

Time Series Econometrics

- Typical data that has been covered up to this point:
 - Cross-sectional data - many individuals at one point in time
 - Panel data - many individuals sampled repeatedly
- Time series data consists of one individual observed in multiple periods
- Why would we step back from panel data to study time series data?
 - Some "individuals" are singular (eg. global temperatures)
 - Decomposing an individual time-series can be helpful for analyses
- One way to look at time-series is the following
 - There are a variety of models that can explain the data. We wish to find the model of best fit without going overboard.
 - For example: Are observables correlated over time, or are unobservables correlated over time (or both)?

ARIMA models

- ARIMA models are a general form of time-series model that incorporate three features:
- **AR:** Autoregressive
 - Outcomes are explicitly correlated over-time
 - GDP this year depends on GDP last year
- **I:** Integrated
 - Not stationary series - drift in average of the outcome
- **MA:** Moving Average
 - The average of an outcome in a current period is a weighted sum of past noise.

Stationarity

- A time series is considered *covariance stationary* if
 - mean reversion around a constant long-run mean
 - finite variance that is time invariant
 - a theoretical correlogram that decreases in lag length (will explain this later)
- A key idea here is that shocks to a stationary system will only be temporary.
 - Without these assumptions, it's very hard to characterize the properties of a variable, as we'll see.
- For these slides the key assumption for our derivations is that all unobserved shocks are Gaussian white noise
 - Normally distributed with **mean zero** and **constant variance**.

The AR(1) process

- The basic AR(1) process is written as follows:

$$Y_t = \phi Y_{t-1} + u_t$$

- Y_t is the observed time-series variable at time t
 - u_t is the unobserved shock at time t - gaussian white noise
 - ϕ determines how much of Y_t is based on past values
- The key (necessary) condition for stationarity is $|\phi| < 1$. To see this, solve for the variance of Y_t and simplify:

$$\begin{aligned} \text{Var}(Y_t) &= \text{Var}(\phi Y_{t-1}) + \text{Var}(u_t) \\ &= \phi^2 \text{Var}(Y_{t-1}) + \text{Var}(u_t) \\ &= \phi^2 \text{Var}(Y_t) + \text{Var}(u_t) \\ \Rightarrow \text{Var}(Y_t) &= \frac{\text{Var}(u_t)}{(1 - \phi^2)} \end{aligned}$$

- Variance of Y_t explodes as ϕ^2 approaches 1.

Code to generate an AR(1) process

- First, create a function to generate an AR1 process

```
AR1<-function(n,phi) {  
  es<-rnorm(n)  
  Y<-rep(0,n)  
  for(i in 2:length(Y)) {  
    Y[i]<-phi*Y[i-1]+es[i]  
  }  
  return(Y)  
}
```

- Plot different time series of length 100 with different ϕ parameters.

```
par(mfrow=c(2,3))  
plot(AR1(100,0.1),type='l',main="phi=0.1",xlab='t',ylab="Y")  
plot(AR1(100,0.5),type='l',main="phi=0.5",xlab='t',ylab="Y")  
plot(AR1(100,0.9),type='l',main="phi=0.9",xlab='t',ylab="Y")  
plot(AR1(100,1),type='l',main="phi=1",xlab='t',ylab="Y")  
plot(AR1(100,1.5),type='l',main="phi=1.5",xlab='t',ylab="Y")  
plot(AR1(100,2),type='l',main="phi=2",xlab='t',ylab="Y")
```

- Try this repeatedly and see what happens to the non stationary plots

The AR(1) process and its properties

- For the stationary AR(1) process with Gaussian white noise for u_t it is straightforward to show that:

$$E(Y_t) = 0$$

- A key property is how the correlation between period t and $t - k$ values decay as k increases. The correlation that we seek is the following:

$$ACF = Cor(Y_t, Y_{t-k}) = \frac{Cov(Y_t, Y_{t-k})}{\sqrt{Var(Y_t)}\sqrt{Var(Y_{t-k})}}$$

- This metric is often called a "correlogram" when plotted against k .
- How do we refine the denominator?
- Since process is stationary, $Var(Y_t) = Var(Y_{t-k})$. So, we have:

$$ACF = \frac{Cov(Y_t, Y_{t-k})}{Var(Y_t)}$$

The AR(1) process and its properties (cont.)

- To simplify the numerator, note that:

$$\text{Cov}(Y_t, Y_{t-k}) = E[(Y_t - E(Y_t))(Y_{t-k} - E(Y_{t-k}))]$$

- Expanding:

$$\text{Cov}(Y_t, Y_{t-k}) = E[Y_t Y_{t-k} - E(Y_t)Y_{t-k} - E(Y_t)E(Y_{t-k}) + Y_t E(Y_{t-k})]$$

- Noting that $E(Y_t) = E(Y_{t-1}) = 0$, we have:

$$\text{Cov}(Y_t, Y_{t-k}) = E[Y_t Y_{t-k}]$$

- Now, we need to solve for Y_t precisely determine the covariance. Recursively using the AR(1) equation k times, we have:

$$\begin{aligned} Y_t &= \phi Y_{t-1} + u_t \\ &= \phi (\phi Y_{t-2} + u_{t-1}) + u_t \\ &= \phi (\phi (\phi Y_{t-3} + u_{t-2}) + u_{t-1}) + u_t \\ &\vdots \\ &= \phi^k Y_{t-k} + u_t + \phi u_{t-1} + \cdots + \phi^{k-2} u_{t-k+2} + \phi^{k-1} u_{t-k+1} \end{aligned}$$

The AR(1) process and its properties (cont.)

- Plugging in to $Cov(Y_t, Y_{t-k})$:

$$\begin{aligned} Cor(Y_t, Y_{t-k}) &= \phi^k E[Y_{t-k} Y_{t-k}] \\ &\quad + E[u_t Y_{t-k}] \\ &\quad + \phi E[u_{t-1} Y_{t-k}] \\ &\quad \vdots \\ &\quad + \phi^{k-2} E[u_{t-k+2} Y_{t-k}] \\ &\quad + \phi^{k-1} E[u_{t-k+1} Y_{t-k}] \end{aligned}$$

- Since future shocks do not determine past Y 's, we are left with:

$$Cor(Y_t, Y_{t-k}) = \phi^k E[Y_{t-k} Y_{t-k}]$$

- Finally, since $E[Y_{t-k}] = 0$, we have:

$$\begin{aligned} Cor(Y_t, Y_{t-k}) &= \phi^k E[(Y_{t-k} - E[Y_{t-k}])(Y_{t-k} - E[Y_{t-k}])] \\ &= \phi^k Var(Y_t) \end{aligned}$$

The AR(1) process and its properties (cont.)

- Overall, we have:

$$ACF = \frac{\phi^k \text{Var}(Y_t)}{\text{Var}(Y_t)} = \phi^k$$

- Since ϕ is between zero and one, ACF asymptotes to zero with higher k .
- Let's now construct some AR(1) processes in \mathbb{R} that are stationary and otherwise and construct their correlogram.

ACF of AR(1) processes in R

- The 'acf' function in R simply constructs and plots ACF functions

```
acf(series, lag.max=10, type="correlation")
```

- "series" is your data
 - "lag.max=10" is the number of lags you want on the plot
 - "type="correlation"" indicates that you want a correlogram
- Use our AR1 function from the previous example within this plotting function:

```
par(mfrow=c(2, 3))  
acf(AR1(100, 0.1), lag.max=10, type="correlation")  
acf(AR1(100, 0.5), lag.max=10, type="correlation")  
acf(AR1(100, 0.9), lag.max=10, type="correlation")  
acf(AR1(100, 1), lag.max=10, type="correlation")  
acf(AR1(100, 1.5), lag.max=10, type="correlation")  
acf(AR1(100, 2), lag.max=10, type="correlation")
```

- This plotting function will also indicate 2 standard deviation confidence intervals for a simple test of significant lags.

The AR(p) process

- The basic AR(p) process is simply an expanded AR(1) to include p lags of the dependent variable

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_{p-1} Y_{t-p+1} + \phi_p Y_{t-p} + u_t$$

- We can program an AR(p) process in R quite easily

```
ARp<-function(n,phi){  
  p<-length(phi)  
  es<-rnorm(n+p)  
  Y<-rep(0,n+p)  
  for(i in (p+1):length(Y)){  
    Y[i]<-t(phi)%*%Y[(i-p):(i-1)]+es[i]  
  }  
  Y<-Y[-(1:p)]  
  return(Y)  
}
```

The Moving Average Models

- MA(1): Current values are a function of current white noise and some function of the last period's white noise

$$Y_t = u_t + \theta u_{t-1}$$

- MA(q): Current values are a function of current white noise and some function prior period noise

$$\begin{aligned} Y_t &= u_t + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_{q-1} u_{t-q+1} + \theta_q u_{t-q} \\ &= u_t + \sum_{j=1}^q \theta_j u_{t-j} \end{aligned}$$

- MA(1) and AR(p) models are similar in the limit, which is a complication for estimation. We now take a look at this.

AR or MA? Theoretical identification issues

- Rearrange MA(1) for u_t .

$$u_t = Y_t - \theta u_{t-1}$$

- For u_{t-1} this is:

$$u_{t-1} = Y_{t-1} - \theta u_{t-2}$$

- Substitute repeatedly into $Y_t = u_t + \theta u_{t-1}$ for the lag error:

$$\begin{aligned} Y_t &= u_t + \theta (Y_{t-1} - \theta u_{t-2}) \\ &= u_t + \theta (Y_{t-1} - \theta (Y_{t-2} - \theta u_{t-3})) \\ &\quad \vdots \\ &= u_t + \theta Y_{t-1} + \theta^2 Y_{t-2} + \cdots + \theta^q Y_{t-q} + \cdots \end{aligned}$$

- But, an AR(p) looks like

$$Y_t = u_t + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_{p-1} Y_{t-p+1} + \phi_p Y_{t-p}$$

- Understanding the subtle differences is central to identification.

MA(1) correlogram

- We once again turn to evaluating the correlation of the observed value in time t with the observed value at time $t - k$.
- For AR(1), the decay of this correlation was log-linear with lag length k : ϕ^k
- To derive for MA(1), we note that:

$$ACF = \frac{Cov(Y_t, Y_{t-k})}{Var(Y_t)}$$

- The variance of Y_t is straightforward

$$\begin{aligned} Var(Y_t) &= Var(u_t) + Var(\theta u_{t-1}) \\ &= Var(u_t) + \theta^2 Var(u_t) \\ &= Var(u_t)(1 + \theta^2) \end{aligned}$$

- $Cov(Y_t, Y_{t-k})$ depends on the length of the lag.

MA(1) correlogram (cont.)

- For $k = 1$

$$\text{Cov}(Y_t, Y_{t-1}) = E[(Y_t - E(Y_t))(Y_{t-1} - E(Y_{t-1}))]$$

- Since our white noise processes are mean zero, so are $E(Y_t)$'s

$$\text{Cov}(Y_t, Y_{t-1}) = E[Y_t Y_{t-1}]$$

- Substituting using the MA(1) equation

$$\begin{aligned}\text{Cov}(Y_t, Y_{t-1}) &= E[(u_t + \theta u_{t-1})(u_{t-1} + \theta u_{t-2})] \\ &= E[u_t u_{t-1} + \theta u_{t-1} u_{t-1} + \theta u_t u_{t-2} + \theta^2 u_{t-1} u_{t-2}]\end{aligned}$$

- Distributing the expectation, $E[u_t u_{t-k}] = 0 \quad \forall k \neq 0$ since shocks are not serially correlated. Hence:

$$\begin{aligned}\text{Cov}(Y_t, Y_{t-1}) &= E[\theta u_{t-1} u_{t-1}] \\ &= \theta E[u_{t-1} u_{t-1}] \\ &= \theta \text{Var}(u_t)\end{aligned}$$

MA(1) correlogram (cont.)

- For $k > 1$

$$\text{Cov}(Y_t, Y_{t-k}) = E[(Y_t - E(Y_t))(Y_{t-k} - E(Y_{t-k}))] = E[Y_t Y_{t-k}]$$

- Substituting using the MA(1) equation

$$\begin{aligned}\text{Cov}(Y_t, Y_{t-k}) &= E[(u_t + \theta u_{t-1})(u_{t-k} + \theta u_{t-k-1})] \\ &= E[u_t u_{t-k} + \theta u_{t-1} u_{t-k} + \theta u_t u_{t-k-1} + \theta^2 u_{t-1} u_{t-k-1}]\end{aligned}$$

- Since $E[u_t u_{t-k}] = 0 \quad \forall k \neq 0$, we have that

$$\text{Cov}(Y_t, Y_{t-k}) = 0$$

- Overall, the ACF for the MA(1) process is written as:

$$\begin{aligned}\text{ACF} &= \frac{\theta^2}{(1 + \theta^2)} \quad \text{if } k = 1 \\ &= 0 \quad \text{if } k > 1\end{aligned}$$

- For AR processes, the correlations persist. For MA they vanish sharply after some period determined by the lag length of errors.

MA(p) process in R

- MA(p) process is similar in coding to the AR(p) process.

```
MAp<-function(n,theta) {  
  d<-length(theta)  
  es<-rnorm(n+d)  
  Y<-rep(0,n+d)  
  for(i in (d+1):length(Y)) {  
    Y[i]<-es[i]+t(theta)%*%es[(i-d):(i-1)]  
  }  
  Y<-Y[-(1:d)]  
  return(Y)  
}
```

- Then, generate a 3X2 correlograms similar to before

```
par(mfrow=c(2,3))  
acf(MAp(100,0.1),lag.max=10,type="correlation")  
acf(MAp(100,0.5),lag.max=10,type="correlation")  
acf(MAp(100,0.9),lag.max=10,type="correlation")  
acf(MAp(100,1),lag.max=10,type="correlation")  
acf(MAp(100,1.5),lag.max=10,type="correlation")  
acf(MAp(100,2),lag.max=10,type="correlation")
```

ARMA(p,q) and ARIMA(p,d,q)

- Not surprisingly, the ARMA model is the combination of AR(p) and MA(q) models

$$Y_t = u_t + \sum_{j=1}^q \theta_j u_{t-j} + \sum_{j=1}^p \phi_j Y_{t-j}$$

- ARIMA(p,d,q) models incorporate an "integrated" term, which is a trend in the average over time.
- Assuming δ is constant, the following model is integrated of order 1:

$$Y_t = \delta + u_t + \sum_{j=1}^q \theta_j u_{t-j} + \sum_{j=1}^p \phi_j Y_{t-j}$$

- To make stationary, take differences (using the Δ operator).

$$\Delta Y_t = \Delta u_t + \sum_{j=1}^q \theta_j \Delta u_{t-j} + \sum_{j=1}^p \phi_j \Delta Y_{t-j}$$

- the 'd' in ARIMA(p,d,q) is the number of times the data must be differenced to make stationary.

Estimating ARIMA models in R

- To estimate ARIMA(p,d,q) models in R, let's first load data with the "quantmod" package

```
install.packages("quantmod")  
library(quantmod)
```

- With quantmod, you can download stock price information from Yahoo finance.
- For example, downloading 7 years of data for google (before the stock split)

```
getSymbols("GOOG", from="2007-01-01", to="2014-01-01")  
prices<-GOOG$GOOG.Open  
rets <- dailyReturn(GOOG)
```

- Use str(GOOG) too see all the information after using getSymbols

Estimating ARIMA models in R

- Use the package "forecast" to provide the optimal ARIMA model

```
install.packages("forecast")  
library(forecast)
```

- The function "auto.arima" gives us the optimal ARIMA model
 - Tries to find the best fit without over parameterizing (eg. Penalized log-likelihood)

- Syntax is simple:

```
auto.arima(prices)  
auto.arima(rets)
```

- The function "forecast.arima" provides predictions with forecast errors.

```
google.arima<-auto.arima(prices)  
forecast.Arima(google.arima,40,level=c(80,95)),  
plot(google.forecast)
```

Modeling Variance - ARCH and GARCH

- Volatility is a key concept in finance
 - Calm vs. Turbulent periods of returns
 - Crucial for modeling and accounting for risk over a particular investing time horizon.
- Heteroskedastic errors are crucial for accounting for volatility.
 - Unlike the first lecture, the variance of white noise will not be constant over time
 - Obviously complicates many of our derivations.

ARCH models

- ARCH stands for "Autoregressive Conditional Heteroskedasticity"
- Consider the following simply time series model

$$Y_t = \beta_0 + \mathbf{X}_t\beta + u_t$$

- outcome variable Y_t
 - vector of explanatory variables \mathbf{X}_t .
 - u_t again is the error term.
- For a ARCH(1), process, we assume that

$$u_t | \Omega_t \sim N(0, h_t)$$

where $h_t = \gamma_0 + \gamma_1 u_{t-1}^2$

- For ARCH(1), the variance of the error term depends on a constant, γ_0 , and a function of the past squared error, $\gamma_1 u_{t-1}^2$.

ARCH (p) and testing

- The ARCH(p) model can be characterized as:

$$Y_t = \beta_0 + \mathbf{X}_t\beta + u_t$$

where $u_t | \Omega_t \sim N(0, h_t)$

$$h_t = \gamma_0 + \sum_{j=1}^p \gamma_j u_{t-j}^2$$

- Testing these models is potentially very simple:
 - $Y_t = \beta_0 + \mathbf{X}_t\beta + u_t$
 - Collect residuals, square them
 - Estimate $u_t^2 = \gamma_0 + \sum_{j=1}^p \gamma_j u_{t-j}^2 + w_t$
 - Evaluate parameters, conduct exclusion test

GARCH(p,q)

- The ARCH(p) in many ways is more of a moving average than autoregressive
 - There is no explicit persistence in h_t - just some lagged function of past observables.
- "Generalized Autoregressive Conditional Heteroskedasticity" allows for dependences on past unobservables, as well as persistence in the variance
- The GARCH(p,q) model is written as:

$$Y_t = \beta_0 + \mathbf{X}_t \beta + u_t$$

where $u_t | \Omega_t \sim N(0, h_t)$

$$h_t = \gamma_0 + \sum_{j=1}^p \delta_j h_{t-j} + \sum_{j=1}^q \gamma_j u_{t-j}^2$$

- \mathbf{X}_t could include lags of Y_t , or residuals like in the ARIMA framework.

Estimating ARCH(p) and GARCH(p,q) models

- In R, there are many functions to estimate ARCH and GARCH models.
- Unsurprisingly, the package "tseries" is one that contains many functions for time series analysis

- To demonstrate, download S&P500 data for 2001 to 2015

```
library(quantmod)
getSymbols('GSPC', from='2001-01-01', to='2015-01-01')
rets = dailyReturn(GSPC)
plot(rets)
```

- To make ourselves sick, let's plot the price series and daily returns.

```
par(mfrow=c(2,1))
plot(GSPC$GSPC.Open) plot(rets)
```

- Install and load the package "tseries" `install.packages("tseries")`

```
library(tseries)
```

Estimating ARCH(p) and GARCH(p,q) models

- The function "garch" estimates both ARCH and GARCH models, assuming no covariates in the regression equation
 - Thus, this is pure variance estimation
- Syntax is as follows

```
garch (data, order=c (p, q) )
```
- p is the GARCH component, and q is the ARCH component
- Using the data we downloaded, estimate a ARCH(1) model and summarize

```
sp500.g=garch (rets, order=c (0, 1) )
summary (sp500.g)
```
- Can predict bounds on values using the predict function

```
u=predict (sp500.g)
```
- This vector has two elements - upper and lower bounds

Estimating ARCH(p) and GARCH(p,q) models

- Can plot the data and estimated bounds using the following code

```
rets_upper<-rets
rets_upper$daily.returns<-u[,1]
rets_lower<-rets
rets_lower$daily.returns<-u[,2]
par(mfrow=c(2,1))
plot(rets, type="l", xlab="time", ylab="daily change",
main="SP500 index 2001-2015")
lines(rets_upper,col="red", lty="dashed",lwd=1.5)
lines(rets_lower,col="red", lty="dashed",lwd=1.5)
```

- What do you notice about this plot, and how might a GARCH model improve the simple ARCH framework?

Estimating ARCH(p) and GARCH(p,q) models

- Can plot the data and estimated bounds using the following code

```
sp500.g2=garch(rets, order=c(1,1))
u2=predict(sp500.g2)
rets_upper2<-rets
rets_upper2$daily.returns<-u2[,1]
rets_lower2<-rets
rets_lower2$daily.returns<-u2[,2]
```

- Finally, plot the old bounds (in black) against the new bounds (blue)

```
plot(rets_upper, type="l", xlab="time", ylab="daily change",
main="SP500 index 2001-2015", ylim=c(-.07, .07))
lines(rets_lower)
lines(rets_upper2, col="blue", lty="dashed", lwd=1)
lines(rets_lower2, col="blue", lty="dashed", lwd=1)
```

Estimating ARCH(p) and GARCH(p,q) models

- The package fGarch contains more bells and whistles
 - GARCH error estimation as above
 - ARMA effects in the original regression equation

- Install the package and load the library

```
install.packages("fGarch")  
library(fGarch)
```

- Syntax for GARCH estimation with ARMA components

```
garchFit(~arma(ar, ma) + garch(p, q), data, trace=FALSE)
```

- Let's run four different models to see how they look

```
garch1<-garchFit(~garch(1, 1), data=rets, trace=FALSE)  
garch2<-garchFit(~arma(1, 1) + garch(2, 1), data=rets, trace=FALSE)  
garch3<-garchFit(~arma(2, 1) + garch(2, 1), data=rets, trace=FALSE)  
garch4<-garchFit(~arma(2, 2) + garch(2, 2), data=rets, trace=FALSE)
```

Estimating ARCH(p) and GARCH(p,q) models

- Summarize works similarly to other regressions

```
summary(garch4)
```

- Predict generates predictions in 'n.ahead' future periods, as well as plots the last 'nx' observations

```
par(mfrow=c(2,2))
```

```
predict(garch1,n.ahead=10,plot=TRUE,nx=20)
```

```
predict(garch2,n.ahead=10,plot=TRUE,nx=20)
```

```
predict(garch3,n.ahead=10,plot=TRUE,nx=20)
```

```
predict(garch4,n.ahead=10,plot=TRUE,nx=20)
```

Estimating ARCH(p) and GARCH(p,q) models

- fGarch also has some cool plotting functions. The main feature is interactive

```
plot(garch4)
```

- Or, you can choose the different elements using "which"

```
par(mfrow=c(2,2))
```

```
plot(garch4, which=1)
```

```
plot(garch4, which=3)
```

```
plot(garch4, which=4)
```

```
plot(garch4, which=10)
```

GARCH-M Models

- In many models of finance, the expected return of an asset is some function of characteristics and then risk.
 - Risk is often viewed through the lens of standard deviation
- The GARCH-M Model allows for this explicit dependence:

$$Y_t = \beta_0 + \mathbf{X}_t\beta + \theta\sqrt{h_t} + u_t$$

where $u_t|\Omega_t \sim N(0, h_t)$

$$h_t = \gamma_0 + \sum_{j=1}^q \delta_j h_{t-j}^2 + \sum_{j=1}^p \gamma_j u_{t-j}^2$$

- Though $\sqrt{h_t}$ may look weird, I like it since it is on the same scale as the dependent variable.
 - Some prefer including variance h_t instead, so it's best to know the results from both.
- Finally, one can also add a set of regressors into the error equation as needed.