

Big Data and Internet Finance

HW 4

zhenyu xu

15620161152289

1. make an R quantlet to solve HW #1 from unit 1 with R and show it on Github (GH)

hint: use the CMB Qs for this work

```
lapply(libraries, library, quietly = TRUE, character.only = TRUE)
```

```
# load dataset  
load(file.choose())  
load(file.choose())  
load(file.choose())
```

```
# three indices return  
ecrix1 = zoo(ecrix, order.by = index(crix1))  
efcrix1 = zoo(efcrix, order.by = index(crix1))
```

```
# plot with different x-axis scales with zoo  
my.panel <- function(x, ...) {  
  lines(x, ...)  
  lines(ecrix1, col = "blue")  
  lines(efcrix1, col = "red")  
}  
plot.zoo(crix1, plot.type = "multiple", type = "l", lwd = 1.5, panel = my.panel,  
  main = "Indices in the CRIX family", xlab = "Date")
```

```
# plot of crix
# plot(as.xts(crix), type="l", auto.grid=FALSE, main = NA)
plot(crix1, ylab = "Price of CRIX", xlab = "Date")
```

```
# plot of crix return
ret = diff(log(crix1))
# plot(as.xts(ret), type="l", auto.grid=FALSE, main = NA)
plot(ret, ylab = "Return of CRIX", xlab = "Date")
```

```
# stationary test
adf.test(ret, alternative = "stationary")
kpss.test(ret, null = "Trend")
```

```
par(mfrow = c(1, 2))
# histogram of returns
hist(ret, col = "grey", breaks = 20, freq = FALSE, ylim = c(0, 25), xlab = "Return of CRIX")
lines(density(ret), lwd = 2)
mu = mean(ret)
sigma = sd(ret)
x = seq(-4, 4, length = 100)
curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add = TRUE, col = "red",
      lwd = 2)
```

```
# qq-plot  
qqnorm(ret)  
qqline(ret, col = "blue", lwd = 3)
```

```
# acf plot  
autocorr = acf(ret, lag.max = 20, ylab = "Sample Autocorrelation", main = "acf plot",  
               lwd = 2, ylim = c(-0.3, 1))
```

```
# pacf plot  
autopcorr = pacf(ret, lag.max = 20, ylab = "Sample Partial Autocorrelation",  
                 main = "pacf plot", ylim = c(-0.3, 0.3), lwd = 2)
```

Q2. Make your R code perfect as in the R examples on quantlet.de i.e. make sure that the code is "time independent" by using actual dimensions of the data that you are collecting from crix.hu-berlin.de Recreate Fig 7 from "Econometrics of CRIX".

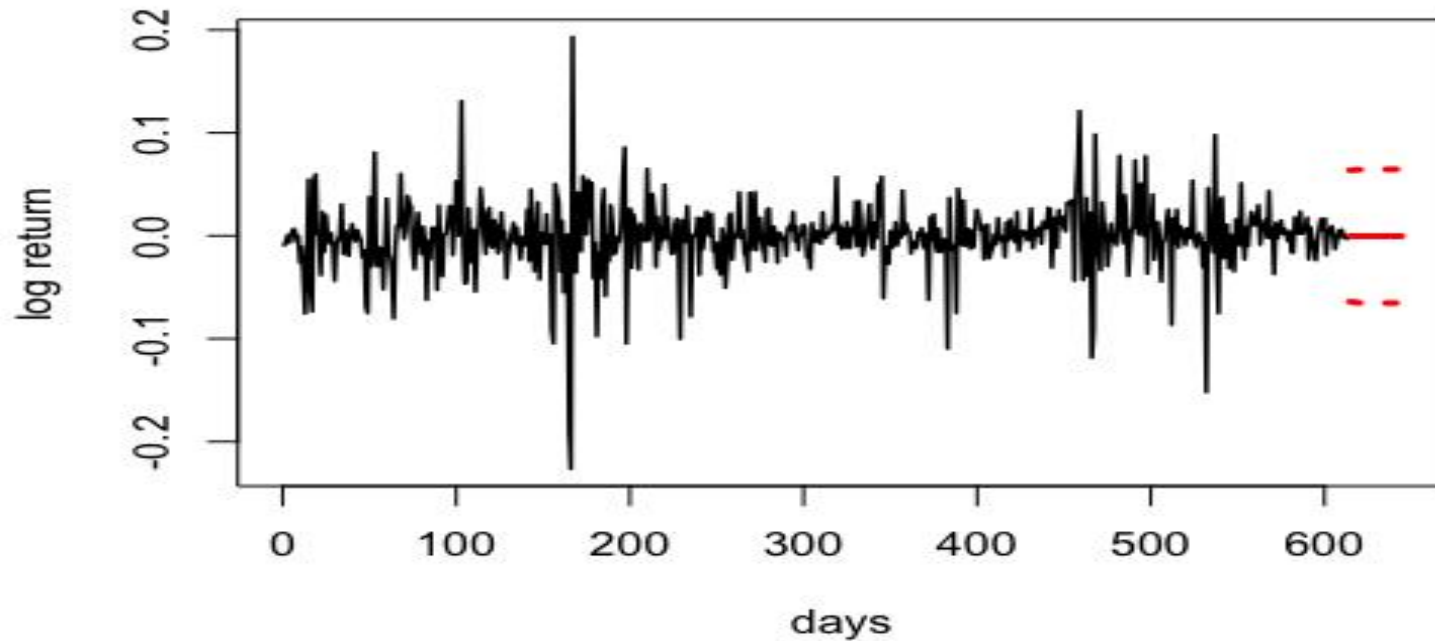


Figure 7: CRIX returns and predicted values.

Codes:

```
# arima model
par(mfrow = c(1, 1))
fit1 = arima(ret, order = c(1, 0, 1))
tsdiag(fit1)
Box.test(fit1$residuals, lag = 1)

# aic
aic = matrix(NA, 6, 6)
for (p in 0:4) {
  for (q in 0:3) {
    a.p.q = arima(ret, order = c(p, 0, q))
    aic.p.q = a.p.q$aic
    aic[p + 1, q + 1] = aic.p.q
  }
}
```

```
# bic
bic = matrix(NA, 6, 6)
for (p in 0:4) {
  for (q in 0:3) {
    b.p.q = arima(ret, order = c(p, 0, q))
    bic.p.q = AIC(b.p.q, k =
log(length(ret)))
    bic[p + 1, q + 1] = bic.p.q
  }
}
```

```
# select p and q order of ARIMA
model
fit4 = arima(ret, order = c(2, 0, 3))
tsdiag(fit4)
Box.test(fit4$residuals, lag = 1)
```

```
fitr4 = arima(ret, order = c(2, 1, 3))
tsdiag(fitr4)
Box.test(fitr4$residuals, lag = 1)
```

```
# to conclude, 202 is better than 213
fit202 = arima(ret, order = c(2, 0, 2))
```

```
AIC(fit202, k = log(length(ret)))
AIC(fit4, k = log(length(ret)))
AIC(fitr4, k = log(length(ret)))
fit202$aic
fit4$aic
fitr4$aic
```

```
# arima202 predict
predict_num = 30
fit202 = arima(ret, order = c(2, 0, 2))
crpre = predict(fit202, n.ahead = predict_num)
```

```
dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(ret))
plot(ret, type = "l", xlim = c(0, length(ret)+predict_num), ylab = "log return", xlab = "days",
     lwd = 1.5, col = "black")
lines(crpre$pred, col = "red", lwd = 3)
lines(crpre$pred + 2 * crpre$se, col = "red", lty = 3, lwd = 3)
lines(crpre$pred - 2 * crpre$se, col = "red", lty = 3, lwd = 3)
```

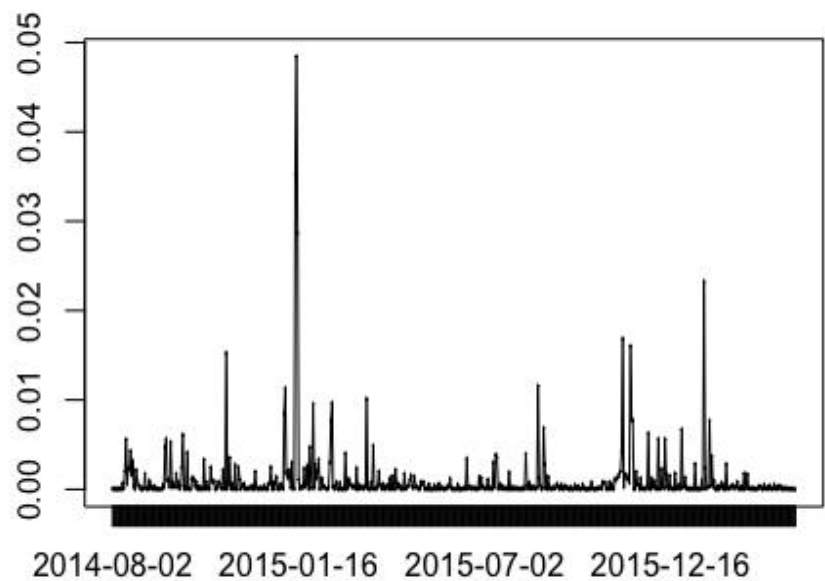


Figure 8: The squared ARIMA(2,0,2) residuals of CRIX returns.

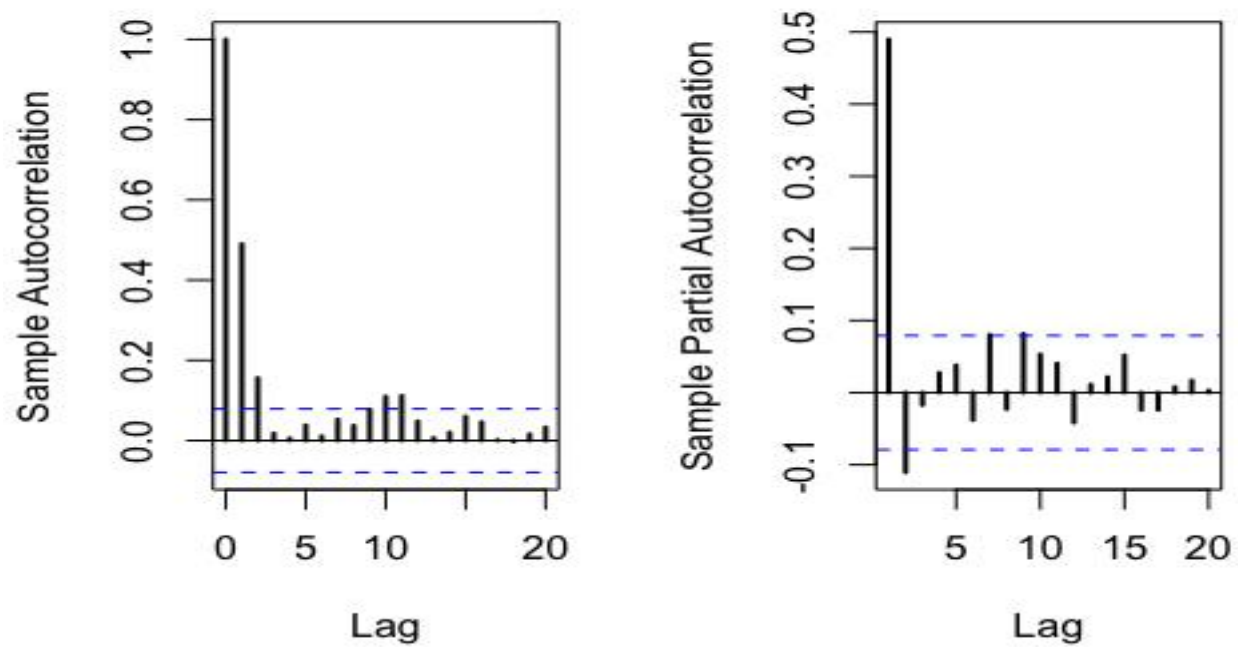


Figure 9: The ACF and PACF of squared ARIMA(2,0,2) residuals

Codes:

```
rm(list = ls(all = TRUE))  
graphics.off()
```

install and load packages

```
libraries = c("tseries")  
lapply(libraries, function(x) if (!(x %in% installed.packages()))  
{  
  install.packages(x)  
})  
lapply(libraries, library, quietly = TRUE, character.only = TRUE)
```

please change your working directory

```
setwd()  
load(file.choose())  
Pr = as.numeric(crix)  
Da = factor(date1)  
crx = data.frame(Da, Pr)  
# plot of crix return  
ret = diff(log(crx$Pr))  
Dare = factor(date1[-1])  
retts = data.frame(Dare, ret)  
# arima202 predict  
fit202 = arima(ret, order = c(2, 0, 2))
```

vola cluster

```
par(mfrow = c(1, 1))  
res = fit202$residuals  
res2 = fit202$residuals^2  
tsres202 = data.frame(Dare, res2)  
plot(tsres202$Dare,  
tsres202$res2, type = "o", ylab =  
NA)  
lines(tsres202$res2)
```

plot(res2, ylab='Squared
residuals', main=NA)

```
par(mfrow = c(1, 2))  
acfres2 = acf(res2, main = NA,  
lag.max = 20, ylab = "Sample  
Autocorrelation", lwd = 2)  
pacfres2 = pacf(res2, lag.max = 20,  
ylab = "Sample Partial  
Autocorrelation", lwd = 2, main =  
NA)
```

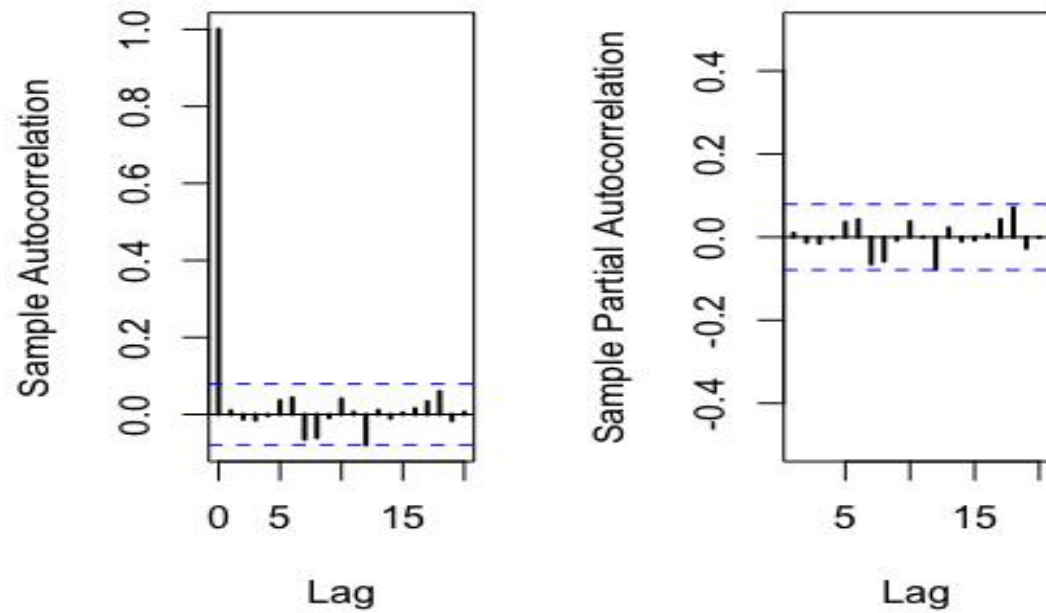


Figure 10: The ACF and PACF of squared ARIMA(2,0,2) residuals

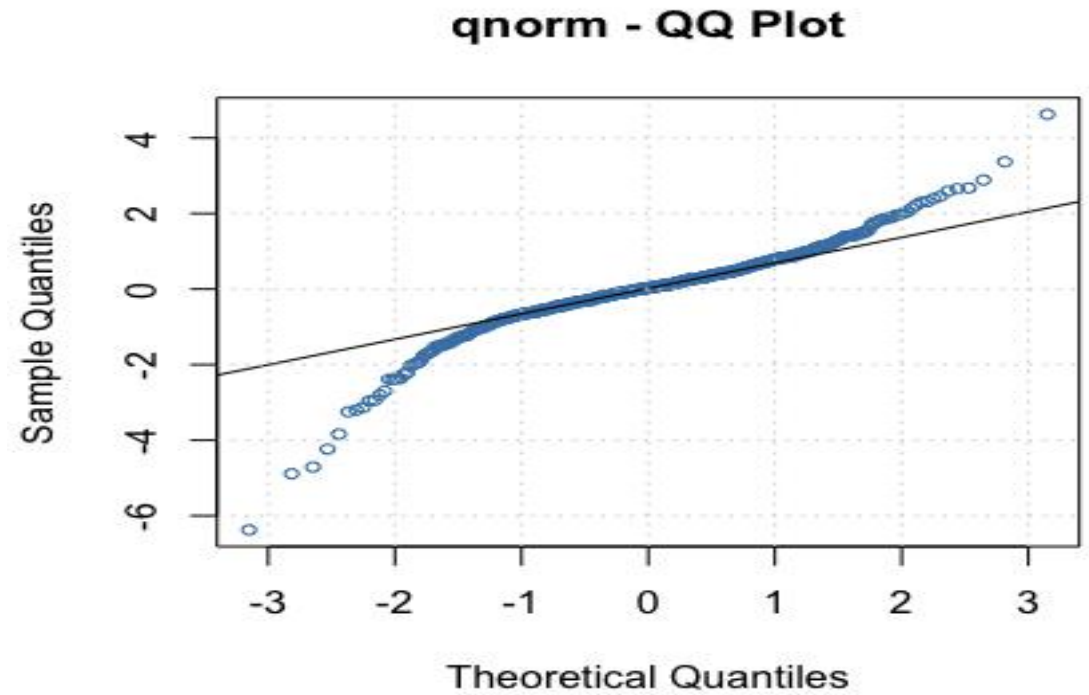


Figure 11: The QQ plots of model residuals of ARIMA-GARCH process.

Codes:

```
rm(list = ls(all = TRUE))
graphics.off()

# install and load packages
libraries = c("forecast", "fGarch")
lapply(libraries, function(x) if (!(x %in% installed.packages()))
{
  install.packages(x)
})
lapply(libraries, library, quietly = TRUE, character.only = TRUE)

# load dataset
load(file.choose())
ret = diff(log(crix1))

# vol cluster
fit202 = arima(ret, order = c(2, 0, 2))
par(mfrow = c(1, 1))
res = fit202$residuals
res2 = fit202$residuals^2
```

```
# different garch model
fg11 = garchFit(data = res, data ~ garch(1, 1))
summary(fg11)
fg12 = garchFit(data = res, data ~ garch(1, 2))
summary(fg12)
fg21 = garchFit(data = res, data ~ garch(2, 1))
summary(fg21)
fg22 = garchFit(data = res, data ~ garch(2, 2))
summary(fg22)

# residual plot
reszo = zoo(fg11$residuals, order.by =
index(crix1))
plot(reszo, ylab = NA, lwd = 2)
```

```
par(mfrow = c(1, 2))
fg11res2 = fg11@residuals
acfres2 = acf(fg11res2, lag.max = 20, ylab = "Sample Autocorrelation",
              main = NA, lwd = 2)
pacfres2 = pacf(fg11res2, lag.max = 20, ylab = "Sample Partial Autocorrelation",
                main = NA, lwd = 2, ylim = c(-0.5, 0.5))
```

```
fg12res2 = fg12@residuals
acfres2 = acf(fg12res2, lag.max = 20, ylab = "Sample Autocorrelation",
              main = NA, lwd = 2)
pacfres2 = pacf(fg12res2, lag.max = 20, ylab = "Sample Partial Autocorrelation",
                main = NA, lwd = 2, ylim = c(-0.5, 0.5))
```

```
# qq plot
par(mfrow = c(1, 1))
plot(fg11, which = 13) #9,10,11,13
```

ACF of Squared Residuals **PACF of Squared Residuals**

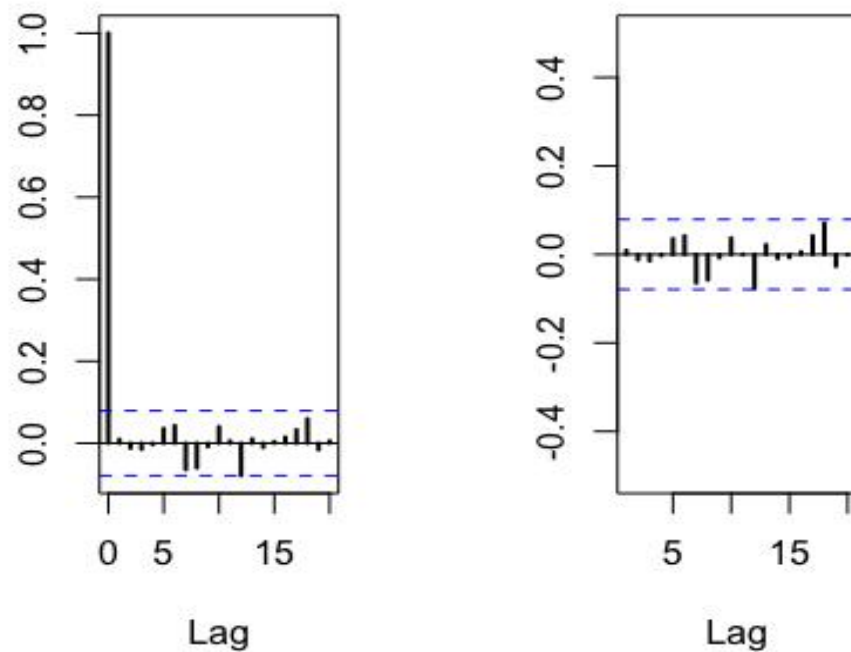


Figure 12: The ACF and PACF plots for model residuals of ARIMA(2,0,2)- t-GARCH(1,1) process.

qstd - QQ Plot

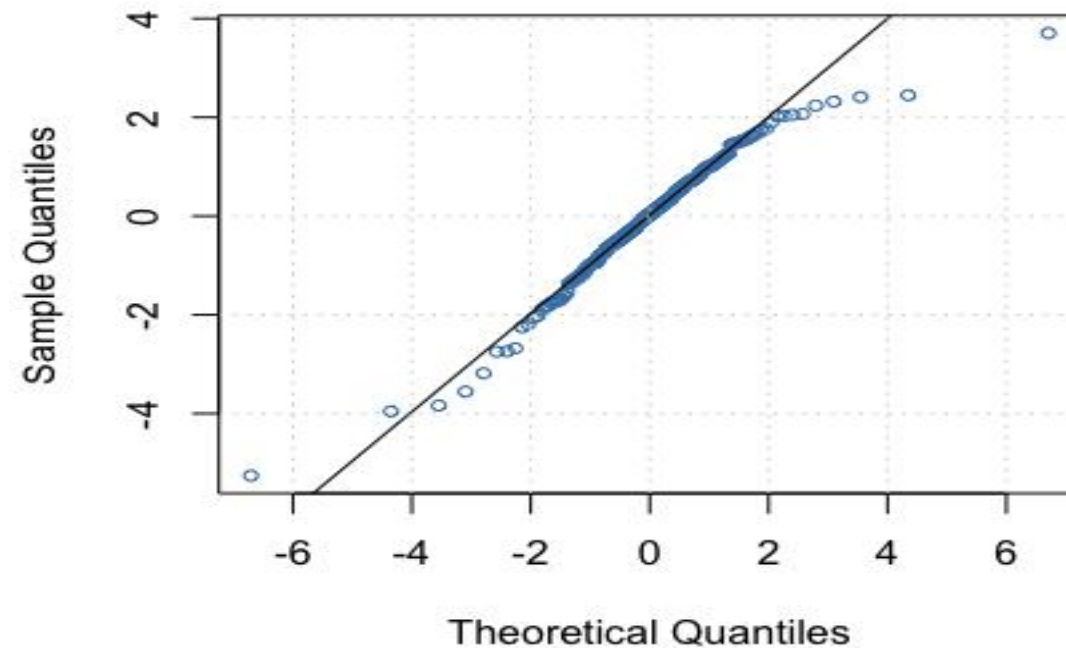


Figure 13: The QQ plots of model residuals of ARIMA-t-GARCH process.

Codes:

```
fg11stu = garchFit(data = res, data ~ garch(1, 1), cond.dist = "std")
```

```
# different forecast with t-garch
```

```
# fg11stufore = predict(fg11stu, n.ahead = 30, plot=TRUE, mse='uncond', auto.grid=FALSE)
```

```
fg11stufore = predict(fg11stu, n.ahead = 30, plot = TRUE, cond.dist = "QMLE",  
                      auto.grid = FALSE)
```

```
par(mfrow = c(1, 2))
```

```
stu.fg11res2 = fg11stu@residuals
```

```
# acf and pacf for t-garch
```

```
stu.acfres2 = acf(stu.fg11res2, ylab = NA, lag.max = 20, main = "ACF of Squared Residuals",  
                  lwd = 2)
```

```
stu.pacfres2 = pacf(stu.fg11res2, lag.max = 20, main = "PACF of Squared Residuals",  
                    lwd = 2, ylab = NA, ylim = c(-0.5, 0.5))
```

```
# ARIMA-t-GARCH qq plot
```

```
par(mfrow = c(1, 1))
```

```
plot(fg11stu, which = 13)
```