STAT509-001-H10-Xinye Xu

Q1

(a) Fit a GARCH(1,1) model assuming iid standard normal innovations to RUTLRet.ts and to RUTLRetdm.ts, and provide a discussion on the two models, and the differences in the results/estimated parameters. Specify which model is best for the data and summarize your reasons for this choice. Carry out the remaining parts of this problem for the model you preferred.

We set include.mean = FALSE in both models, as the RUTLRetdm.ts has zero mean. As most estimated significant of coefficients, information criteria values are similiar to each other, we compare the sum of alpha1 and beata1. As the sum of them for model 'RUTLRet.ts' is 0.9216, which is less than 0.9248 of 'RUTLRetdm.ts'. So the 'RUTLRet.ts' is more stable, because the sum is less closed to 1. We should use'RUTLRet.ts'.

```r
# load data
RUT = read.csv("RUT_03_2015-03_2019.csv", header = TRUE)
RUT.ts <- ts(data=RUT$Adj.Close,start=c(2015,11),frequency=52,names=c('AdjClosePrice'))
RUTLRet = diff(log(RUT$Adj.Close))
RUTLRet.ts <- ts(data=RUTLRet,start=c(2015,12),frequency=52,names=c('Log Return'))
RUTLRetdm.ts = RUTLRet.ts-mean(RUTLRet)
```

```r
# fit RUTLRet.ts (log return)
library('fGarch')
```

```
## Warning: package 'fGarch' was built under R version 3.4.4

## Loading required package: timeDate

## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'zone/tz/2018i.
## 1.0/zoneinfo/America/Detroit'

## Loading required package: timeSeries

## Loading required package: fBasics

##

## Rmetrics Package fBasics

## Analysing Markets and calculating Basic Statistics

## Copyright (C) 2005-2014 Rmetrics Association Zurich

## Educational Software for Financial Engineering and Computational Science

## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.

## https://www.rmetrics.org --- Mail to: info@rmetrics.org
```

```r
garchFit_ARGARCH <- garchFit(~garch(1,1), data=RUTLRet.ts,
                             cond.dist = c("norm"), include.mean = FALSE,
                             trace = FALSE, algorithm = c("nlminb"),
                             hessian = c("ropt"))

# fit RUTLRetdm.ts (true return)
garchFit_ARGARCH_mean <- garchFit(~garch(1,1), data=RUTLRetdm.ts,
                             cond.dist = c("norm"), include.mean = FALSE,
                             trace = FALSE, algorithm = c("nlminb"),
                              hessian = c("ropt"))
summary(garchFit_ARGARCH)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = RUTLRet.ts, cond.dist = c("norm"),
##     include.mean = FALSE, trace = FALSE, algorithm = c("nlminb"),
##     hessian = c("ropt"))
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x7fb2439d4df0>
##  [data = RUTLRet.ts]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##      omega      alpha1       beta1
## 6.4527e-05  2.7911e-01  6.4251e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  6.453e-05   2.852e-05    2.263  0.02364 *
## alpha1 2.791e-01   1.007e-01    2.772  0.00558 **
## beta1  6.425e-01   8.537e-02    7.526 5.22e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  496.9412    normalized:  2.377709
##
## Description:
##  Tue Apr 16 00:17:46 2019 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  25.61928  2.734284e-06
##  Shapiro-Wilk Test  R    W      0.9622763 2.32518e-05
##  Ljung-Box Test     R    Q(10)  8.229038  0.6064758
##  Ljung-Box Test     R    Q(15)  14.8537   0.4620061
##  Ljung-Box Test     R    Q(20)  15.86875  0.7247326
##  Ljung-Box Test     R^2  Q(10)  7.505944  0.6769717
##  Ljung-Box Test     R^2  Q(15)  9.071873  0.8737295
##  Ljung-Box Test     R^2  Q(20)  11.01976  0.9457085
##  LM Arch Test       R    TR^2   7.775473  0.802421
##
```

```
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.726710 -4.678733 -4.727114 -4.707313
```

```r
summary(garchFit_ARGARCH_mean)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = RUTLRetdm.ts, cond.dist = c("norm"),
##     include.mean = FALSE, trace = FALSE, algorithm = c("nlminb"),
##     hessian = c("ropt"))
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x7fb2460854b0>
##  [data = RUTLRetdm.ts]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##      omega      alpha1       beta1
## 6.3218e-05  2.8466e-01  6.4014e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  6.322e-05   2.727e-05    2.318   0.0204 *
## alpha1 2.847e-01   9.864e-02    2.886   0.0039 **
## beta1  6.401e-01   8.382e-02    7.637 2.22e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  497.9445    normalized:  2.38251
##
## Description:
##  Tue Apr 16 00:17:47 2019 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  21.34246  2.3203e-05
##  Shapiro-Wilk Test  R    W      0.9631456 2.925054e-05
##  Ljung-Box Test     R    Q(10)  8.437136  0.5862186
##  Ljung-Box Test     R    Q(15)  15.2582   0.4329836
##  Ljung-Box Test     R    Q(20)  16.31415  0.6969533
##  Ljung-Box Test     R^2  Q(10)  8.504219  0.5797093
##  Ljung-Box Test     R^2  Q(15)  9.94601   0.8231222
##  Ljung-Box Test     R^2  Q(20)  12.06664  0.9137633
##  LM Arch Test       R    TR^2   8.675848  0.7303395
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.736311 -4.688335 -4.736716 -4.716914
```

```
# garchFit parameter:
# include.mean -- this flag determines if the parameter for the mean will
# be estimated or not. If include.mean=TRUE this will be the case,
# hessian -- a string denoting how the Hessian matrix should be evaluated,
# algorithm -- a string parameter that determines the algorithm used for maximum likelihood estimation.

show(c(2.791e-01 + 6.425e-01 , 2.847e-01 + 6.401e-01))
```

```
## [1] 0.9216 0.9248
```

(b) Specify your final model for the log-returns in detail and provide standard errors of the parameter estimates $\alpha0, \alpha1, \beta1$, also specify the half-life of the volatility based on the estimated parameters.

To calculate 'volatility' half life, by making $\lambda^{k-1} \leq 1/2$, so $(k1) * log(\lambda) \leq log(2)$, then $k \geq 1 - log(2)/log(\lambda)$.

```
# lambda is alpha0 + beta1
lambda = 2.791e-01 + 6.425e-01
halflife = ceiling((1 - log(2)/log(lambda)))
halflife
```

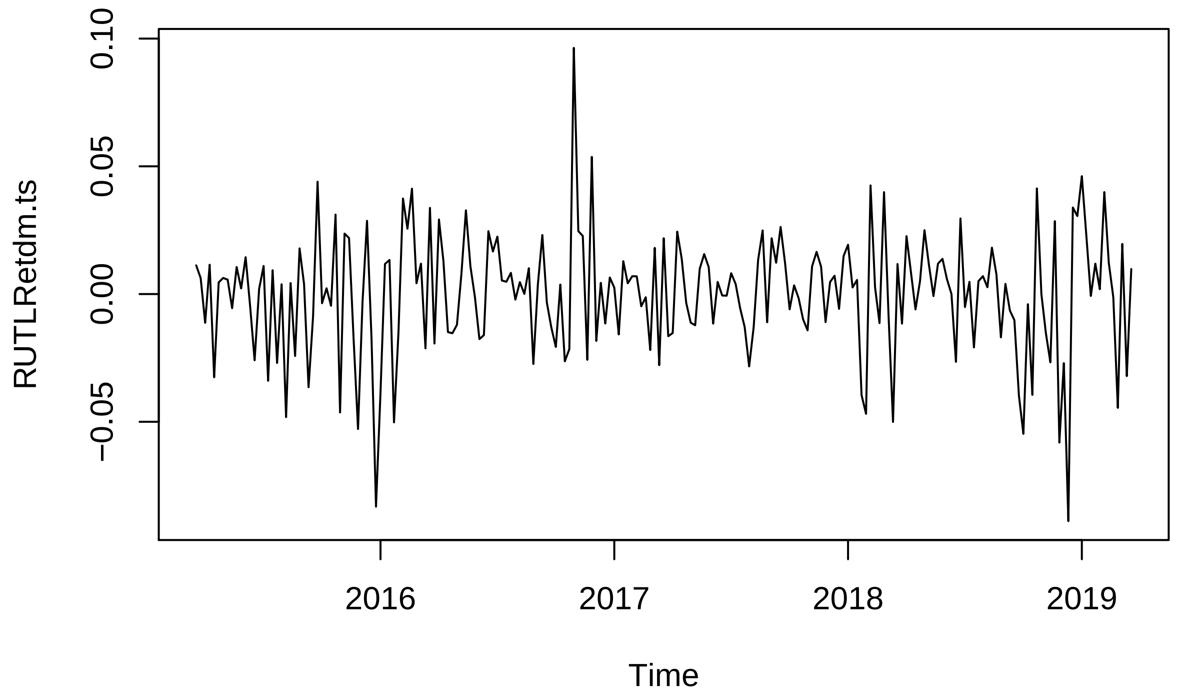
```
## [1] 10
```

```
# ceiling containing the smallest integers not less than
# the corresponding elements of x.
```

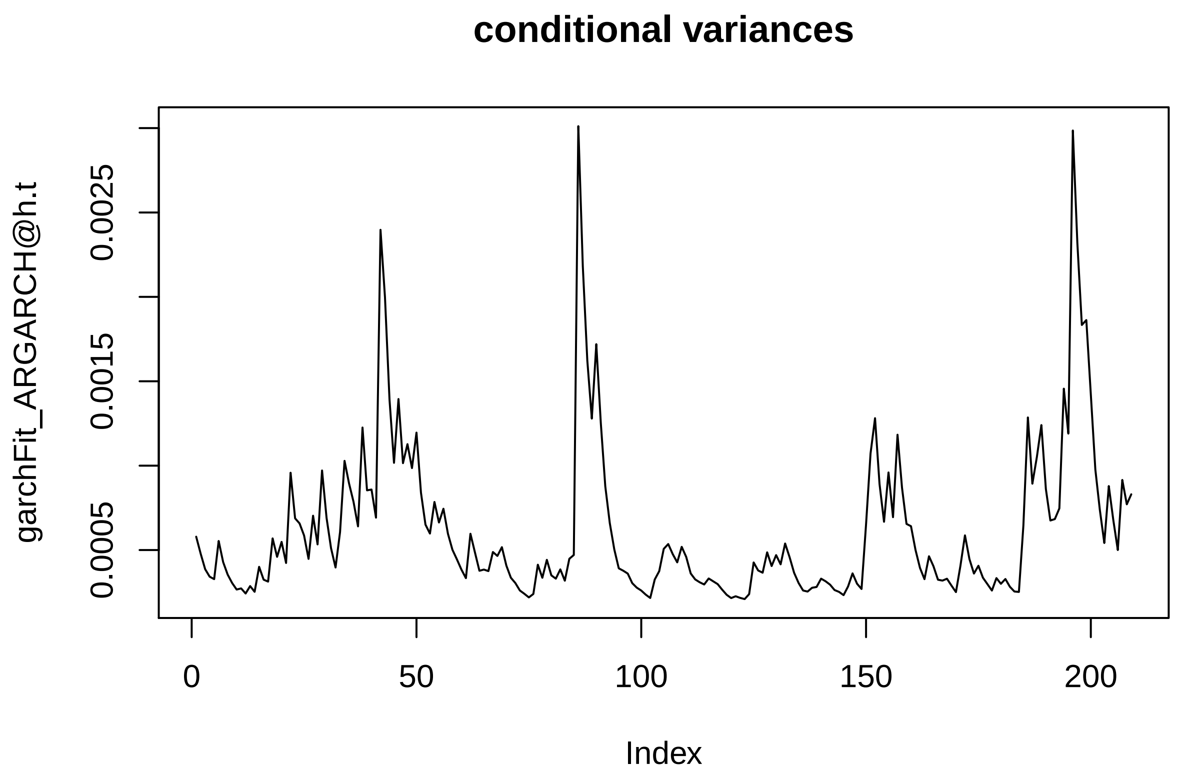
(c) Generate plots of the estimated conditional volatilities σ^n and of the estimated innovations (residuals) ϵ^n.

According to the plots, the original time series seems to have vulnerable estimated conditional volatilities. The residuals (innovations) seem to be good since they are more similar to uniform distribution over time.

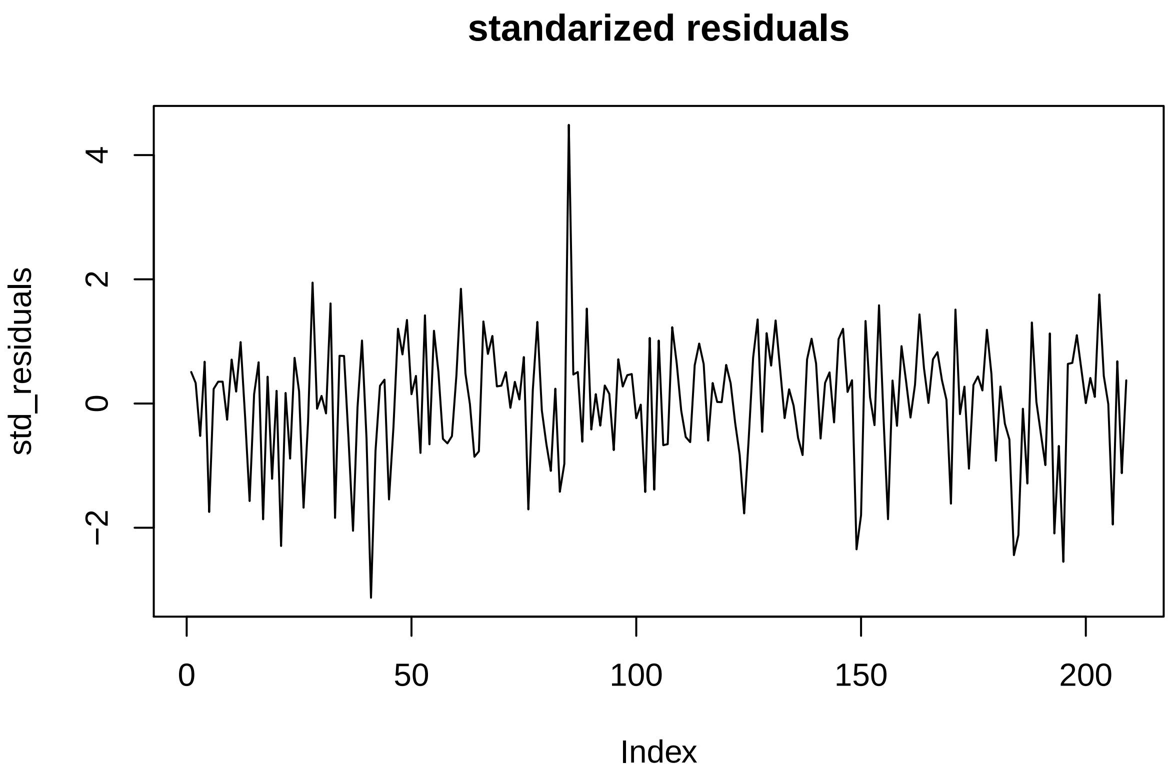
```
plot(RUTLRetdm.ts)
```



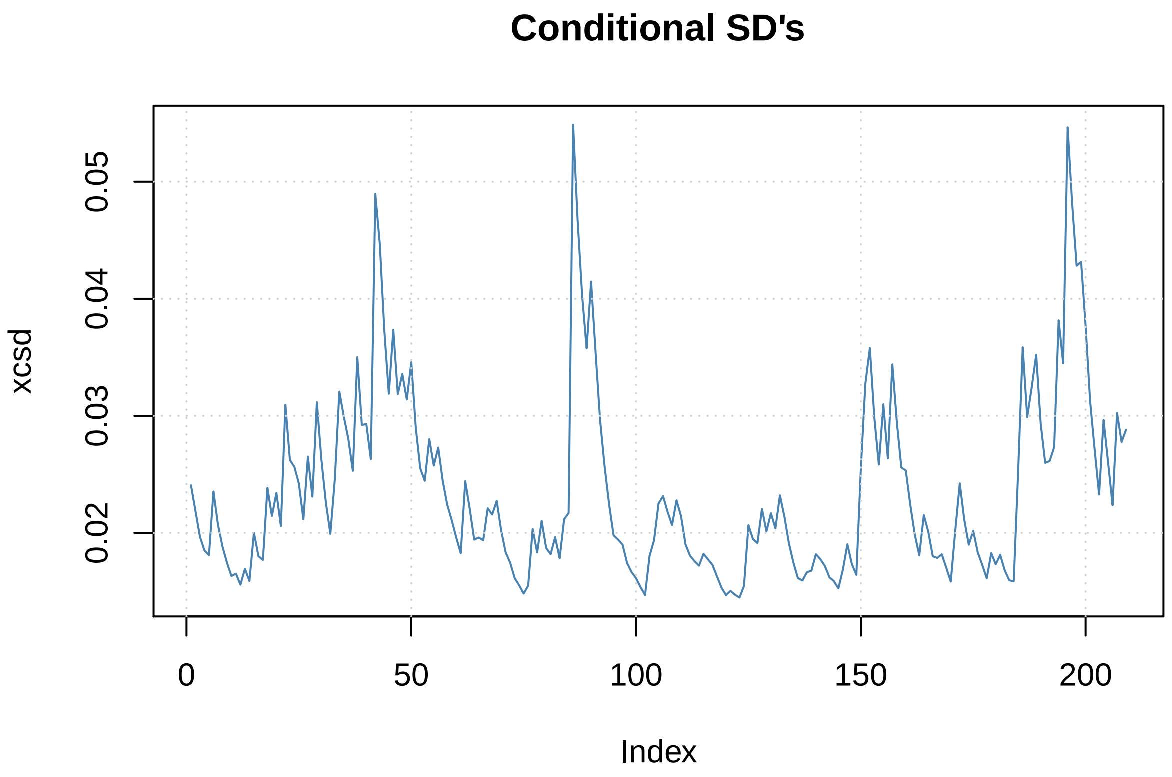
```r
#  conditional variances
plot(garchFit_ARGARCH@h.t, type = 'l', main = 'conditional variances')
```



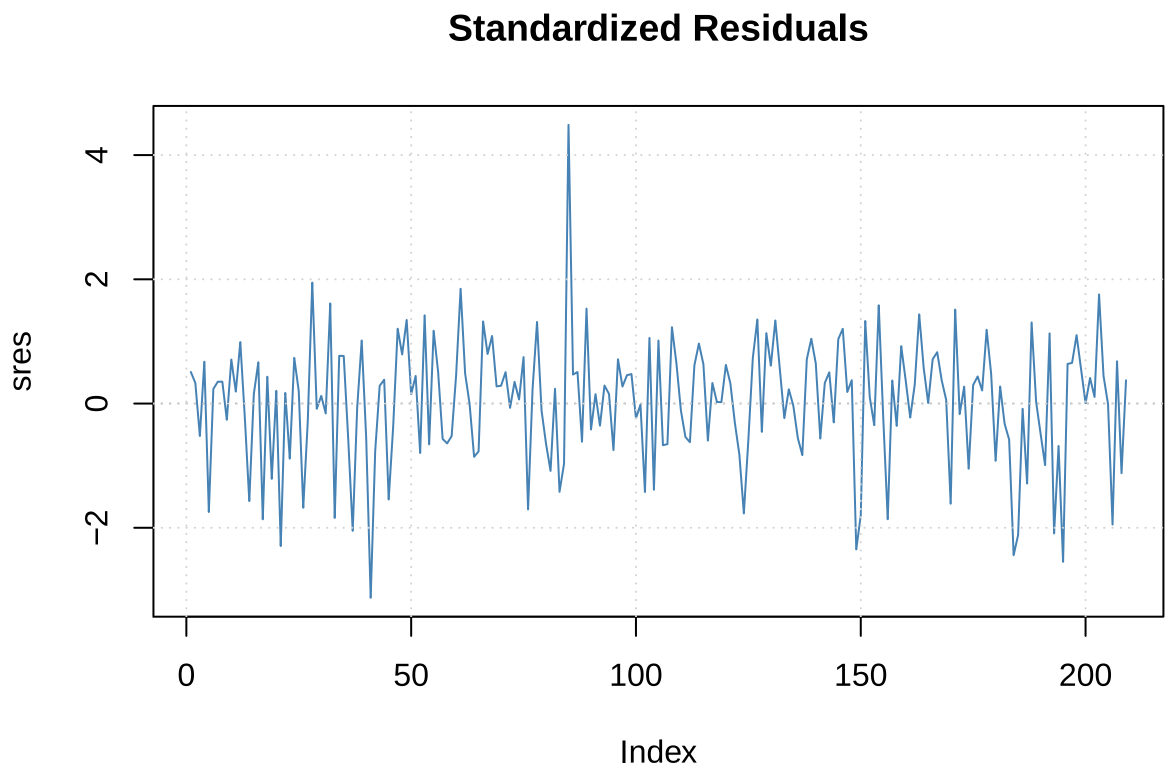
```r
# standarized residuals
std_residuals= residuals(garchFit_ARGARCH, standardize = T)
plot(std_residuals, type = 'l', main = 'standarized residuals')
```



```
# plot methods in garchFit
# The generic function plot allows to display 13 graphs. These are the
# Time SeriesPlot
# Conditional Standard Deviation Plot
# Series Plot with 2 Conditional SD Superimposed
# Autocorrelation function Plot of Observations
# Autocorrelation function Plot of Squared Observations
# Cross Correlation Plot
# Residuals Plot
# Conditional Standard Deviations Plot
# Standardized Residuals Plot
# ACF Plot of Standardized Residuals
# ACF Plot of Squared Standardized Residuals
# Cross Correlation Plot between $r^2$ and r
# Quantile-Quantile Plot of Standardized Residuals
# https://cran.r-project.org/web/packages/fGarch/fGarch.pdf

# anohte way to plot: (same results)
plot(garchFit_ARGARCH,which=8) # Conditional Standard Deviations
```



```r
plot(garchFit_ARGARCH,which=9) # Standardized Residuals
```



(d) Carry out the appropriate diagnostics on your GARCH estimation and summarize your findings.

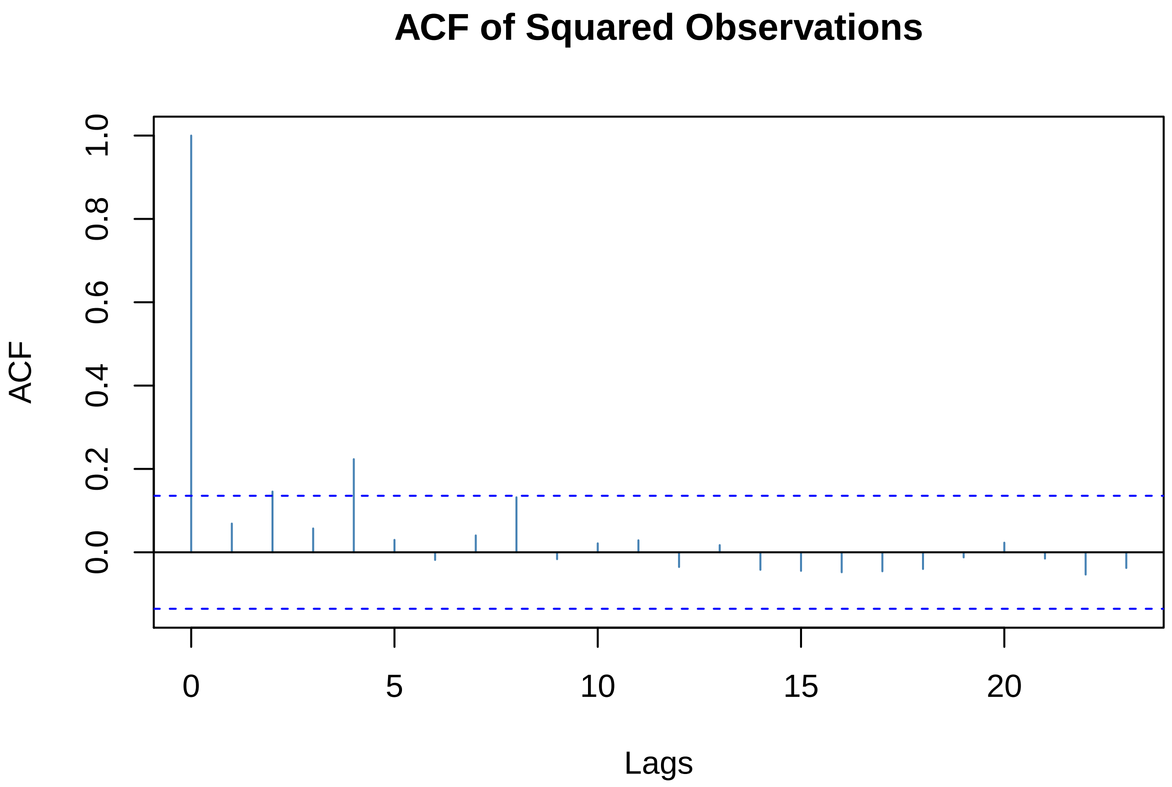
With the summary in (a), the Jarque-Bera Test had a p-value that is pretty small (2.3203e-05). p-value of the Shapiro-Wilk test for normality is also quite small (2.925054e-05), meaning that the standardized residuals might not be a normal distribution.

Then, Ljung-Box Test p-values for both residuals and squared residuals for some lages are also very samll, suggesting this model seems to have a good fit and without correlatins among lag residuals.

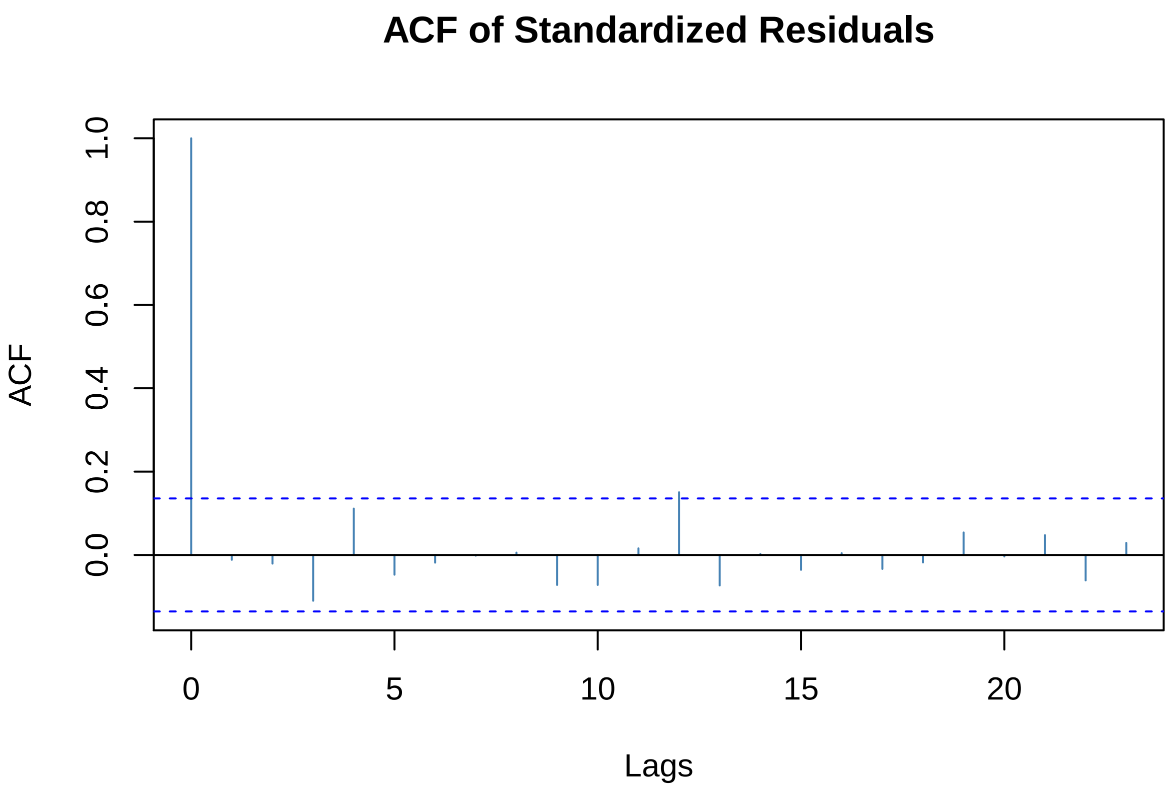
Next step, we check the ACF plots, for the squared observation, there are some sticks outside of the significant boundary. However, with fitted GARCH model, ACF test shows that there is no obvious lag autocorrelation among residuals. From the qqplots, we can notice that it has a little slight tail, and also it seems to be not symmetric for left and right tail.

In conclusion, as we can see in the (b), the standardized residuals plot is similar to white noise, which is qutie good. Also, it was confimed by ACF plot that after fitting GARCH, confirming that there is not much correlation left in the residuals.

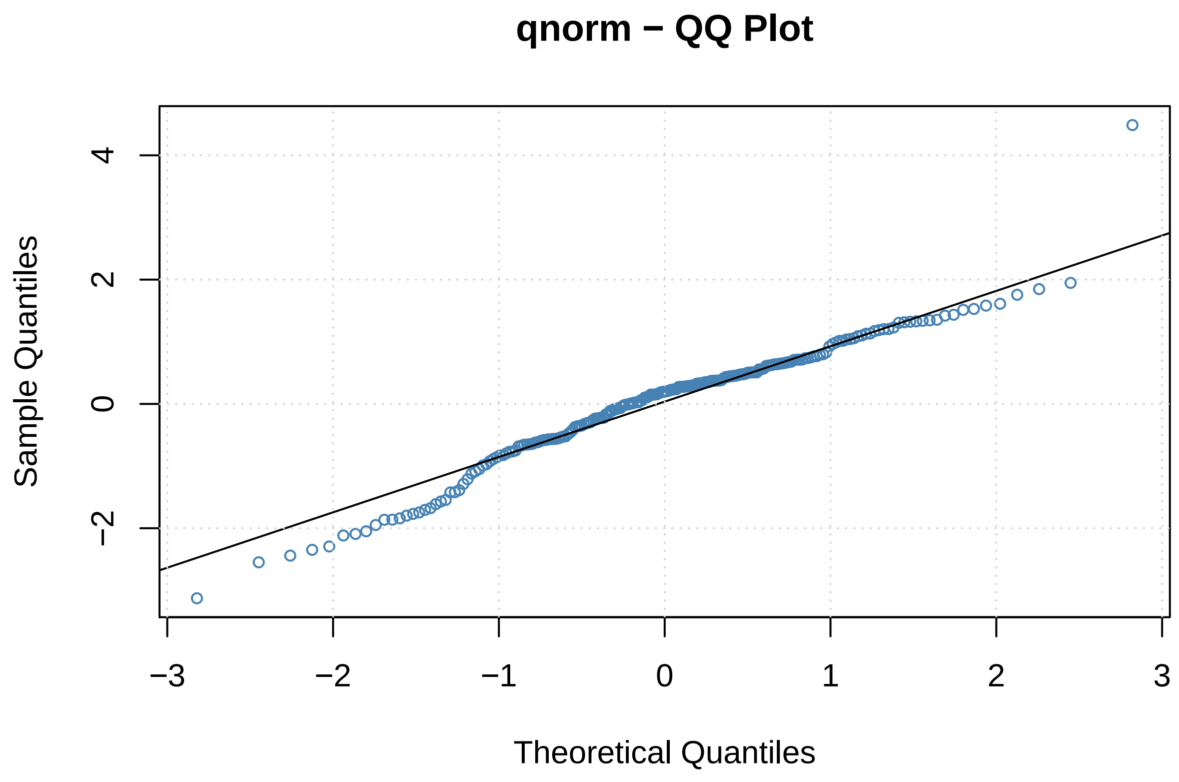
```r
# Autocorrelation function Plot of Squared Observations
plot(garchFit_ARGARCH,which=5)
```



```r
# ACF Plot of Standardized Residuals
plot(garchFit_ARGARCH,which=10)
```



```r
# Quantile-Quantile Plot of Standardized Residuals
plot(garchFit_ARGARCH,which=13)
```



(e) Be specific about more general GARCH methods/models that are likely to be better fit to this data. By directly working with the residuals from your GARCH estimation, analyze/estimate the appropriate t-distribution for the white noise process, and carry out the appropriate diagnostics for how well this fits the residuals – provide a discussion/interpretation of the results.

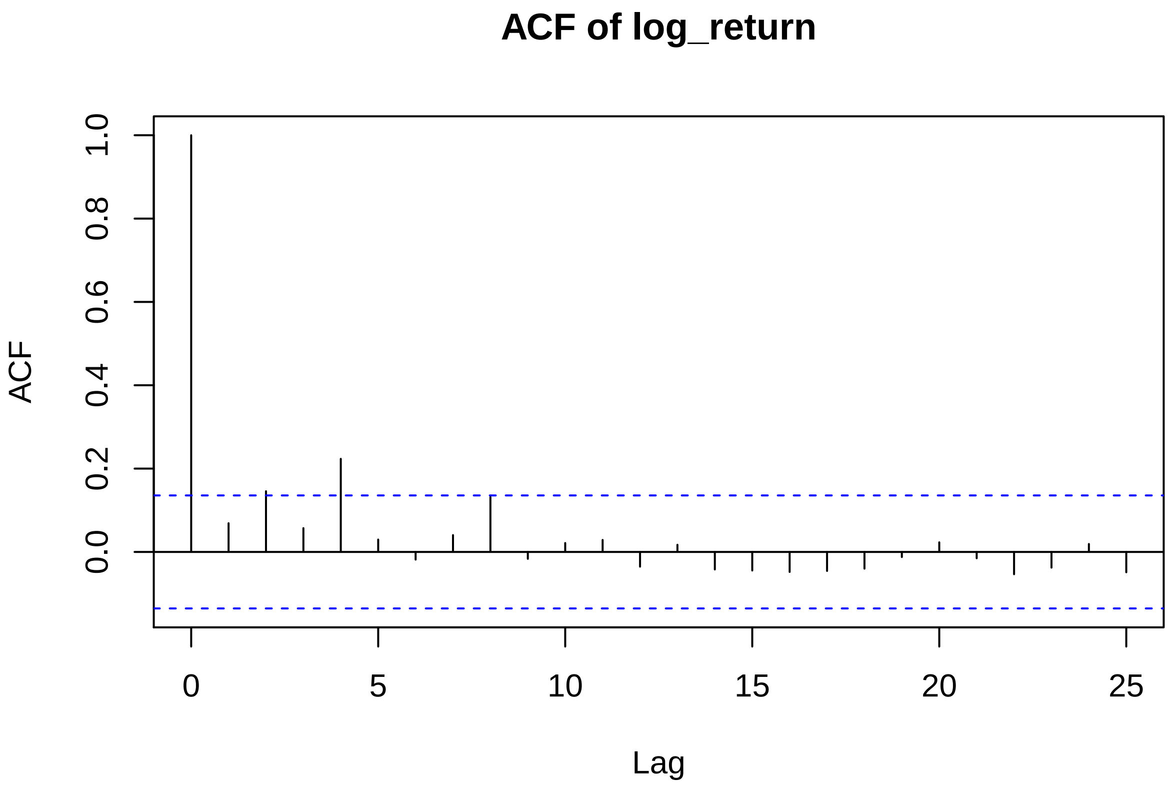
After fitting with t-distribution for the white noise process, qqplots of new model fits bettwe although there is assymetric and some heavier tail in the left side, light in the right hand side. Also, by comparing the ACI values, in t-dist white noise, -4.755527 AIC is less than -4.726710 quite lot, meaning a better choice of model.

```
# fit garch by t dist
garchFit_ARGARCH_tdis <- garchFit(~garch(1,1), data=RUTLRet.ts,
                             cond.dist = c("std"), include.mean = FALSE,
                             trace = FALSE, algorithm = c("nlminb"),
                             hessian = c("ropt"))
summary(garchFit_ARGARCH_tdis)
```

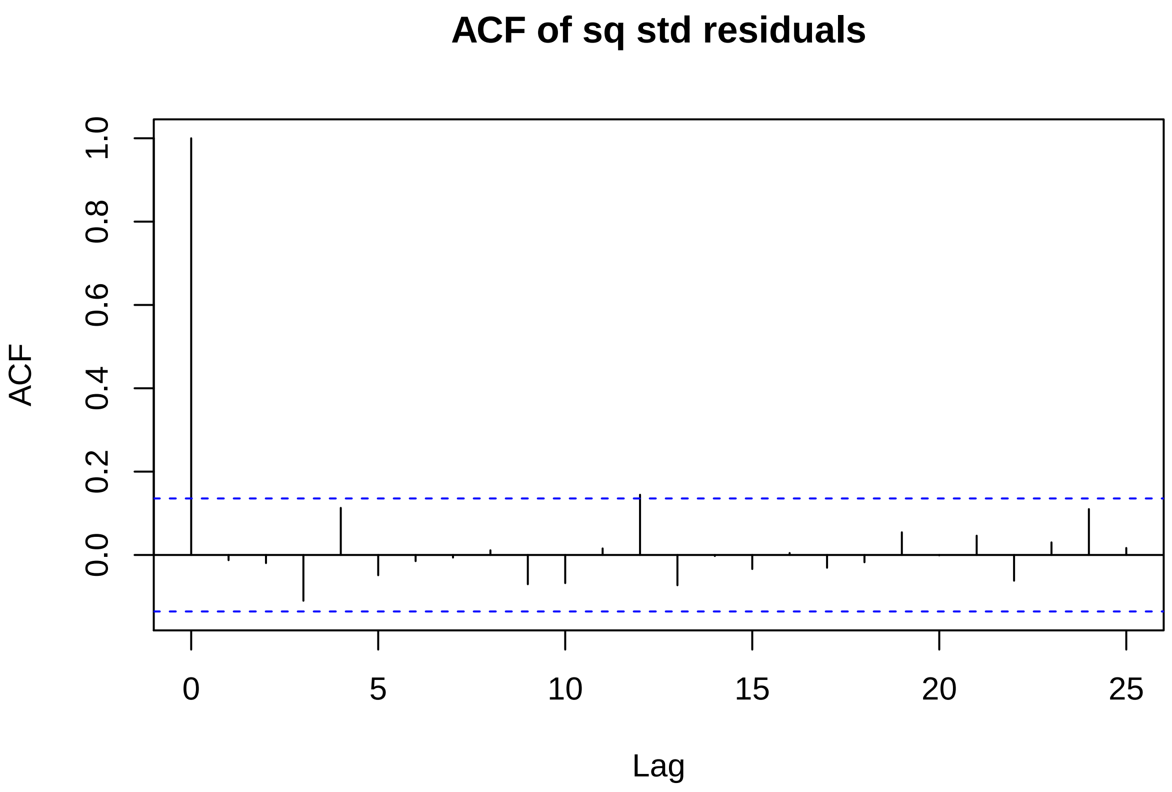
```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = RUTLRet.ts, cond.dist = c("std"),
##     include.mean = FALSE, trace = FALSE, algorithm = c("nlminb"),
##     hessian = c("ropt"))
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x7fb244f3cf90>
##  [data = RUTLRet.ts]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##      omega      alpha1       beta1       shape
## 6.3227e-05  2.2350e-01  6.8519e-01  7.2272e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  6.323e-05   3.306e-05    1.913   0.0558 .
## alpha1 2.235e-01   9.771e-02    2.288   0.0222 *
## beta1  6.852e-01   9.785e-02    7.002 2.52e-12 ***
## shape  7.227e+00   3.397e+00    2.128   0.0334 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  500.9526    normalized:  2.396902
##
## Description:
##  Tue Apr 16 00:17:49 2019 by user:
##
##
```

```
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  29.81235  3.359931e-07
##  Shapiro-Wilk Test  R    W      0.9609261 1.636221e-05
##  Ljung-Box Test     R    Q(10)  8.136026  0.615552
##  Ljung-Box Test     R    Q(15)  14.31097  0.5020871
##  Ljung-Box Test     R    Q(20)  15.28952  0.7596084
##  Ljung-Box Test     R^2  Q(10)  6.816398  0.742657
##  Ljung-Box Test     R^2  Q(15)  8.518656  0.9013124
##  Ljung-Box Test     R^2  Q(20)  10.38431  0.9606562
##  LM Arch Test       R    TR^2   7.338818  0.8344352
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.755527 -4.691559 -4.756242 -4.729665
```

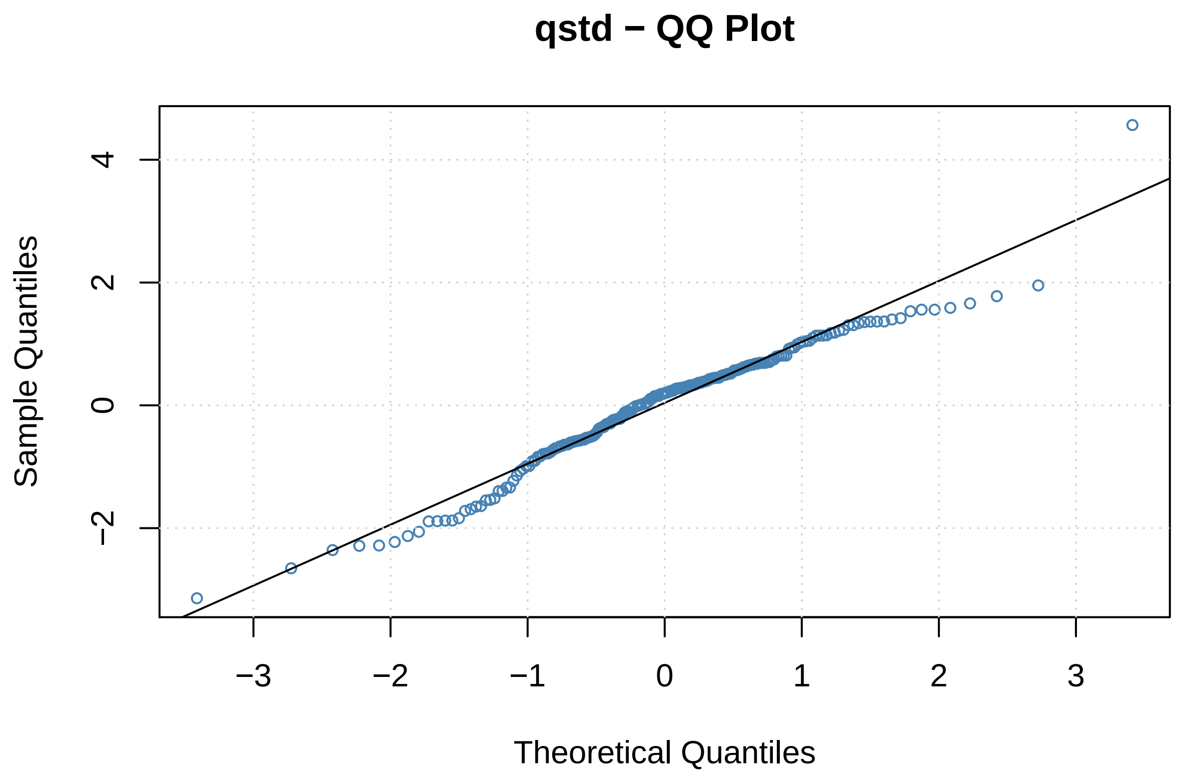
```
# include mean
# this flag determines if the parameter for the mean will be estimated or not. If include.mean=TRUE thi


# diagnostics
# ACF for Standardized Residuals
acf(as.vector(RUTLRet.ts^2), lag=25, main='ACF of log_return')
```



```
acf(as.vector(residuals(garchFit_ARGARCH_tdis, standardize = T)),
    lag=25, main='ACF of sq std residuals')
```



```r
# qqplot of Standardized Residuals
plot(garchFit_ARGARCH_tdis,which=13)
```



Q2

2. Following up on problem 1, utilize the estimated GARCH model to derive the relative VaR for the 5 weeks following the last week of data for q = .005. Do this for two cases: (i) assuming the white noise process is normal and (ii) assuming white noise process follows the estimated t-distribution from part (e).

The future 5 days relative VaR is the following.

```r
# w0 from from above summary fun
omega_norm = 6.453e-05
omega_t = 6.3227e-05
last_ind = nrow(RUT)
n = 5 # predicted freq

# for white noise process is normal
sigma_model = predict(garchFit_ARGARCH, n)$standardDeviation
norm = rep(NA,n)
VaR_norm = rep(NA,n)
for (i in (1:n)){
  norm[i] = -qnorm(.005, mean = omega_norm, sd = sigma_model[i])
  VaR_norm[i] = exp(norm[i])-1
}

# for white noise process is t-dist
sigma_model = predict(garchFit_ARGARCH_tdis, n)$standardDeviation
tdist = rep(NA,n)
VaR_tdist = rep(NA,n)
for (i in (1:n)){
  tdist[i] = -qnorm(.005, mean = omega_t, sd = sigma_model[i])
  VaR_tdist[i] = exp(tdist[i])-1
}
show(c('with norm white noise', VaR_norm))
```

```
## [1] "with norm white noise" "0.0667459122291596"    "0.067568090573221"
## [4] "0.0683178993362188"    "0.0690024180349891"    "0.0696279047285104"
```

```r
show(c('with t-dist white noise', VaR_tdist))
```

```
## [1] "with t-dist white noise" "0.0666873299050035"
## [3] "0.0670018600480407"      "0.067286515762679"
## [5] "0.0675442377661655"      "0.0677776580684937"
```