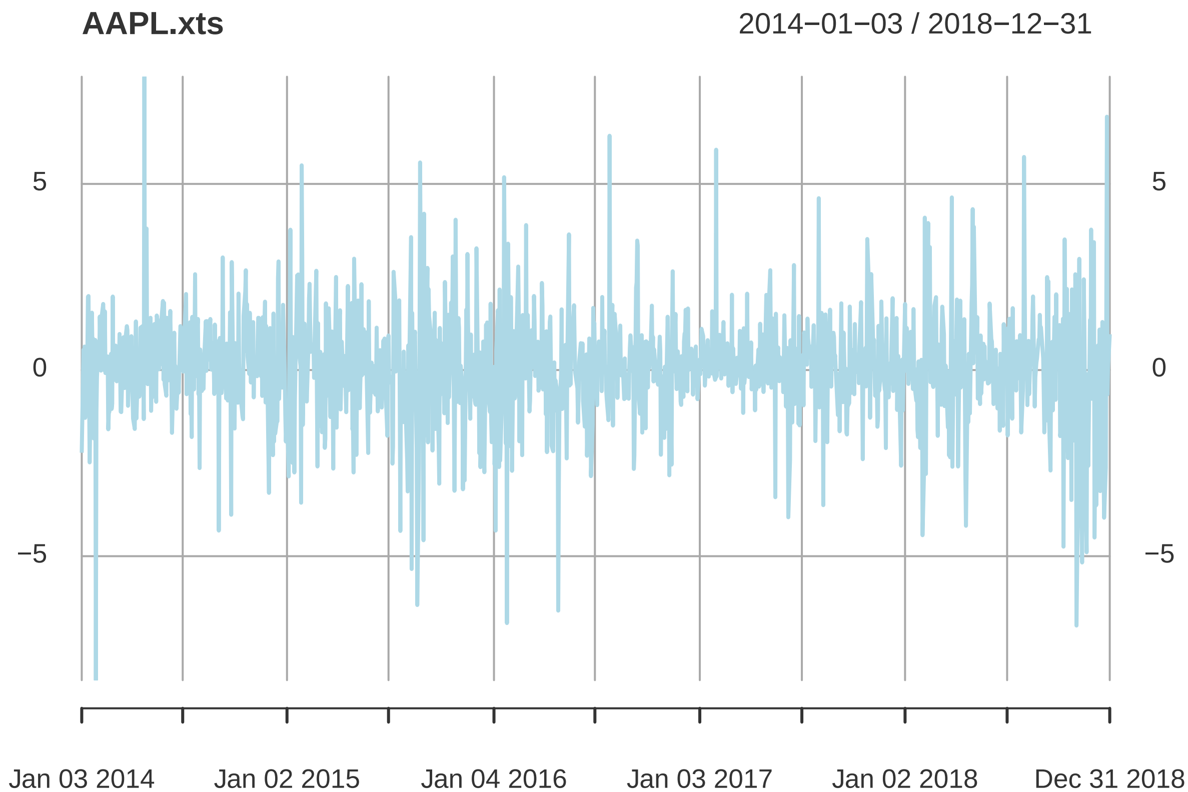
5261Project

yz3380

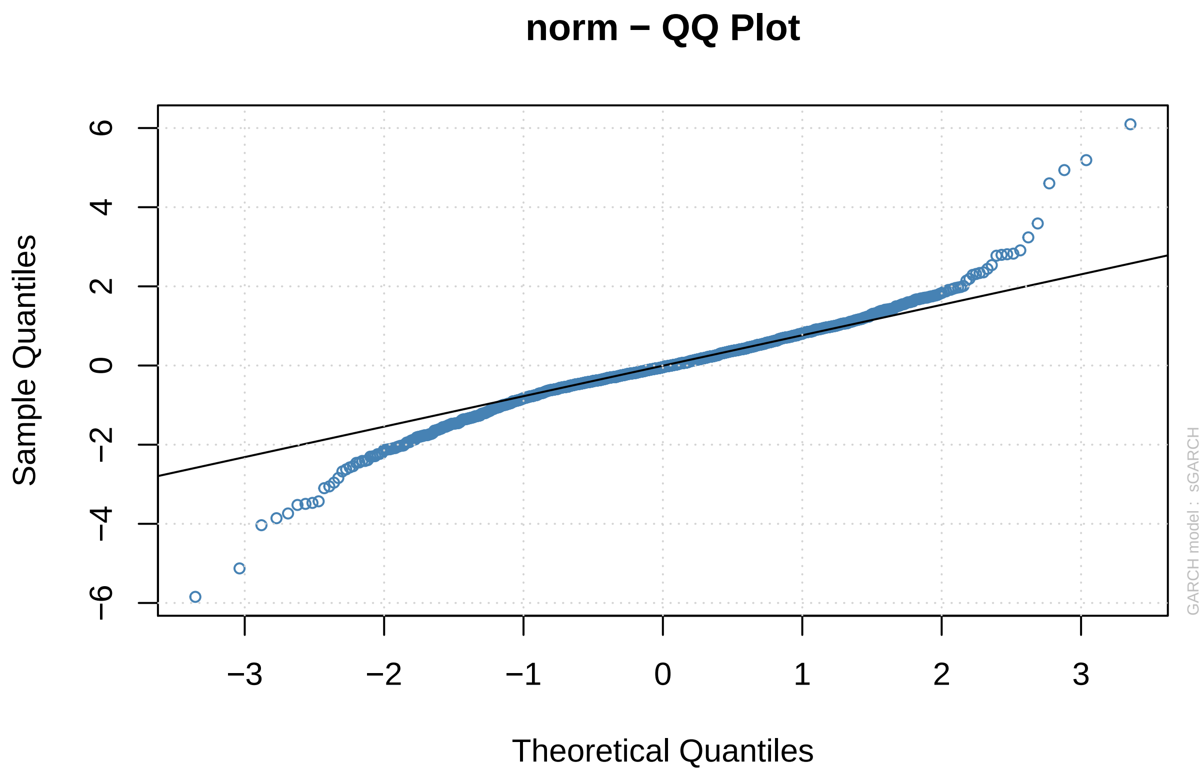
April 14, 2019

GARCH Model in estimating VaR

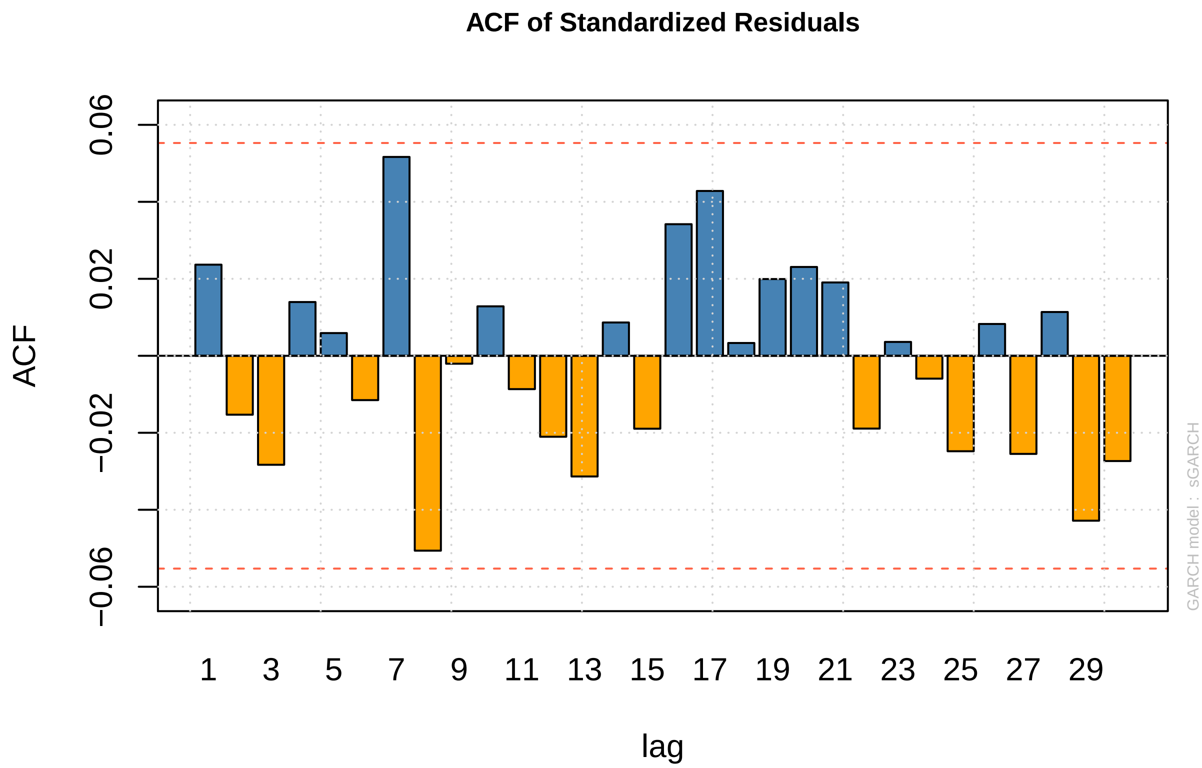
```
# read stock price data
return <- read.csv("closedata_return.csv")
return[-1] <- return[-1] + 1
return$Date <- as.Date(return$Date, "%m/%d/%Y")
colnames(return) <- c("Date", "AAPL", "AMZN", "FB", "GM", "NFLX", "SPY")
Date <- return[,1] # save the date
rownames(return) <- return[,1]
return <- return[,-1]
n <- nrow(return)

# transform price data into 100*log return Time series
return<- 100*log(return)
AAPL.xts <- na.omit(xts(x = return$AAPL, order.by = Date))
AMZN.xts <- na.omit(xts(x = return$AMZN, order.by = Date))
FB.xts <- na.omit(xts(x = return$FB, order.by = Date))
GM.xts <- na.omit(xts(x = return$GM, order.by = Date))
NFLX.xts <- na.omit(xts(x = return$NFLX, order.by = Date))
SPY.xts <- na.omit(xts(x = return$SPY, order.by = Date))
# Other types of financial data we would like to use
# Exchange rate, USD/JPY, USD/CNY
# Bond price 10yrs TSY
# Market Index, SP500, HSI, NIKKEI225, DAX30, CAC40, FTSE100


# plot the return time series
#plot(AAPL~Date, data=return, type='l', xlab='Date', ylab='Log Return')
plot(AAPL.xts, col='lightblue')
```



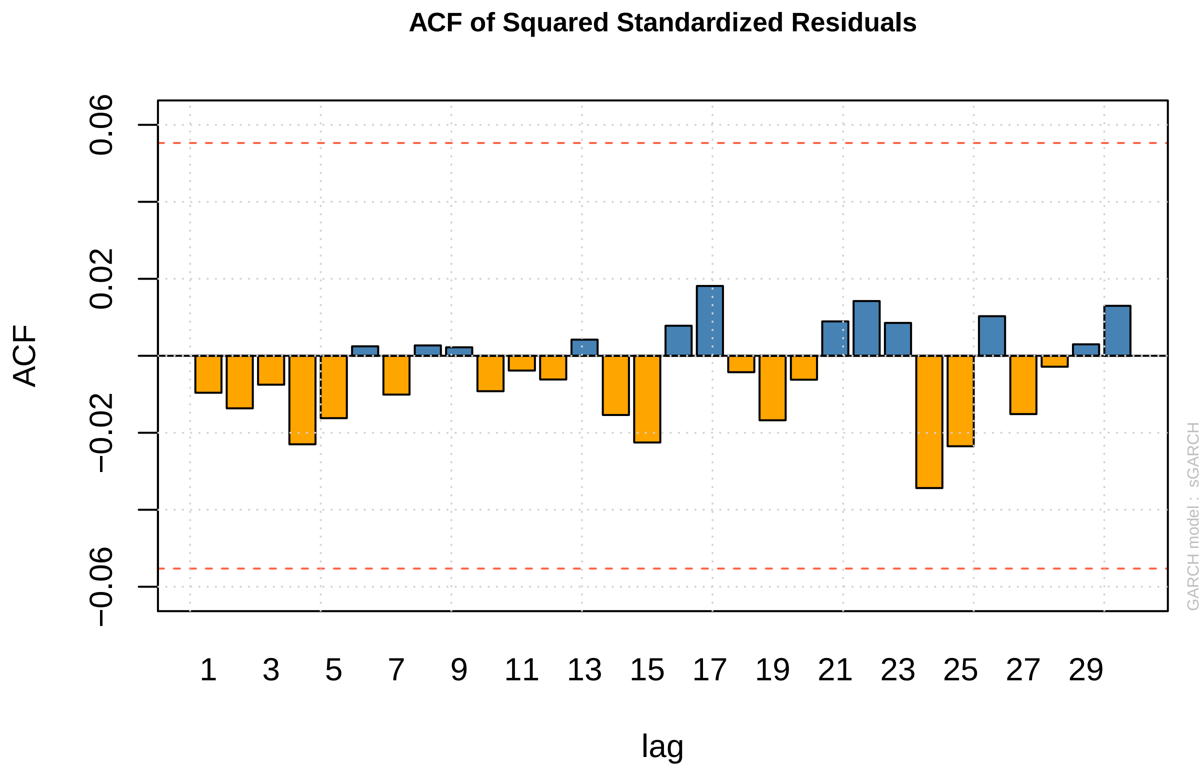

```r
# fit the Garch model for AAPL

# specify our model
# fit ARMA-GARCH model with normal error
model1 <- ugarchspec(variance.model=list(model="sGARCH", garchOrder=c(1, 1)), mean.model=list(armaOrder=
model1_fit <- ugarchfit(spec=model1, data=AAPL.xts, solver.control=list(trace=0))

# Model goodness of fit
#qqnorm(residuals(model1_fit))
#abline(0,1)
plot(model1_fit, which=9) # qqplot
```




```
plot(model1_fit, which=10) # acf of residuals
```




```r
plot(model1_fit, which=11) # acf of squared residuals
```




```r
# back testing VaR
VaR_1 <- quantile(model1_fit, 0.01) # train VaR / in-sample VaR
mean(return$AAPL<VaR_1) # VaR violation rate / back-testing
```

```
## [1] 0.01750199
```

```r
# fit ARMA-GARCH model with t error
modelt <- ugarchspec(mean.model=list(armaOrder=c(1,1)), variance.model=list(garchOrder=c(1,1)), distribu
modelt_fit <- ugarchfit(data=AAPL.xts, spec=modelt)
show(modelt_fit) # check the Pearson Goodness-of-Fit test, it works well
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.10631    0.032478   3.2734 0.001063
## ar1     -0.84500    0.273705  -3.0873 0.002020
```

```
## ma1      0.85155    0.267797   3.1798 0.001474
## omega    0.16270    0.066592   2.4432 0.014560
## alpha1   0.13618    0.039232   3.4710 0.000518
## beta1    0.81203    0.051787  15.6804 0.000000
## shape    3.92927    0.457960   8.5799 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.10631    0.032344   3.2869 0.001013
## ar1     -0.84500    0.121460  -6.9570 0.000000
## ma1      0.85155    0.117299   7.2597 0.000000
## omega    0.16270    0.080258   2.0271 0.042647
## alpha1   0.13618    0.041345   3.2937 0.000989
## beta1    0.81203    0.063686  12.7507 0.000000
## shape    3.92927    0.453645   8.6616 0.000000
##
## LogLikelihood : -2160.308
##
## Information Criteria
## ------------------------------------
##
## Akaike       3.4484
## Bayes        3.4770
## Shibata      3.4483
## Hannan-Quinn 3.4591
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.119  0.2901
## Lag[2*(p+q)+(p+q)-1][5]     2.208  0.9050
## Lag[4*(p+q)+(p+q)-1][9]     4.146  0.6564
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2482  0.6183
## Lag[2*(p+q)+(p+q)-1][5]    1.2191  0.8084
## Lag[4*(p+q)+(p+q)-1][9]    1.8948  0.9172
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.3092 0.500 2.000  0.5782
## ARCH Lag[5]    1.2069 1.440 1.667  0.6727
## ARCH Lag[7]    1.5316 2.315 1.543  0.8150
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.3557
## Individual Statistics:
```

```
## mu     0.05198
## ar1    0.03132
## ma1    0.03145
## omega  0.40020
## alpha1 0.33895
## beta1  0.37946
## shape  0.44975
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.3943 0.6934
## Negative Sign Bias  0.2277 0.8199
## Positive Sign Bias  0.8600 0.3900
## Joint Effect        2.5929 0.4587
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     21.49       0.3104
## 2    30     26.70       0.5879
## 3    40     38.05       0.5130
## 4    50     43.20       0.7064
##
##
## Elapsed time : 0.287267
```

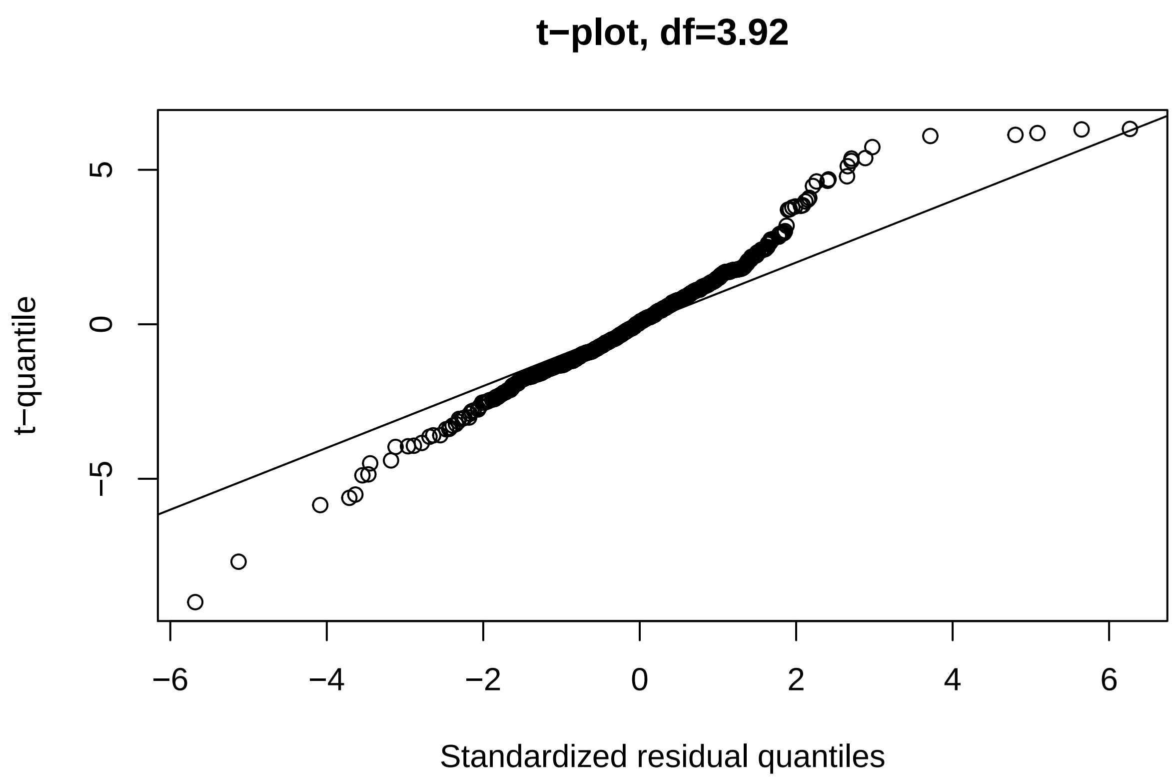
```
# MLE t fit to residuals
err1 <- as.vector(residuals(modelt_fit, standardize=TRUE))
fitdistr(err1,"t")
```

```
##         m              s             df
##   -0.01788028    0.70076277    3.91246206
##  ( 0.02342861) ( 0.02528813) ( 0.46069466)
```

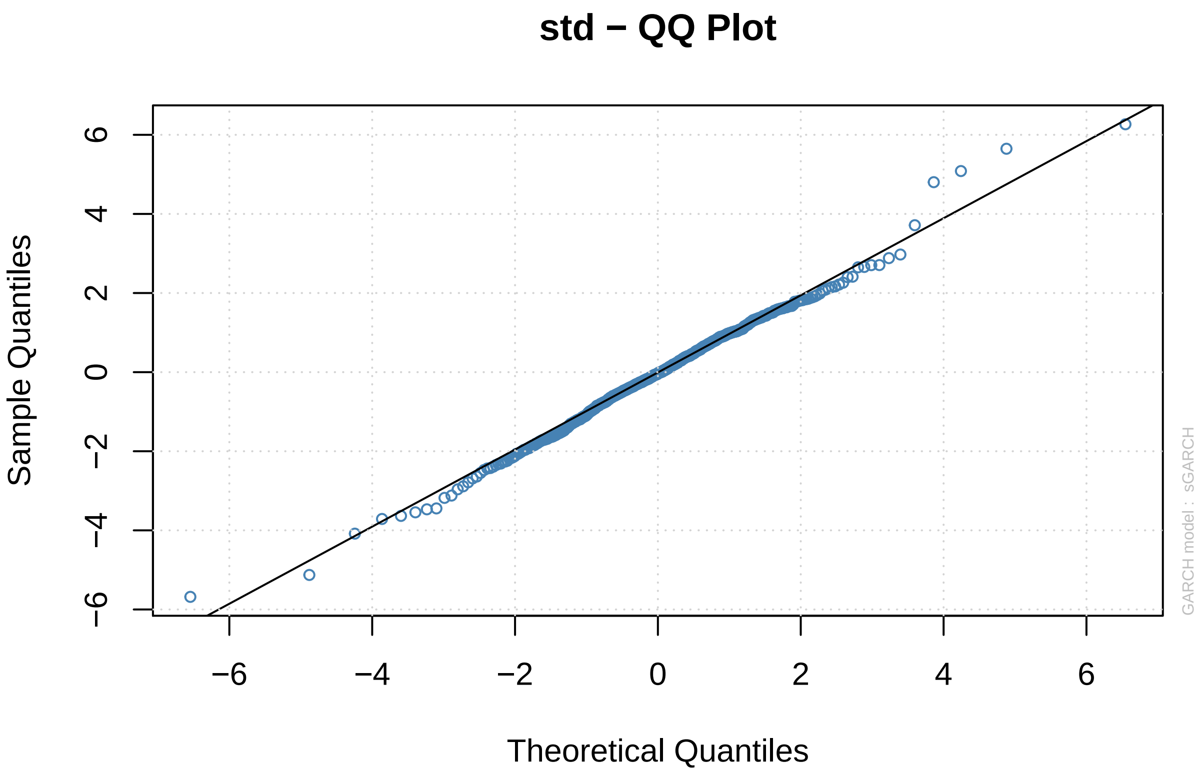
```
modelt_fit@fit$coef
```

```
##         mu        ar1        ma1      omega     alpha1      beta1
##  0.1063130 -0.8449987  0.8515487  0.1626952  0.1361755  0.8120345
##      shape
##  3.9292715
```

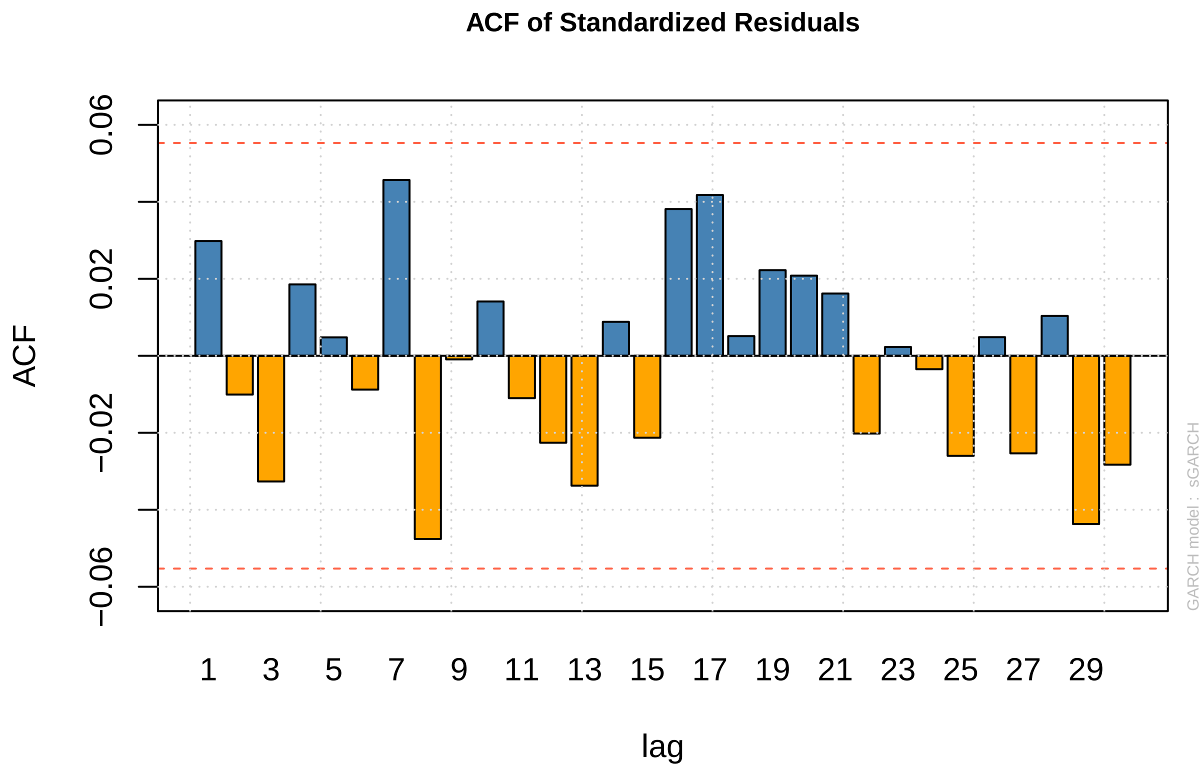
```
qqplot(sort(err1), sort(rt(1257,df=modelt_fit@fit$coef[7])), main='t-plot, df=3.92', ylab='t-quantile',
abline(0, 1)
```



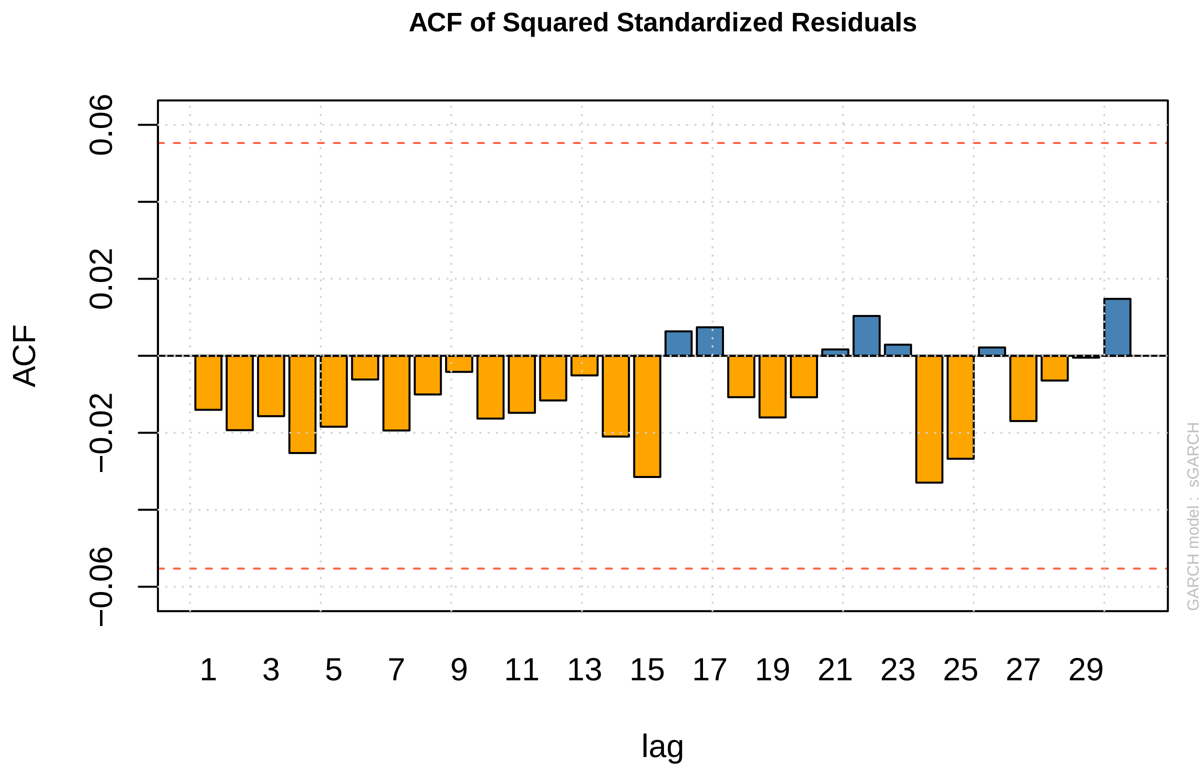

```
plot(modelt_fit, which=9)

# acf of residuals, not satisfying , but from Ljung-Box test we can assume no serial correlation in res
plot(modelt_fit, which=9) # qqplot
```




```
plot(modelt_fit, which=10) # acf of residuals
```

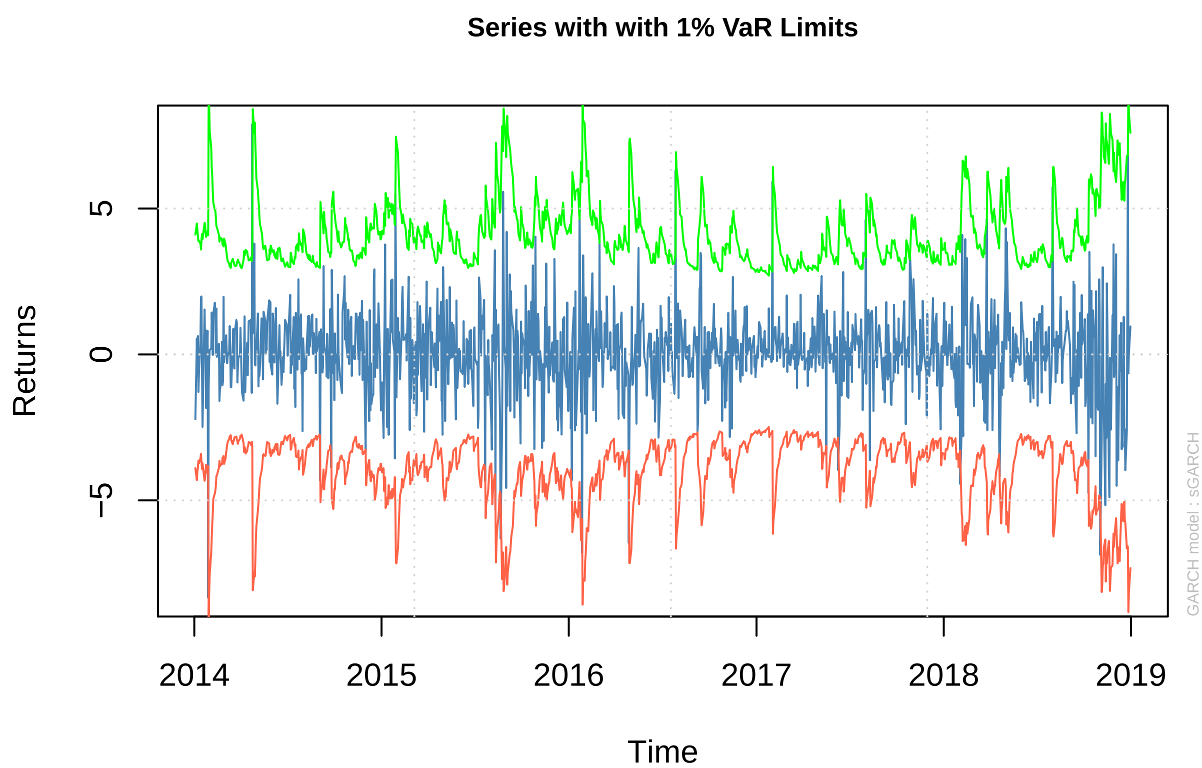



```
plot(modelt_fit, which=11) # acf of squared residuals
```




```
plot(modelt_fit,which=2) # Plot series with 1% VaR limits
```

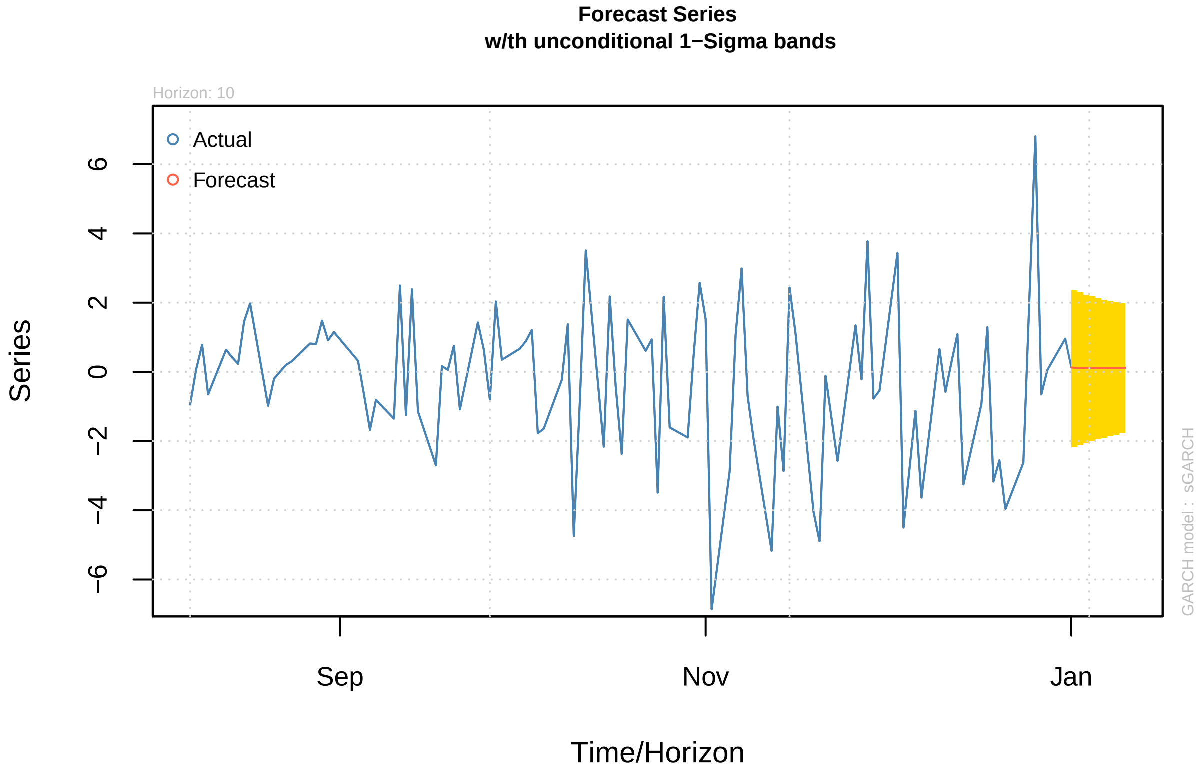
```
##
## please wait...calculating quantiles...
```




```r
VaR_2 <- quantile(modelt_fit, 0.01) # train VaR / in-sample VaR
mean(return$AAPL<VaR_2) # better than normal assumption!
```

```
## [1] 0.01113763
```

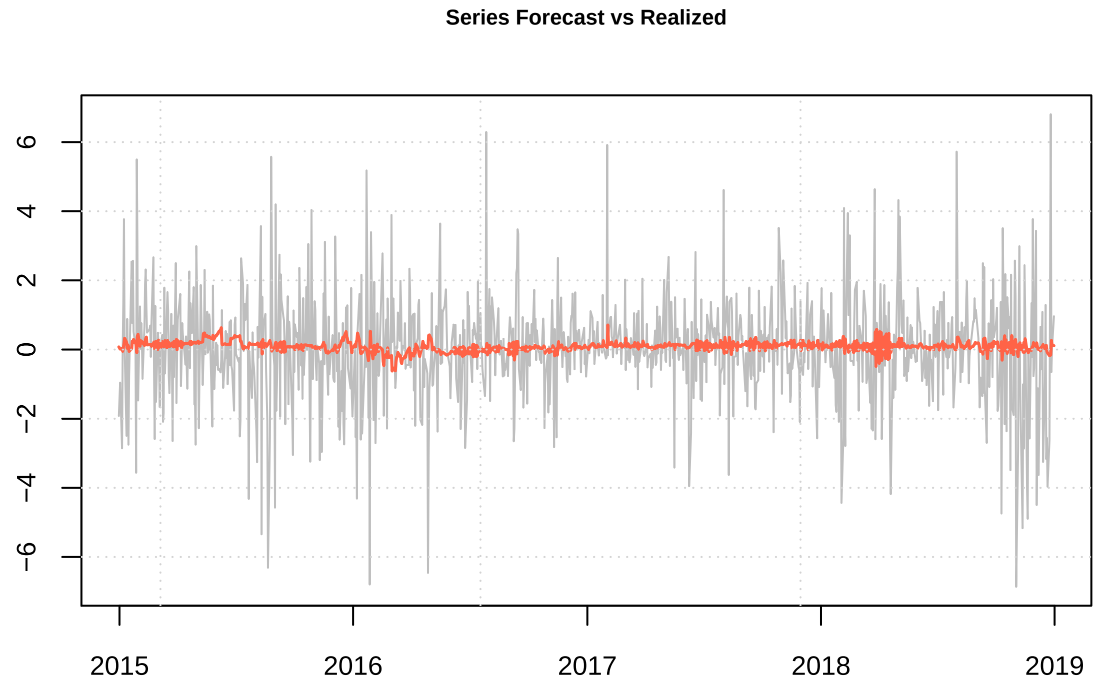
```r
# forecasting
model1_for <- ugarchforecast(model1_fit, data=NULL, n.ahead=10, n.roll=0, out.sample=0) # sigma: condit
plot(model1_for, which=1) # plot forecast
```



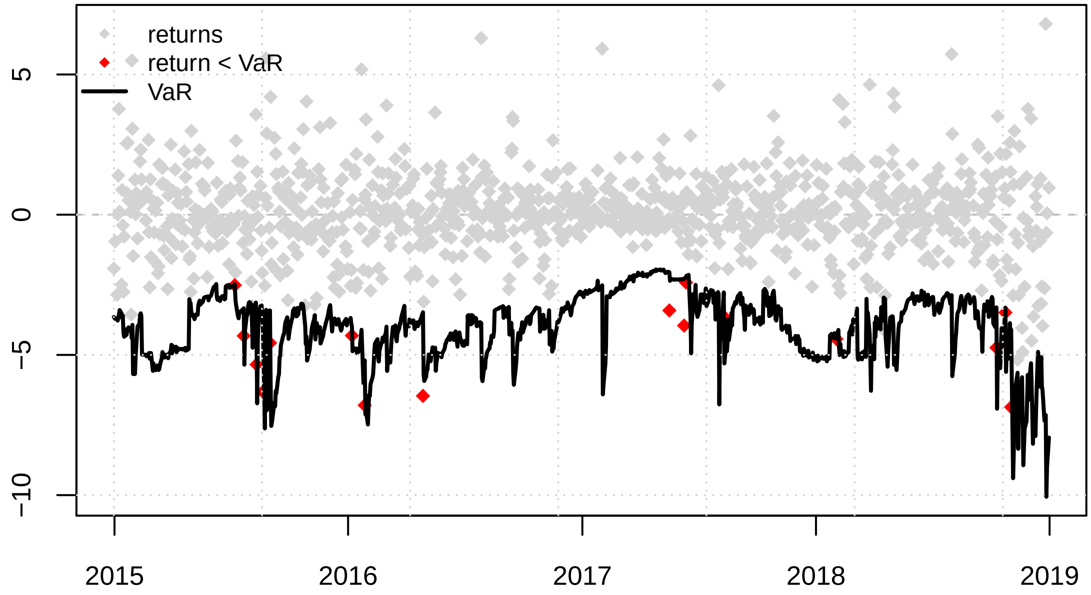

```r
# Sample forecast
model2 <- ugarchspec(variance.model=list(model="sGARCH", garchOrder=c(1, 1)), mean.model=list(armaOrder=
model2_fit <- ugarchfit(model2, data=AAPL.xts, out.sample=2)
model2_for <- ugarchforecast(model2_fit, data=NULL, n.ahead=1, n.roll=2, out.sample=2)
#qnorm(0.05)*sigma(model2_for)+fitted(model2_for)
quantile(model2_for,0.05) # estimated 0.05 VaR
```

```
##      2018-12-27 2018-12-28 2018-12-31
## T+1   -4.595023  -4.111515  -3.809166
```

```r
# rolling forecast
model <- ugarchspec(variance.model=list(model="sGARCH", garchOrder=c(1, 1)), mean.model=list(armaOrder=
modelroll1 <- ugarchroll(model, data=AAPL.xts, n.ahead=1, n.start=250, refit.every=10, refit.window=c("
plot(modelroll1, which=3)
```




```
plot(modelroll1, which=4)
```




```
# Analysis functions
# Back-test VaR function with ES
Backtest_VaR <- function(data, ar=1, ma=1, alpha=1, beta=1, dist='std'){
  model <- ugarchspec(variance.model=list(model="sGARCH", garchOrder=c(alpha, beta)), mean.model=list(a
  model_fit <- ugarchfit(model, data=data)
  VaR95 <- quantile(model_fit, 0.05)
  VaR99 <- quantile(model_fit, 0.01)
  rate95 <- mean(data<VaR95)
  rate99 <- mean(data<VaR99)
  shortfall95 <- mean(data[data<VaR95])
  shortfall99 <- mean(data[data<VaR99])
  result <-list(VaR95=VaR95, VaR99=VaR99, rate95=rate95, rate99=rate99, shortfall95=shortfall95, shortf
  return(result)
}

# Rolling VaR estimation function(ARMA+GARCH)
Rolling_VaR <- function(data, ar=1, ma=1, alpha=1, beta=1, dist='std', rolling, refit){
  #n <- length(data)
  model <- ugarchspec(variance.model=list(model="sGARCH", garchOrder=c(alpha, beta)), mean.model=list(a
  modelroll <- ugarchroll(model, data=data, n.ahead=1, n.start=rolling, refit.every=refit, refit.window
  VaR95 <- modelroll@forecast$VaR[,2]
  VaR99 <- modelroll@forecast$VaR[,1]
  real <- modelroll@forecast$VaR[,3]
  rate95 <- mean(real<VaR95)
  rate99 <- mean(real<VaR99)
  shortfall95 <- mean(real[real<VaR95])
```

```r
  shortfall99 <- mean(real[real<VaR99])
  result <-list(VaR95=VaR95, VaR99=VaR99, rate95=rate95, rate99=rate99, shortfall95=shortfall95, shortfa
  return(result)
}
```

```r
# BacktestResults on different data

# input data
Stocks <- list('AAPL'=AAPL.xts, 'AMZN'=AMZN.xts, 'FB'=FB.xts, 'GM'=GM.xts, 'NFLX'=NFLX.xts, 'SPY'=SPY.xt

# implement function
Violation <- function(datalist){
  n <- length(datalist)
  BktVaRn <- matrix(0, nrow=n, ncol=2)
  stocknames <- names(datalist)
  rownames(BktVaRn) <- stocknames
  colnames(BktVaRn) <- c("95VaR Violation%", "99VaR Violation%")
  BktVaRt <- matrix(0, nrow=n, ncol=2)
  rownames(BktVaRt) <- stocknames
  colnames(BktVaRt) <- c("95VaR Violation%", "99VaR Violation%")

  # normal assumption
  for (i in 1:n){
    result <- Backtest_VaR(Stocks[[i]], dist='norm')
    BktVaRn[i,1] <- result$rate95
    BktVaRn[i,2] <- result$rate99
  }
  # t assumption
  for (i in 1:n){
    result <- Backtest_VaR(Stocks[[i]])
    BktVaRt[i,1] <- result$rate95
    BktVaRt[i,2] <- result$rate99
  }

  BktVaRn[] <- percent(BktVaRn)
  BktVaRt[] <- percent(BktVaRt)
  return(list(BktVaRn=BktVaRn, BktVaRt=BktVaRt))
}

Violation(Stocks)
```

```
## $BktVaRn
##      95VaR Violation% 99VaR Violation%
## AAPL "5.17%"          "1.75%"
## AMZN "4.22%"          "1.35%"
## FB   "4.53%"          "1.67%"
## GM   "5.57%"          "2.07%"
## NFLX "4.22%"          "1.59%"
## SPY  "6.21%"          "2.86%"
##
## $BktVaRt
##      95VaR Violation% 99VaR Violation%
## AAPL "6.44%"          "1.11%"
## AMZN "5.81%"          "0.95%"
```

```
## FB   "6.36%"           "1.43%"
## GM   "6.28%"           "1.35%"
## NFLX "5.81%"           "0.72%"
## SPY  "7.24%"           "1.59%"
```

```r
# Rolling estimation results on different data

Rolling <- function(datalist, rolling, refit){
  n <- length(datalist)
  RollVaRn <- matrix(0, nrow=n, ncol=2)
  stocknames <- names(datalist)
  rownames(RollVaRn) <- stocknames
  colnames(RollVaRn) <- c("95VaR Violation%", "99VaR Violation%")
  RollVaRt <- matrix(0, nrow=n, ncol=2)
  rownames(RollVaRt) <- stocknames
  colnames(RollVaRt) <- c("95VaR Violation%", "99VaR Violation%")

  # normal assumption
  for (i in 1:n){
    result <- Rolling_VaR(Stocks[[i]], dist='norm', rolling=rolling, refit=refit)
    RollVaRn[i,1] <- result$rate95
    RollVaRn[i,2] <- result$rate99
  }
  # t assumption
  for (i in 1:n){
    result <- Rolling_VaR(Stocks[[i]], rolling=rolling, refit=refit)
    RollVaRt[i,1] <- result$rate95
    RollVaRt[i,2] <- result$rate99
  }

  RollVaRn[] <- percent(RollVaRn)
  RollVaRt[] <- percent(RollVaRt)
  return(list(RollVaRn=RollVaRn, RollVaRt=RollVaRt))
}

Rolling(Stocks, 250, 10)
```

```
## $RollVaRn
##      95VaR Violation% 99VaR Violation%
## AAPL "6.65%"           "2.28%"
## AMZN "5.76%"           "2.48%"
## FB   "6.95%"           "2.58%"
## GM   "6.06%"           "2.18%"
## NFLX "4.67%"           "2.18%"
## SPY  "5.86%"           "2.98%"
##
## $RollVaRt
##      95VaR Violation% 99VaR Violation%
## AAPL "6.95%"           "1.59%"
## AMZN "6.06%"           "1.19%"
## FB   "6.26%"           "1.79%"
## GM   "6.65%"           "1.59%"
## NFLX "5.26%"           "1.09%"
## SPY  "6.75%"           "2.18%"
```