Time Series Analysis

Modeling Heteroskedasticity

Nicoleta Serban, Ph.D.
Professor
Stewart School of Industrial and Systems Engineering

GARCH Model: Data Example

About This Lesson

PDC Energy, Inc (PDCE)

Summary:
- Crude oil and natural gas producer with headquartered in Denver, Colorado
- PDC's portfolio is comprised of the Wattenberg Field in Colorado, the Delaware Basin in West Texas and the Utica Shale in Ohio

Time Series Data:
- Daily stock price for more than 12 years of data starting with January 2007
- Largely dependent on the crude oil price

Georgia Tech

ARMA-GARCH Fit

##Divide time series into training and testing
Predict August & September 1-16

pdcert2 = pdcert[-1]
n=length(pdcert2)
pdcert.test = pdcert2[3419:n]
pdcert.train = pdcert2[-c(3419:n)]

ARMA(4,4) & GARCH(1,1)

library(fGarch)
garchFit.ts = garchFit(~ arma(4,4)+ garch(1,1), data=pdcert.train, trace = FALSE)
fore_garch11 = predict(garchFit.ts, n.ahead = 32)

Prediction of 'garchFit' does not work properly when considering joint arma+garch

Georgia Tech

ARMA-GARCH Fit: Different Implementation

```
## ugrach from rugarch library (more computationally expensive)
## ARMA(4,4) & GARCH(1,1)
library(rugarch)
spec = ugarchspec(variance.model=list(garchOrder=c(1,1)),
                  mean.model=list(armaOrder=c(4,4), include.mean=T),
                  distribution.model="std")
fit = ugarchfit(spec, pdcert.train, solver = 'hybrid’)
fore = ugarchforecast(fit, n.ahead=32)
```

Prediction using ‘ugarchfit’ works properly when considering joint models

ARMA+GARCH Order Selection

ARMA & GARCH – simultaneous fit and model selection for efficient estimators of the model parameters.

ARMA(p,q)+GARCH(m,n) – computationally expensive to search over all combinations of (p,q)x(m,n) orders, hence apply a heuristic algorithm

- Step 1: Apply ARMA to model the time series and select orders ⇒ selected initial orders (p_0, q_0)
- Step 2: Apply ARMA(p_0, q_0)+GARCH(m,n) with varying m & n orders (consider only small values) ⇒ selected initial orders (m_0, n_0)
- Step 3: Apply ARMA(p,q)+GARCH(m_0, n_0) with varying p & q orders ⇒ selected initial orders (p_1, q_1)
- Step 4: Apply ARMA(p_1, q_1)+GARCH(m,n) with varying m & n orders (consider only small values) ⇒ selected initial orders (m_1, n_1)

Step 2: GARCH Order Selection

```
## R function to be used across multiple combinations of (m,n) orders
test_modelAGG <- function(m,n){
    spec <- ugarchspec(variance.model=list(garchOrder=c(m,n)),
                    mean.model=list(armaOrder=c(4,4),
                                include.mean=T),
                    distribution.model="std")
    fit <- ugarchfit(spec, pdcert.train, solver = 'hybrid')
    current.bic <- infocriteria(fit)[2]
    df <- data.frame(m,n,current.bic)
    names(df) <- c("m","n","BIC")
    print(paste(m,n,current.bic,sep=" "))
    return(df)
}
```

Fix the orders for modeling the conditional mean: ARMA(4,4)

Step 2: GARCH Order Selection (cont'd)

```
## Consider all combinations of m & n between 0 to 2
ordersAGG = data.frame(Inf,Inf,Inf)
names(ordersAGG) <- c("m","n","BIC")
for (m in 0:2){
  for (n in 0:2){
    possibleError <- tryCatch(
      ordersAGG<-rbind(ordersAGG,test_modelAGG(m,n)),
      error=function(e) e
    )
    if(inherits(possibleError, "error")) next
  }
}
ordersAGG <- ordersAGG[order(-ordersAGG$BIC),]
```

Selected Orders for modeling conditional variance: GARCH(2,2)

Step 3: ARMA Order Selection Update

```
## R function to be used across multiple combinations of (p,q) orders
test_modelAGA <- function(p,q){
    spec = ugarchspec(variance.model=list(garchOrder=c(2,2)),
                      mean.model=list(armaOrder=c(p,q),
                                      include.mean=T),
                      distribution.model="std")
    fit = ugarchfit(spec, pdcert.train, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(p,q,current.bic)
    names(df) <- c("p","q","BIC")
    print(paste(p,q,current.bic,sep=" "))
    return(df)
}
```

Fix the orders for modeling the conditional variance: GARCH(2,2)

Step 3: ARMA Order Selection Update (cont'd)

```
## Update the ARMA order
ordersAGA = data.frame(Inf,Inf,Inf)
names(ordersAGA) <- c("p","q","BIC")
for (p in 0:4){
    for (q in 0:4){
        possibleError <- tryCatch(
            ordersAGA<-rbind(ordersAGA,test_modelAGA(p,q)),
            error=function(e) e
        )
        if(inherits(possibleError, "error")) next
    }
}
ordersAGA <- ordersAGA[order(-ordersAGA$BIC),]
tail(ordersAGA)
```

Selected Orders for modeling the conditional mean: ARMA(0,0)

Choose: AR=0, MA=1 instead

Step 4: GARCH Order Selection Update

```
## R function to be used across multiple combinations of (m,n) orders
test_modelAGG <- function(m,n){
    spec = ugarchspec(variance.model=list(garchOrder=c(m,n)),
                mean.model=list(armaOrder=c(0,1),
               include.mean=T), distribution.model="std")
    fit = ugarchfit(spec, pdcert.train, solver = 'hybrid')
    current.bic = infocriteria(fit)[2]
    df = data.frame(m,n,current.bic)
    names(df) <- c("m","n","BIC")
    print(paste(m,n,current.bic,sep=" "))
    return(df)
}
.............
```

Fix the orders for modeling the conditional mean: ARMA(0,1)

Selected Orders for modeling conditional variance: GARCH(1,1)

Georgia Tech

ARMA+GARCH Fit: Model Evaluation

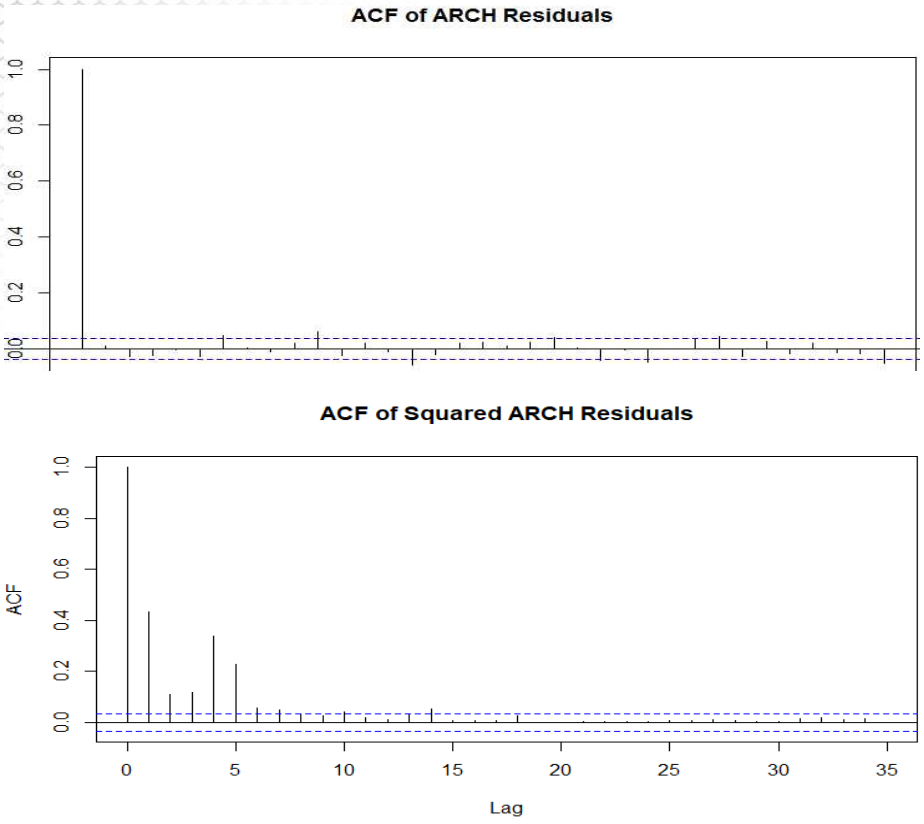
```
## Fit all three models and compare
spec.1 = ugarchspec(variance.model=list(garchOrder=c(2,2)),
                mean.model=list(armaOrder=c(4, 4),
                include.mean=T), distribution.model="std")
final.model.1 = ugarchfit(spec.1, pdcert.train, solver = 'hybrid')
……
## Compare Information Criteria
infocriteria(final.model.1)             ## ARMA(4,4)+GARCH(2,2)
infocriteria(final.model.2)             ## ARMA(0,1)+GARCH(2,2)
infocriteria(final.model.3)             ## ARMA(0,1)+GARCH(1,1)
```

ARMA+GARCH Fit: Model Evaluation (cont'd)

```
> ## compare Information Criteria
> infocri teria(final.model.1)

Akaike        - 4.060253
Bayes         - 4.033321
Shibata       - 4.060291
Hannan-Quinn  - 4.050629
> infocriteria(final.model.2)

Akaike        - 4.055557
Bayes         - 4.041193
Shibata       - 4.055568
Hannan-Quinn  - 4.050424
> infocriteria(final.model.3)

Akaike        - 4.056774
Bayes         - 4.046001
Shibata       - 4.056780
Hannan-Quinn  - 4.052925
```

All models perform similarly: choose least complex model

Residual Analysis



White Noise

Not White Noise

Forecasting: Mean & Volatility

```
## Prediction of the return time series and the volatility sigma
nfore = length(pdcert.test)
fore.series.1 = NULL
fore.sigma.1 = NULL
for(f in 1: nfore){
    ## Fit models
    data = pdcert.train
    if(f>2)
       data = c(pdcert.train,pdcert.test[1:(f-1)])
    final.model.1 = ugarchfit(spec.1, data, solver = 'hybrid')
   ## Forecast
    fore = ugarchforecast(final.model.1, n.ahead=1)
    fore.series.1 = c(fore.series.1, fore@forecast$seriesFor)
    fore.sigma.1 = c(fore.sigma.1, fore@forecast$sigmaFor)
    }
```

Georgia Tech

Prediction Accuracy Comparison

```
> ### Mean Squared Prediction Error (MSPE)
> mean((fore.series.1 - pdcert.test)^2)
[1] 0.003071104
> mean((fore.series.2 - pdcert.test)^2)
[1] 0.003134157
> mean((fore.series.3 - pdcert.test)^2)
[1] 0.003133087
> ### Mean Absolute Prediction Error (MAE)
> mean(abs(fore.series.1 - pdcert.test))
[1] 0.03535444
> mean(abs(fore.series.2 - pdcert.test))
[1] 0.03671894
> mean(abs(fore.series.3 - pdcert.test))
[1] 0.03671133
```

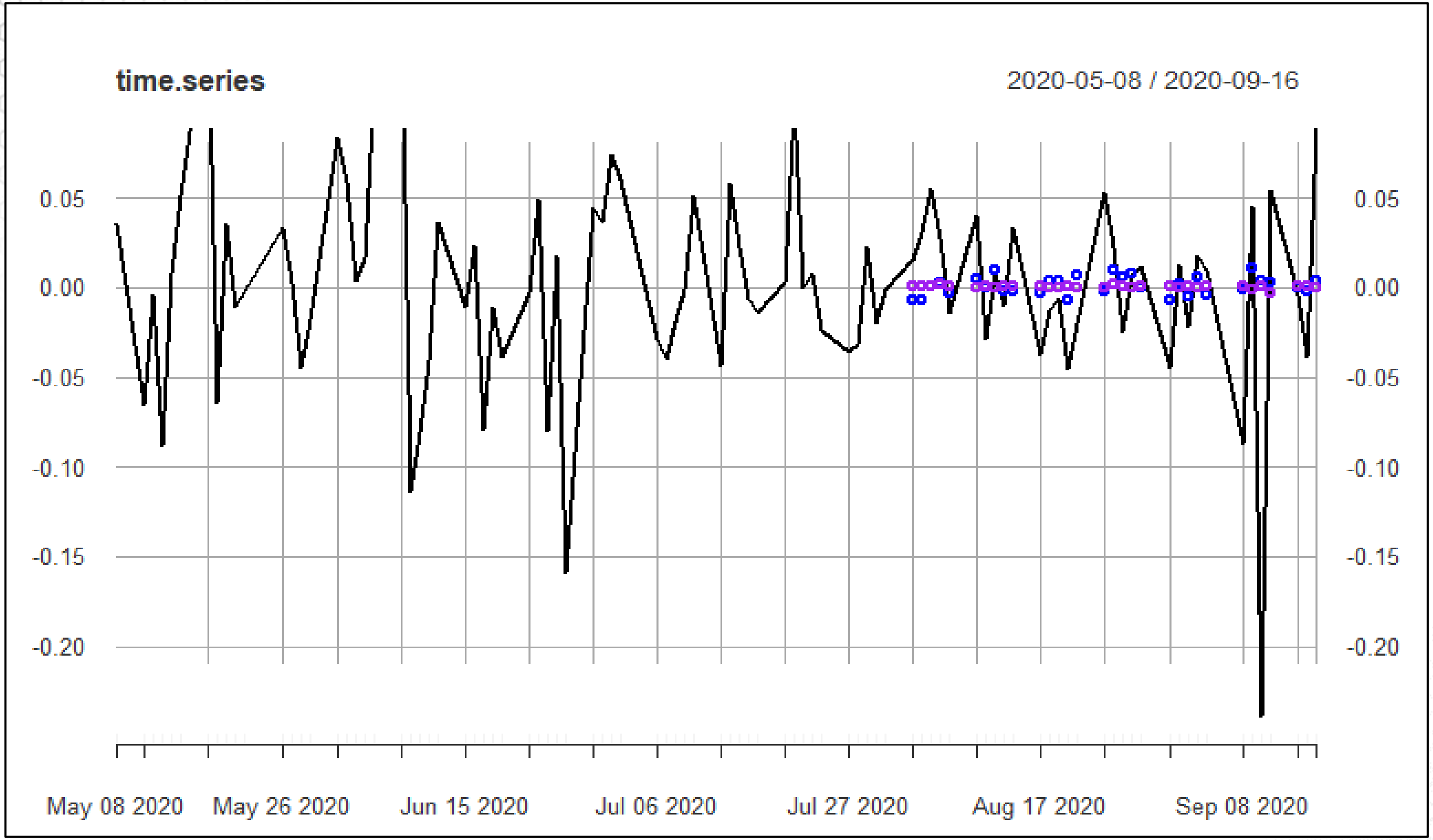
Model 1 performs best across all measures

```
> ### Mean Absolute Percentage Error (MAPE)
> mean(abs(fore.series.1 - pdcert.test)/abs(pdcert.test))
[1] 0.9554577
> mean(abs(fore.series.2 - pdcert.test)/abs(pdcert.test))
[1] 1.013034
> mean(abs(fore.series.3 - pdcert.test)/abs(pdcert.test))
[1] 1.012645
> ### Precision Measure (PM)
> sum((fore.series.1 - pdcert.test)^2)/sum((pdcert.test-mean(pdcert.test))^2)
[1] 0.9944229
> sum((fore.series.2 - pdcert.test)^2)/sum((pdcert.test-mean(pdcert.test))^2)
[1] 1.01484
> sum((fore.series.3 - pdcert.test)^2)/sum((pdcert.test-mean(pdcert.test))^2)
[1] 1.014493
```

Mean Prediction Comparison

```
## Create a similar data structure for the forecasts
data.plot = pdcert.test
names(data.plot)="Fore"
## Compare observed time series with mean forecasts
plot(pdcert[c(n-90):n],type="l", ylim=c(ymin,ymax), xlab="Time", ylab="Return Price")
data.plot$Fore=fore.series.1
points(data.plot,lwd= 2, col="blue")
data.plot$Fore=fore.series.2
points(data.plot,lwd= 2, col="brown")
data.plot$Fore=fore.series.3
points(data.plot,lwd= 2, col="purple")
```

Mean Prediction Comparison

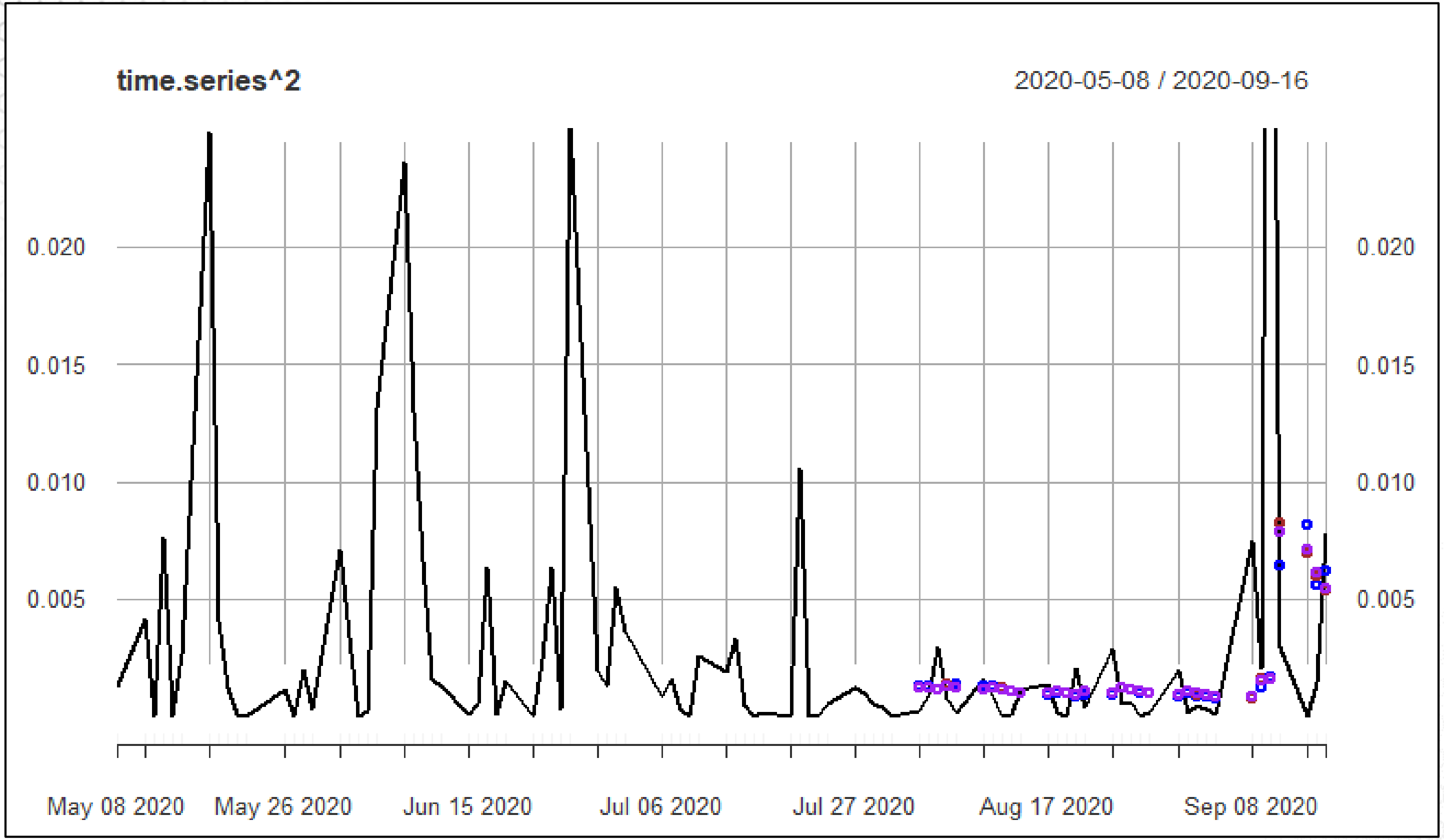


Georgia Tech

Variance Prediction Comparison

```
## Compare squared observed time series with variance forecasts
plot(time.series^2,type="l", ylim=c(ymin,ymax), xlab="Time", ylab="Return Price")
data.plot$Fore=fore.sigma.1^2
points(data.plot,lwd= 2, col="blue")
data.plot$Fore=fore.sigma.2^2
points(data.plot,lwd= 2, col="brown")
data.plot$Fore=fore.sigma.3^2
points(data.plot,lwd= 2, col="purple")
```

Variance Prediction Comparison



Georgia Tech

Summary