STAT619 project Analysis 2 - Nasdaq-100/QQQ Pricing

Erik Duus

5/5/2021

Retrieve data from Yahoo finance:

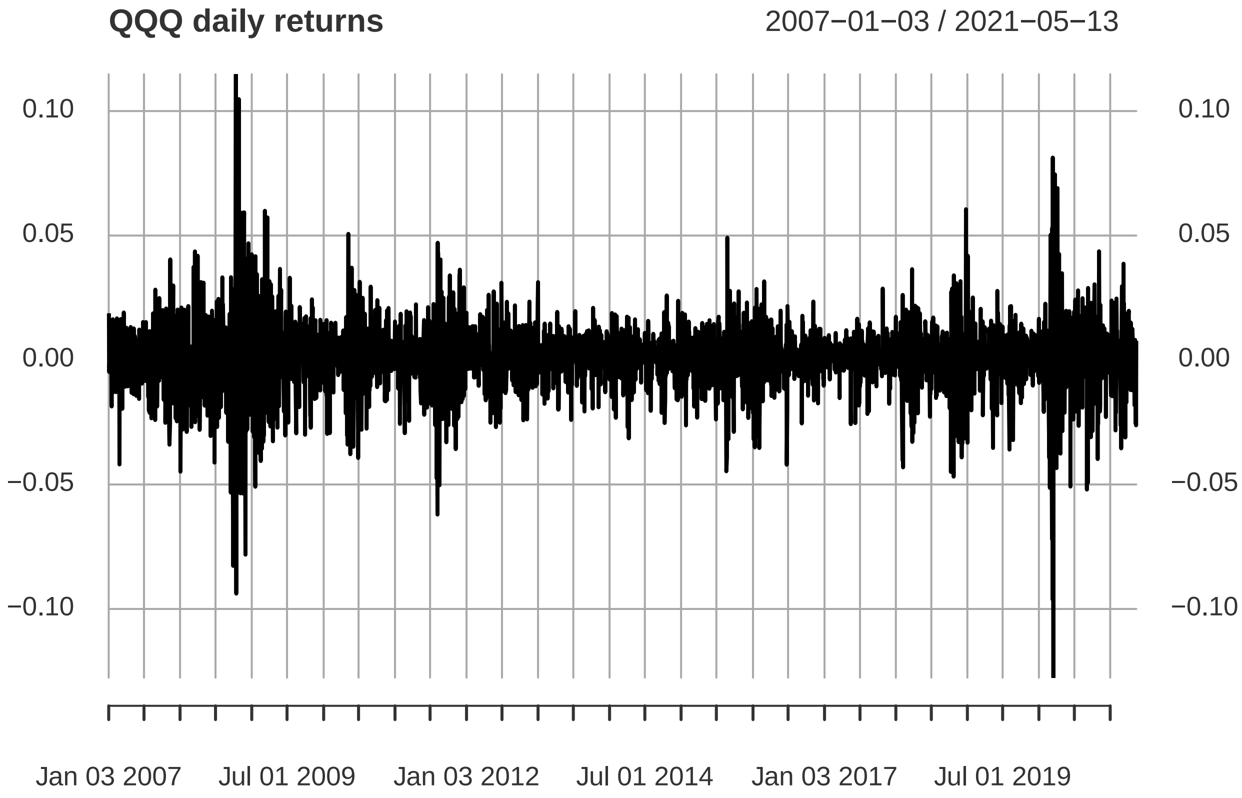
- use simple interpolation to approximate missing values.
- save as a local file so download not required each time

```
options("getSymbols.warning4.0"=FALSE)
options("getSymbols.yahoo.warning"=FALSE)

# get data from YHOO and save to disk

qqq <- getSymbols("QQQ", auto.assign = FALSE)

# which(is.na(btc$'BTC-USD.Close'))
qqq <- na.approx(qqq)

#saveRDS(qqq, './qqq.rds')
```

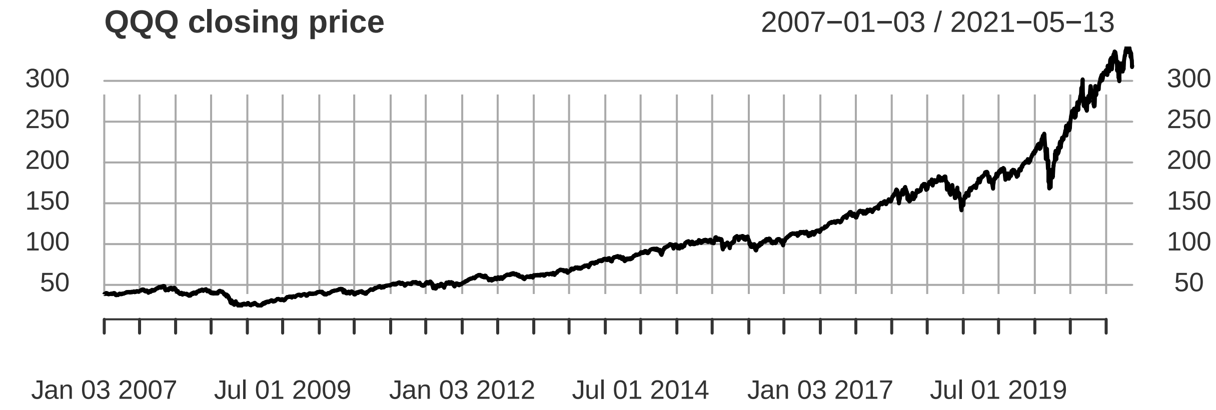
Load local copy of pricing data

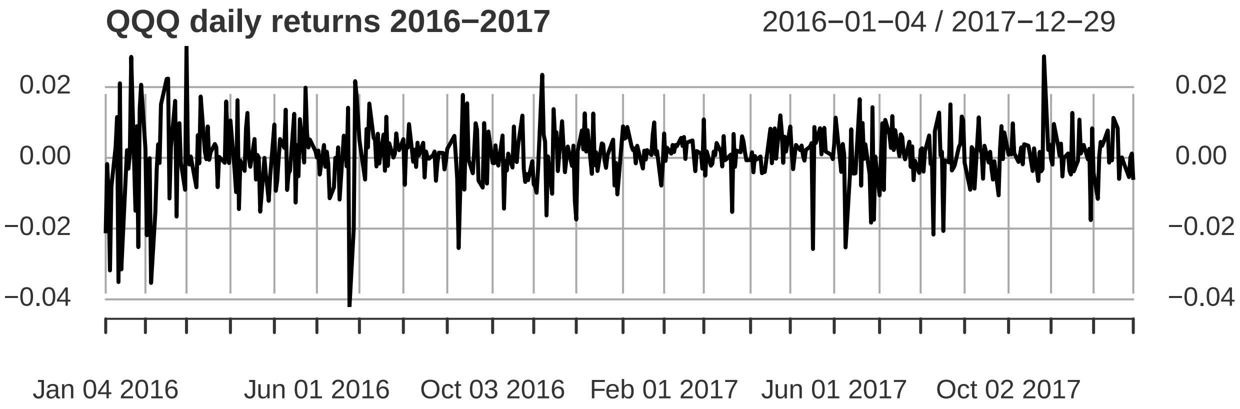
- Examine time series; it is growing exponentially
- Log transform and difference to convert to a returns series
- Mean looks stationary, variance is not constant
- Returns show volatility - GARCH may be needed
- Extract the 2016-2017 window for analysis
- Plot ACF/PACF of returns and squared returns; squared returns confirm GARCH

```
# load saved Yahoo data; use adjusted closing prices
qqqall = readRDS('./qqq.rds')
qqqall <- qqqall$QQQ.Adjusted

qqqr <- diff(log(qqqall))
plot(qqqr, main='QQQ daily returns')
```

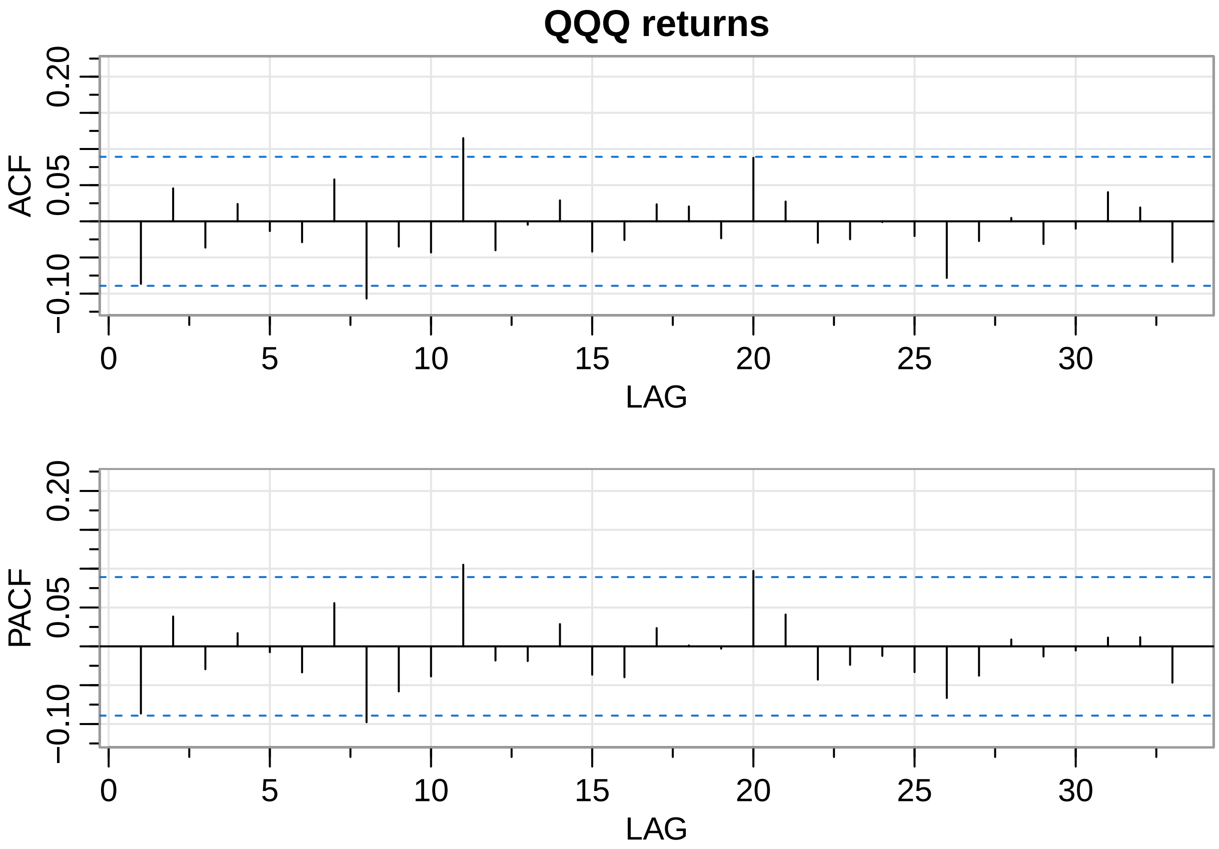


```r
qqq <- window(qqqall, start='2016-01-01',end='2018-01-01')
qqqr <- window(qqqr, start='2016-01-01',end='2018-01-01')

layout(matrix(c(1,1,1,1,
                2,2,2,2),
              nrow=2, byrow=TRUE))

plot(qqqall, main='QQQ closing price')
plot(qqqr, main='QQQ daily returns 2016-2017')
```



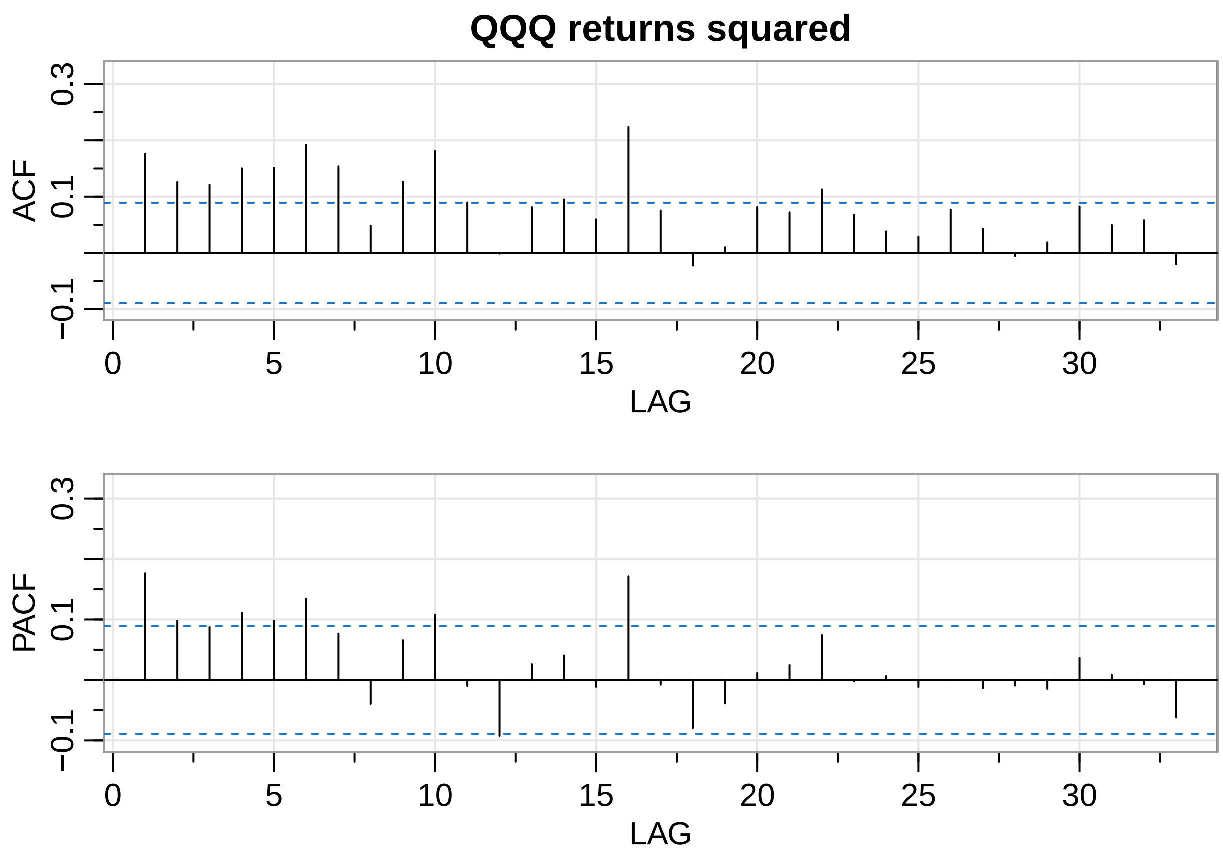


```r
acf2(qqqr, main='QQQ returns')
```



```
##       [,1] [,2]  [,3] [,4]  [,5]  [,6] [,7]  [,8]  [,9] [,10] [,11] [,12] [,13]
## ACF  -0.09 0.05 -0.04 0.02 -0.01 -0.03 0.06 -0.11 -0.03 -0.04  0.12 -0.04  0.00
## PACF -0.09 0.04 -0.03 0.02 -0.01 -0.03 0.06 -0.10 -0.06 -0.04  0.11 -0.02 -0.02
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.03 -0.04 -0.03  0.02  0.02 -0.02  0.09  0.03 -0.03 -0.02  0.00 -0.02
## PACF  0.03 -0.04 -0.04  0.02  0.00  0.00  0.10  0.04 -0.04 -0.02 -0.01 -0.03
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33]
## ACF  -0.08 -0.03  0.00 -0.03 -0.01  0.04  0.02 -0.06
## PACF -0.07 -0.04  0.01 -0.01 -0.01  0.01  0.01 -0.05
```

```
acf2(qqqrˆ2, main=’QQQ returns squared’)
```



```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  0.18 0.13 0.12 0.15 0.15 0.19 0.15  0.05 0.13  0.18  0.09  0.00  0.08
## PACF 0.18 0.10 0.09 0.11 0.10 0.13 0.08 -0.04 0.07  0.11 -0.01 -0.09  0.03
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.10  0.06  0.22  0.08 -0.02  0.01  0.08  0.07  0.11  0.07  0.04  0.03
## PACF  0.04 -0.01  0.17 -0.01 -0.08 -0.04  0.01  0.02  0.07  0.00  0.01 -0.01
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33]
## ACF   0.08  0.04 -0.01  0.02  0.08  0.05  0.06 -0.02
## PACF  0.00 -0.01 -0.01 -0.01  0.04  0.01 -0.01 -0.06
```

confirm stationarity of mean through unit root tests

```
# null hypothesis not stationary
adf.test(qqqr, k=0)
```

```
## Warning in adf.test(qqqr, k = 0): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  qqqr
## Dickey-Fuller = -24.551, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(qqqr)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  qqqr
## Dickey-Fuller = -9.3445, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
pp.test(qqqr)
```

```
## Warning in pp.test(qqqr): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  qqqr
## Dickey-Fuller Z(alpha) = -543.77, Truncation lag parameter = 5, p-value
## = 0.01
## alternative hypothesis: stationary
```

```r
# null hypothesis stationary
kpss.test(qqqr)
```

```
## Warning in kpss.test(qqqr): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  qqqr
## KPSS Level = 0.14991, Truncation lag parameter = 5, p-value = 0.1
```

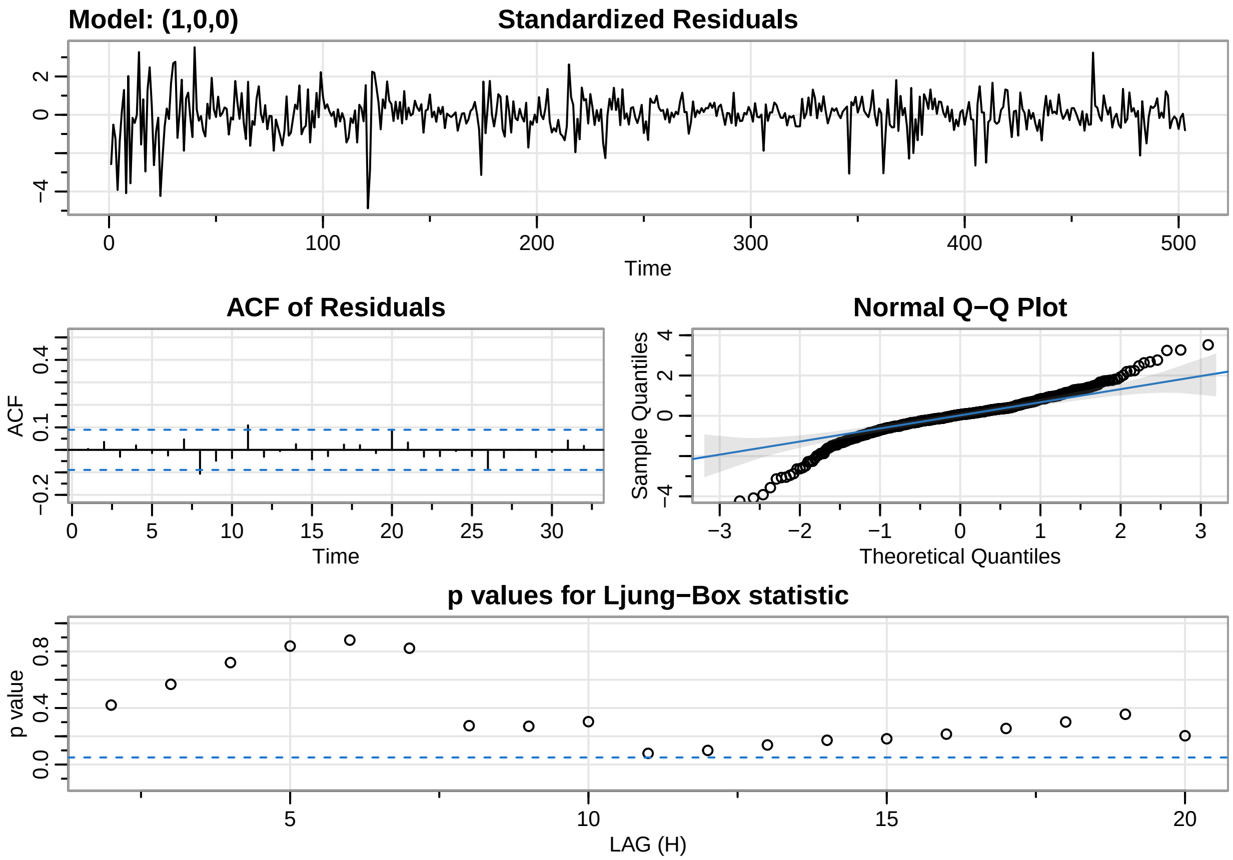
Run auto.arima to see what it suggests;diagnostics of fitted model also show variance in residuals, confirming garch.

```r
auto.arima(qqqr, seasonal=FALSE)
```

```
## Series: qqqr
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##           ar1   mean
##       -0.0876  7e-04
## s.e.   0.0447  4e-04
##
## sigma^2 estimated as 7.302e-05:  log likelihood=1682.76
## AIC=-3359.52   AICc=-3359.47   BIC=-3346.86
```

```r
sarima(qqqr, 1,0,0)
```

```
## initial  value -4.766209
## iter   2 value -4.770027
## iter   3 value -4.770031
## iter   4 value -4.770036
## iter   5 value -4.770037
## iter   5 value -4.770037
## iter   5 value -4.770037
## final  value -4.770037
## converged
## initial  value -4.764377
## iter   2 value -4.764382
## iter   3 value -4.764388
## iter   4 value -4.764388
## iter   4 value -4.764388
## iter   4 value -4.764388
## final  value -4.764388
## converged
```



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = xmean, include.mean = FALSE, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1  xmean
##       -0.0876  7e-04
## s.e.   0.0447  4e-04
##
## sigma^2 estimated as 7.273e-05:  log likelihood = 1682.76,  aic = -3359.52
##
## $degrees_of_freedom
## [1] 501
##
## $ttable
##       Estimate     SE t.value p.value
## ar1    -0.0876 0.0447 -1.9602  0.0505
## xmean   0.0007 0.0004  1.9938  0.0467
##
## $AIC
## [1] -6.678971
##
## $AICc
## [1] -6.678924
##
## $BIC
## [1] -6.653799
```

ACF and PACF of series shows a tiny bit of auto-correlation, suggesting ARMA(1,1)

ACF and PACF of squared series shows both decaying, suggesting GARCH(1,1)

Fit GARCH(1,1) to start as a baseline

```
gf <- garchFit(~garch(1,1), data=qqqr, cond.dist='std', trace=FALSE)
```

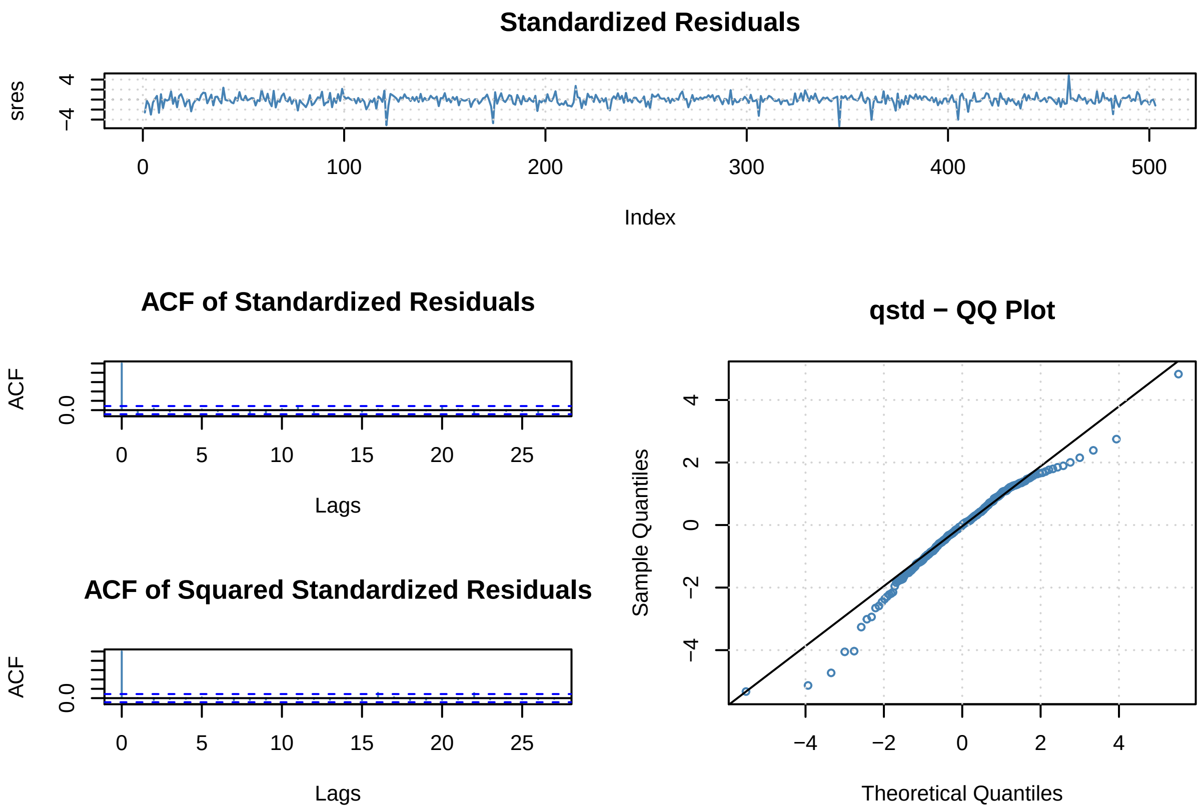
```
## Warning: Using formula(x) is deprecated when x is a character vector of length > 1.
##   Consider formula(paste(x, collapse = " ")) instead.
```

```
summary(gf)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = qqqr, cond.dist = "std",
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x0000000021ff0328>
##  [data = qqqr]
##
```

```
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu       omega      alpha1       beta1       shape
## 1.1094e-03  2.5634e-06  1.3668e-01  8.5113e-01  3.4234e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.109e-03   2.560e-04    4.334 1.46e-05 ***
## omega  2.563e-06   1.683e-06    1.523  0.12767
## alpha1 1.367e-01   5.250e-02    2.604  0.00923 **
## beta1  8.511e-01   4.806e-02   17.709  < 2e-16 ***
## shape  3.423e+00   5.608e-01    6.104 1.03e-09 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Log Likelihood:
##  1763.356    normalized:  3.505678
##
## Description:
##  Tue May 25 19:42:04 2021 by user: orovi
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  718.8286  0
##  Shapiro-Wilk Test  R    W      0.9216501 1.64025e-15
##  Ljung-Box Test     R    Q(10)  10.40282  0.4058908
##  Ljung-Box Test     R    Q(15)  15.14452  0.4410579
##  Ljung-Box Test     R    Q(20)  17.57307  0.6155093
##  Ljung-Box Test     R^2  Q(10)  3.521033  0.9663794
##  Ljung-Box Test     R^2  Q(15)  5.491048  0.9870892
##  Ljung-Box Test     R^2  Q(20)  14.55135  0.8014781
##  LM Arch Test       R    TR^2   5.347455  0.945356
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -6.991476 -6.949522 -6.991671 -6.975018
```

```r
layout(matrix(c(1,1,1,1,
                1,1,1,1,
                2,2,4,4,
                2,2,4,4,
                3,3,4,4,
                3,3,4,4),nrow=6, byrow=TRUE))
plot(gf, which=9)
plot(gf, which=10)
plot(gf, which=11)
plot(gf, which=13)
```



model coefficients are significant, but QQ-plot is not that normal. perhaps ARMA features are required. several iterations get us to ARMA(5,3)+GARCH(1,1)

** (NOTE ARMA(5,3) is best model but wont predict correctly) **

```
gf53 <- garchFit(~arma(5,3)+garch(1,1), data=qqqr ,cond.dist='std', trace=FALSE)
```

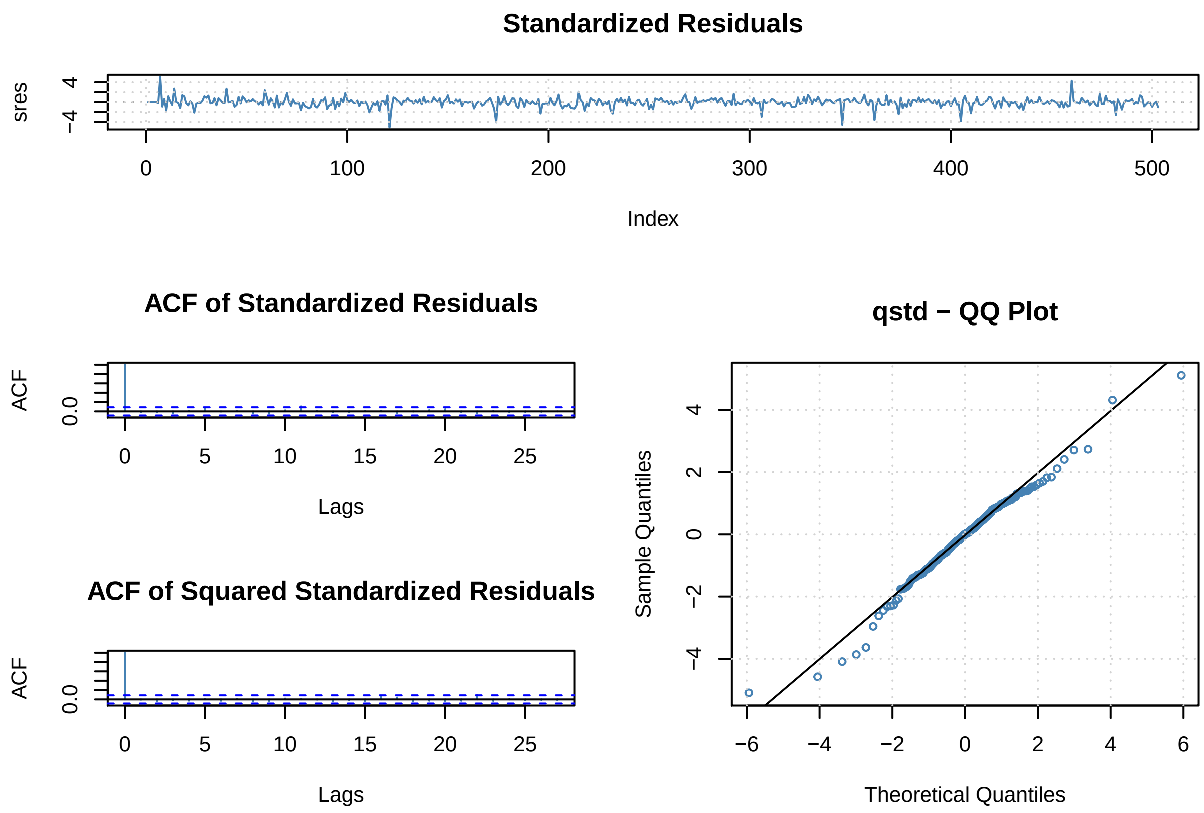
```
## Warning: Using formula(x) is deprecated when x is a character vector of length > 1.
##   Consider formula(paste(x, collapse = " ")) instead.
```

```
summary(gf53)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(5, 3) + garch(1, 1), data = qqqr, cond.dist = "std",
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(5, 3) + garch(1, 1)
## <environment: 0x00000000203f5f40>
##  [data = qqqr]
```

```
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##          mu          ar1          ar2          ar3          ar4          ar5
##  2.6551e-04   4.2359e-01  -3.9607e-01   7.7965e-01   9.3830e-02  -1.2449e-01
##         ma1          ma2          ma3        omega       alpha1        beta1
## -5.1537e-01   5.2407e-01  -8.9357e-01   4.4330e-06   1.9124e-01   8.1233e-01
##       shape
##  2.9658e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      2.655e-04   7.909e-05    3.357 0.000788 ***
## ar1     4.236e-01   5.116e-02    8.280 2.22e-16 ***
## ar2    -3.961e-01   5.100e-02   -7.767 7.99e-15 ***
## ar3     7.796e-01   4.196e-02   18.581  < 2e-16 ***
## ar4     9.383e-02   4.548e-02    2.063 0.039090 *
## ar5    -1.245e-01   4.165e-02   -2.989 0.002797 **
## ma1    -5.154e-01   2.697e-02  -19.112  < 2e-16 ***
## ma2     5.241e-01   2.898e-02   18.086  < 2e-16 ***
## ma3    -8.936e-01   2.901e-02  -30.804  < 2e-16 ***
## omega   4.433e-06   2.821e-06    1.572 0.116065
## alpha1  1.912e-01   8.732e-02    2.190 0.028522 *
## beta1   8.123e-01   6.315e-02   12.863  < 2e-16 ***
## shape   2.966e+00   5.067e-01    5.854 4.81e-09 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Log Likelihood:
##  1783.878    normalized:  3.546477
##
## Description:
##  Tue May 25 19:42:05 2021 by user: orovi
##
##
## Standardised Residuals Tests:
##                                  Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  993.4853  0
##  Shapiro-Wilk Test  R    W      0.9085347 0
##  Ljung-Box Test     R    Q(10)  6.499543  0.7716946
##  Ljung-Box Test     R    Q(15)  12.9058   0.6095736
##  Ljung-Box Test     R    Q(20)  15.32202  0.7576944
##  Ljung-Box Test     R^2  Q(10)  3.644229  0.9619728
##  Ljung-Box Test     R^2  Q(15)  5.527343  0.9866417
##  Ljung-Box Test     R^2  Q(20)  10.38313  0.9606809
##  LM Arch Test       R    TR^2   4.768475  0.9652682
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -7.041263 -6.932182 -7.042555 -6.998471
```

```
layout(matrix(c(1,1,1,1,
                1,1,1,1,
                2,2,4,4,
                2,2,4,4,
                3,3,4,4,
                3,3,4,4),nrow=6, byrow=TRUE))
plot(gf53, which=9)
plot(gf53, which=10)
plot(gf53, which=11)
plot(gf53, which=13)
```



QQ plot is much better. residual plot and ACF plots suggest residuals are white noise. Ljung-Box tests are also suggestive of white noise residuals.

Conclude this model is accurate and use for predictions.

NOTE: fgarch package produces errors on this model when forecasting! revert to ARMA(2,2)+GARCH(1,1) for forecast

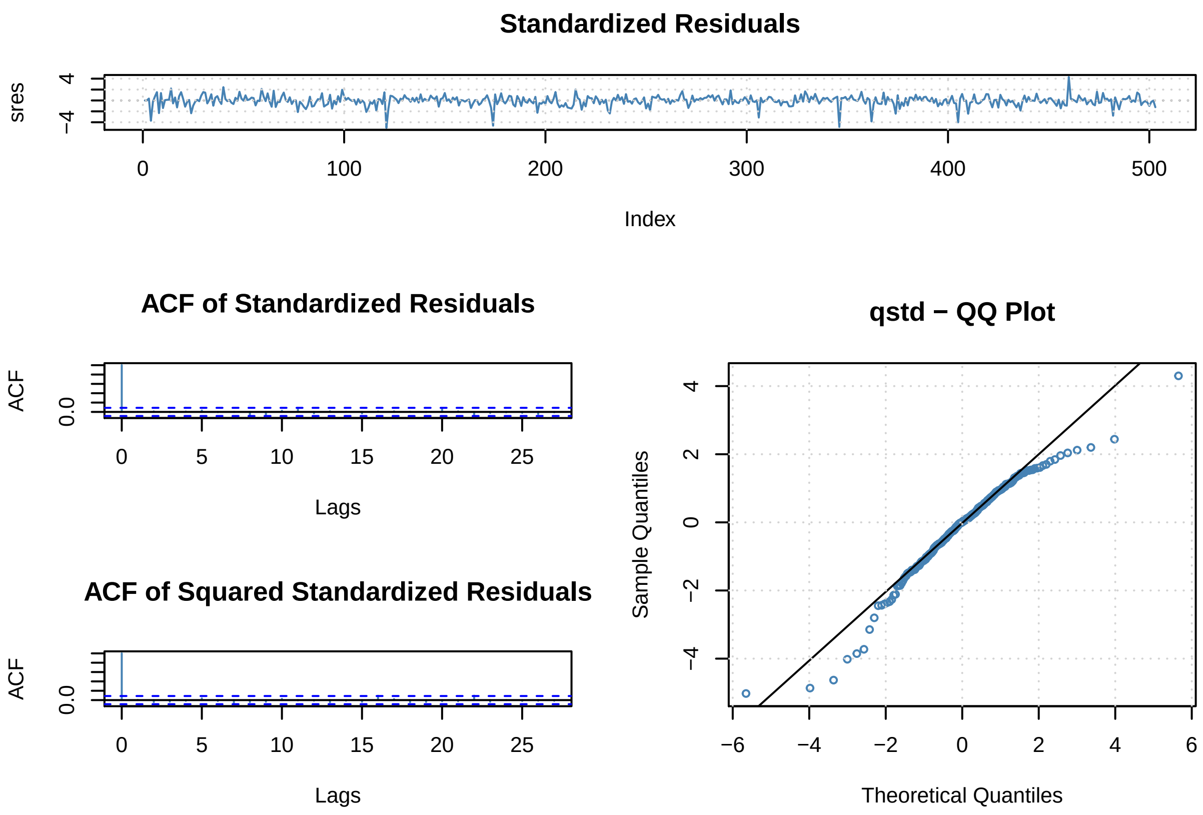
```
gf1 <- garchFit(~arma(2,2)+garch(1,1), data=qqqr ,cond.dist='std', trace=FALSE)
```

```
## Warning in log(s2): NaNs produced
## Warning: Using formula(x) is deprecated when x is a character vector of length > 1.
##   Consider formula(paste(x, collapse = " ")) instead.
```

```r
summary(gf1)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(2, 2) + garch(1, 1), data = qqqr, cond.dist = "std",
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(2, 2) + garch(1, 1)
## <environment: 0x000000002028f0c8>
##  [data = qqqr]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##           mu          ar1          ar2          ma1          ma2        omega
##   0.00013786   0.32645418   0.55550751  -0.44149180  -0.50703767   0.00000318
##       alpha1        beta1        shape
##   0.15315068   0.83699753   3.27412877
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      1.379e-04   7.813e-05    1.764   0.0777 .
## ar1     3.265e-01   3.373e-01    0.968   0.3332
## ar2     5.555e-01   2.868e-01    1.937   0.0528 .
## ma1    -4.415e-01   3.408e-01   -1.296   0.1951
## ma2    -5.070e-01   3.019e-01   -1.680   0.0930 .
## omega   3.180e-06   2.086e-06    1.525   0.1273
## alpha1  1.532e-01   6.236e-02    2.456   0.0140 *
## beta1   8.370e-01   5.450e-02   15.357  < 2e-16 ***
## shape   3.274e+00   5.515e-01    5.937  2.9e-09 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Log Likelihood:
##  1772.294    normalized:  3.523447
##
## Description:
##  Tue May 25 19:42:05 2021 by user: orovi
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  645.5674  0
##  Shapiro-Wilk Test  R    W      0.9224773 2.011307e-15
##  Ljung-Box Test     R    Q(10)  5.670213  0.8421679
##  Ljung-Box Test     R    Q(15)  10.91722  0.7584403
##  Ljung-Box Test     R    Q(20)  13.63381  0.848567
##  Ljung-Box Test     R^2  Q(10)  5.907832  0.8229464
##  Ljung-Box Test     R^2  Q(15)  7.748837  0.9334568
##  Ljung-Box Test     R^2  Q(20)  14.57169  0.8003688
##  LM Arch Test       R    TR^2   6.279338  0.901351
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -7.011109 -6.935592 -7.011735 -6.981484
```

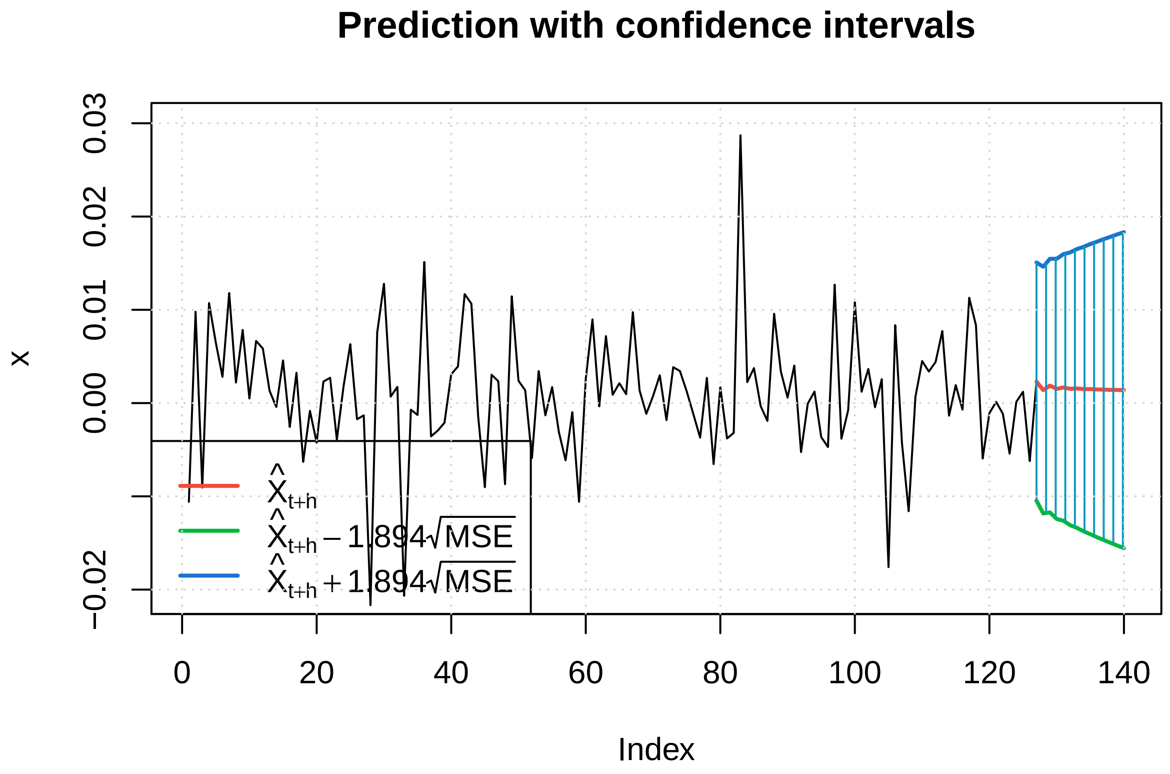
```r
layout(matrix(c(1,1,1,1,
                1,1,1,1,
                2,2,4,4,
                2,2,4,4,
                3,3,4,4,
                3,3,4,4),nrow=6, byrow=TRUE))
plot(gf1, which=9)
plot(gf1, which=10)
plot(gf1, which=11)
plot(gf1, which=13)
```



predict 14 days of future returns. recall this is a daily returns series, so these are predictions of future daily returns.

```
### 14 day predictions of returns - PLOT
preds <- predict(gf1, n.ahead=14, plot=TRUE)
```

```
## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length

## Warning in a_vec[i] <- ar[1:min(u2, i - 1)] * a_vec[(i - 1):(i - u2)] + : number
## of items to replace is not a multiple of replacement length
```

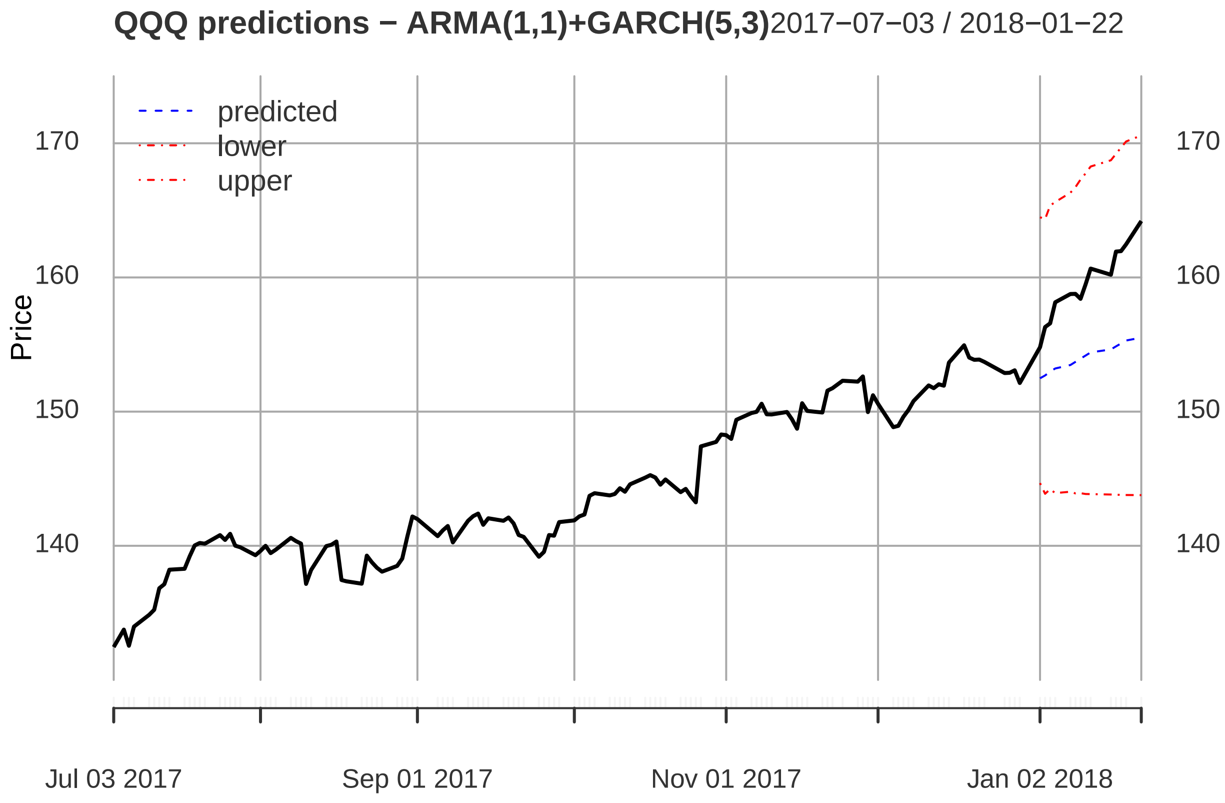


```
preds
```

```
##    meanForecast   meanError standardDeviation lowerInterval upperInterval
## 1   0.002305051 0.006739107       0.006739107   -0.01046200    0.01507210
## 2   0.001392226 0.006982061       0.006938888   -0.01183509    0.01461954
## 3   0.001872829 0.007176102       0.007131188   -0.01172209    0.01546775
## 4   0.001522642 0.007362891       0.007316616   -0.01242615    0.01547143
## 5   0.001675301 0.007543253       0.007495699   -0.01261518    0.01596578
## 6   0.001530606 0.007717683       0.007668898   -0.01309033    0.01615154
## 7   0.001568173 0.007886595       0.007836621   -0.01337276    0.01650910
## 8   0.001500058 0.008050350       0.007999227   -0.01375110    0.01675122
## 9   0.001498690 0.008209275       0.008157038   -0.01405355    0.01705093
## 10  0.001460404 0.008363659       0.008310342   -0.01438431    0.01730512
## 11  0.001447146 0.008513765       0.008459398   -0.01468194    0.01757623
## 12  0.001421550 0.008659828       0.008604443   -0.01498425    0.01782735
## 13  0.001405829 0.008802065       0.008745689   -0.01526943    0.01808109
## 14  0.001386479 0.008940671       0.008883331   -0.01555137    0.01832433
```

to produce pricing predictions the returns series needs to be applied to the tail of the price series, and then undifferenced and exp() applied.

```
# convert forecast means and intervals to prices
mf <- diffinv(preds$meanForecast, xi=log(qqq[503]))
mf <- mf[2:15] # drop first seed value (from original series)
li <- mf * (1 + preds$lowerInterval)
ui <- mf * (1 + preds$upperInterval)
mf <- exp(mf)
li <- exp(li)
ui <- exp(ui)

# get original time series plus the actual values for prediction interval
# create prediction, upper and lower interval series aligned with actuals
qqq_true <- window(qqqall, start=’2017-07-01’,end=’2018-01-22’)
qqq_mf <- tail(qqq_true, 14)
qqq_mf[,1] <- mf
qqq_li <- tail(qqq_true, 14)
qqq_li[,1] <- li
qqq_ui <- tail(qqq_true, 14)
qqq_ui[,1] <- ui


plt <- plot(ylim=c(130,175), qqq_true, type=’l’, main=’QQQ predictions - ARMA(1,1)+GARCH(5,3)’, ylab=’P
plt <- lines(qqq_mf, lty=2, lwd=1, col=’blue’)
plt <- lines(qqq_li, lty=4, lwd=1, col=’red’)
plt <- lines(qqq_ui, lty=4, lwd=1, col=’red’)
plt <- addLegend(’topleft’, legend.names=c(’predicted’,’lower’, ’upper’),
          lty=c(2,4,4), lwd=c(1,1,1),col=c(’blue’, ’red’, ’red’))
plt
```



compute RMSE of predictions

```
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 4.0.5
```

```
##
## Attaching package: 'Metrics'
```

```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
rmse(tail(qqq_true,14), tail(qqq_mf,14))
```

```
## [1] 5.678657
```