In [2]:
```
using CSV, DataFrames
using Plots
using Distributions, Statistics
using Optim, NLopt
```

Time Series Analysis

Definition (Stochastic process) A stochastic process is a sequence of random variables, indexed by time: $\{Y_t\}_{t=-\infty}^{\infty}$

Definition (Times series) A time series is one observation of a stochastic process, over a specific interval: $\{y_t\}_{t=1}^{n}$.

So a time series is a sample of size n from a stochastic process. It's important to keep in mind that conceptually, one could draw another sample, and that the values would be different.

Definition (Autocovariance) The j^{th} autocovariance of a stochastic process is $\gamma_{jt} = \mathcal{E}(y_t - \mu_t)(y_{t-j} - \mu_{t-j})$ where $\mu_t = \mathcal{E}\left(y_t\right)$.

Definition (Covariance (weak) stationarity) A stochastic process is covariance stationary if it has time constant mean and autocovariances of all orders:

$$\begin{aligned} \mu_t &= \mu, \ \forall t \\ \gamma_{jt} &= \gamma_j, \ \forall t \end{aligned}$$

As we've seen, this implies that $\gamma_j = \gamma_{-j}$: the autocovariances depend only one the interval between observations, but not the time of the observations.

Definition (Strong stationarity) A stochastic process is strongly stationary if the joint distribution of an arbitrary collection of the $\{Y_t\}$, e.g., $(Y_{t-j}, Y_{t-k}, \ldots, Y_t, \ldots, Y_{t+l}, Y_{t+m}\}$, doesn't depend on t.

Since moments are determined by the distribution, strong stationarity⇒weak stationarity.

The time series is one sample from the stochastic process, and each of the random variables over the sample interval is sampled only once. One could think of M repeated samples from the stoch. proc., e.g., $\{y_{tm}\}_{m=1}^{M}$ By a LLN, we would expect that

$$\frac{1}{M}\sum_{m=1}^{M} y_{tm} \overset{p}{\to} \mathcal{E}(Y_t)$$

as M gets large. The problem is, we have only one sample to work with, since we can't go back in time and collect another. How can $\mathcal{E}(Y_t)$ be estimated then? It turns out that **ergodicity** is the needed property.

Definition (Ergodicity) A stationary stochastic process is ergodic (for the mean) if the time average converges to the mean

$$\frac{1}{n}\sum_{t=1}^{n} y_t \overset{p}{\to} \mu \tag{1}$$

A sufficient condition for ergodicity is that the autocovariances be absolutely summable:

$$\sum_{j=0}^{\infty} |\gamma_j| < \infty$$

This implies that the autocovariances die off.

Definition (Autocorrelation) The j^{th} autocorrelation, ρ_j is just the j^{th} autocovariance divided by the variance:

$$\rho_j = \frac{\gamma_j}{\gamma_0} \tag{2}$$

Definition (White noise) White noise is just the time series literature term for a classical error. ϵ_t is white noise if i) $\mathcal{E}(\epsilon_t) = 0, \forall t$, ii) $V(\epsilon_t) = \sigma^2, \forall t$ and iii) ϵ_t and ϵ_s are independent, $t \neq s$. Gaussian white noise just adds a normality assumption.

AR(p)

An AR(p) process can be represented as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

where ϵ_t is white noise. This is just a linear regression model, and assuming stationarity, we can estimate the parameters by OLS.

What is needed for stationarity?

The dynamic behavior of an AR(p) process can be studied by writing this p^{th} order difference equation as a vector first order difference equation (this is known as the companion form):

$$\begin{bmatrix} y_t \\ y_{t-1} \\ \vdots \\ y_{t-p+1} \end{bmatrix} = \begin{bmatrix} c \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} \phi_1 & \phi_2 & \cdots & & \phi_p \\ 1 & 0 & 0 & & 0 \\ 0 & 1 & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \cdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_{t-1} \\ y_{t-2} \\ \vdots \\ y_{t-p} \end{bmatrix} + \begin{bmatrix} \varepsilon_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

or

$$Y_t = C + FY_{t-1} + E_t$$

With this, we can recursively work forward in time:

$$\begin{aligned} Y_{t+1} &= C + FY_t + E_{t+1} \\ &= C + F(C + FY_{t-1} + E_t) + E_{t+1} \\ &= C + FC + F^2 Y_{t-1} + FE_t + E_{t+1} \end{aligned}$$

and

$$\begin{aligned} Y_{t+2} &= C + FY_{t+1} + E_{t+2} \\ &= C + F\left(C + FC + F^2 Y_{t-1} + FE_t + E_{t+1}\right) + E_{t+2} \\ &= C + FC + F^2 C + F^3 Y_{t-1} + F^2 E_t + FE_{t+1} + E_{t+2} \end{aligned}$$

or in general

$$Y_{t+j} = C + FC + \cdots + F^j C + F^{j+1} Y_{t-1} + F^j E_t + F^{j-1} E_{t+1} + \cdots + FE_{t+j-1} + E_{t+j}$$

Consider the impact of a shock in period t on y_{t+j}. This is simply

$$\frac{\partial Y_{t+j}}{\partial E_t'}_{(1,1)} = F^j_{(1,1)}$$

If the system is to be stationary, then as we move forward in time this impact must die off. Otherwise a shock causes a permanent change in the mean of y_t. Therefore, stationarity requires that

$$\lim_{j \to \infty} F^j_{(1,1)} = 0$$

Save this result, we'll need it in a minute.

Consider the eigenvalues of the matrix F. These are the λ such that

$$|F - \lambda I_P| = 0$$

The determinant here can be expressed as a polynomial. For example, for $p = 1$, the matrix F is simply

$$F = \phi_1$$

so

$$|\phi_1 - \lambda| = 0$$

can be written as

$$\phi_1 - \lambda = 0$$

When $p = 2$, the matrix F is

$$F = \begin{bmatrix} \phi_1 & \phi_2 \\ 1 & 0 \end{bmatrix}$$

so

$$F - \lambda I_P = \begin{bmatrix} \phi_1 - \lambda & \phi_2 \\ 1 & -\lambda \end{bmatrix}$$

and

$$|F - \lambda I_P| = \lambda^2 - \lambda\phi_1 - \phi_2$$

So the eigenvalues are the roots of the polynomial

$$\lambda^2 - \lambda\phi_1 - \phi_2$$

which can be found using the quadratic equation. This generalizes. For a p^{th} order AR process, the eigenvalues are the roots of

$$\lambda^p - \lambda^{p-1}\phi_1 - \lambda^{p-2}\phi_2 - \cdots - \lambda\phi_{p-1} - \phi_p = 0$$

Supposing that all of the roots of this polynomial are distinct, then the matrix F can be factored as, using Eigen decomposition,}

$$F = T\Lambda T^{-1}$$

where T is the matrix which has as its columns the eigenvectors of F, and Λ is a diagonal matrix with the eigenvalues on the main diagonal. Using this decomposition, we can write

$$F^j = \left(T\Lambda T^{-1}\right)\left(T\Lambda T^{-1}\right)\cdots\left(T\Lambda T^{-1}\right)$$

where $T\Lambda T^{-1}$ is repeated j times. This gives

$$F^j = T\Lambda^j T^{-1}$$

and

$$\Lambda^j = \begin{bmatrix} \lambda_1^j & 0 & & 0 \\ 0 & \lambda_2^j & & \\ & & \ddots & \\ 0 & & & \lambda_p^j \end{bmatrix}$$

Supposing that the $\lambda_i \; i = 1, 2, \ldots, p$ are all real valued, it is clear that

$$\lim_{j \to \infty} F^j_{(1,1)} = 0$$

requires that

$$|\lambda_i| < 1, i = 1, 2, \ldots, p$$

e.g., the eigenvalues must be less than one in absolute value.

- It may be the case that some eigenvalues are complex-valued. The previous result generalizes to the requirement that the eigenvalues be less than one in **modulus**, where the modulus of a complex number $a + bi$ is

  $$mod(a + bi) = \sqrt{a^2 + b^2}$$

  This leads to the famous statement that "stationarity requires the roots of the determinantal polynomial to lie inside the complex unit circle."

- When there are roots on \ the unit circle (unit roots) or outside the unit circle, we leave the world of stationary processes.

- Dynamic multipliers: $\partial y_{t+j}/\partial \varepsilon_t = F^j_{(1,1)}$ is a \emph{dynamic multiplier} or an **impulse-response** function. Real eigenvalues lead to steady movements, whereas complex eigenvalues lead to oscillatory behavior. Of course, when there are multiple eigenvalues the overall effect can be a mixture.

Moments of AR(p) process

The AR(p) process is

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Assuming stationarity, $\mathcal{E}(y_t) = \mu, \forall t$, so

$$\mu = c + \phi_1 \mu + \phi_2 \mu + \ldots + \phi_p \mu$$

so

$$\mu = \frac{c}{1 - \phi_1 - \phi_2 - \ldots - \phi_p}$$

and

$$c = \mu - \phi_1 \mu - \ldots - \phi_p \mu$$

so

$$\begin{aligned} y_t - \mu &= \mu - \phi_1 \mu - \ldots - \phi_p \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t - \mu \\ &= \phi_1 (y_{t-1} - \mu) + \phi_2 (y_{t-2} - \mu) + \ldots + \phi_p (y_{t-p} - \mu) + \varepsilon_t \end{aligned}$$

With this, the second moments are easy to find: The variance is

$$\gamma_0 = \phi_1 \gamma_1 + \phi_2 \gamma_2 + \ldots + \phi_p \gamma_p + \sigma^2$$

The autocovariances of orders $j \geq 1$ follow the rule

$$\begin{aligned} \gamma_j &= \mathcal{E}\left[(y_t - \mu)\left(y_{t-j} - \mu\right))\right] \\ &= \mathcal{E}\left[(\phi_1 (y_{t-1} - \mu) + \phi_2 (y_{t-2} - \mu) + \ldots + \phi_p (y_{t-p} - \mu) + \varepsilon_t)\left(y_{t-j} - \mu\right)\right] \\ &= \phi_1 \gamma_{j-1} + \phi_2 \gamma_{j-2} + \ldots + \phi_p \gamma_{j-p} \end{aligned}$$

Using the fact that $\gamma_{-j} = \gamma_j$, one can take the $p+1$ equations for $j = 0, 1, \ldots, p$, which have $p+1$ unknowns (σ^2, $\gamma_0, \gamma_1, \ldots, \gamma_p$) and solve for the unknowns. With these, the γ_j for $j > p$ can be solved for recursively.

MA(q)

A q^{th} order moving average (MA) process is

$$y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where ε_t is white noise. The variance is

$$\begin{aligned}\gamma_0 &= \mathcal{E}(y_t-\mu)^2\\ &= \mathcal{E}(\varepsilon_t+\theta_1\varepsilon_{t-1}+\theta_2\varepsilon_{t-2}+\cdots+\theta_q\varepsilon_{t-q})^2\\ &= \sigma^2\left(1+\theta_1^2+\theta_2^2+\cdots+\theta_q^2\right)\end{aligned}$$

Similarly, the autocovariances are

$$\begin{aligned}\gamma_j &= \mathcal{E}\left[(y_t-\mu)\left(y_{t-j}-\mu\right)\right]\\ &= \sigma^2(\theta_j+\theta_{j+1}\theta_1+\theta_{j+2}\theta_2+\cdots+\theta_q\theta_{q-j}), j\le q\\ &= 0, j>q\end{aligned}$$

Therefore an MA(q) process is necessarily covariance stationary and ergodic, as long as σ^2 and all of the θ_j are finite.

For example, if we have an MA(1) model, then $E(y_t)=\mu$, $V(y_t)=\sigma^2(1+\theta_1^2)$, and $\gamma_1=\sigma^2\theta_1$. The higher order autocovariances are zero.

- Thus, if the model is MA(1) with normally distributed shocks, the density of the vector of n observations, y, is

$$f_Y(y|\rho)=\frac{1}{\sqrt{(2\pi)^n\,|\Sigma|}}\exp\left(-\frac{1}{2}(y-\mu)'\Sigma^{-1}(y-\mu)\right) \tag{3}$$

  where

$$\Sigma=\sigma^2\begin{bmatrix}1+\theta_1^2 & \theta_1 & 0 & \cdots & 0\\ \theta_1 & \ddots & \ddots & \ddots & \vdots\\ 0 & \ddots & \ddots & \ddots & 0\\ \vdots & \ddots & \ddots & \ddots & \theta_1\\ 0 & \cdots & 0 & \theta_1 & 1+\theta_1^2\end{bmatrix}.$$

  With this, it is very easy to program the log-likelihood function. For higher order MA models, the only difference is the structure of Σ becomes more complicated. In this form, one needs a lot of computer memory. A more economical approach uses the Kalman filter, which we'll see in the discussion of state space models.

- If we don't make assumptions on the distribution of the shocks, then method of moments estimation can be used.

Notes:

- An issue to be aware of is that MA models are not identified, in that there exist multiple parameter values that give the same value of the likelihood function.
- For example, the MA(1) model with $\tilde{\sigma}^2=\theta^2\sigma^2$ and $\tilde{\theta}_1=\frac{1}{\theta_1}$ has identical first and second moments to the original model, so the likelihood function has the same value.
- Normally, the parameterization that leads to an invertible MA model is the one that is selected. An invertible MA model is one that has a representation as a AR(∞) model. For the MA(1)

model, the invertible parameterization has $|\theta_1| < 1$.

- This implies that parameter restrictions will need to be imposed when estimating the MA model, to enforce selection of the invertible model.
- Maximization of the conditional likelihood is also used for estimation, sometimes. Assuming that ϵ_0 is known (for example, equal to zero), then one can compute ϵ_1, given the parameters. Then one works forward recursively to get all of the ϵ_t. With these, the likelihood function is very easy to compute. This is a convenient shortcut, but it's not recommended if the sample is not large, especially since it's not hard to compute the exact likelihood function.

ARMA(p,q) Model

An ARMA(p, q) model is $(1 + \phi_1 L + \phi_2 L^2 + \ldots + \phi_p L^p) y_t = c + (1 + \theta_1 L + \theta_2 L^2 + \ldots + \theta_q L^q)\epsilon_t$. These are popular in applied time series analysis. A high order AR process may be well approximated by a low order MA process, and a high order MA process may be well approximated by a low order AR process. By combining low order AR and MA processes in the same model, one can hope to fit a wide variety of time series using a parsimonious number of parameters. There is much literature on how to choose p and q, which is outside the scope of this course. Estimation can be done using the Kalman filter, assuming that the errors are normally distributed.

ARCH and GARCH

ARCH (autoregressive conditionally heteroscedastic) models appeared in the literature in 1982, in Engle, Robert F. (1982). Autoregressive Conditional Heteroskedasticity with Estimates of Variance of United Kingdom Inflation, Econometrica 50:987-1008. This paper stimulated a very large growth in the literature for a number of years afterward. The related GARCH (generalized ARCH) model is now one of the most widely used models for financial time series.

Financial time series often exhibit several type of behavior:

- volatility clustering: periods of low variation can be followed by periods of high variation
- fat tails, or excess kurtosis,: the marginal density of a series is more strongly peaked and has fatter tails than does a normal distribution with the same mean and variance.
- leverage (negative correlation between returns and volatility), which often shows up as negative skewness of returns
- perhaps slight autocorrelation within the bounds allowed by arbitrage

The data set "nysewk" provides an example. If we compute 100 times the growth rate of the series, using log differences, we can obtain the plots below:

In [21]:
```
# weekly close price of NSYE, data provided with GRETL

data = DataFrame(CSV.File("../data/nysewk.csv",header=false))
first(data,6)
```

Out[21]: 6 rows × 1 columns

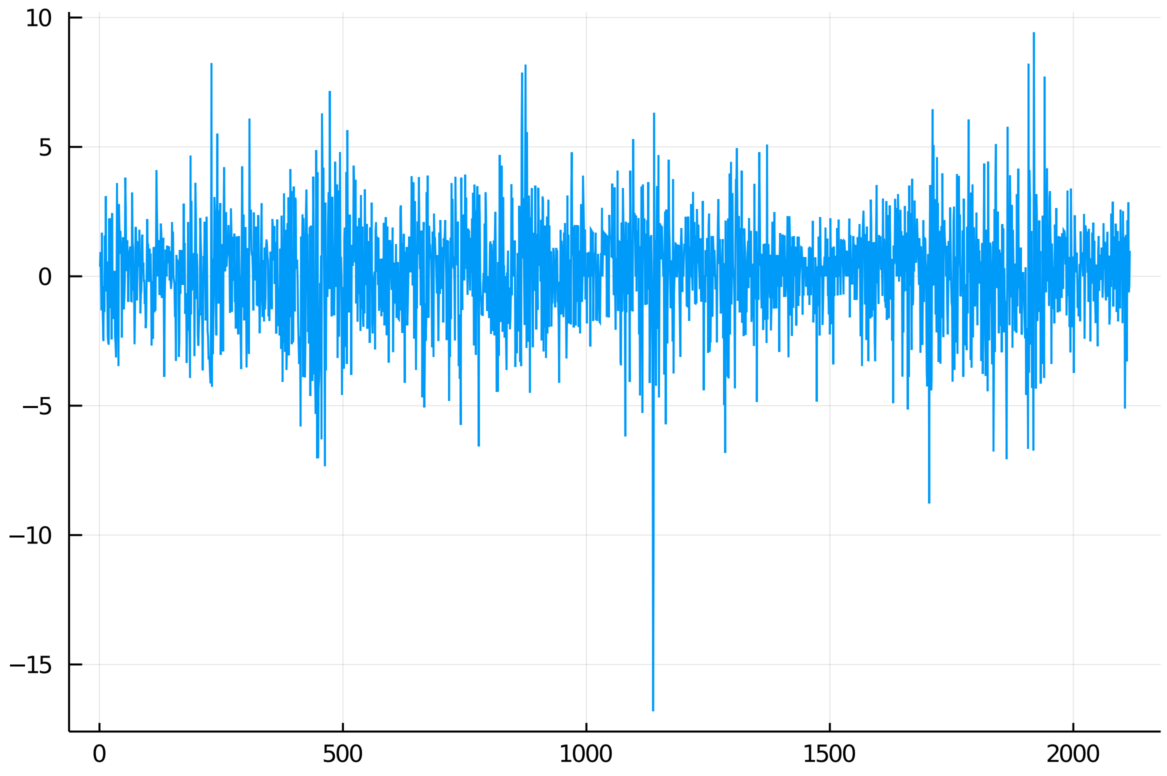
Column1
Float64

	Column1
	Float64
1	531.0
2	533.0
3	538.0
4	538.0
5	531.0
6	540.0

In [23]:
```
# compute weekly percentage growth
data = convert(Array,data)
y = 100.0 * log.(data[2:end] ./ data[1:end-1]);
```

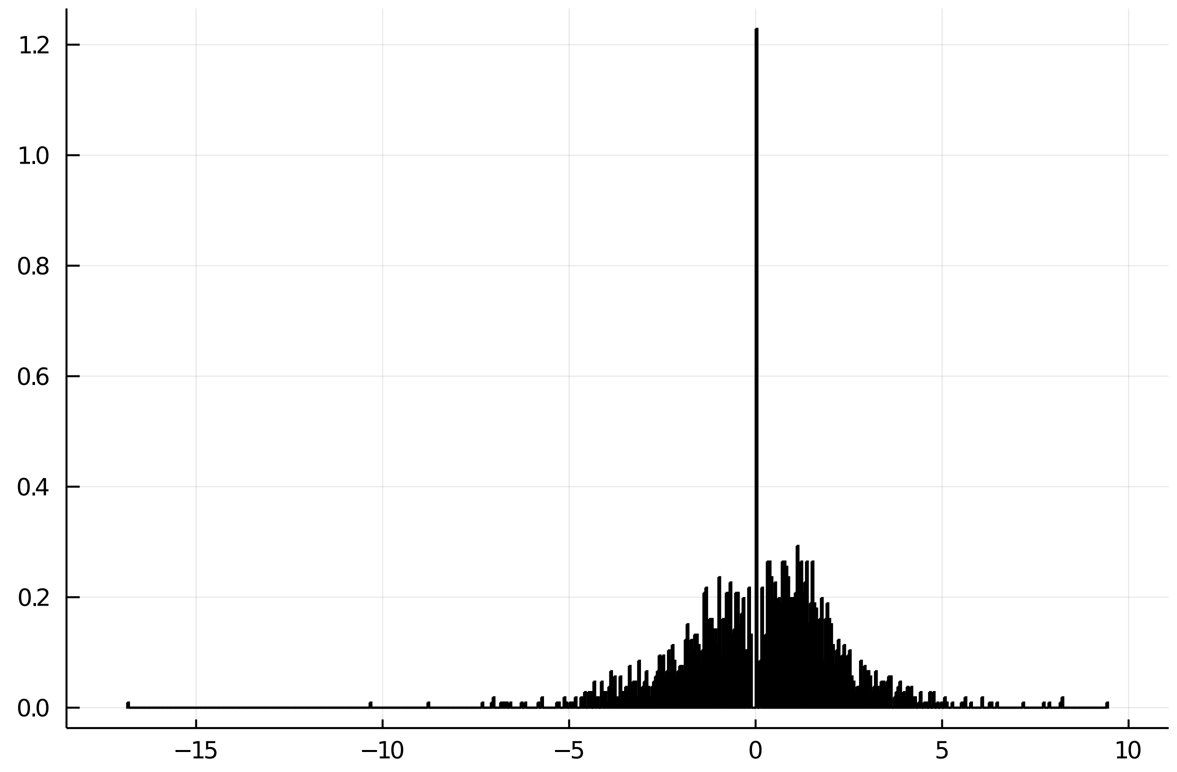
In [6]:
```
p1 = plot(y,legend=false)
```

Out[6]:



In [12]:
```
p2 = histogram(y, normed=true,legend=false,bins=500)
```

Out[12]:



In the first we clearly see volatility clusters, and in the second, we see excess kurtosis, skew, and tails fatter than the normal distribution.

ARCH

A basic Autoregressive conditional heteroskedasticity (ARCH) specification is

$$\begin{aligned} y_t &= \mu + \rho y_{t-1} + \epsilon_t \\ &\equiv g_t + \epsilon_t \\ \epsilon_t &= \sigma_t u_t \\ \sigma_t^2 &= \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 \end{aligned}$$

where the u_t are Gaussian white noise shocks. The ARCH variance is a moving average process. Previous large shocks to the series cause the conditional variance of the series to increase. There is no leverage: negative shocks have the same impact on the future variance as do positive shocks..

- for σ_t^2 to be positive for all realizations of $\{\epsilon_t\}$, we need $\omega > 0$, $\alpha_i \geq 0$, $\forall i$.
- to ensure that the model is covariance stationary, we need $\sum_i \alpha_i < 1$. Otherwise, the variances will explode off to infinity.

To find the likelihood in terms of the observable y_t instead of the unobservable ϵ_t, first note that the series $u_t = (y_t - g_t) / \sigma_t = \frac{\epsilon_t}{\sigma_t}$ is iid Gaussian, so the likelihood is simply the product of standard normal densities.

$$u \sim N(0, I), \text{ so}$$
$$f(u) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u_t^2}{2}\right)$$

The joint density for y can be constructed using a change of variables:

We have $u_t = (y_t - \mu - \rho y_{t-1}) / \sigma_t$, so $\frac{\partial u_t}{\partial y_t} = \frac{1}{\sigma_t}$ and $|\frac{\partial u}{\partial y'}| = \prod_{t=1}^{n} \frac{1}{\sigma_t}$, doing a change of variables,

$$f(y;\theta) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi}} \frac{1}{\sigma_t} \exp\left(-\frac{1}{2}\left(\frac{y_t - \mu - \rho y_{t-1}}{\sigma_t}\right)^2\right)$$

where θ is the vector of all parameters (the parameters in g_t, and the ω and alpha parameters of the ARCH specification. Taking logs,

$$\ln L(\theta) = -n \ln \sqrt{2\pi} - \sum_{t=1}^{n} \ln \sigma_t - \frac{1}{2} \sum_{t=1}^{n} \left(\frac{y_t - \mu - \rho y_{t-1}}{\sigma_t}\right)^2.$$

In principle, this is easy to maximize. Some complications can arise when the restrictions for positivity and stationarity are imposed. Consider a fairly short data series with low volatility in the initial part, and high volatility at the end. This data appears to have a nonstationary variance sequence. If one attempts to estimate and ARCH model with stationarity imposed, the data and the restrictions are saying two different things, which can make maximization of the likelihood function difficult.

GARCH

Note that an ARCH model specifies the variance process as a moving average. For the same reason that an ARMA model may be used to parsimoniously model a series instead of a high order AR or MA, one can do the same thing for the variance series. A basic GARCH(p,q) specification is

$$\begin{aligned} y_t &= \mu + \rho y_{t-1} + \epsilon_t \\ \epsilon_t &= \sigma_t u_t \\ \sigma_t^2 &= \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 \end{aligned}$$

The idea is that a GARCH model with low values of p and q may fit the data as well or better than an ARCH model with large q. The model also requires restrictions for positive variance and stationarity, which are:

- $\omega > 0$
- $\alpha_i \geq 0, \, i = 1, \ldots, q$
- $\beta_i \geq 0, \, i = 1, \ldots, p$
- $\sum_{i=1}^{q} \alpha_i + \sum_{i=1}^{p} \beta_i < 1.$

To estimate a GARCH model, you need to initialize σ_0^2 at some value. The sample unconditional variance is one possibility. Another choice could be the sample variance of the initial elements of the sequence. One can also ''backcast'' the conditional variance.

Notes:

- The GARCH model also requires restrictions on the parameters to ensure stationarity and positivity of the variance.
- A useful modification is the EGARCH model (exponential GARCH). This model treats the logarithm of the variance as an ARMA process, so the variance will be positive without restrictions on the parameters.
- There are many variants that introduce asymmetry (leverage) and non-normality.
- GARCH(1,1) is a highly popular model in financial analysis.

In [18]:
```
function garch11(theta, y)
    # dissect the parameter vector
    mu = theta[1]
    rho = theta[2]
    omega = theta[3]
    alpha = theta[4]
    beta = theta[5]
    resid = y[2:end] .- mu .- rho*y[1:end-1]
    n = size(resid,1)
    h = zeros(n)
    # initialize variance; either of these next two are reasonable choices
    h[1] = var(y[1:10])
    #h[1] = var(y)
    rsq = resid.^2.0
    for t = 2:n
        h[t] = omega + alpha*rsq[t-1] + beta*h[t-1]
    end
    logL = -log(sqrt(2.0*pi)) .- 0.5*log.(h) .- 0.5*rsq./h
end
```

Out[18]:
```
garch11 (generic function with 1 method)
```

In [24]:
```
thetastart = [mean(y); 0.0; var(y); 0.1; 0.1]
obj = theta -> -sum(garch11(theta, y))
thetahat, logL, junk  = fmincon(obj, thetastart, [], [], [-Inf, -1.0, 0.0, 0.0, 0.0], [
```

Out[24]:
```
([0.17509446603259488, -0.0017974081950996854, 0.1565082410267794, 0.11201475043965671,
0.8548995338432239], 4402.20524546603, :FTOL_REACHED)
```

In [25]:
```
model = θ -> garch11(θ, y)
avg_obj = θ -> -mean(vec(model(θ))) # average log likelihood
thetahat, objvalue, converged = fminunc(avg_obj, thetahat) # do the minimization of -lo
```

Out[25]:
```
([0.17509289904293293, -0.0017974048437616162, 0.15650738859226337, 0.11200543356269112,
0.8549074823073503], 2.081420919751293, true)
```

How do you select p and q?

Model Selection

Generally, as you add more variables to a regression, the bias of the predictions decreases and the variance increases. Too few covariates yields high bias; this called **underfitting**. Too many covariates yields high variance; this called **overfitting**. Good predictions result from achieving a good balance between bias and variance.

In model selection there are two problems:

1. assigning a "score" to each model which measures, in some sense, how good the model is, and
2. searching through all the models to find the model with the best score.

Let $S \subset 1, \ldots, k$ and let $\mathcal{X}_S = X_j : j \in S$ denote a subset of the covariates. Let β_S denote the coefficients of the corresponding set of covariates. Define $\hat{r_S}(x)$ to be the estimated regression function. The fitted/predicted values from model S are denoted by $\hat{Y}_i(S) = \hat{r_S}(X_i)$. The **prediction risk** is defined to be

$$R(S) = \sum_{i=1}^{n} \mathbb{E}(\hat{Y}_i(S) - Y_i^*)^2$$

where Y_i^* denotes the value of a future observation of Y_i at covariate value X_i. Our goal is to choose S to make $R(S)$ small.

The **training error** is defined to be

$$\hat{R_{tr}}(S) = \sum_{i=1}^{n} \mathbb{E}(\hat{Y}_i(S) - Y_i)^2$$

Finding a good model involves trading off fit and complexity.

AIC (Akaike Information Criterion)

The idea is to choose S to maximize:

$$AIC(S) = l_S - |S|$$

where l_S is the log-likelihood of the model evaluated at the MLE. This can be thought of "goodness of fit" minus "complexity."

BIC (Bayesian information criterion)

$$BIC(S) = l_S - \frac{|S|}{2} \log n$$

The BIC score has a Bayesian interpretation. Let $S = S_1, \ldots, S_m$ denote a set of models. Suppose we assign the prior $P(S_j) = 1/m$ over the models. Also, assume we put a smooth prior on the parameters within each model. It can be shown that the posterior probability for a model is approximately,

$$P(S_j|\text{data}) \approx \frac{e^{BIC(S_j)}}{\sum_r e^{BIC(S_r)}}$$

Hence, choosing the model with highest BIC is like choosing the model with highest posterior probability. The BIC score also has an information-theoretic interpretation in terms of something called minimum description length.

Leave-one-out cross-validation

The risk estimator is

$$\hat{R}_{CV}(S) = \sum_{i=1}^{n}(Y_i - \hat{Y_{(i)}})^2$$

where $\hat{Y_{(i)}}$ is the prediction for Y_i obtained by fitting the model Y_i with omitted. It can be shown that

$$\hat{R}_{CV}(S) = \sum_{i=1}^{n}\left(\frac{Y_i - \hat{Y}_i(S)}{1 - U_{ii}(S)}\right)^2$$

where $U_{ii}(S)$ is the i^{th} diagonal element of the matrix

$$U(S) = X_S(X_S^T X_S)^{-1} X_S^T$$

Thus, one need not actually drop each observation and re-fit the model. A generalization is **k-fold cross-validation**. Here we divide the data into k groups; often people take k = 10. We omit one group of data and fit the models to the remaining data. We use the fitted model to predict the data in the group that was omitted. We then estimate the risk by $\sum_i (Y_i - \hat{Y}_i)^2$ where the sum is over the the data points in the omitted group. This process is repeated for each of the k groups and the resulting risk estimates are averaged.

Now let us turn to the problem of model search. If there are k covariates then there are 2^k possible models. We need to search through all these models, assign a score to each one, and choose the model with the best score. If k is not too large we can do a complete search over all the models. When k is large, this is infeasible. In that case we need to search over a subset of all the models. Two common methods are forward and backward stepwise regression. In forward stepwise regression, we start with no covariates in the model. We then add the one variable that leads to the best score. We continue adding variables one at a time until the score does not improve. Backwards stepwise regression is the same except that we start with the biggest model and drop one variable at a time. Both are greedy searches; nether is guaranteed to find the model with the best score. Another popular method is to do random searching through the set of all models. However, there is no reason to expect this to be superior to a deterministic search.

Function Load:

In [14]:
```
function fminunc(obj, x; tol = 1e-10)
    results = Optim.optimize(obj, x, LBFGS(),
                             Optim.Options(
                             g_tol = tol,
                             x_tol=tol,
                             f_tol=tol))
    return results.minimizer, results.minimum, Optim.converged(results)
    #xopt, objvalue, flag = fmincon(obj, x, tol=tol)
    #return xopt, objvalue, flag
end
```

Out[14]:
```
fminunc (generic function with 1 method)
```

In [15]:
```
function fmincon(obj, startval, R=[], r=[], lb=[], ub=[]; tol = 1e-10, iterlim=0)
    # the objective is an anonymous function
    function objective_function(x::Vector{Float64}, grad::Vector{Float64})
        obj_func_value = obj(x)[1,1]
        return(obj_func_value)
    end
    # impose the linear restrictions
    function constraint_function(x::Vector, grad::Vector, R, r)
        result = R*x .- r
        return result[1,1]
    end
    opt = Opt(:LN_COBYLA, size(startval,1))
    min_objective!(opt, objective_function)
    # impose lower and/or upper bounds
    if lb != [] lower_bounds!(opt, lb) end
    if ub != [] upper_bounds!(opt, ub) end
    # impose linear restrictions, by looping over the rows
    if R != []
        for i = 1:size(R,1)
            equality_constraint!(opt, (theta, g) -> constraint_function(theta, g, R[i:i
        end
    end
    xtol_rel!(opt, tol)
    ftol_rel!(opt, tol)
    maxeval!(opt, iterlim)
    (objvalue, xopt, flag) = NLopt.optimize(opt, startval)
    return xopt, objvalue, flag
end
```

Out[15]:
```
fmincon (generic function with 5 methods)
```