HW4

Ze YANG

10/1/2018

R Lab from Ruppert

Problem 1

```
data(Tbrate, package="Ecdat")
library(forecast)
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'default/
## America/New_York'
```

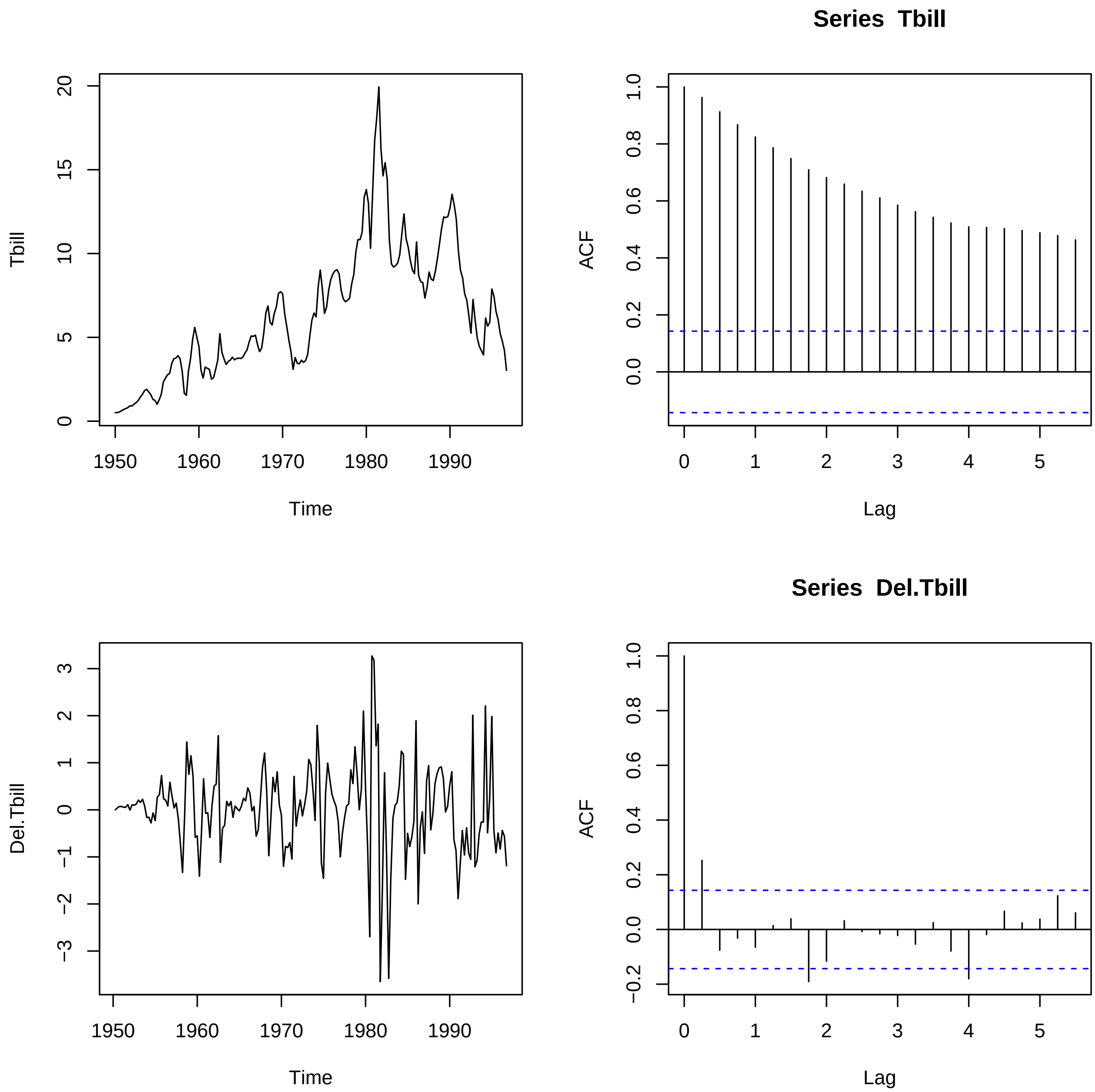
```
library(tseries)
library(fGarch)
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
## Loading required package: fBasics
```

```
# r = the 91-day treasury bill rate
# y = the log of real GDP
# pi = the inflation rate
Tbill = Tbrate[,1]
Del.Tbill = diff(Tbill)

par(mfrow=c(2,2))
plot(Tbill)
acf(Tbill)
plot(Del.Tbill)
acf(Del.Tbill)
```




```r
adf.test(Tbill)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Tbill
## Dickey-Fuller = -1.925, Lag order = 5, p-value = 0.6075
## alternative hypothesis: stationary
```

```r
adf.test(Del.Tbill)
```

```
## Warning in adf.test(Del.Tbill): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
```

```
## data:  Del.Tbill
## Dickey-Fuller = -5.2979, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Comments - `Tbill`'s plot has a clear trend in it, and the acf plot decays slowly. The ADF test failed to reject null H_0 : the series is nonstationary. Hence we believe it's nonstationary.

- `Del.Tbill`'s plot displays mean-reverting behavior; the acf plot decays fast. The ADF test also reject the nonstationarity hypothesis with confidence level 0.01. So we think it's stationary.
- `Del.Tbill` exhibits two types of heteroscedasticity patterns: First, it has a volatility clustering behavior. Second, the volatility is correlated to the value of `Tbill`: the vol's higher in early 1980s when the local mean of `Tbill` itself is high.

Problem 2

```r
garch.model.Tbill = garchFit(formula= ~arma(1,0) + garch(1,0),Del.Tbill)
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(1, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 0)
##  ARMA Order:                1 0
##  Max ARMA Order:            1
##  GARCH Order:               1 0
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          norm
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          187
##  Recursion Init:            mci
##  Series Scale:              0.9422364
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                      U           V     params includes
##     mu     -0.14290725   0.1429072 0.01197006     TRUE
##     ar1    -0.99999999   1.0000000 0.25347489     TRUE
##     omega   0.00000100 100.0000000 0.10000000     TRUE
##     alpha1  0.00000001   1.0000000 0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000 0.10000000    FALSE
##     delta   0.00000000   2.0000000 2.00000000    FALSE
##     skew    0.10000000  10.0000000 1.00000000    FALSE
##     shape   1.00000000  10.0000000 4.00000000    FALSE
##  Index List of Parameters to be Optimized:
##     mu    ar1  omega alpha1
##      1      2      3      4
##  Persistence:                  0.1
```

```
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     456.21058: 0.0119701 0.253475 0.100000 0.100000
##   1:     251.86762: 0.0119754 0.236936  1.02079 0.489709
##   2:     250.90930: 0.0120591 0.473682 0.845081 0.893495
##   3:     246.37888: 0.0132650 0.303699 0.757428  1.00000
##   4:     244.30653: 0.0172433 0.228377 0.691915  1.00000
##   5:     240.23776: 0.0270411 0.161803 0.543304  1.00000
##   6:     236.37169: 0.0367592 0.175592 0.370918  1.00000
##   7:     236.31833: 0.0446835 0.210080 0.425054 0.537516
##   8:     235.93894: 0.0447059 0.288586 0.470834 0.622965
##   9:     235.66475: 0.0444909 0.288456 0.377870 0.704783
##  10:     235.36527: 0.0456870 0.217083 0.405800 0.756581
##  11:     235.19182: 0.0473954 0.248964 0.405265 0.772573
##  12:     235.13214: 0.0513735 0.255349 0.382404 0.810199
##  13:     235.08391: 0.0553971 0.257352 0.383930 0.823996
##  14:     234.96674: 0.0712269 0.253195 0.382822 0.857205
##  15:     234.92037: 0.0846539 0.246053 0.380919 0.862586
##  16:     234.91025: 0.0885175 0.242328 0.380676 0.848471
##  17:     234.90819: 0.0890151 0.241257 0.380809 0.836077
##  18:     234.90814: 0.0886908 0.241545 0.380857 0.834926
##  19:     234.90814: 0.0886149 0.241631 0.380886 0.834829
##  20:     234.90814: 0.0886153 0.241635 0.380889 0.834828
##
## Final Estimate of the Negative LLH:
##  LLH:  223.7818    norm LLH:  1.196694
##         mu        ar1      omega     alpha1
## 0.08349652 0.24163450 0.33815662 0.83482811
##
## R-optimhess Difference Approximated Hessian Matrix:
##                mu        ar1      omega    alpha1
## mu     -379.01802  -68.15986  -37.51437   4.76325
## ar1     -68.15986 -209.40200   51.00153   8.23281
## omega   -37.51437   51.00153 -399.68185 -54.31253
## alpha1    4.76325    8.23281  -54.31253 -24.58337
## attr(,"time")
## Time difference of 0.01015687 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.05084801 secs
```

```
summary(garch.model.Tbill)
```

```
##
## Title:
##  GARCH Modelling
##
```

```
## Call:
##  garchFit(formula = ~arma(1, 0) + garch(1, 0), data = Del.Tbill)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 0)
## <environment: 0x7fadd0b80830>
##  [data = Del.Tbill]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu       ar1     omega    alpha1
## 0.083497  0.241635  0.338157  0.834828
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.08350     0.05391    1.549 0.121395
## ar1      0.24163     0.07280    3.319 0.000902 ***
## omega    0.33816     0.06145    5.503 3.73e-08 ***
## alpha1   0.83483     0.24295    3.436 0.000590 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -223.7818    normalized:  -1.196694
##
## Description:
##  Mon Oct  1 15:38:53 2018 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  26.96616  1.394352e-06
##  Shapiro-Wilk Test  R    W      0.957289  1.957134e-05
##  Ljung-Box Test     R    Q(10)  10.8275   0.3711143
##  Ljung-Box Test     R    Q(15)  13.10815  0.5939444
##  Ljung-Box Test     R    Q(20)  16.11144  0.7096868
##  Ljung-Box Test     R^2  Q(10)  12.18313  0.2729873
##  Ljung-Box Test     R^2  Q(15)  14.31743  0.5016035
##  Ljung-Box Test     R^2  Q(20)  17.28754  0.634231
##  LM Arch Test       R    TR^2   9.685432  0.6435358
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 2.436169 2.505284 2.435279 2.464174
```

```
garch.model.Tbill@fit$matcoef
```

```
##          Estimate  Std. Error  t value     Pr(>|t|)
## mu     0.08349652  0.05390549 1.548943 1.213955e-01
## ar1    0.24163450  0.07279700 3.319292 9.024598e-04
```

```
## omega  0.33815662  0.06144724 5.503203 3.729535e-08
## alpha1 0.83482811  0.24295106 3.436199 5.899382e-04
```

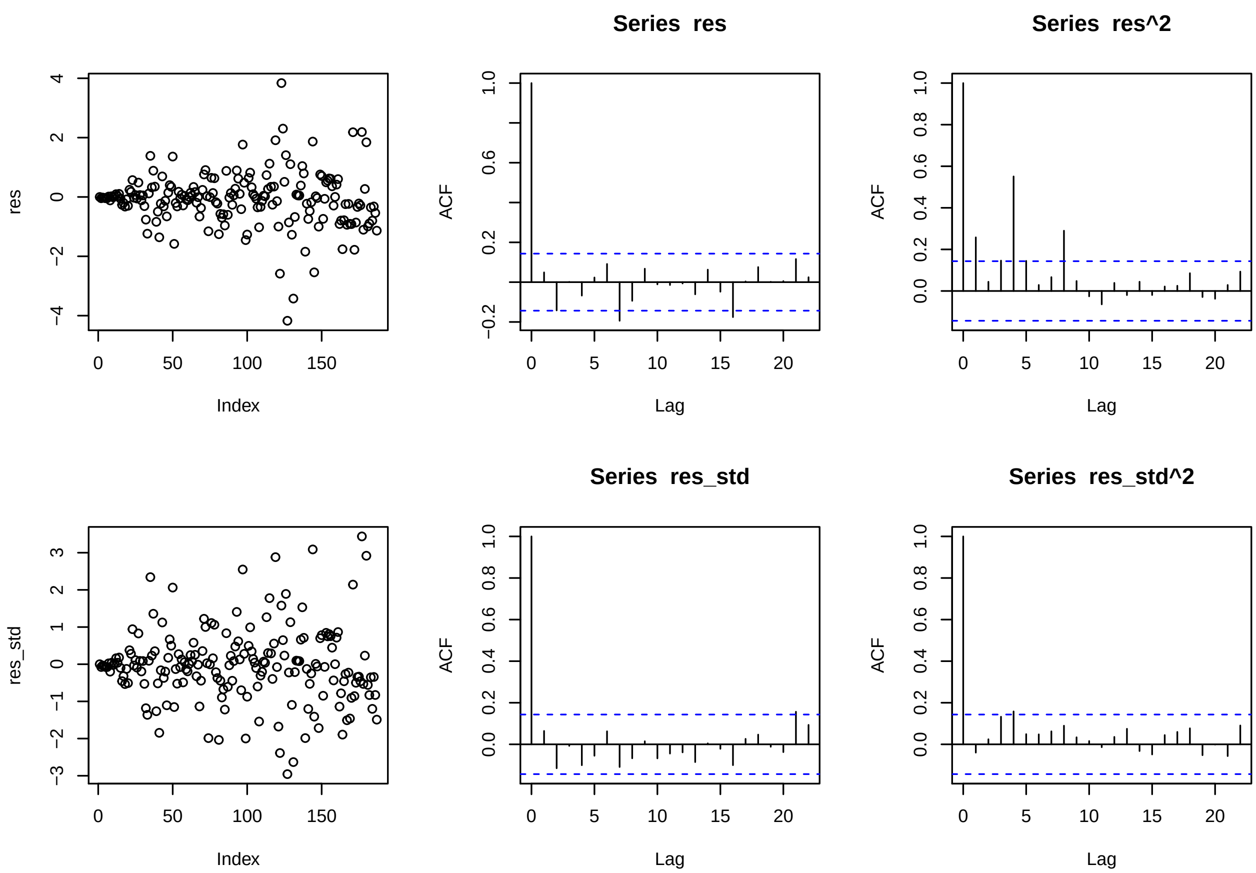
(a) The ARMA/GARCH model is fitted for `Del.Tbill`. The model that corresponds to R parameter names is:

$$
\begin{aligned}
&X_t - \text{mu} = \text{ar1} \cdot (X_{t-1} - \text{mu}) + a_t \\
&a_t = \sigma_t \epsilon_t \\
&\sigma_t^2 = \text{omega} + \text{alpha1} \cdot a_{t-1}^2 \\
&\epsilon_t \sim \text{i.i.d white noise, mean 0, unit variance}
\end{aligned}
$$

(b) The parameter estimates are shown in the `summary(garch.model.Tbill)`, where we have: `mu` $= 0.08350$, `ar1` $= 0.24163$, `omega` $= 0.33816$, and `alpha1` $= 0.83483$.

Problem 3

```
res = residuals(garch.model.Tbill)
res_std = res / garch.model.Tbill@sigma.t
par(mfrow=c(2,3))
plot(res)
acf(res)
acf(res^2)
plot(res_std)
acf(res_std)
acf(res_std^2)
```



Answers (I'd like to answer (d) first, since (c) is actually based upon (d))

- (d): `garch.model.Tbill@sigma.t` contains the estimates of the conditional standard deviation of a_t from the GARCH model. Therefore, `res_std` is the realizations of $\epsilon_t = a_t/\sigma_t$.
- (a): `acf(res)` shows there is no significant autocorrelation in the residuals, which implies that the model fit of the conditional mean (AR(1)) is adequete.
- (b): `acf(res^2)` shows significant autocorrelation in the squared residuals a_t^2, which implies a conditional heteroscedasticity - the reason why we are fitting the GARCH part.
- (c): `acf(res_std^2)` shows no significant autocorrelation in the standardized squared residuals $\epsilon_t^2 = (a_t/\sigma_t)^2$, which is consistant to the white noise assumption about ϵ_t, and implies that GARCH(1,0) fit for the conditional standard deviation is again adequate.
- (e): The time series plot of `res_std` shows the empirical distribution of `res_std` have heavier tail than normal distribution - there are quite a few points with absolute value > 1.96, i.e. beyond 97.5% two-sided percentile of standard normal distribution.

Problem 4

```
# fit the model
garch.model.log.Tbill = garchFit(formula= ~arma(1,0) + garch(1,0), diff(log(Tbill)))
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(1, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 0)
##  ARMA Order:                1 0
##  Max ARMA Order:            1
##  GARCH Order:               1 0
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          norm
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          187
##  Recursion Init:            mci
##  Series Scale:              0.1506804
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                        U           V     params includes
##     mu     -0.63215833   0.6321583 0.05810348     TRUE
##     ar1    -0.99999999   1.0000000 0.29238716     TRUE
##     omega   0.00000100 100.0000000 0.10000000     TRUE
##     alpha1  0.00000001   1.0000000 0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000 0.10000000    FALSE
##     delta   0.00000000   2.0000000 2.00000000    FALSE
##     skew    0.10000000  10.0000000 1.00000000    FALSE
##     shape   1.00000000  10.0000000 4.00000000    FALSE
##  Index List of Parameters to be Optimized:
```

```
##      mu    ar1  omega alpha1
##       1      2      3      4
##  Persistence:                  0.1
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     540.68757: 0.0581035 0.292387 0.100000 0.100000
##   1:     259.25047: 0.0581586 0.351171  1.04735 0.414755
##   2:     258.16987: 0.0582028 0.456570  1.03558 0.402981
##   3:     252.49959: 0.0574411 0.545312 0.727946 0.684987
##   4:     250.06907: 0.0574976 0.428738 0.487576 0.613163
##   5:     249.49697: 0.0730634 0.539003 0.542145 0.585597
##   6:     249.37315: 0.0770830 0.479971 0.564196 0.647037
##   7:     249.26377: 0.0814361 0.486281 0.508052 0.660433
##   8:     249.22636: 0.0809935 0.488008 0.527309 0.622664
##   9:     249.22557: 0.0800861 0.491156 0.531795 0.627067
##  10:     249.22447: 0.0791458 0.490668 0.531751 0.621237
##  11:     249.22437: 0.0788238 0.489868 0.531039 0.621091
##  12:     249.22436: 0.0788350 0.490084 0.531231 0.620527
##  13:     249.22436: 0.0788424 0.490071 0.531222 0.620629
##  14:     249.22436: 0.0788416 0.490071 0.531221 0.620624
##
## Final Estimate of the Negative LLH:
##  LLH:  -104.6908    norm LLH:  -0.5598439
##         mu        ar1      omega     alpha1
## 0.01187989 0.49007057 0.01206114 0.62062414
##
## R-optimhess Difference Approximated Hessian Matrix:
##                  mu         ar1        omega       alpha1
## mu     -12197.74425 -168.269073   -7034.6225    77.646879
## ar1      -168.26907 -234.866231    -192.8231    -6.463916
## omega   -7034.62251 -192.823054 -371223.5397 -1802.624159
## alpha1     77.64688   -6.463916   -1802.6242   -32.483982
## attr(,"time")
## Time difference of 0.009238958 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.03180599 secs
```

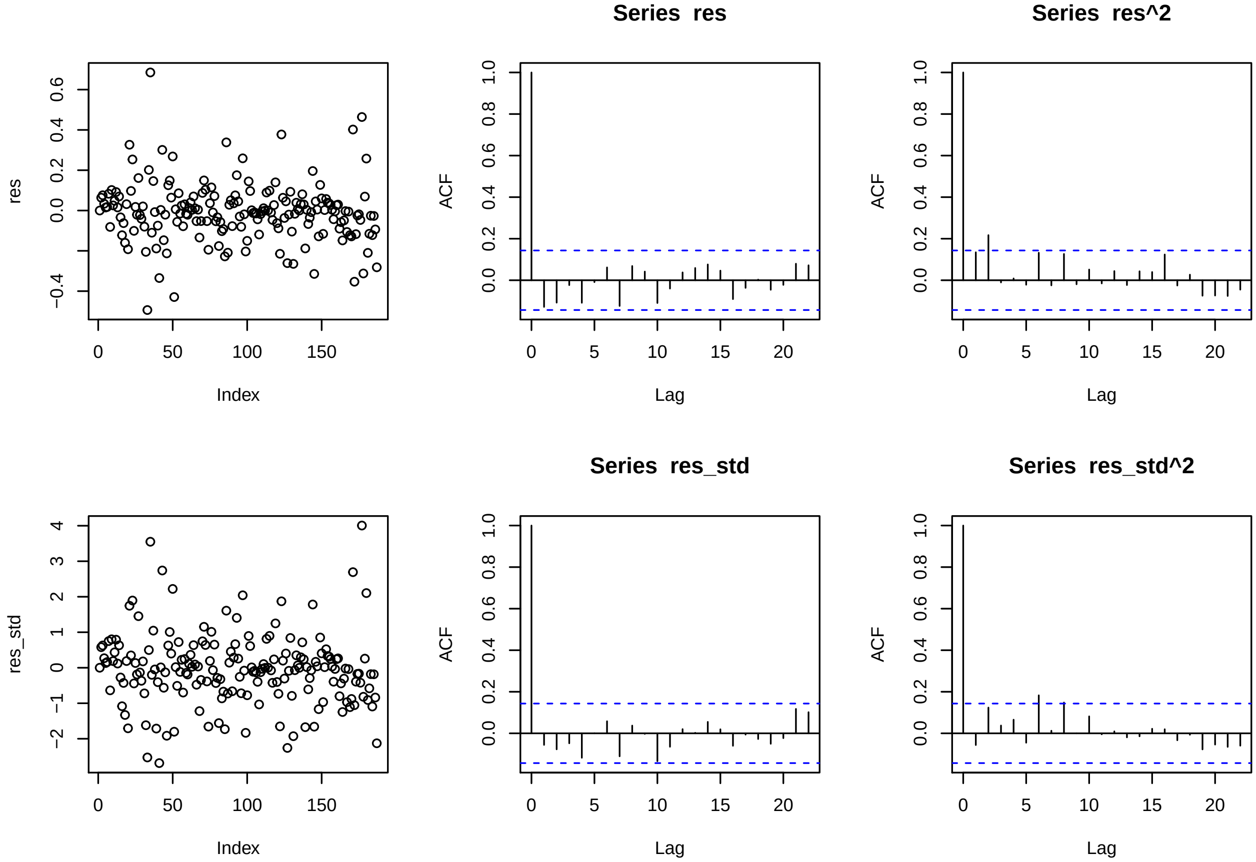
```
summary(garch.model.log.Tbill)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(1, 0) + garch(1, 0), data = diff(log(Tbill)))
##
```

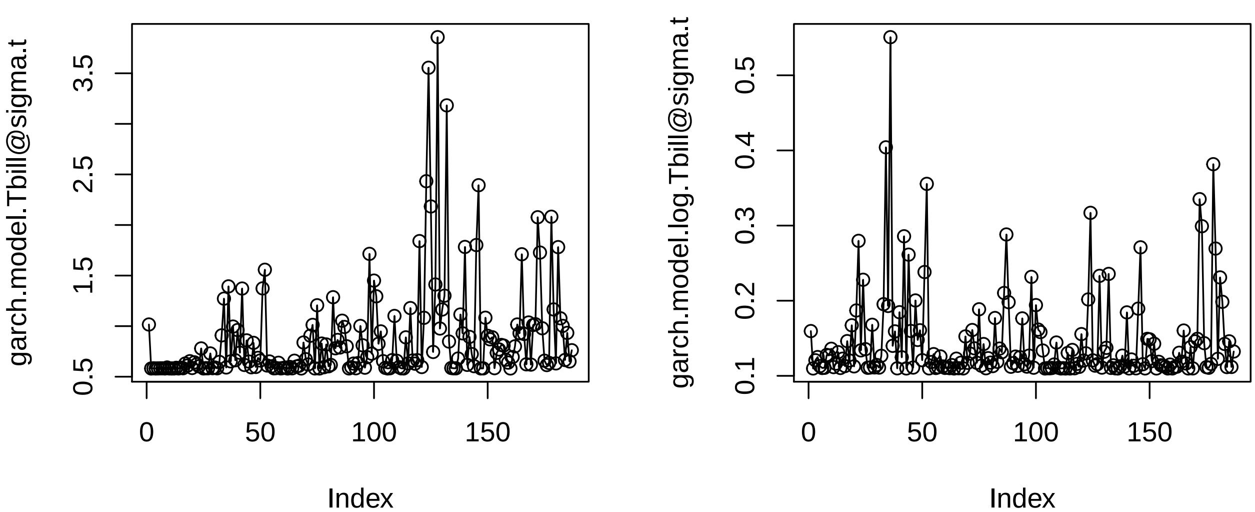
```
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 0)
## <environment: 0x7fadcda0e108>
##  [data = diff(log(Tbill))]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu       ar1     omega    alpha1
## 0.011880  0.490071  0.012061  0.620624
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.011880    0.009373    1.267  0.20500
## ar1     0.490071    0.065891    7.438 1.03e-13 ***
## omega   0.012061    0.001961    6.150 7.76e-10 ***
## alpha1  0.620624    0.210910    2.943  0.00325 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  104.6908    normalized:  0.5598439
##
## Description:
##  Mon Oct  1 15:38:54 2018 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  45.79453  1.137217e-10
##  Shapiro-Wilk Test  R    W      0.9540278 9.29422e-06
##  Ljung-Box Test     R    Q(10)  11.76569  0.3010438
##  Ljung-Box Test     R    Q(15)  13.38437  0.5726355
##  Ljung-Box Test     R    Q(20)  14.94135  0.7797538
##  Ljung-Box Test     R^2  Q(10)  17.25489  0.06891087
##  Ljung-Box Test     R^2  Q(15)  17.49449  0.2901728
##  Ljung-Box Test     R^2  Q(20)  19.69902  0.4768933
##  LM Arch Test       R    TR^2   12.22925  0.4274482
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -1.076907 -1.007792 -1.077797 -1.048902
```

```r
# diagnostic plots
res = residuals(garch.model.log.Tbill)
res_std = res / garch.model.log.Tbill@sigma.t
par(mfrow=c(2,3))
plot(res)
acf(res)
acf(res^2)
```

```
plot(res_std)
acf(res_std)
acf(res_std^2)
```




```
par(mfrow=c(1,2))
plot(garch.model.Tbill@sigma.t)
lines(garch.model.Tbill@sigma.t)
plot(garch.model.log.Tbill@sigma.t)
lines(garch.model.log.Tbill@sigma.t)
```



Comments

- The main difference between the two is the conditional standard deviation series. Shown in the time series plot above.
- The σ_t of the log difference model is more "stable", and the spike associated with large `Tbill` value around early 1980s also disappeared in the log transformed model.

Ford Data

(a)

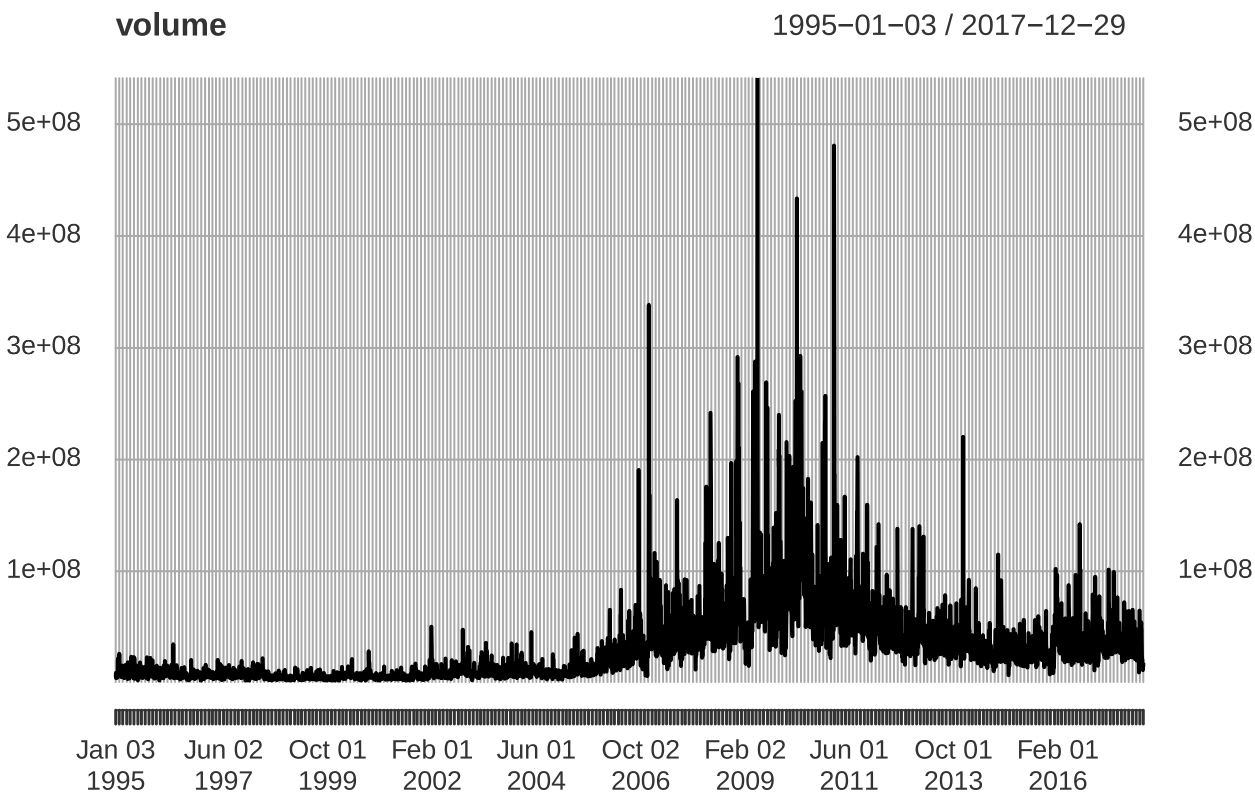
```
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:timeSeries':
##
##     time<-

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

##
## Attaching package: 'TTR'

## The following object is masked from 'package:fBasics':
##
##     volatility

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
FORD = getSymbols('F', from='1995-1-1', to='2017-12-31', auto.assign = F)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

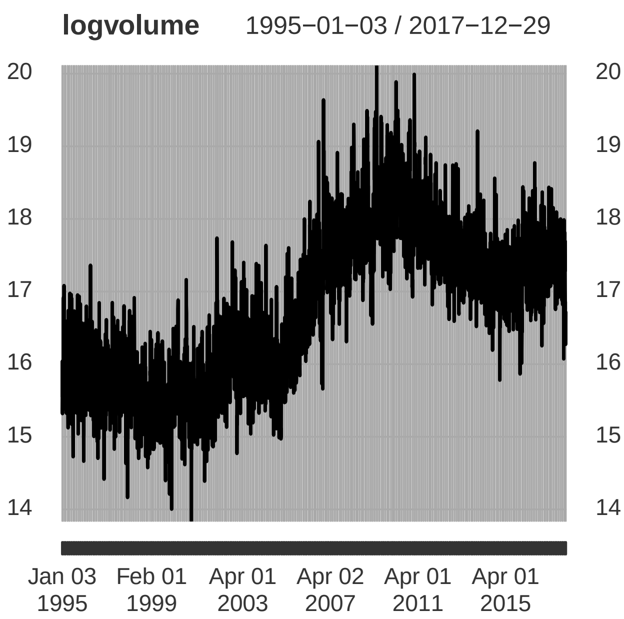
```
volume = FORD$F.Volume
plot(volume)
```

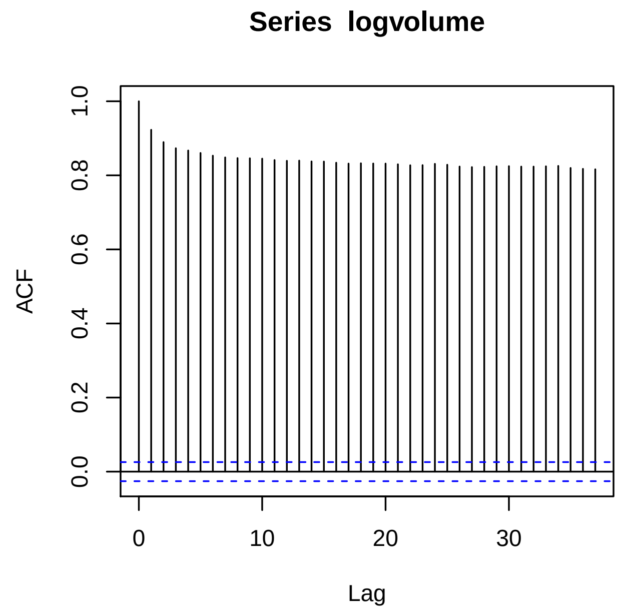


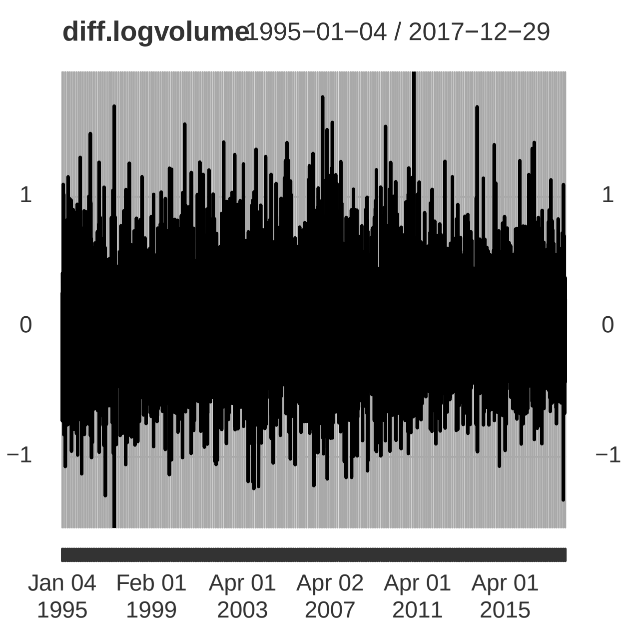
The volume has an extremely unstable size of variability. In particular, the variability of the series spikes among 2009 - 2011. Therefore, it would be a good idea to apply Box-Cox transformation to stablize its variability.

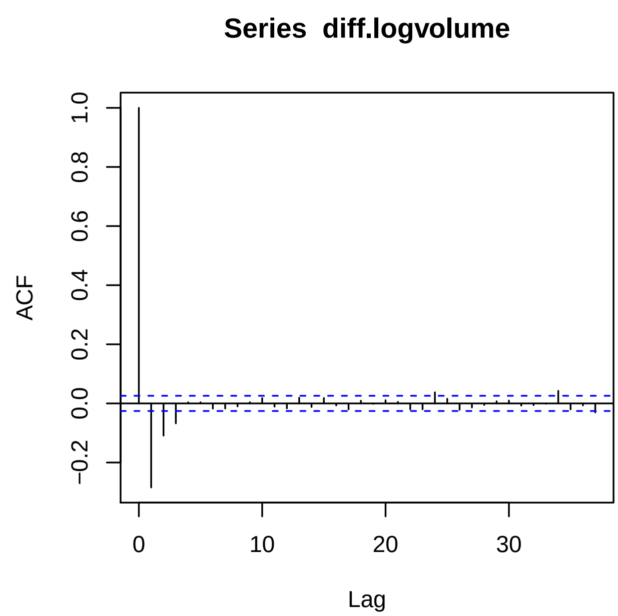
(b)

```
logvolume = log(volume)
diff.logvolume = diff(logvolume)
diff.logvolume = diff.logvolume[!is.na(diff.logvolume)]
par(mfrow=c(2,2))
plot(logvolume)
acf(logvolume)
plot(diff.logvolume)
acf(diff.logvolume)
```









- The `logvolume` series has a clear trend in it, and the ACF decays slowly. It's not stationary.
- The first difference of `logvolume` exhibits a more stable behavior, and the ACF of it decays very fast. So we believe `diff.logvolume` is stationary.

(c)

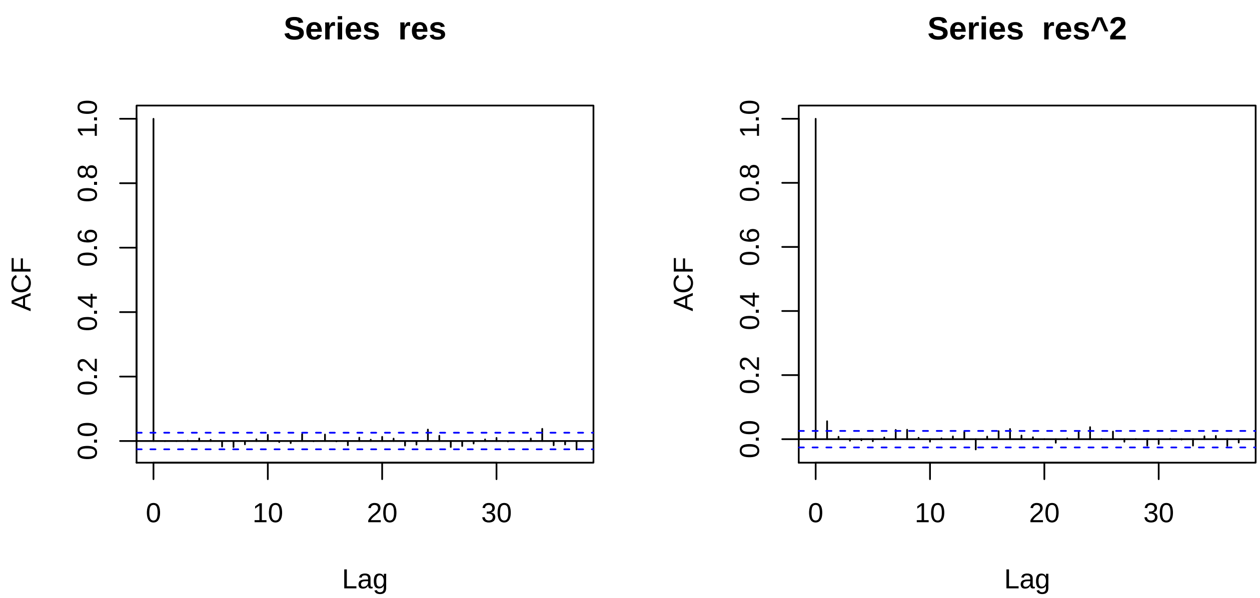
```
arima.fit = auto.arima(diff.logvolume, ic='aicc', approximation=F, step=F)
summary(arima.fit)
```

```
## Series: diff.logvolume
## ARIMA(1,0,4) with zero mean
##
```

```
## Coefficients:
##          ar1      ma1     ma2     ma3    ma4
##       0.8598  -1.3442  0.2199  0.0650  0.067
## s.e.  0.0248   0.0284  0.0245  0.0221  0.017
##
## sigma^2 estimated as 0.1242:  log likelihood=-2176.21
## AIC=4364.42   AICc=4364.43   BIC=4404.4
##
## Training set error measures:
##                       ME      RMSE       MAE     MPE     MAPE      MASE
## Training set 0.004279167 0.352317 0.2698948 39.7425 277.3254 0.5384615
##                       ACF1
## Training set -0.0001589462
```

Auto arima with AICC criteria finds optimal model ARIMA(1, 0, 4). The parameter estimates are shown in the summary above.

```r
res = arima.fit$residuals
par(mfrow=c(1,2))
acf(res)
acf(res^2)
```



- `acf(res)` shows there is no significant autocorrelation in the residuals, which implies that the model fit of the conditional mean ARIMA(1, 0, 4) is adequete.
- `acf(res^2)` shows some autocorrelation in the squared residuals a_t^2 at lag=1, which implies a conditional heteroscedasticity. We can try GARCH fit to this part.

```r
garch.model.logvolume = garchFit(
  formula= ~arma(1,4) + garch(1,1), diff.logvolume)
```

```r
garch.model.logvolume@fit$coef
```

```
##            mu           ar1           ma1           ma2           ma3
##  6.853411e-05  5.257637e-01 -1.000000e+00  5.024231e-02  2.999536e-03
##           ma4         omega        alpha1         beta1
## -7.899813e-03  1.931606e-04  3.749038e-03  9.946603e-01
```

The fitted parameters of ARIMA(1,0,4)/GARCH(1,1) are shown in the list above.

```
arima.fit$aic
```

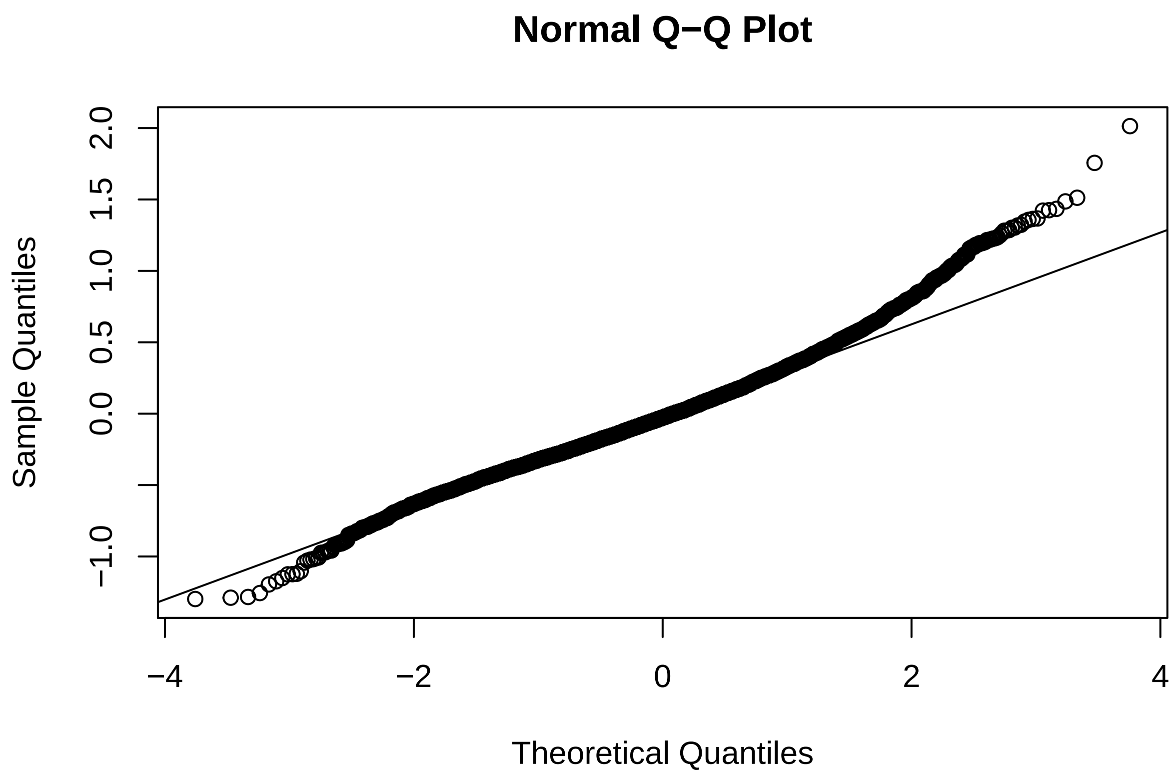
```
## [1] 4364.42
```

```
(garch.model.logvolume@fit$ics) * length(diff.logvolume)
```

```
##      AIC      BIC      SIC     HQIC
## 4363.129 4423.104 4363.101 4383.994
```

The ARIMA(1,0,4) conditional mean model alone has an AIC of 4364.42, while the ARIMA(1,0,4)/GARCH(1,1) model has a smaller AIC of 4363.129 Thus AIC implies that including the GARCH(1,1) component is useful indeed.

```
qqnorm(garch.model.logvolume@residuals)
qqline(garch.model.logvolume@residuals)
```



The QQ plot suggests that the residuals from the ARIMA/GARCH fit is still heavier than normal.

```
aparch.model.logvolume = garchFit(
  formula= ~arma(1,4) + aparch(1,1), diff.logvolume)
```

```
## Warning in sqrt(diag(fit$cvar)): NaNs produced
```

```
aparch.model.logvolume@fit$coef
```

```
##            mu           ar1           ma1           ma2           ma3
##  0.0000529336  0.5255967948 -0.9999999900  0.0503433988  0.0028967156
##           ma4         omega        alpha1        gamma1         beta1
## -0.0075848452  0.0003364403  0.0053547278  0.1443256624  0.9930301131
##         delta
##  1.7692790767
```

```r
arima.fit$aic
```

```
## [1] 4364.42
```

```r
(garch.model.logvolume@fit$ics) * length(diff.logvolume)
```

```
##      AIC      BIC      SIC     HQIC 
## 4363.129 4423.104 4363.101 4383.994
```

```r
(aparch.model.logvolume@fit$ics) * length(diff.logvolume)
```

```
##      AIC      BIC      SIC     HQIC 
## 4371.712 4445.014 4371.670 4397.213
```

The APARCH(1,1) model has a larger AIC of 4371.712. Thus the AIC does not justify this additional complexity. The ARIMA/GARCH model is still the perferable one.