TS] **tsset**.
depvar and *varlist* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.
`by`, `collect`, `fp`, `rolling`, `statsby`, and `xi` are allowed; see [U] **11.1.10 Prefix commands**.
`iweight`s are allowed; see [U] **11.1.6 weight**.
`collinear` and `coeflegend` do not appear in the dialog box.
See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

To fit an ARCH($\#_m$) model with Gaussian errors, type

```
. arch depvar ..., arch(1/#m)
```

To fit a GARCH($\#_m$,$\#_k$) model assuming that the errors follow Student's t distribution with 7 degrees of freedom, type

```
. arch depvar ..., arch(1/#m) garch(1/#k) distribution(t 7)
```

You can also fit many other models.

Details of syntax

The basic model `arch` fits is

$$\begin{aligned} y_t &= \mathbf{x}_t\boldsymbol{\beta} + \epsilon_t \\ \text{Var}(\epsilon_t) &= \sigma_t^2 = \gamma_0 + A(\boldsymbol{\sigma}, \boldsymbol{\epsilon}) + B(\boldsymbol{\sigma}, \boldsymbol{\epsilon})^2 \end{aligned} \tag{1}$$

The y_t equation may optionally include ARCH-in-mean and ARMA terms:

$$y_t = \mathbf{x}_t\boldsymbol{\beta} + \sum_i \psi_i g(\sigma_{t-i}^2) + \text{ARMA}(p, q) + \epsilon_t$$

If no options are specified, $A() = B() = 0$, and the model collapses to linear regression. The following options add to $A()$ (α, γ, and κ represent parameters to be estimated):

Option	Terms added to $A()$
`arch()`	$A() = A() + \alpha_{1,1}\epsilon_{t-1}^2 + \alpha_{1,2}\epsilon_{t-2}^2 + \cdots$
`garch()`	$A() = A() + \alpha_{2,1}\sigma_{t-1}^2 + \alpha_{2,2}\sigma_{t-2}^2 + \cdots$
`saarch()`	$A() = A() + \alpha_{3,1}\epsilon_{t-1} + \alpha_{3,2}\epsilon_{t-2} + \cdots$
`tarch()`	$A() = A() + \alpha_{4,1}\epsilon_{t-1}^2(\epsilon_{t-1} > 0) + \alpha_{4,2}\epsilon_{t-2}^2(\epsilon_{t-2} > 0) + \cdots$
`aarch()`	$A() = A() + \alpha_{5,1}(\lvert\epsilon_{t-1}\rvert + \gamma_{5,1}\epsilon_{t-1})^2 + \alpha_{5,2}(\lvert\epsilon_{t-2}\rvert + \gamma_{5,2}\epsilon_{t-2})^2 + \cdots$
`narch()`	$A() = A() + \alpha_{6,1}(\epsilon_{t-1} - \kappa_{6,1})^2 + \alpha_{6,2}(\epsilon_{t-2} - \kappa_{6,2})^2 + \cdots$
`narchk()`	$A() = A() + \alpha_{7,1}(\epsilon_{t-1} - \kappa_7)^2 + \alpha_{7,2}(\epsilon_{t-2} - \kappa_7)^2 + \cdots$

The following options add to $B()$:

Option	Terms added to $B()$
`abarch()`	$B() = B() + \alpha_{8,1}\lvert\epsilon_{t-1}\rvert + \alpha_{8,2}\lvert\epsilon_{t-2}\rvert + \cdots$
`atarch()`	$B() = B() + \alpha_{9,1}\lvert\epsilon_{t-1}\rvert(\epsilon_{t-1} > 0) + \alpha_{9,2}\lvert\epsilon_{t-2}\rvert(\epsilon_{t-2} > 0) + \cdots$
`sdgarch()`	$B() = B() + \alpha_{10,1}\sigma_{t-1} + \alpha_{10,2}\sigma_{t-2} + \cdots$

Each option requires a *numlist* argument (see [U] **11.1.8 numlist**), which determines the lagged terms included. `arch(1)` specifies $\alpha_{1,1}\epsilon_{t-1}^2$, `arch(2)` specifies $\alpha_{1,2}\epsilon_{t-2}^2$, `arch(1,2)` specifies $\alpha_{1,1}\epsilon_{t-1}^2 + \alpha_{1,2}\epsilon_{t-2}^2$, `arch(1/3)` specifies $\alpha_{1,1}\epsilon_{t-1}^2 + \alpha_{1,2}\epsilon_{t-2}^2 + \alpha_{1,3}\epsilon_{t-3}^2$, etc.

If the `earch()` or `egarch()` option is specified, the basic model fit is

$$\begin{aligned} y_t &= \mathbf{x}_t\boldsymbol{\beta} + \sum_i \psi_i g(\sigma_{t-i}^2) + \text{ARMA}(p, q) + \epsilon_t \\ \ln\text{Var}(\epsilon_t) &= \ln\sigma_t^2 = \gamma_0 + C(\ln\boldsymbol{\sigma}, \mathbf{z}) + A(\boldsymbol{\sigma}, \boldsymbol{\epsilon}) + B(\boldsymbol{\sigma}, \boldsymbol{\epsilon})^2 \end{aligned} \tag{2}$$

where $z_t = \epsilon_t/\sigma_t$. $A()$ and $B()$ are given as above, but $A()$ and $B()$ now add to $\ln\sigma_t^2$ rather than σ_t^2. (The options corresponding to $A()$ and $B()$ are rarely specified here.) $C()$ is given by

Option	Terms added to $C()$
`earch()`	$C() = C() + \alpha_{11,1} z_{t-1} + \gamma_{11,1}(\lvert z_{t-1}\rvert - \sqrt{2/\pi}) + \alpha_{11,2} z_{t-2} + \gamma_{11,2}(\lvert z_{t-2}\rvert - \sqrt{2/\pi}) + \cdots$
`egarch()`	$C() = C() + \alpha_{12,1} \ln\sigma^2_{t-1} + \alpha_{12,2} \ln\sigma^2_{t-2} + \cdots$

Instead, if the `parch()`, `tparch()`, `aparch()`, `nparch()`, `nparchk()`, or `pgarch()` option is specified, the basic model fit is

$$
\begin{aligned}
y_t &= \mathbf{x}_t\boldsymbol{\beta} + \sum_i \psi_i g(\sigma^2_{t-i}) + \text{ARMA}(p,q) + \epsilon_t \\
\{\text{Var}(\epsilon_t)\}^{\varphi/2} &= \sigma_t^{\varphi} = \gamma_0 + D(\boldsymbol{\sigma},\boldsymbol{\epsilon}) + A(\boldsymbol{\sigma},\boldsymbol{\epsilon}) + B(\boldsymbol{\sigma},\boldsymbol{\epsilon})^2
\end{aligned}
\tag{3}
$$

where φ is a parameter to be estimated. $A()$ and $B()$ are given as above, but $A()$ and $B()$ now add to σ_t^{φ}. (The options corresponding to $A()$ and $B()$ are rarely specified here.) $D()$ is given by

Option	Terms added to $D()$
`parch()`	$D() = D() + \alpha_{13,1}\epsilon^{\varphi}_{t-1} + \alpha_{13,2}\epsilon^{\varphi}_{t-2} + \cdots$
`tparch()`	$D() = D() + \alpha_{14,1}\epsilon^{\varphi}_{t-1}(\epsilon_{t-1} > 0) + \alpha_{14,2}\epsilon^{\varphi}_{t-2}(\epsilon_{t-2} > 0) + \cdots$
`aparch()`	$D() = D() + \alpha_{15,1}(\lvert\epsilon_{t-1}\rvert + \gamma_{15,1}\epsilon_{t-1})^{\varphi} + \alpha_{15,2}(\lvert\epsilon_{t-2}\rvert + \gamma_{15,2}\epsilon_{t-2})^{\varphi} + \cdots$
`nparch()`	$D() = D() + \alpha_{16,1}\lvert\epsilon_{t-1} - \kappa_{16,1}\rvert^{\varphi} + \alpha_{16,2}\lvert\epsilon_{t-2} - \kappa_{16,2}\rvert^{\varphi} + \cdots$
`nparchk()`	$D() = D() + \alpha_{17,1}\lvert\epsilon_{t-1} - \kappa_{17}\rvert^{\varphi} + \alpha_{17,2}\lvert\epsilon_{t-2} - \kappa_{17}\rvert^{\varphi} + \cdots$
`pgarch()`	$D() = D() + \alpha_{18,1}\sigma^{\varphi}_{t-1} + \alpha_{18,2}\sigma^{\varphi}_{t-2} + \cdots$

Common models

Common term	Options to specify
ARCH (Engle 1982)	`arch()`
GARCH (Bollerslev 1986)	`arch() garch()`
ARCH-in-mean (Engle, Lilien, and Robins 1987)	`archm arch()` [`garch()`]
GARCH with ARMA terms	`arch() garch() ar() ma()`
EGARCH (Nelson 1991)	`earch() egarch()`
TARCH, threshold ARCH (Zakoian 1994)	`abarch() atarch() sdgarch()`
GJR, form of threshold ARCH (Glosten, Jagannathan, and Runkle 1993)	`arch() tarch()` [`garch()`]
SAARCH, simple asymmetric ARCH (Engle 1990)	`arch() saarch()` [`garch()`]
PARCH, power ARCH (Higgins and Bera 1992)	`parch()` [`pgarch()`]
NARCH, nonlinear ARCH	`narch()` [`garch()`]
NARCHK, nonlinear ARCH with one shift	`narchk()` [`garch()`]
A-PARCH, asymmetric power ARCH (Ding, Granger, and Engle 1993)	`aparch()` [`pgarch()`]
NPARCH, nonlinear power ARCH	`nparch()` [`pgarch()`]

In all cases, you type

arch *depvar* [*indepvars*], *options*

where *options* are chosen from the table above. Each option requires that you specify as its argument a *numlist* that specifies the lags to be included. For most ARCH models, that value will be 1. For instance, to fit the classic first-order GARCH model on cpi, you would type

```
. arch cpi, arch(1) garch(1)
```

If you wanted to fit a first-order GARCH model of cpi on wage, you would type

```
. arch cpi wage, arch(1) garch(1)
```

If, for any of the options, you want first- and second-order terms, specify *optionname*(1/2). Specifying garch(1) arch(1/2) would fit a GARCH model with first- and second-order ARCH terms. If you specified arch(2), only the lag 2 term would be included.

Reading arch output

The regression table reported by `arch` when using the normal distribution for the errors will appear as

op.depvar	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]
depvar					
x1	# ...				
x2					
L1.	# ...				
L2.	# ...				
_cons	# ...				
ARCHM					
sigma2	# ...				
ARMA					
ar					
L1.	# ...				
ma					
L1.	# ...				
HET					
z1	# ...				
z2					
L1.	# ...				
L2.	# ...				
ARCH					
arch					
L1.	# ...				
garch					
L1.	# ...				
aparch					
L1.	# ...				
etc.					
_cons	# ...				
POWER					
power	# ...				

Dividing lines separate "equations".

The first one, two, or three equations report the mean model:

$$y_t = \mathbf{x}_t\boldsymbol{\beta} + \sum_i \psi_i g(\sigma^2_{t-i}) + \text{ARMA}(p,q) + \epsilon_t$$

The first equation reports $\boldsymbol{\beta}$, and the equation will be named [*depvar*]; if you fit a model on `d.cpi`, the first equation would be named `[cpi]`. In Stata, the coefficient on `x1` in the above example could be referred to as [*depvar*]`_b[x1]`. The coefficient on the lag 2 value of `x2` would be referred to as [*depvar*]`_b[L2.x2]`. Such notation would be used, for instance, in a later `test` command; see [R] **test**.

The [ARCHM] equation reports the ψ coefficients if your model includes ARCH-in-mean terms; see options discussed under the **Model 2** tab below. Most ARCH-in-mean models include only a contemporaneous variance term, so the term $\sum_i \psi_i g(\sigma^2_{t-i})$ becomes $\psi\sigma^2_t$. The coefficient ψ will be [ARCHM]_b[sigma2]. If your model includes lags of σ^2_t, the additional coefficients will be [ARCHM]_b[L1.sigma2], and so on. If you specify a transformation $g()$ (option archmexp()), the coefficients will be [ARCHM]_b[sigma2ex], [ARCHM]_b[L1.sigma2ex], and so on. sigma2ex refers to $g(\sigma^2_t)$, the transformed value of the conditional variance.

The [ARMA] equation reports the ARMA coefficients if your model includes them; see options discussed under the **Model 2** tab below. This equation includes one or two "variables" named ar and ma. In later test statements, you could refer to the coefficient on the first lag of the autoregressive term by typing [ARMA]_b[L1.ar] or simply [ARMA]_b[L.ar] (the L operator is assumed to be lag 1 if you do not specify otherwise). The second lag on the moving-average term, if there were one, could be referred to by typing [ARMA]_b[L2.ma].

The next one, two, or three equations report the variance model.

The [HET] equation reports the multiplicative heteroskedasticity if the model includes it. When you fit such a model, you specify the variables (and their lags), determining the multiplicative heteroskedasticity; after estimation, their coefficients are simply [HET]_b[*op.varname*].

The [ARCH] equation reports the ARCH, GARCH, etc., terms by referring to "variables" arch, garch, and so on. For instance, if you specified arch(1) garch(1) when you fit the model, the conditional variance is given by $\sigma^2_t = \gamma_0 + \alpha_{1,1}\epsilon^2_{t-1} + \alpha_{2,1}\sigma^2_{t-1}$. The coefficients would be named [ARCH]_b[_cons] (γ_0), [ARCH]_b[L.arch] ($\alpha_{1,1}$), and [ARCH]_b[L.garch] ($\alpha_{2,1}$).

The [POWER] equation appears only if you are fitting a variance model in the form of (3) above; the estimated φ is the coefficient [POWER]_b[power].

Also, if you use the distribution() option and specify either Student's t or the generalized error distribution but do not specify the degree-of-freedom or shape parameter, then you will see two additional rows in the table. The final row contains the estimated degree-of-freedom or shape parameter. Immediately preceding the final row is a transformed version of the parameter that arch used during estimation to ensure that the degree-of-freedom parameter is greater than two or that the shape parameter is positive.

The naming convention for estimated ARCH, GARCH, etc., parameters is as follows (definitions for parameters α_i, γ_i, and κ_i can be found in the tables for $A()$, $B()$, $C()$, and $D()$ above):

Option	1st parameter	2nd parameter	Common parameter
arch()	α_1 = [ARCH]_b[arch]		
garch()	α_2 = [ARCH]_b[garch]		
saarch()	α_3 = [ARCH]_b[saarch]		
tarch()	α_4 = [ARCH]_b[tarch]		
aarch()	α_5 = [ARCH]_b[aarch]	γ_5 = [ARCH]_b[aarch_e]	
narch()	α_6 = [ARCH]_b[narch]	κ_6 = [ARCH]_b[narch_k]	
narchk()	α_7 = [ARCH]_b[narch]	κ_7 = [ARCH]_b[narch_k]	
abarch()	α_8 = [ARCH]_b[abarch]		
atarch()	α_9 = [ARCH]_b[atarch]		
sdgarch()	α_{10} = [ARCH]_b[sdgarch]		
earch()	α_{11} = [ARCH]_b[earch]	γ_{11} = [ARCH]_b[earch_a]	
egarch()	α_{12} = [ARCH]_b[egarch]		
parch()	α_{13} = [ARCH]_b[parch]		φ = [POWER]_b[power]
tparch()	α_{14} = [ARCH]_b[tparch]		φ = [POWER]_b[power]
aparch()	α_{15} = [ARCH]_b[aparch]	γ_{15} = [ARCH]_b[aparch_e]	φ = [POWER]_b[power]
nparch()	α_{16} = [ARCH]_b[nparch]	κ_{16} = [ARCH]_b[nparch_k]	φ = [POWER]_b[power]
nparchk()	α_{17} = [ARCH]_b[nparch]	κ_{17} = [ARCH]_b[nparch_k]	φ = [POWER]_b[power]
pgarch()	α_{18} = [ARCH]_b[pgarch]		φ = [POWER]_b[power]

Options

Model

noconstant; see [R] **Estimation options**.

arch(*numlist*) specifies the ARCH terms (lags of ϵ_t^2).

Specify arch(1) to include first-order terms, arch(1/2) to specify first- and second-order terms, arch(1/3) to specify first-, second-, and third-order terms, etc. Terms may be omitted. Specify arch(1/3 5) to specify terms with lags 1, 2, 3, and 5. All the options work this way.

arch() may not be specified with aarch(), narch(), narchk(), nparchk(), or nparch(), as this would result in collinear terms.

garch(*numlist*) specifies the GARCH terms (lags of σ_t^2).

saarch(*numlist*) specifies the simple asymmetric ARCH terms. Adding these terms is one way to make the standard ARCH and GARCH models respond asymmetrically to positive and negative innovations. Specifying saarch() with arch() and garch() corresponds to the SAARCH model of Engle (1990).

saarch() may not be specified with narch(), narchk(), nparchk(), or nparch(), as this would result in collinear terms.

tarch(*numlist*) specifies the threshold ARCH terms. Adding these is another way to make the standard ARCH and GARCH models respond asymmetrically to positive and negative innovations. Specifying tarch() with arch() and garch() corresponds to one form of the GJR model (Glosten, Jagannathan, and Runkle 1993).

tarch() may not be specified with tparch() or aarch(), as this would result in collinear terms.

aarch(*numlist*) specifies lags of the two-parameter term $\alpha_i(|\epsilon_t| + \gamma_i\epsilon_t)^2$. This term provides the same underlying form of asymmetry as including arch() and tarch(), but it is expressed in a different way.

aarch() may not be specified with arch() or tarch(), as this would result in collinear terms.

`narch(`*numlist*`)` specifies lags of the two-parameter term $\alpha_i(\epsilon_t - \kappa_i)^2$. This term allows the minimum conditional variance to occur at a value of lagged innovations other than zero. For any term specified at lag L, the minimum contribution to conditional variance of that lag occurs when $\epsilon_{t-L}^2 = \kappa_L$—the squared innovations at that lag are equal to the estimated constant κ_L.

`narch()` may not be specified with `arch()`, `saarch()`, `narchk()`, `nparchk()`, or `nparch()`, as this would result in collinear terms.

`narchk(`*numlist*`)` specifies lags of the two-parameter term $\alpha_i(\epsilon_t - \kappa)^2$; this is a variation of `narch()` with κ held constant for all lags.

`narchk()` may not be specified with `arch()`, `saarch()`, `narch()`, `nparchk()`, or `nparch()`, as this would result in collinear terms.

`abarch(`*numlist*`)` specifies lags of the term $|\epsilon_t|$.

`atarch(`*numlist*`)` specifies lags of $|\epsilon_t|(\epsilon_t > 0)$, where $(\epsilon_t > 0)$ represents the indicator function returning 1 when true and 0 when false. Like the TARCH terms, these ATARCH terms allow the effect of unanticipated innovations to be asymmetric about zero.

`sdgarch(`*numlist*`)` specifies lags of σ_t. Combining `atarch()`, `abarch()`, and `sdgarch()` produces the model by Zakoian (1994) that the author called the TARCH model. The acronym TARCH, however, refers to any model using thresholding to obtain asymmetry.

`earch(`*numlist*`)` specifies lags of the two-parameter term $\alpha z_t + \gamma(|z_t| - \sqrt{2/\pi})$. These terms represent the influence of news—lagged innovations—in Nelson's (1991) EGARCH model. For these terms, $z_t = \epsilon_t/\sigma_t$, and `arch` assumes $z_t \sim N(0,1)$. Nelson derived the general form of an EGARCH model for any assumed distribution and performed estimation assuming a generalized error distribution (GED). See Hamilton (1994) for a derivation where z_t is assumed normal. The z_t terms can be parameterized in either of these two equivalent ways. `arch` uses Nelson's original parameterization; see Hamilton (1994) for an equivalent alternative.

`egarch(`*numlist*`)` specifies lags of $\ln(\sigma_t^2)$.

For the following options, the model is parameterized in terms of $h(\epsilon_t)^\varphi$ and σ_t^φ. One φ is estimated, even when more than one option is specified.

`parch(`*numlist*`)` specifies lags of $|\epsilon_t|^\varphi$. `parch()` combined with `pgarch()` corresponds to the class of nonlinear models of conditional variance suggested by Higgins and Bera (1992).

`tparch(`*numlist*`)` specifies lags of $(\epsilon_t > 0)|\epsilon_t|^\varphi$, where $(\epsilon_t > 0)$ represents the indicator function returning 1 when true and 0 when false. As with `tarch()`, `tparch()` specifies terms that allow for a differential impact of "good" (positive innovations) and "bad" (negative innovations) news for lags specified by *numlist*.

`tparch()` may not be specified with `tarch()`, as this would result in collinear terms.

`aparch(`*numlist*`)` specifies lags of the two-parameter term $\alpha(|\epsilon_t| + \gamma\epsilon_t)^\varphi$. This asymmetric power ARCH model, A-PARCH, was proposed by Ding, Granger, and Engle (1993) and corresponds to a Box–Cox function in the lagged innovations. The authors fit the original A-PARCH model on more than 16,000 daily observations of the Standard and Poor's 500, and for good reason. As the number of parameters and the flexibility of the specification increase, more data are required to estimate the parameters of the conditional heteroskedasticity. See Ding, Granger, and Engle (1993) for a discussion of how seven popular ARCH models nest within the A-PARCH model.

When γ goes to 1, the full term goes to zero for many observations and can then be numerically unstable.

`nparch(`*numlist*`)` specifies lags of the two-parameter term $\alpha|\epsilon_t - \kappa_i|^{\varphi}$.

`nparch()` may not be specified with `arch()`, `saarch()`, `narch()`, `narchk()`, or `nparchk()`, as this would result in collinear terms.

`nparchk(`*numlist*`)` specifies lags of the two-parameter term $\alpha|\epsilon_t - \kappa|^{\varphi}$; this is a variation of `nparch()` with κ held constant for all lags. This is the direct analog of `narchk()`, except for the power of φ. `nparchk()` corresponds to an extended form of the model of Higgins and Bera (1992) as presented by Bollerslev, Engle, and Nelson (1994). `nparchk()` would typically be combined with the `pgarch()` option.

`nparchk()` may not be specified with `arch()`, `saarch()`, `narch()`, `narchk()`, or `nparch()`, as this would result in collinear terms.

`pgarch(`*numlist*`)` specifies lags of σ_t^{φ}.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

Model 2

`archm` specifies that an ARCH-in-mean term be included in the specification of the mean equation. This term allows the expected value of *depvar* to depend on the conditional variance. ARCH-in-mean is most commonly used in evaluating financial time series when a theory supports a tradeoff between asset risk and return. By default, no ARCH-in-mean terms are included in the model.

`archm` specifies that the contemporaneous expected conditional variance be included in the mean equation. For example, typing

```
. arch y x, archm arch(1)
```

specifies the model

$$\mathtt{y}_t = \beta_0 + \beta_1 \mathtt{x}_t + \psi\sigma_t^2 + \epsilon_t$$
$$\sigma_t^2 = \gamma_0 + \gamma\epsilon_{t-1}^2$$

`archmlags(`*numlist*`)` is an expansion of `archm` that includes lags of the conditional variance σ_t^2 in the mean equation. To specify a contemporaneous and once-lagged variance, specify either `archm archmlags(1)` or `archmlags(0/1)`.

`archmexp(`*exp*`)` applies the transformation in *exp* to any ARCH-in-mean terms in the model. The expression should contain an `X` wherever a value of the conditional variance is to enter the expression. This option can be used to produce the commonly used ARCH-in-mean of the conditional standard deviation. With the example from `archm`, typing

```
. arch y x, archm arch(1) archmexp(sqrt(X))
```

specifies the mean equation $\mathtt{y}_t = \beta_0 + \beta_1 \mathtt{x}_t + \psi\sigma_t + \epsilon_t$. Alternatively, typing

```
. arch y x, archm arch(1) archmexp(1/sqrt(X))
```

specifies $\mathtt{y}_t = \beta_0 + \beta_1 \mathtt{x}_t + \psi/\sigma_t + \epsilon_t$.

`arima(#`p`,#`d`,#`q`)` is an alternative, shorthand notation for specifying autoregressive models in the dependent variable. The dependent variable and any independent variables are differenced $\#_d$ times, 1 through $\#_p$ lags of autocorrelations are included, and 1 through $\#_q$ lags of moving averages are included. For example, the specification

```
. arch y, arima(2,1,3)
```

is equivalent to

```
. arch D.y, ar(1/2) ma(1/3)
```

The former is easier to write for classic ARIMA models of the mean equation, but it is not nearly as expressive as the latter. If gaps in the AR or MA lags are to be modeled, or if different operators are to be applied to independent variables, the latter syntax is required.

`ar(`*numlist*`)` specifies the autoregressive terms of the structural model disturbance to be included in the model. For example, `ar(1/3)` specifies that lags 1, 2, and 3 of the structural disturbance be included in the model. `ar(1,4)` specifies that lags 1 and 4 be included, possibly to account for quarterly effects.

If the model does not contain regressors, these terms can also be considered autoregressive terms for the dependent variable; see [TS] **arima**.

`ma(`*numlist*`)` specifies the moving-average terms to be included in the model. These are the terms for the lagged innovations or white-noise disturbances.

Model 3

`distribution(`*dist* `[` *#* `])` specifies the distribution to assume for the error term. *dist* may be `gaussian`, `normal`, `t`, or `ged`. `gaussian` and `normal` are synonyms, and *#* cannot be specified with them.

If `distribution(t)` is specified, `arch` assumes that the errors follow Student's t distribution, and the degree-of-freedom parameter is estimated along with the other parameters of the model. If `distribution(t` *#*`)` is specified, then `arch` uses Student's t distribution with *#* degrees of freedom. *#* must be greater than 2.

If `distribution(ged)` is specified, `arch` assumes that the errors have a generalized error distribution, and the shape parameter is estimated along with the other parameters of the model. If `distribution(ged` *#*`)` is specified, then `arch` uses the generalized error distribution with shape parameter *#*. *#* must be positive. The generalized error distribution is identical to the normal distribution when the shape parameter equals 2.

`het(`*varlist*`)` specifies that *varlist* be included in the specification of the conditional variance. *varlist* may contain time-series operators. This varlist enters the variance specification collectively as multiplicative heteroskedasticity; see Judge et al. (1985). If `het()` is not specified, the model will not contain multiplicative heteroskedasticity.

Assume that the conditional variance depends on variables `x` and `w` and has an ARCH(1) component. We request this specification by using the `het(x w) arch(1)` options, and this corresponds to the conditional-variance model

$$\sigma_t^2 = \exp(\lambda_0 + \lambda_1 \mathtt{x}_t + \lambda_2 \mathtt{w}_t) + \alpha\epsilon_{t-1}^2$$

Multiplicative heteroskedasticity enters differently with an EGARCH model because the variance is already specified in logs. For the `het(x w) earch(1) egarch(1)` options, the variance model is

$$\ln(\sigma_t^2) = \lambda_0 + \lambda_1 \mathtt{x}_t + \lambda_2 \mathtt{w}_t + \alpha z_{t-1} + \gamma(|z_{t-1}| - \sqrt{2/\pi}) + \delta \ln(\sigma_{t-1}^2)$$

`savespace` conserves memory by retaining only those variables required for estimation. The original dataset is restored after estimation. This option is rarely used and should be specified only if there is insufficient memory to fit a model without the option. `arch` requires considerably more temporary storage during estimation than most estimation commands in Stata.

Priming

`arch0(`*cond_method*`)` is a rarely used option that specifies how to compute the conditioning (presample or priming) values for σ_t^2 and ϵ_t^2. In the presample period, it is assumed that $\sigma_t^2 = \epsilon_t^2$ and that this value is constant. If `arch0()` is not specified, the priming values are computed as the expected unconditional variance given the current estimates of the $\boldsymbol{\beta}$ coefficients and any ARMA parameters.

`arch0(xb)`, the default, specifies that the priming values are the expected unconditional variance of the model, which is $\sum_1^T \widehat{\epsilon}_t^2/T$, where $\widehat{\epsilon}_t$ is computed from the mean equation and any ARMA terms.

`arch0(xb0)` specifies that the priming values are the estimated variance of the residuals from an OLS estimate of the mean equation.

`arch0(xbwt)` specifies that the priming values are the weighted sum of the $\widehat{\epsilon}_t^2$ from the current conditional mean equation (and ARMA terms) that places more weight on estimates of ϵ_t^2 at the beginning of the sample.

`arch0(xb0wt)` specifies that the priming values are the weighted sum of the $\widehat{\epsilon}_t^2$ from an OLS estimate of the mean equation (and ARMA terms) that places more weight on estimates of ϵ_t^2 at the beginning of the sample.

`arch0(zero)` specifies that the priming values are 0. Unlike the priming values for ARIMA models, 0 is generally not a consistent estimate of the presample conditional variance or squared innovations.

`arch0(#)` specifies that $\sigma_t^2 = \epsilon_t^2 = \#$ for any specified nonnegative #. Thus `arch0(0)` is equivalent to `arch0(zero)`.

`arma0(`*cond_method*`)` is a rarely used option that specifies how the ϵ_t values are initialized at the beginning of the sample for the ARMA component, if the model has one. This option has an effect only when AR or MA terms are included in the model (the `ar()`, `ma()`, or `arima()` option is specified).

`arma0(zero)`, the default, specifies that all priming values of ϵ_t be taken as 0. This fits the model over the entire requested sample and takes ϵ_t as its expected value of 0 for all lags required by the ARMA terms; see Judge et al. (1985).

`arma0(p)`, `arma0(q)`, and `arma0(pq)` specify that estimation begin after priming the recursions for a certain number of observations. `p` specifies that estimation begin after the pth observation in the sample, where p is the maximum AR lag in the model; `q` specifies that estimation begin after the qth observation in the sample, where q is the maximum MA lag in the model; and `pq` specifies that estimation begin after the $(p+q)$th observation in the sample.

During the priming period, the recursions necessary to generate predicted disturbances are performed, but results are used only to initialize preestimation values of ϵ_t. To understand the definition of preestimation, say that you fit a model `in 10/100`. If the model is specified with `ar(1,2)`, preestimation refers to observations 10 and 11.

The ARCH terms σ_t^2 and ϵ_t^2 are also updated over these observations. Any required lags of ϵ_t before the priming period are taken to be their expected value of 0, and ϵ_t^2 and σ_t^2 take the values specified in `arch0()`.

`arma0(#)` specifies that the presample values of ϵ_t are to be taken as # for all lags required by the ARMA terms. Thus `arma0(0)` is equivalent to `arma0(zero)`.

`condobs(#)` is a rarely used option that specifies a fixed number of conditioning observations at the start of the sample. Over these priming observations, the recursions necessary to generate predicted disturbances are performed, but only to initialize preestimation values of ϵ_t, ϵ_t^2, and σ_t^2.

Any required lags of ϵ_t before the initialization period are taken to be their expected value of 0 (or the value specified in `arma0()`), and required values of ϵ_t^2 and σ_t^2 assume the values specified by `arch0()`. `condobs()` can be used if conditioning observations are desired for the lags in the ARCH terms of the model. If `arma()` is also specified, the maximum number of conditioning observations required by `arma()` and `condobs(#)` is used.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are robust to some kinds of misspecification (`robust`) and that are derived from asymptotic theory (`oim`, `opg`); see [R] ***vce_option***.

For ARCH models, the robust or quasi–maximum likelihood estimates (QMLE) of variance are robust to symmetric nonnormality in the disturbances. The robust variance estimates generally are not robust to functional misspecification of the mean equation; see Bollerslev and Wooldridge (1992).

The robust variance estimates computed by `arch` are based on the full Huber/White/sandwich formulation, as discussed in [P] **_robust**. Many other software packages report robust estimates that set some terms to their expectations of zero (Bollerslev and Wooldridge 1992), which saves them from calculating second derivatives of the log-likelihood function.

Reporting

`level(#)`; see [R] **Estimation options**.

`detail` specifies that a detailed list of any gaps in the series be reported, including gaps due to missing observations or missing data for the dependent variable or independent variables.

`nocnsreport`; see [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `vsquish`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `gtolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **Maximize** for all options except `gtolerance()`, and see below for information on `gtolerance()`.

These options are often more important for ARCH models than for other maximum likelihood models because of convergence problems associated with ARCH models—ARCH model likelihoods are notoriously difficult to maximize.

Setting `technique()` to something other than the default or BHHH changes the *vcetype* to `vce(oim)`.

The following options are all related to maximization and are either particularly important in fitting ARCH models or not available for most other estimators.

`gtolerance(#)` specifies the tolerance for the gradient relative to the coefficients. When $|g_i\, b_i| \leq$ `gtolerance()` for all parameters b_i and the corresponding elements of the gradient g_i, the gradient tolerance criterion is met. The default gradient tolerance for `arch` is `gtolerance(.05)`.

`gtolerance(999)` may be specified to disable the gradient criterion. If the optimizer becomes stuck with repeated "(backed up)" messages, the gradient probably still contains substantial values, but an uphill direction cannot be found for the likelihood. With this option, results can often be obtained, but whether the global maximum likelihood has been found is unclear.

When the maximization is not going well, it is also possible to set the maximum number of iterations (see [R] **Maximize**) to the point where the optimizer appears to be stuck and to inspect the estimation results at that point.

`from(`*init_specs*`)` specifies the initial values of the coefficients. ARCH models may be sensitive to initial values and may have coefficient values that correspond to local maximums. The default starting values are obtained via a series of regressions, producing results that, on the basis of asymptotic theory, are consistent for the $\boldsymbol{\beta}$ and ARMA parameters and generally reasonable for the rest. Nevertheless, these values may not always be feasible in that the likelihood function cannot be evaluated at the initial values `arch` first chooses. In such cases, the estimation is restarted with ARCH and ARMA parameters initialized to zero. It is possible, but unlikely, that even these values will be infeasible and that you will have to supply initial values yourself.

The standard syntax for `from()` accepts a matrix, a list of values, or coefficient name value pairs; see [R] **Maximize**. `arch` also allows the following:

`from(archb0)` sets the starting value for all the ARCH/GARCH/... parameters in the conditional-variance equation to 0.

`from(armab0)` sets the starting value for all ARMA parameters in the model to 0.

`from(archb0 armab0)` sets the starting value for all ARCH/GARCH/... and ARMA parameters to 0.

The following options are available with `arch` but are not shown in the dialog box:

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

stata.com

The volatility of a series is not constant through time; periods of relatively low volatility and periods of relatively high volatility tend to be grouped together. This is a commonly observed characteristic of economic time series and is even more pronounced in many frequently sampled financial series. ARCH models seek to estimate this time-dependent volatility as a function of observed prior volatility. Sometimes the model of volatility is of more interest than the model of the conditional mean. As implemented in `arch`, the volatility model may also include regressors to account for a structural component in the volatility—usually referred to as multiplicative heteroskedasticity.

ARCH models were introduced by Engle (1982) in a study of inflation rates, and there has since been a barrage of proposed parametric and nonparametric specifications of autoregressive conditional heteroskedasticity. Overviews of the literature can found in Bollerslev, Engle, and Nelson (1994) and Bollerslev, Chou, and Kroner (1992). Introductions to basic ARCH models appear in many general econometrics texts, including Davidson and MacKinnon (1993), Kmenta (1997), Stock and Watson (2019), and Wooldridge (2020). Harvey (1989) and Enders (2004) provide introductions to ARCH in the larger context of econometric time-series modeling, and Hamilton (1994) gives considerably more detail in the same context. Becketti (2020, chap. 8) provides a simple introduction to ARCH modeling with an emphasis on how to use Stata's `arch` command.

`arch` fits models of autoregressive conditional heteroskedasticity (ARCH, GARCH, etc.) using conditional maximum likelihood. By "conditional", we mean that the likelihood is computed based on an assumed or estimated set of priming values for the squared innovations ϵ_t^2 and variances σ_t^2 prior to the estimation sample; see Hamilton (1994) or Bollerslev (1986). Sometimes more conditioning is done on the first a, g, or $a+g$ observations in the sample, where a is the maximum ARCH term lag and g is the maximum GARCH term lag (or the maximum lags from the other ARCH family terms).

The original ARCH model proposed by Engle (1982) modeled the variance of a regression model's disturbances as a linear function of lagged values of the squared regression disturbances. We can write an ARCH(m) model as

$$y_t = \mathbf{x}_t\boldsymbol{\beta} + \epsilon_t \qquad \text{(conditional mean)}$$
$$\sigma_t^2 = \gamma_0 + \gamma_1\epsilon_{t-1}^2 + \gamma_2\epsilon_{t-2}^2 + \cdots + \gamma_m\epsilon_{t-m}^2 \qquad \text{(conditional variance)}$$

where

ϵ_t^2 is the squared residuals (or innovations)

γ_i are the ARCH parameters

The ARCH model has a specification for both the conditional mean and the conditional variance, and the variance is a function of the size of prior unanticipated innovations—ϵ_t^2. This model was generalized by Bollerslev (1986) to include lagged values of the conditional variance—a GARCH model. The GARCH(m, k) model is written as

$$y_t = \mathbf{x}_t\boldsymbol{\beta} + \epsilon_t$$
$$\sigma_t^2 = \gamma_0 + \gamma_1\epsilon_{t-1}^2 + \gamma_2\epsilon_{t-2}^2 + \cdots + \gamma_m\epsilon_{t-m}^2 + \delta_1\sigma_{t-1}^2 + \delta_2\sigma_{t-2}^2 + \cdots + \delta_k\sigma_{t-k}^2$$

where

γ_i are the ARCH parameters

δ_i are the GARCH parameters

In his pioneering work, Engle (1982) assumed that the error term, ϵ_t, followed a Gaussian (normal) distribution: $\epsilon_t \sim N(0, \sigma_t^2)$. However, as Mandelbrot (1963) and many others have noted, the distribution of stock returns appears to be leptokurtotic, meaning that extreme stock returns are more frequent than would be expected if the returns were normally distributed. Researchers have therefore assumed other distributions that can have fatter tails than the normal distribution; `arch` allows you to fit models assuming the errors follow Student's t distribution or the generalized error distribution. The t distribution has fatter tails than the normal distribution; as the degree-of-freedom parameter approaches infinity, the t distribution converges to the normal distribution. The generalized error distribution's tails are fatter than the normal distribution's when the shape parameter is less than two and are thinner than the normal distribution's when the shape parameter is greater than two.

The GARCH model of conditional variance can be considered an ARMA process in the squared innovations, although not in the variances as the equations might seem to suggest; see Hamilton (1994). Specifically, the standard GARCH model implies that the squared innovations result from

$$\epsilon_t^2 = \gamma_0 + (\gamma_1 + \delta_1)\epsilon_{t-1}^2 + (\gamma_2 + \delta_2)\epsilon_{t-2}^2 + \cdots + (\gamma_k + \delta_k)\epsilon_{t-k}^2 + w_t - \delta_1 w_{t-1} - \delta_2 w_{t-2} - \delta_3 w_{t-3}$$

where

$w_t = \epsilon_t^2 - \sigma_t^2$

w_t is a white-noise process that is fundamental for ϵ_t^2

One of the primary benefits of the GARCH specification is its parsimony in identifying the conditional variance. As with ARIMA models, the ARMA specification in GARCH allows the conditional variance to be modeled with fewer parameters than with an ARCH specification alone. Empirically, many series with a conditionally heteroskedastic disturbance have been adequately modeled with a GARCH(1,1) specification.

An ARMA process in the disturbances can easily be added to the mean equation. For example, the mean equation can be written with an ARMA$(1,1)$ disturbance as

$$y_t = \mathbf{x}_t\boldsymbol{\beta} + \rho(y_{t-1} - \mathbf{x}_{t-1}\boldsymbol{\beta}) + \theta\epsilon_{t-1} + \epsilon_t$$

with an obvious generalization to ARMA(p,q) by adding terms; see [TS] **arima** for more discussion of this specification. This change affects only the conditional-variance specification in that ϵ_t^2 now results from a different specification of the conditional mean.

Much of the literature on ARCH models focuses on alternative specifications of the variance equation. `arch` allows many of these specifications to be requested using the `saarch()` through `pgarch()` options, which imply that one or more terms may be changed or added to the specification of the variance equation.

These alternative specifications also address asymmetry. Both the ARCH and GARCH specifications imply a symmetric impact of innovations. Whether an innovation ϵ_t^2 is positive or negative makes no difference to the expected variance σ_t^2 in the ensuing periods; only the size of the innovation matters—good news and bad news have the same effect. Many theories, however, suggest that positive and negative innovations should vary in their impact. For risk-averse investors, a large unanticipated drop in the market is more likely to lead to higher volatility than a large unanticipated increase (see Black [1976], Nelson [1991]). `saarch()`, `tarch()`, `aarch()`, `abarch()`, `earch()`, `aparch()`, and `tparch()` allow various specifications of asymmetric effects.

`narch()`, `narchk()`, `nparch()`, and `nparchk()` imply an asymmetric impact of a specific form. All the models considered so far have a minimum conditional variance when the lagged innovations are all zero. "No news is good news" when it comes to keeping the conditional variance small. `narch()`, `narchk()`, `nparch()`, and `nparchk()` also have a symmetric response to innovations, but they are not centered at zero. The entire news-response function (response to innovations) is shifted horizontally so that minimum variance lies at some specific positive or negative value for prior innovations.

ARCH-in-mean models allow the conditional variance of the series to influence the conditional mean. This is particularly convenient for modeling the risk–return relationship in financial series; the riskier an investment, with all else equal, the lower its expected return. ARCH-in-mean models modify the specification of the conditional mean equation to be

$$y_t = \mathbf{x_t}\boldsymbol{\beta} + \psi\sigma_t^2 + \epsilon_t \qquad \text{(ARCH-in-mean)}$$

Although this linear form in the current conditional variance has dominated the literature, `arch` allows the conditional variance to enter the mean equation through a nonlinear transformation $g()$ and for this transformed term to be included contemporaneously or lagged.

$$y_t = \mathbf{x}_t\boldsymbol{\beta} + \psi_0 g(\sigma_t^2) + \psi_1 g(\sigma_{t-1}^2) + \psi_2 g(\sigma_{t-2}^2) + \cdots + \epsilon_t$$

Square root is the most commonly used $g()$ transformation because researchers want to include a linear term for the conditional standard deviation, but any transform $g()$ is allowed.

▷ Example 1: ARCH model

Consider a simple model of the U.S. Wholesale Price Index (WPI) (Enders 2004, 87–93), which we also consider in [TS] **arima**. The data are quarterly over the period 1960q1 through 1990q4.

In [TS] **arima**, we fit a model of the continuously compounded rate of change in the WPI, $\ln(\text{WPI}_t) - \ln(\text{WPI}_{t-1})$. The graph of the differenced series—see [TS] **arima**—clearly shows periods of high volatility and other periods of relative tranquility. This makes the series a good candidate for ARCH modeling. Indeed, price indices have been a common target of ARCH models. Engle (1982) presented the original ARCH formulation in an analysis of U.K. inflation rates.

First, we fit a constant-only model by OLS and test ARCH effects by using Engle's Lagrange multiplier test (`estat archlm`).

```
. use https://www.stata-press.com/data/r17/wpi1
. regress D.ln_wpi
```

Source	SS	df	MS	Number of obs	=	123
				F(0, 122)	=	0.00
Model	0	0	.	Prob > F	=	.
Residual	.02521709	122	.000206697	R-squared	=	0.0000
				Adj R-squared	=	0.0000
Total	.02521709	122	.000206697	Root MSE	=	.01438

D.ln_wpi	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
_cons	.0108215	.0012963	8.35	0.000	.0082553	.0133878

```
. estat archlm, lags(1)
LM test for autoregressive conditional heteroskedasticity (ARCH)
```

lags(*p*)	chi2	df	Prob > chi2
1	8.366	1	0.0038

H0: no ARCH effects *vs.* H1: ARCH(*p*) disturbance

Because the LM test shows a p-value of 0.0038, which is well below 0.05, we reject the null hypothesis of no ARCH(1) effects. Thus we can further estimate the ARCH(1) parameter by specifying `arch(1)`. See [R] **regress postestimation time series** for more information on Engle's LM test.

The first-order generalized ARCH model (GARCH, Bollerslev 1986) is the most commonly used specification for the conditional variance in empirical work and is typically written GARCH(1, 1). We can estimate a GARCH(1, 1) process for the log-differenced series by typing

```
. arch D.ln_wpi, arch(1) garch(1)
(setting optimization to BHHH)
Iteration 0:   log likelihood =  355.23458
Iteration 1:   log likelihood =  365.64586
  (output omitted)
Iteration 10:  log likelihood =  373.23397
ARCH family regression
Sample: 1960q2 thru 1990q4                      Number of obs     =        123
                                                Wald chi2(.)      =          .
Log likelihood = 373.234                        Prob > chi2       =          .
```

D.ln_wpi	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
ln_wpi						
_cons	.0061167	.0010616	5.76	0.000	.0040361	.0081974
ARCH						
arch						
L1.	.4364123	.2437428	1.79	0.073	-.0413147	.9141394
garch						
L1.	.4544606	.1866606	2.43	0.015	.0886127	.8203086
_cons	.0000269	.0000122	2.20	0.028	2.97e-06	.0000508

We have estimated the ARCH(1) parameter to be 0.436 and the GARCH(1) parameter to be 0.454, so our fitted GARCH(1, 1) model is

$$y_t = 0.0061 + \epsilon_t$$
$$\sigma_t^2 = 0.436\,\epsilon_{t-1}^2 + 0.454\,\sigma_{t-1}^2$$

where $y_t = \ln(\texttt{wpi}_t) - \ln(\texttt{wpi}_{t-1})$.

The model Wald test and probability are both reported as missing (.). By convention, Stata reports the model test for the mean equation. Here and fairly often for ARCH models, the mean equation consists only of a constant, and there is nothing to test. ◁

▷ Example 2: ARCH model with ARMA process

We can retain the GARCH(1, 1) specification for the conditional variance and model the mean as an ARMA process with AR(1) and MA(1) terms as well as a fourth-lag MA term to control for quarterly seasonal effects by typing

```
. arch D.ln_wpi, ar(1) ma(1 4) arch(1) garch(1)
(setting optimization to BHHH)
Iteration 0:   log likelihood =   380.9997
Iteration 1:   log likelihood =  388.57823
Iteration 2:   log likelihood =  391.34143
Iteration 3:   log likelihood =  396.36991
Iteration 4:   log likelihood =  398.01098
(switching optimization to BFGS)
Iteration 5:   log likelihood =  398.23668
BFGS stepping has contracted, resetting BFGS Hessian (0)
Iteration 6:   log likelihood =  399.21497
Iteration 7:   log likelihood =  399.21537  (backed up)
  (output omitted)
(switching optimization to BHHH)
Iteration 15:  log likelihood =  399.51441
Iteration 16:  log likelihood =  399.51443
Iteration 17:  log likelihood =  399.51443
ARCH family regression -- ARMA disturbances
Sample: 1960q2 thru 1990q4                      Number of obs     =        123
                                                Wald chi2(3)      =     153.56
Log likelihood = 399.5144                       Prob > chi2       =     0.0000
```

D.ln_wpi	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
ln_wpi						
_cons	.0069541	.0039517	1.76	0.078	-.000791	.0146992
ARMA						
ar						
L1.	.7922674	.1072225	7.39	0.000	.5821153	1.00242
ma						
L1.	-.341774	.1499943	-2.28	0.023	-.6357574	-.0477905
L4.	.2451724	.1251131	1.96	0.050	-.0000447	.4903896
ARCH						
arch						
L1.	.2040449	.1244991	1.64	0.101	-.0399688	.4480587
garch						
L1.	.6949687	.1892176	3.67	0.000	.3241091	1.065828
_cons	.0000119	.0000104	1.14	0.253	-8.52e-06	.0000324

To clarify exactly what we have estimated, we could write our model as

$$y_t = 0.007 + 0.792\,(y_{t-1} - 0.007) - 0.342\,\epsilon_{t-1} + 0.245\,\epsilon_{t-4} + \epsilon_t$$
$$\sigma_t^2 = 0.204\,\epsilon_{t-1}^2 + .695\,\sigma_{t-1}^2$$

where $y_t = \ln(\texttt{wpi}_t) - \ln(\texttt{wpi}_{t-1})$.

The ARCH(1) coefficient, 0.204, is not significantly different from zero, but the ARCH(1) and GARCH(1) coefficients are significant collectively. If you doubt this, you can check with `test`.

```
. test [ARCH]L1.arch [ARCH]L1.garch
 ( 1)  [ARCH]L.arch = 0
 ( 2)  [ARCH]L.garch = 0
           chi2(  2) =   84.92
         Prob > chi2 =    0.0000
```

(For comparison, we fit the model over the same sample used in example 1 of [TS] **arima**; Enders fits this GARCH model but over a slightly different sample.)

◁

❑ Technical note

The rather ugly iteration log on the previous result is typical, as difficulty in converging is common in ARCH models. This is actually a fairly well-behaved likelihood for an ARCH model. The "switching optimization to . . . " messages are standard messages from the default optimization method for `arch`. The "backed up" messages are typical of `BFGS` stepping as the BFGS Hessian is often overoptimistic, particularly during early iterations. These messages are nothing to be concerned about.

Nevertheless, watch out for the messages "BFGS stepping has contracted, resetting BFGS Hessian" and "backed up", which can flag problems that may result in an iteration log that goes on and on. Stata will never report convergence and will never report final results. The question is, when do you give up and press *Break*, and if you do, what then?

If the "BFGS stepping has contracted" message occurs repeatedly (more than, say, five times), it often indicates that convergence will never be achieved. Literally, it means that the BFGS algorithm was stuck and reset its Hessian and take a steepest-descent step.

The "backed up" message, if it occurs repeatedly, also indicates problems, but only if the likelihood value is simultaneously not changing. If the message occurs repeatedly but the likelihood value is changing, as it did above, all is going well; it is just going slowly.

If you have convergence problems, you can specify options to assist the current maximization method or try a different method. Or, your model specification and data may simply lead to a likelihood that is not concave in the allowable region and thus cannot be maximized.

If you see the "backed up" message with no change in the likelihood, you can reset the gradient tolerance to a larger value. Specifying the `gtolerance(999)` option disables gradient checking, allowing convergence to be declared more easily. This does not guarantee that convergence will be declared, and even if it is, the global maximum likelihood may not have been found.

You can also try to specify initial values.

Finally, you can try a different maximization method; see options discussed under the **Maximization** tab above.

ARCH models are notorious for having convergence difficulties. Unlike in most estimators in Stata, it is common for convergence to require many steps or even to fail. This is particularly true of the explicitly nonlinear terms such as `aarch()`, `narch()`, `aparch()`, or `archm` (ARCH-in-mean), and of any model with several lags in the ARCH terms. There is not always a solution. You can try other maximization methods or different starting values, but if your data do not support your assumed ARCH structure, convergence simply may not be possible.

ARCH models can be susceptible to irrelevant regressors or unnecessary lags, whether in the specification of the conditional mean or in the conditional variance. In these situations, `arch` will often continue to iterate, making little to no improvement in the likelihood. We view this conservative approach as better than declaring convergence prematurely when the likelihood has not been fully

maximized. `arch` is estimating the conditional form of second sample moments, often with flexible functions, and that is asking much of the data.

❑

❑ Technical note

`if` *exp* and `in` *range* are interpreted differently with commands accepting time-series operators. The time-series operators are resolved *before* the conditions are tested, which may lead to some confusion. Note the results of the following `list` commands:

```
. use https://www.stata-press.com/data/r17/archxmpl
. list t y l.y in 5/10
```

	t	y	L. y
5.	1961q1	30.8	30.7
6.	1961q2	30.5	30.8
7.	1961q3	30.5	30.5
8.	1961q4	30.6	30.5
9.	1962q1	30.7	30.6
10.	1962q2	30.6	30.7

```
. keep in 5/10
(118 observations deleted)
. list t y l.y
```

	t	y	L. y
1.	1961q1	30.8	.
2.	1961q2	30.5	30.8
3.	1961q3	30.5	30.5
4.	1961q4	30.6	30.5
5.	1962q1	30.7	30.6
6.	1962q2	30.6	30.7

We have one more lagged observation for `y` in the first case: `l.y` was resolved before the `in` restriction was applied. In the second case, the dataset no longer contains the value of `y` to compute the first lag. This means that

```
. use https://www.stata-press.com/data/r17/archxmpl, clear
. arch y l.x if twithin(1962q2, 1990q3), arch(1)
```

is not the same as

```
. keep if twithin(1962q2, 1990q3)
. arch y l.x, arch(1)
```

❑

Example 3: Asymmetric effects—EGARCH model

Continuing with the WPI data, we might be concerned that the economy as a whole responds differently to unanticipated increases in wholesale prices than it does to unanticipated decreases. Perhaps unanticipated increases lead to cash flow issues that affect inventories and lead to more volatility. We can see if the data support this supposition by specifying an ARCH model that allows an asymmetric effect of "news"—innovations or unanticipated changes. One of the most popular such models is EGARCH (Nelson 1991). The full first-order EGARCH model for the WPI can be specified as follows:

```
. use https://www.stata-press.com/data/r17/wpi1, clear
. arch D.ln_wpi, ar(1) ma(1 4) earch(1) egarch(1)
(setting optimization to BHHH)
Iteration 0:   log likelihood =   227.5251
Iteration 1:   log likelihood =  381.68412
  (output omitted)
Iteration 23:  log likelihood =  405.31453
ARCH family regression -- ARMA disturbances
Sample: 1960q2 thru 1990q4                      Number of obs     =        123
                                                Wald chi2(3)      =     156.02
Log likelihood = 405.3145                       Prob > chi2       =     0.0000
```

D.ln_wpi	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
ln_wpi						
_cons	.0087342	.0034004	2.57	0.010	.0020695	.0153989
ARMA						
ar						
L1.	.7692139	.0968393	7.94	0.000	.5794124	.9590154
ma						
L1.	-.3554623	.1265721	-2.81	0.005	-.6035391	-.1073855
L4.	.2414626	.0863834	2.80	0.005	.0721543	.4107709
ARCH						
earch						
L1.	.4063939	.11635	3.49	0.000	.1783521	.6344358
earch_a						
L1.	.2467327	.1233357	2.00	0.045	.0049993	.4884662
egarch						
L1.	.8417332	.0704074	11.96	0.000	.7037372	.9797291
_cons	-1.488366	.6604354	-2.25	0.024	-2.782795	-.1939363

Our result for the variance is

$$\ln(\sigma_t^2) = -1.49 + .406\, z_{t-1} + .247\,(\left|z_{t-1}\right| - \sqrt{2/\pi}\,) + .842\, \ln(\sigma_{t-1}^2)$$

where $z_t = \epsilon_t/\sigma_t$, which is distributed as $N(0,1)$.

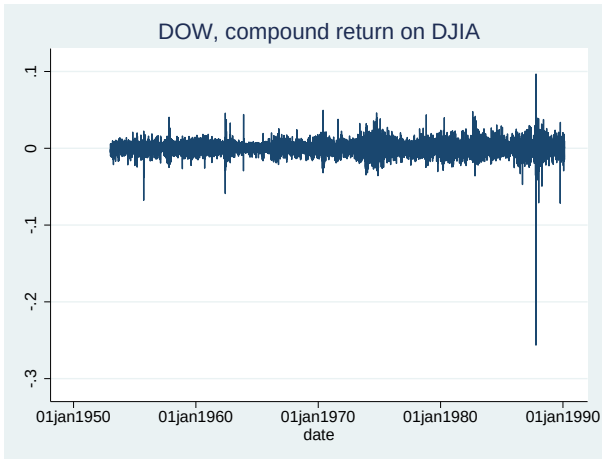
This is a strong indication for a leverage effect. The positive `L1.earch` coefficient implies that positive innovations (unanticipated price increases) are more destabilizing than negative innovations. The effect appears strong (0.406) and is substantially larger than the symmetric effect (0.247). In fact, the relative scales of the two coefficients imply that the positive leverage completely dominates the symmetric effect.

This can readily be seen if we plot what is often referred to as the news-response or news-impact function. This curve shows the resulting conditional variance as a function of unanticipated news, in the form of innovations, that is, the conditional variance σ_t^2 as a function of ϵ_t. Thus we must evaluate σ_t^2 for various values of ϵ_t—say, -4 to 4—and then graph the result.

◁

▷ Example 4: Asymmetric power ARCH model

As an example of a frequently sampled, long-run series, consider the daily closing indices of the Dow Jones Industrial Average, variable `dowclose`. To avoid the first half of the century, when the New York Stock Exchange was open for Saturday trading, only data after 1jan1953 are used. The compound return of the series is used as the dependent variable and is graphed below.



We formed this difference by referring to `D.ln_dow`, but only after playing a trick. The series is daily, and each observation represents the Dow closing index for the day. Our data included a time variable recorded as a daily date. We wanted, however, to model the log differences in the series, and we wanted the span from Friday to Monday to appear as a single-period difference. That is, the day before Monday is Friday. Because our dataset was `tsset` with `date`, the span from Friday to Monday was 3 days. The solution was to create a second variable that sequentially numbered the observations. By `tsset`ing the data with this new variable, we obtained the desired differences.

```
. generate t = _n
. tsset t
```

Now our data look like this:

```
. use https://www.stata-press.com/data/r17/dow1, clear
. generate dayofwk = dow(date)
. list date dayofwk t ln_dow D.ln_dow in 1/8
```

	date	dayofwk	t	ln_dow	D. ln_dow
1.	02jan1953	5	1	5.677096	.
2.	05jan1953	1	2	5.682899	.0058026
3.	06jan1953	2	3	5.677439	-.0054603
4.	07jan1953	3	4	5.672636	-.0048032
5.	08jan1953	4	5	5.671259	-.0013762
6.	09jan1953	5	6	5.661223	-.0100365
7.	12jan1953	1	7	5.653191	-.0080323
8.	13jan1953	2	8	5.659134	.0059433

```
. list date dayofwk t ln_dow D.ln_dow in -8/l
```

	date	dayofwk	t	ln_dow	D. ln_dow
9334.	08feb1990	4	9334	7.880188	.0016198
9335.	09feb1990	5	9335	7.881635	.0014472
9336.	12feb1990	1	9336	7.870601	-.011034
9337.	13feb1990	2	9337	7.872665	.0020638
9338.	14feb1990	3	9338	7.872577	-.0000877
9339.	15feb1990	4	9339	7.88213	.009553
9340.	16feb1990	5	9340	7.876863	-.0052676
9341.	20feb1990	2	9341	7.862054	-.0148082

The difference operator `D` spans weekends because the specified time variable, `t`, is not a true date and has a difference of 1 for all observations. We must leave this contrived time variable in place during estimation, or `arch` will be convinced that our dataset has gaps. If we were using calendar dates, we would indeed have gaps.

Ding, Granger, and Engle (1993) fit an A-PARCH model of daily returns of the Standard and Poor's 500 (S&P 500) for 3jan1928–30aug1991. We will fit the same model for the Dow data shown above. The model includes an AR(1) term as well as the A-PARCH specification of conditional variance.

```
. arch D.ln_dow, ar(1) aparch(1) pgarch(1)
(setting optimization to BHHH)
Iteration 0:   log likelihood =  31139.547
Iteration 1:   log likelihood =  31350.751
  (output omitted)
ARCH family regression -- AR disturbances
Sample: 2 thru 9341                             Number of obs     =       9340
                                                Wald chi2(1)      =     175.46
Log likelihood = 32273.56                       Prob > chi2       =     0.0000
```

D.ln_dow	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
ln_dow						
_cons	.0001786	.0000875	2.04	0.041	7.15e-06	.00035
ARMA						
ar						
L1.	.1410944	.0106519	13.25	0.000	.1202171	.1619716
ARCH						
aparch						
L1.	.0626323	.0034307	18.26	0.000	.0559082	.0693564
aparch_e						
L1.	-.3645093	.0378485	-9.63	0.000	-.4386909	-.2903277
pgarch						
L1.	.9299015	.0030998	299.99	0.000	.923826	.935977
_cons	7.19e-06	2.53e-06	2.84	0.004	2.23e-06	.0000121
POWER						
power	1.585187	.0629186	25.19	0.000	1.461869	1.708505

In the iteration log, the final iteration reports the message "backed up". For most estimators, ending on a "backed up" message would be a cause for great concern, but not with `arch` or, for that matter, `arima`, as long as you do not specify the `gtolerance()` option. `arch` and `arima`, by default, monitor the gradient and declare convergence only if, in addition to everything else, the gradient is small enough.

The fitted model demonstrates substantial asymmetry, with the large negative `L1.aparch_e` coefficient indicating that the market responds with much more volatility to unexpected drops in returns (bad news) than it does to increases in returns (good news).

◁

▷ Example 5: ARCH model with nonnormal errors

Stock returns tend to be leptokurtotic, meaning that large returns (either positive or negative) occur more frequently than one would expect if returns were in fact normally distributed. Here we refit the previous A-PARCH model assuming the errors follow the generalized error distribution, and we let `arch` estimate the shape parameter of the distribution.

```
. use https://www.stata-press.com/data/r17/dow1, clear
. arch D.ln_dow, ar(1) aparch(1) pgarch(1) distribution(ged)
(setting optimization to BHHH)
Iteration 0:   log likelihood =  31139.547
Iteration 1:   log likelihood =   31348.13
  (output omitted)
ARCH family regression -- AR disturbances
Sample: 2 thru 9341                             Number of obs     =       9340
                                                Wald chi2(1)      =     178.22
Log likelihood = 32486.46                       Prob > chi2       =     0.0000
```

D.ln_dow	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
ln_dow						
_cons	.0002735	.000078	3.51	0.000	.0001207	.0004264
ARMA						
ar						
L1.	.1337479	.0100187	13.35	0.000	.1141116	.1533842
ARCH						
aparch						
L1.	.0641772	.0049401	12.99	0.000	.0544949	.0738595
aparch_e						
L1.	-.405225	.0573059	-7.07	0.000	-.5175426	-.2929074
pgarch						
L1.	.9341739	.0045668	204.56	0.000	.9252231	.9431247
_cons	.0000216	.0000117	1.84	0.066	-1.39e-06	.0000446
POWER						
power	1.32524	.1030699	12.86	0.000	1.123227	1.527253
/lnshape	.3527019	.0094819	37.20	0.000	.3341177	.371286
shape	1.422907	.0134919			1.396707	1.449598

The ARMA and ARCH coefficients are similar to those we obtained when we assumed normally distributed errors, though we do note that the power term is now closer to 1. The estimated shape parameter for the generalized error distribution is shown at the bottom of the output. Here the shape parameter is 1.42; because it is less than 2, the distribution of the errors has tails that are fatter than they would be if the errors were normally distributed.

◁

▷ Example 6: ARCH model with constraints

Engle's (1982) original model, which sparked the interest in ARCH, provides an example requiring constraints. Most current ARCH specifications use GARCH terms to provide flexible dynamic properties without estimating an excessive number of parameters. The original model was limited to ARCH terms, and to help cope with the collinearity of the terms, a declining lag structure was imposed in the parameters. The conditional variance equation was specified as

$$\begin{aligned}\sigma_t^2 &= \alpha_0 + \alpha(0.4\,\epsilon_{t-1} + 0.3\,\epsilon_{t-2} + 0.2\,\epsilon_{t-3} + 0.1\,\epsilon_{t-4}) \\ &= \alpha_0 + 0.4\,\alpha\epsilon_{t-1} + 0.3\,\alpha\epsilon_{t-2} + 0.2\,\alpha\epsilon_{t-3} + 0.1\,\alpha\epsilon_{t-4}\end{aligned}$$

From the earlier `arch` output, we know how the coefficients will be named. In Stata, the formula is

$$\sigma_t^2 = \texttt{[ARCH]\_cons} + 0.4\,\texttt{[ARCH]L1.arch}\,\epsilon_{t-1} + 0.3\,\texttt{[ARCH]L2.arch}\,\epsilon_{t-2} + 0.2\,\texttt{[ARCH]L3.arch}\,\epsilon_{t-3} + 0.1\,\texttt{[ARCH]L4.arch}\,\epsilon_{t-4}$$

We could specify these linear constraints many ways, but the following seems fairly intuitive; see [R] **constraint** for syntax.

```
. use https://www.stata-press.com/data/r17/wpi1, clear
. constraint 1 (3/4)*[ARCH]l1.arch = [ARCH]l2.arch
. constraint 2 (2/4)*[ARCH]l1.arch = [ARCH]l3.arch
. constraint 3 (1/4)*[ARCH]l1.arch = [ARCH]l4.arch
```

The original model was fit on U.K. inflation; we will again use the WPI data and retain our earlier specification of the mean equation, which differs from Engle's U.K. inflation model. With our constraints, we type

```
. arch D.ln_wpi, ar(1) ma(1 4) arch(1/4) constraints(1/3)
(setting optimization to BHHH)
Iteration 0:   log likelihood =  396.80198
Iteration 1:   log likelihood =  399.07809
  (output omitted)
ARCH family regression -- ARMA disturbances
Sample: 1960q2 thru 1990q4                      Number of obs     =       123
                                                Wald chi2(3)      =    123.32
Log likelihood = 399.4624                       Prob > chi2       =    0.0000
 ( 1)  .75*[ARCH]L.arch - [ARCH]L2.arch = 0
 ( 2)  .5*[ARCH]L.arch - [ARCH]L3.arch = 0
 ( 3)  .25*[ARCH]L.arch - [ARCH]L4.arch = 0
```

D.ln_wpi	Coefficient	OPG std. err.	z	P>|z|	[95% conf.	interval]
ln_wpi						
_cons	.0077204	.0034531	2.24	0.025	.0009525	.0144883
ARMA						
ar						
L1.	.7388168	.1126811	6.56	0.000	.5179659	.9596676
ma						
L1.	-.2559691	.1442861	-1.77	0.076	-.5387646	.0268264
L4.	.2528923	.1140185	2.22	0.027	.02942	.4763645
ARCH						
arch						
L1.	.2180138	.0737787	2.95	0.003	.0734101	.3626174
L2.	.1635103	.055334	2.95	0.003	.0550576	.2719631
L3.	.1090069	.0368894	2.95	0.003	.0367051	.1813087
L4.	.0545034	.0184447	2.95	0.003	.0183525	.0906544
_cons	.0000483	7.66e-06	6.30	0.000	.0000333	.0000633

`L1.arch`, `L2.arch`, `L3.arch`, and `L4.arch` coefficients have the constrained relative sizes.

◁

Stored results

`arch` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_gaps)`	number of gaps
`e(condobs)`	number of conditioning observations
`e(k)`	number of parameters
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_dv)`	number of dependent variables
`e(k_aux)`	number of auxiliary parameters
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(chi2)`	χ^2
`e(p)`	p-value for model test
`e(archi)`	$\sigma_0^2=\epsilon_0^2$, priming values
`e(archany)`	`1` if model contains ARCH terms, `0` otherwise
`e(tdf)`	degrees of freedom for Student's t distribution
`e(shape)`	shape parameter for generalized error distribution
`e(tmin)`	minimum time
`e(tmax)`	maximum time
`e(power)`	φ for power ARCH terms
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`arch`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(covariates)`	list of covariates
`e(eqnames)`	names of equations
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(title)`	title in estimation output
`e(tmins)`	formatted minimum time
`e(tmaxs)`	formatted maximum time
`e(dist)`	distribution for error term: `gaussian`, `t`, or `ged`
`e(mhet)`	`1` if multiplicative heteroskedasticity
`e(dfopt)`	`yes` if degrees of freedom for t distribution or shape parameter for GED distribution was estimated, `no` otherwise
`e(chi2type)`	`Wald`; type of model χ^2 test
`e(vce)`	*vcetype* specified in `vce()`
`e(vcetype)`	title used to label Std. err.
`e(ma)`	lags for moving-average terms
`e(ar)`	lags for autoregressive terms
`e(arch)`	lags for ARCH terms
`e(archm)`	ARCH-in-mean lags
`e(archmexp)`	ARCH-in-mean exp
`e(earch)`	lags for EARCH terms
`e(egarch)`	lags for EGARCH terms
`e(aarch)`	lags for AARCH terms
`e(narch)`	lags for NARCH terms
`e(aparch)`	lags for A-PARCH terms
`e(nparch)`	lags for NPARCH terms
`e(saarch)`	lags for SAARCH terms
`e(parch)`	lags for PARCH terms
`e(tparch)`	lags for TPARCH terms
`e(abarch)`	lags for ABARCH terms
`e(tarch)`	lags for TARCH terms

e(atarch)	lags for ATARCH terms
e(sdgarch)	lags for SDGARCH terms
e(pgarch)	lags for PGARCH terms
e(garch)	lags for GARCH terms
e(opt)	type of optimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(tech)	maximization technique, including number of iterations
e(tech_steps)	number of iterations performed before switching techniques
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

The mean equation for the model fit by arch and with ARMA terms can be written as

$$y_t = \mathbf{x_t}\boldsymbol{\beta} + \sum_{i=1}^{p} \psi_i g(\sigma^2_{t-i}) + \sum_{j=1}^{p} \rho_j \left\{ y_{t-j} - x_{t-j}\boldsymbol{\beta} - \sum_{i=1}^{p} \psi_i g(\sigma^2_{t-j-i}) \right\} + \sum_{k=1}^{q} \theta_k \epsilon_{t-k} + \epsilon_t \qquad \text{(conditional mean)}$$

where

$\boldsymbol{\beta}$ are the regression parameters,

$\boldsymbol{\psi}$ are the ARCH-in-mean parameters,

$\boldsymbol{\rho}$ are the autoregression parameters,

$\boldsymbol{\theta}$ are the moving-average parameters, and

$g()$ is a general function, see the archmexp() option.

Any of the parameters in this full specification of the conditional mean may be zero. For example, the model need not have moving-average parameters ($\boldsymbol{\theta} = 0$) or ARCH-in-mean parameters ($\boldsymbol{\psi} = 0$).

The variance equation will be one of the following:

$$\sigma^2 = \gamma_0 + A(\boldsymbol{\sigma}, \boldsymbol{\epsilon}) + B(\boldsymbol{\sigma}, \boldsymbol{\epsilon})^2 \tag{1}$$

$$\ln \sigma_t^2 = \gamma_0 + C(\ln\boldsymbol{\sigma}, \mathbf{z}) + A(\boldsymbol{\sigma}, \boldsymbol{\epsilon}) + B(\boldsymbol{\sigma}, \boldsymbol{\epsilon})^2 \tag{2}$$

$$\sigma_t^{\varphi} = \gamma_0 + D(\boldsymbol{\sigma}, \boldsymbol{\epsilon}) + A(\boldsymbol{\sigma}, \boldsymbol{\epsilon}) + B(\boldsymbol{\sigma}, \boldsymbol{\epsilon})^2 \tag{3}$$

where $A(\boldsymbol{\sigma}, \boldsymbol{\epsilon})$, $B(\boldsymbol{\sigma}, \boldsymbol{\epsilon})$, $C(\ln\boldsymbol{\sigma}, \mathbf{z})$, and $D(\boldsymbol{\sigma}, \boldsymbol{\epsilon})$ are linear sums of the appropriate ARCH terms; see *Details of syntax* for more information. Equation (1) is used if no EGARCH or power ARCH terms are included in the model, (2) if EGARCH terms are included, and (3) if any power ARCH terms are included; see *Details of syntax*.

Methods and formulas are presented under the following headings:

Priming values
Likelihood from prediction error decomposition
Missing data

Priming values

The above model is recursive with potentially long memory. It is necessary to assume preestimation sample values for ϵ_t, ϵ_t^2, and σ_t^2 to begin the recursions, and the remaining computations are therefore conditioned on these priming values, which can be controlled using the `arch0()` and `arma0()` options. See options discussed under the **Priming** tab above.

The `arch0(xb0wt)` and `arch0(xbwt)` options compute a weighted sum of estimated disturbances with more weight on the early observations. With either of these options,

$$\sigma^2_{t_0-i} = \epsilon^2_{t_0-i} = (1 - 0.7) \sum_{t=0}^{T-1} 0.7^{T-t-1}\, \epsilon^2_{T-t} \qquad \forall i$$

where t_0 is the first observation for which the likelihood is computed; see options discussed under the **Priming** tab above. The ϵ_t^2 are all computed from the conditional mean equation. If `arch0(xb0wt)` is specified, $\boldsymbol{\beta}$, ψ_i, ρ_j, and θ_k are taken from initial regression estimates and held constant during optimization. If `arch0(xbwt)` is specified, the current estimates of $\boldsymbol{\beta}$, ψ_i, ρ_j, and θ_k are used to compute ϵ_t^2 on every iteration. If any ψ_i is in the mean equation (ARCH-in-mean is specified), the estimates of ϵ_t^2 from the initial regression estimates are not consistent.

Likelihood from prediction error decomposition

The likelihood function for ARCH has a particularly simple form. Given priming (or conditioning) values of ϵ_t, ϵ_t^2, and σ_t^2, the mean equation above can be solved recursively for every ϵ_t (prediction error decomposition). Likewise, the conditional variance can be computed recursively for each observation by using the variance equation. Using these predicted errors, their associated variances, and the assumption that $\epsilon_t \sim N(0, \sigma_t^2)$, we find that the log likelihood for each observation t is

$$\ln L_t = -\frac{1}{2}\left\{ \ln(2\pi\sigma_t^2) + \frac{\epsilon_t^2}{\sigma_t^2} \right\}$$

If we assume that $\epsilon_t \sim t(\text{df})$, then as given in Hamilton (1994, 662),

$$\ln L_t = \ln\Gamma\left(\frac{\text{df}+1}{2}\right) - \ln\Gamma\left(\frac{\text{df}}{2}\right) - \frac{1}{2}\left[\ln\left\{(\text{df}-2)\pi\sigma_t^2\right\} + (\text{df}+1)\ln\left\{1+\frac{\epsilon_t^2}{(\text{df}-2)\sigma_t^2}\right\}\right]$$

The likelihood is not defined for $\text{df} \leq 2$, so instead of estimating df directly, we estimate $m = \ln(\text{df}-2)$. Then $\text{df} = \exp(m)+2 > 2$ for any m.

Following Bollerslev, Engle, and Nelson (1994, 2978), the log likelihood for the tth observation, assuming $\epsilon_t \sim \text{GED}(s)$, is

$$\ln L_t = \ln s - \ln\lambda - \frac{s+1}{s}\ln 2 - \ln\Gamma\left(s^{-1}\right) - \frac{1}{2}\left|\frac{\epsilon_t}{\lambda\sigma_t}\right|^s$$

where

$$\lambda = \left\{\frac{\Gamma\left(s^{-1}\right)}{2^{2/s}\Gamma\left(3s^{-1}\right)}\right\}^{1/2}$$

To enforce the restriction that $s > 0$, we estimate $r = \ln s$.

This command supports the Huber/White/sandwich estimator of the variance using `vce(robust)`. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*.

Missing data

ARCH allows missing data or missing observations but does not attempt to condition on the surrounding data. If a dynamic component cannot be computed—ϵ_t, ϵ_t^2, and/or σ_t^2—its priming value is substituted. If a covariate, the dependent variable, or the entire observation is missing, the observation does not enter the likelihood, and its dynamic components are set to their priming values for that observation. This is acceptable only asymptotically and should not be used with a great deal of missing data.

Robert Fry Engle (1942–) was born in Syracuse, New York. He earned degrees in physics and economics at Williams College and Cornell and then worked at MIT and the University of California, San Diego, before moving to New York University Stern School of Business in 2000. He was awarded the 2003 Nobel Prize in Economics for research on autoregressive conditional heteroskedasticity and is a leading expert in time-series analysis, especially the analysis of financial markets.

References

Adkins, L. C., and R. C. Hill. 2011. *Using Stata for Principles of Econometrics*. 4th ed. Hoboken, NJ: Wiley.

Baum, C. F., and S. Hurn. 2021a. *Environmental Econometrics Using Stata*. College Station, TX: Stata Press.

——. 2021b. "What good is a volatility model?" A reexamination after 20 years. *Stata Journal* 21: 295–319.

Becketti, S. 2020. *Introduction to Time Series Using Stata*. Rev. ed. College Station, TX: Stata Press.

Berndt, E. K., B. H. Hall, R. E. Hall, and J. A. Hausman. 1974. Estimation and inference in nonlinear structural models. *Annals of Economic and Social Measurement* 3/4: 653–665.

Black, F. S. 1976. Studies of stock price volatility changes. *Proceedings of the American Statistical Association, Business and Economics Statistics* 177–181.

Boffelli, S., and G. Urga. 2016. *Financial Econometrics Using Stata*. College Station, TX: Stata Press.

Bollerslev, T. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31: 307–327. https://doi.org/10.1016/0304-4076(86)90063-1.

Bollerslev, T., R. Y. Chou, and K. F. Kroner. 1992. ARCH modeling in finance. *Journal of Econometrics* 52: 5–59. https://doi.org/10.1016/0304-4076(92)90064-X.

Bollerslev, T., R. F. Engle, and D. B. Nelson. 1994. ARCH models. In Vol. 4 of *Handbook of Econometrics*, ed. R. F. Engle and D. L. McFadden. Amsterdam: Elsevier. https://doi.org/10.1016/S1573-4412(05)80018-2.

Bollerslev, T., and J. M. Wooldridge. 1992. Quasi-maximum likelihood estimation and inference in dynamic models with time-varying covariances. *Econometric Reviews* 11: 143–172. https://doi.org/10.1080/07474939208800229.

Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.

Dicle, M. F., and J. D. Levendis. 2017. Technical financial analysis tools for Stata. *Stata Journal* 17: 736–747.

Diebold, F. X. 2003. The *ET* Interview: Professor Robert F. Engle. *Econometric Theory* 19: 1159–1193. https://doi.org/10.1017/S0266466603196120.

Ding, Z., C. W. J. Granger, and R. F. Engle. 1993. A long memory property of stock market returns and a new model. *Journal of Empirical Finance* 1: 83–106. https://doi.org/10.1016/0927-5398(93)90006-D.

Enders, W. 2004. *Applied Econometric Time Series*. 2nd ed. New York: Wiley.

Engle, R. F. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* 50: 987–1007. https://doi.org/10.2307/1912773.

——. 1990. Discussion: Stock volatility and the crash of '87. *Review of Financial Studies* 3: 103–106. https://doi.org/10.1093/rfs/3.1.103.

Engle, R. F., D. M. Lilien, and R. P. Robins. 1987. Estimating time varying risk premia in the term structure: The ARCH-M model. *Econometrica* 55: 391–407. https://doi.org/10.2307/1913242.

Glosten, L. R., R. Jagannathan, and D. E. Runkle. 1993. On the relation between the expected value and the volatility of the nominal excess return on stocks. *Journal of Finance* 48: 1779–1801. https://doi.org/10.1111/j.1540-6261.1993.tb05128.x.

Hamilton, J. D. 1994. *Time Series Analysis*. Princeton, NJ: Princeton University Press.

Harvey, A. C. 1989. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge: Cambridge University Press.

——. 1990. *The Econometric Analysis of Time Series*. 2nd ed. Cambridge, MA: MIT Press.

Higgins, M. L., and A. K. Bera. 1992. A class of nonlinear ARCH models. *International Economic Review* 33: 137–158. https://doi.org/10.2307/2526988.

Hill, R. C., W. E. Griffiths, and G. C. Lim. 2018. *Principles of Econometrics*. 5th ed. Hoboken, NJ: Wiley.

Judge, G. G., W. E. Griffiths, R. C. Hill, H. Lütkepohl, and T.-C. Lee. 1985. *The Theory and Practice of Econometrics*. 2nd ed. New York: Wiley.

Kmenta, J. 1997. *Elements of Econometrics*. 2nd ed. Ann Arbor: University of Michigan Press.

Mandelbrot, B. B. 1963. The variation of certain speculative prices. *Journal of Business* 36: 394–419. https://doi.org/10.1086/294632.

Nelson, D. B. 1991. Conditional heteroskedasticity in asset returns: A new approach. *Econometrica* 59: 347–370. https://doi.org/10.2307/2938260.

Pickup, M. 2015. *Introduction to Time Series Analysis*. Thousand Oaks, CA: SAGE.

Stock, J. H., and M. W. Watson. 2019. *Introduction to Econometrics*. 4th ed. New York: Pearson.

Wooldridge, J. M. 2020. *Introductory Econometrics: A Modern Approach*. 7th ed. Boston: Cengage.

Zakoian, J. M. 1994. Threshold heteroskedastic models. *Journal of Economic Dynamics and Control* 18: 931–955. https://doi.org/10.1016/0165-1889(94)90039-6.

Also see

[TS] **arch postestimation** — Postestimation tools for arch

[TS] **arima** — ARIMA, ARMAX, and other dynamic regression models

[TS] **mgarch** — Multivariate GARCH models

[TS] **tsset** — Declare data to be time-series data

[R] **regress** — Linear regression

[U] 20 Estimation and postestimation commands