Conditional Heteroscedasticity (CH)

- So far, our models are for the *conditional mean*.

- For instance, the Gaussian AR(1) model

\[ y_t - \mu = \phi (y_{t-1} - \mu) + \epsilon_t \]

may be written:

Conditionally on \( y_{t-1}, y_{t-2}, \ldots \), \( y_t \sim N[\mu + \phi (y_{t-1} - \mu), \sigma_w^2] \).

- The conditional *mean* depends on the past, the conditional *variance* does not.
• Three key features:
  – The conditional distribution is normal;
  – The conditional mean is a linear function of $y_{t-1}, y_{t-2}, \ldots$;
  – The conditional variance is constant: *conditional homoscedasticity*.

• All three features could be changed.
Non-normal noise: typically longer tails; for fitting, provided the variance is finite, changes the likelihood function, but not much else.

Nonlinear mean function: Modeling a nonlinear mean is quite difficult; for instance, ensuring stationarity is restrictive. *Threshold* models are perhaps most feasible.

Non-constant variance. Two approaches:

- ARCH (AutoRegressive CH), GARCH (Generalized ARCH), ...
- Stochastic volatility.
**ARCH Models**

- Simplest is ARCH(1):

  \[
  y_t = \sigma_t \epsilon_t \\
  \sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2
  \]

  where \( \epsilon_t \) is Gaussian white noise with variance 1.

- Alternatively:

  Conditionally on \( y_{t-1}, y_{t-2}, \ldots \), \( y_t \sim N\left(0, \alpha_0 + \alpha_1 y_{t-1}^2\right) \).
• If $|y_{t-1}|$ happens to be large, $\sigma_t$ is increased, so $|y_t|$ also tends to be large.

• Conversely, if $|y_{t-1}|$ happens to be small, $\sigma_t$ is decreased, so $|y_t|$ also tends to be small.

• $\Rightarrow$ volatility clusters and long tails.

• $n = 1000$; alpha1 = 0.9; alpha0 = 1 - alpha1;
y = epsilon = ts(rnorm(n));
par(mfcol = c(2, 1));
plot(epsilon);
for (i in 2:n) y[i] = epsilon[i] * sqrt(alpha0 + alpha1 * y[i - 1]^2);
plot(y);
ARCH as AR

- The ARCH(1) model for $y_t$ implies:

\[
y_t^2 = \sigma_t^2 \epsilon_t^2
= \sigma_t^2 + \sigma_t^2 (\epsilon_t^2 - 1)
= \alpha_0 + \alpha_1 y_{t-1}^2 + \sigma_t^2 (\epsilon_t^2 - 1)
\]

or

\[
y_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + v_t,
\]

where

\[
v_t = \sigma_t^2 (\epsilon_t^2 - 1).
\]
• Note that

\[ E(v_t | y_{t-1}, y_{t-2}, \ldots) = 0, \]

and hence that for \( h > 0 \),

\[
E(vtv_{t-h}) = E[E(vtv_{t-h} | y_{t-1}, y_{t-2}, \ldots)] \\
= E[v_{t-h}E(v_t | y_{t-1}, y_{t-2}, \ldots)] \\
= 0,
\]

so \( v_t \) is (highly nonnormal) white noise, and \( y_t^2 \) is AR(1).

• For positivity and stationarity, \( \alpha_0 > 0 \) and \( 0 \leq \alpha_1 < 1 \), and unconditionally,

\[
E(y_t^2) = \text{var}(y_t) = \frac{\alpha_0}{1 - \alpha_1}.
\]
Extensions and Generalizations

- Extend to ARCH($m$):
  \[
  y_t = \sigma_t \epsilon_t \\
  \sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \alpha_2 y_{t-2}^2 + \cdots + \alpha_m y_{t-m}^2.
  \]
  Now $y_t^2$ is AR($m$) $\Rightarrow$ usual restrictions on $\alpha$'s.

- Generalize to GARCH($m, r$):
  \[
  y_t = \sigma_t \epsilon_t \\
  \sigma_t^2 = \alpha_0 + \sum_{j=1}^{m} \alpha_j y_{t-j}^2 + \sum_{j=1}^{r} \beta_j \sigma_{t-j}^2.
  \]
  Now $y_t^2$ is ARMA[$m, \max(m, r)$] $\Rightarrow$ corresponding restrictions on $\alpha$'s and $\beta$'s.
Simplest GARCH model: GARCH(1, 1)

- The GARCH(1, 1) model is widely used:
  \[
  \sigma_t^2 = \alpha_0 + \alpha_1 y_{t-1}^2 + \beta_1 \sigma_{t-1}^2
  \]
  with
  \[
  \alpha_1 + \beta_1 < 1
  \]
  for stationarity.

- The unconditional variance is now
  \[
  \mathbb{E}(y_t^2) = \text{var}(y_t) = \frac{\alpha_0}{1 - \alpha_1 - \beta_1}.
  \]
n = 1000; alpha1 = 0.5; beta1 = 0.4; alpha0 = 1 - alpha1 - beta1;
y = epsilon = ts(rnorm(n));
par(mfcol = c(2, 1));
plot(epsilon);
sigmatsq = 1;
for (i in 2:n) {
    sigmatsq = alpha0 + alpha1 * y[i - 1]^2 + beta1 * sigmatsq;
y[i] = epsilon[i] * sqrt(sigmatsq);
}
plot(y);

Volatility clusters are more sustained.
In SAS, use `proc autoreg` and the `garch` option on the `model` statement.

In R, explore and describe volatility:

```r
nyse = ts(scan("nyse.dat"));
par(mfcol = c(2, 1));
plot(nyse);
plot(nyse);
plot(abs(nyse));
lines(lowess(time(nyse), abs(nyse), f = .005), col = "red");

par(mfcol = c(2, 2));
acf(nyse);
acf(abs(nyse));
acf(abs(nyse));
acf(nyse^2);
```
In R, fit GARCH (default is 1,1):

```r
library(tseries);
ylse.g = garch(nyse);
summary(nyse.g);
plot(nyse.g);

par(mfcol = c(1, 1));
plot(nyse);
matlines(predict(nyse.g), col = "red", lty = 1);
```
GARCH with a unit root: IGARCH

• A special case: IGARCH(1, 1) = GARCH(1, 1) with $\alpha_1 + \beta_1 = 1$:

$$y_t = \sigma_t \epsilon_t$$

$$\sigma_t^2 = \alpha_0 + (1 - \beta_1) y_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

• Solving recursively with $\alpha_0 = 0$:

$$\sigma_t^2 = (1 - \beta_1) \sum_{j=1}^{\infty} \beta_1^{j-1} y_{t-j}^2$$

an exponentially weighted moving average of $y_t^2$. 

13
Tail Length

• All xARCH models give $y_t$ with “fat tails”:

  $y_t = \sigma_t \epsilon_t$ where $\epsilon_t \sim N(0, 1)$ \Rightarrow

  $$f_y(y) = \int f_\sigma(\sigma) \times \frac{1}{\sigma} \phi \left( \frac{y}{\sigma} \right) d\sigma.$$  

  $f_y(\cdot)$ is a mixture of Gaussian densities with the same mean and different variances.

• In practice, residuals in xARCH models may not be normal, but are usually closer to normal than the original data.
Shumway and Stoffer’s code for Example 5.3 does not work with the R `garch` function.

The `fGarch` package provides another method, `garchFit`, which allows simultaneous fitting of ARMA and GARCH models.
gnp96 = read.table("http://www.stat.pitt.edu/stoffer/tsa2/data/gnp96.dat");
gnpr = ts(diff(log(gnp96[, 2])), frequency = 4, start = c(1947, 1));
library(fGarch);
gnpr.mod = garchFit(gnpr ~ arma(1, 0) + garch(1, 0), data.frame(gnpr = gnpr));
summary(gnpr.mod);

Title:
   GARCH Modelling

Call:
   garchFit(formula = gnpr ~ arma(1, 0) + garch(1, 0),
            data = data.frame(gnpr = gnpr))

Mean and Variance Equation:
   data ~ arma(1, 0) + garch(1, 0)
   [data = data.frame(gnpr = gnpr)]

Conditional Distribution:
   norm
Coefficient(s):

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mu</td>
<td>ar1</td>
<td>omega</td>
<td>alpha1</td>
</tr>
<tr>
<td>0.00527795</td>
<td>0.36656255</td>
<td>0.00007331</td>
<td>0.19447134</td>
</tr>
</tbody>
</table>

Std. Errors:

based on Hessian

Error Analysis:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate</td>
<td>Std. Error</td>
<td>t value</td>
<td>Pr(&gt;</td>
<td>t</td>
</tr>
<tr>
<td>mu</td>
<td>5.278e-03</td>
<td>8.996e-04</td>
<td>5.867</td>
<td>4.44e-09 ***</td>
</tr>
<tr>
<td>ar1</td>
<td>3.666e-01</td>
<td>7.514e-02</td>
<td>4.878</td>
<td>1.07e-06 ***</td>
</tr>
<tr>
<td>omega</td>
<td>7.331e-05</td>
<td>9.011e-06</td>
<td>8.135</td>
<td>4.44e-16 ***</td>
</tr>
<tr>
<td>alpha1</td>
<td>1.945e-01</td>
<td>9.554e-02</td>
<td>2.035</td>
<td>0.0418 *</td>
</tr>
</tbody>
</table>

Signif. codes:  0 ***  0.001 **  0.01 *  0.05 .  0.1  1

Log Likelihood:

722.2849 normalized: 3.253536
### Standardised Residuals Tests:

<table>
<thead>
<tr>
<th>Test</th>
<th>Type</th>
<th>Statistic</th>
<th>p-Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jarque-Bera Test</td>
<td>R</td>
<td>Chi^2</td>
<td>9.118036</td>
</tr>
<tr>
<td>Shapiro-Wilk Test</td>
<td>R</td>
<td>W</td>
<td>0.9842405</td>
</tr>
<tr>
<td>Ljung-Box Test</td>
<td>R</td>
<td>Q(10)</td>
<td>9.874326</td>
</tr>
<tr>
<td>Ljung-Box Test</td>
<td>R</td>
<td>Q(15)</td>
<td>17.55855</td>
</tr>
<tr>
<td>Ljung-Box Test</td>
<td>R</td>
<td>Q(20)</td>
<td>23.41363</td>
</tr>
<tr>
<td>Ljung-Box Test</td>
<td>R^2</td>
<td>Q(10)</td>
<td>19.2821</td>
</tr>
<tr>
<td>Ljung-Box Test</td>
<td>R^2</td>
<td>Q(15)</td>
<td>33.23648</td>
</tr>
<tr>
<td>Ljung-Box Test</td>
<td>R^2</td>
<td>Q(20)</td>
<td>37.74259</td>
</tr>
<tr>
<td>LM Arch Test</td>
<td>R</td>
<td>TR^2</td>
<td>25.41625</td>
</tr>
</tbody>
</table>

### Information Criterion Statistics:

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>-6.471035</td>
</tr>
<tr>
<td>BIC</td>
<td>-6.409726</td>
</tr>
<tr>
<td>SIC</td>
<td>-6.471669</td>
</tr>
<tr>
<td>HQIC</td>
<td>-6.446282</td>
</tr>
</tbody>
</table>
• `garchFit` also provides many diagnostic plots:

```
plot(gnpr.mod);
```