Time series Project 2

Carlos Perez,

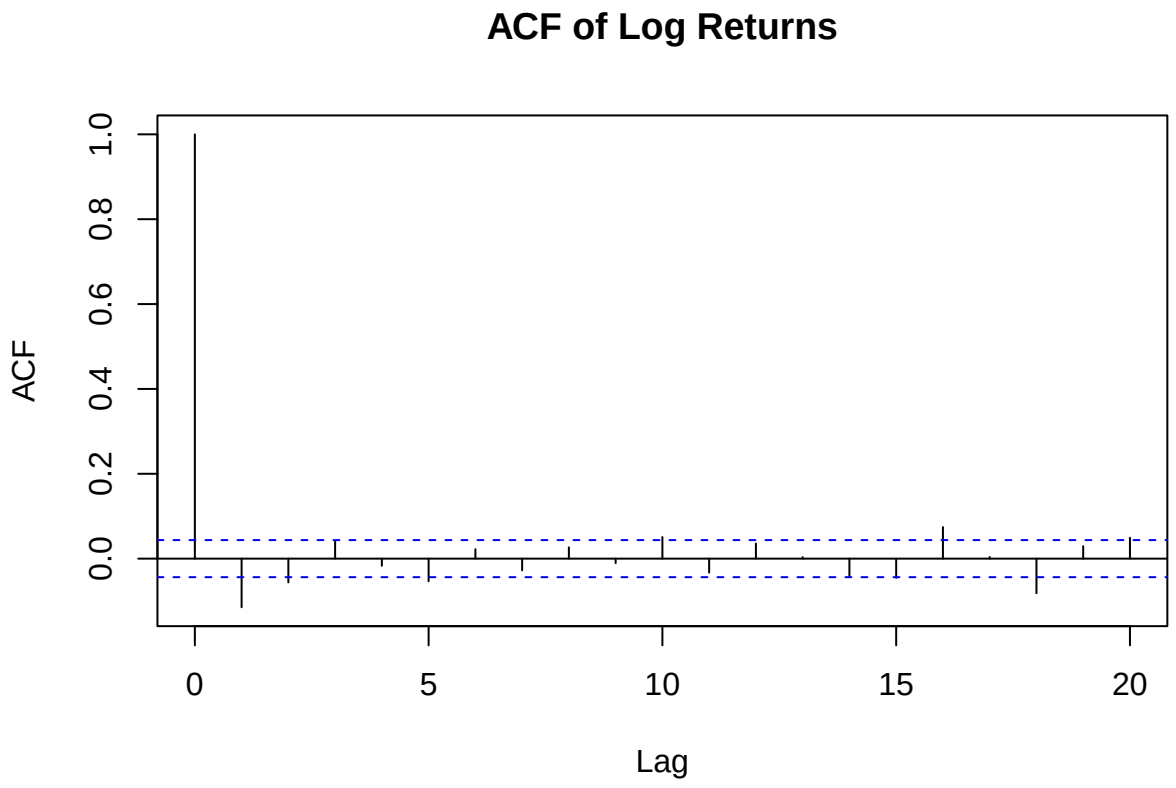
2023-06-02

Environment

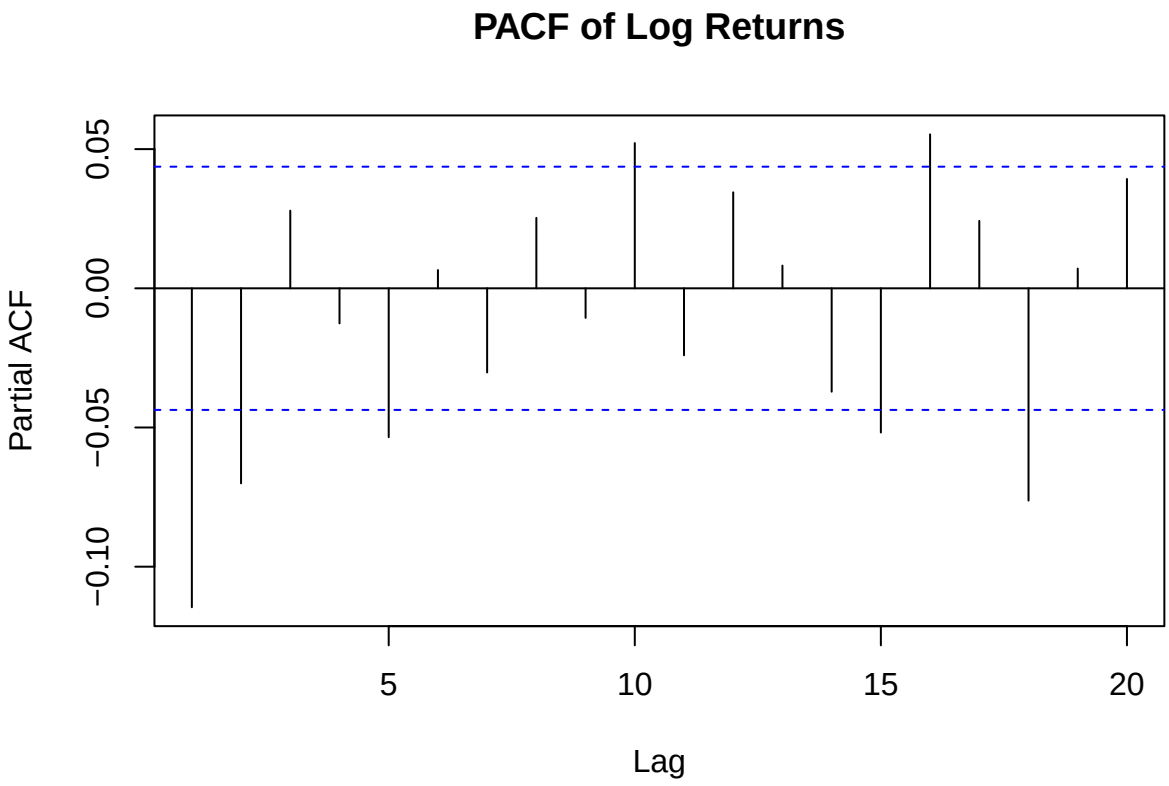
```
#Load the packages
library(pacman)
p_load(data.table, fixest, multcomp,stargazer, ggplot2, multcomp, knitr, stats, tidyr, haven, rlang, dpl

#create the dataset
data <- read.csv("sp500.csv")
datats <- ts(data)
```

Exercise 1

```
# Initialize an empty vector to store the log-returns
LR <- numeric(length = nrow(data))

# Compute the log-returns individually for each time point
for (t in 2:nrow(data)) {
  LR[t] <- log(data$DATA[t]) - log(data$DATA[t - 1])
}

# Remove the first element since it does not have a lag
LR <- LR[-1]

#Plotting the ACF and PACF
acf(LR, lag.max = 20, main = "ACF of Log Returns")
```




```
pacf(LR, lag.max = 20, main = "PACF of Log Returns")
```




```
#Ljung-Box test
Box.test(LR, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  LR
## X-squared = 96.455, df = 20, p-value = 5.399e-12
```

When observing the path that both the ACF and PACF plots take we can see that a an ARMA(1) model would be fitting to the data, since neither the ACF nor the PACF seem like they are trailing off, so choosing a AR(p) or MA(q) model is not a good fit. However by the small p values on the Ljung-Box test we can state that the residual are highly correlated and therefore a conditional hetoroskedastic model would fit better.

Exercise 2

```
# Initialize variables
Baic <- Inf
Border <- c(0, 0)

# Iterate over different p and q values
for (p in 1:5) {
```

```
  for (q in 1:5) {
    # Fit ARMA model
    arma_model <- arima(LR, order = c(p, 0, q))

    # Calculate AIC
    aic <- AIC(arma_model)

    # Update best AIC and order if necessary
    if (aic < Baic) {
      Baic <- aic
      Border <- c(p, 0, q)
    }
  }
}
```

```
## Warning in arima(LR, order = c(p, 0, q)): possible convergence problem: optim
## gave code = 1
```

```
# Print the best AIC and order
cat("Best AIC:", Baic)
```

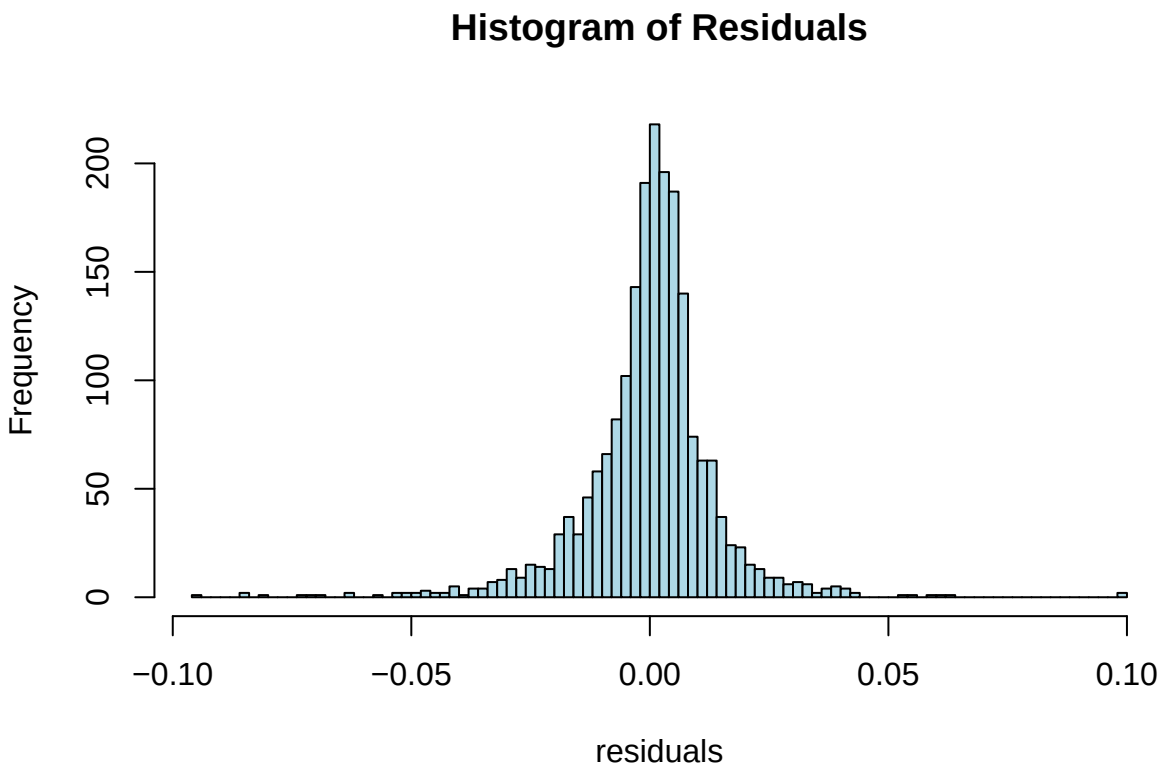
```
## Best AIC: -11503.43
```

```
cat("Best Order (p, d, q):", Border)
```

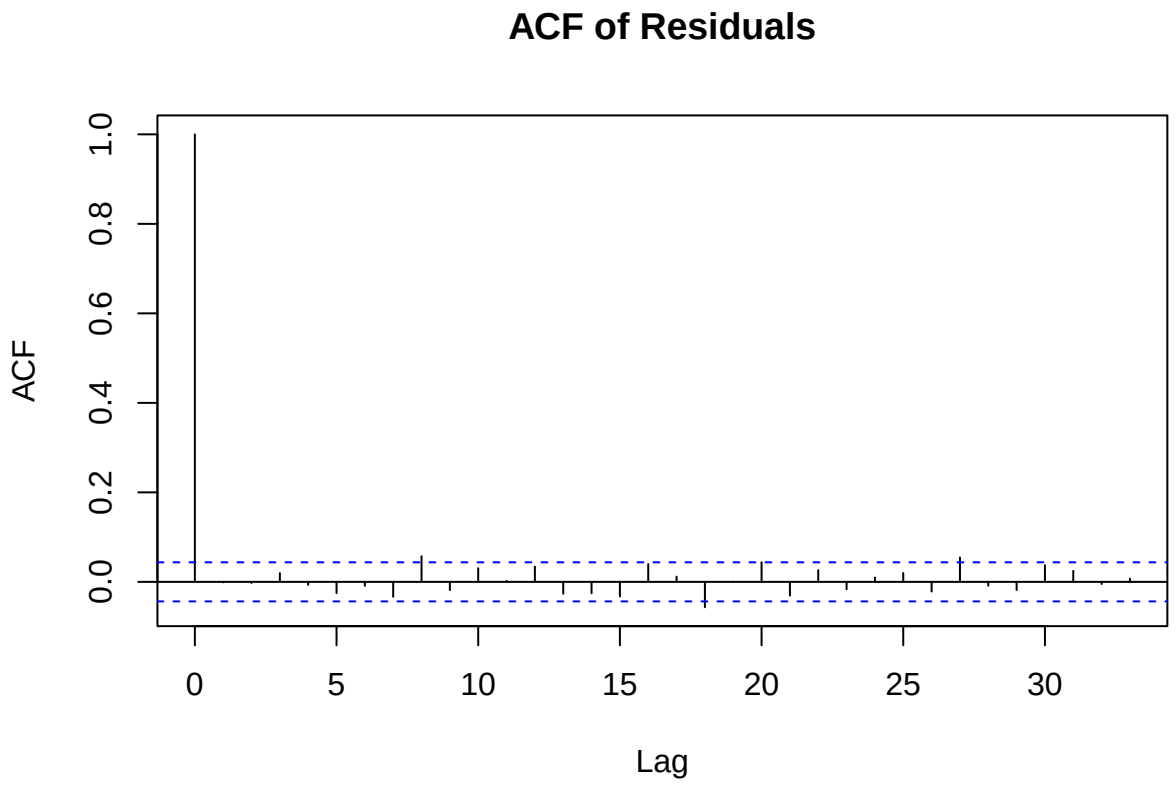
```
## Best Order (p, d, q): 3 0 3
```

Exercise 3

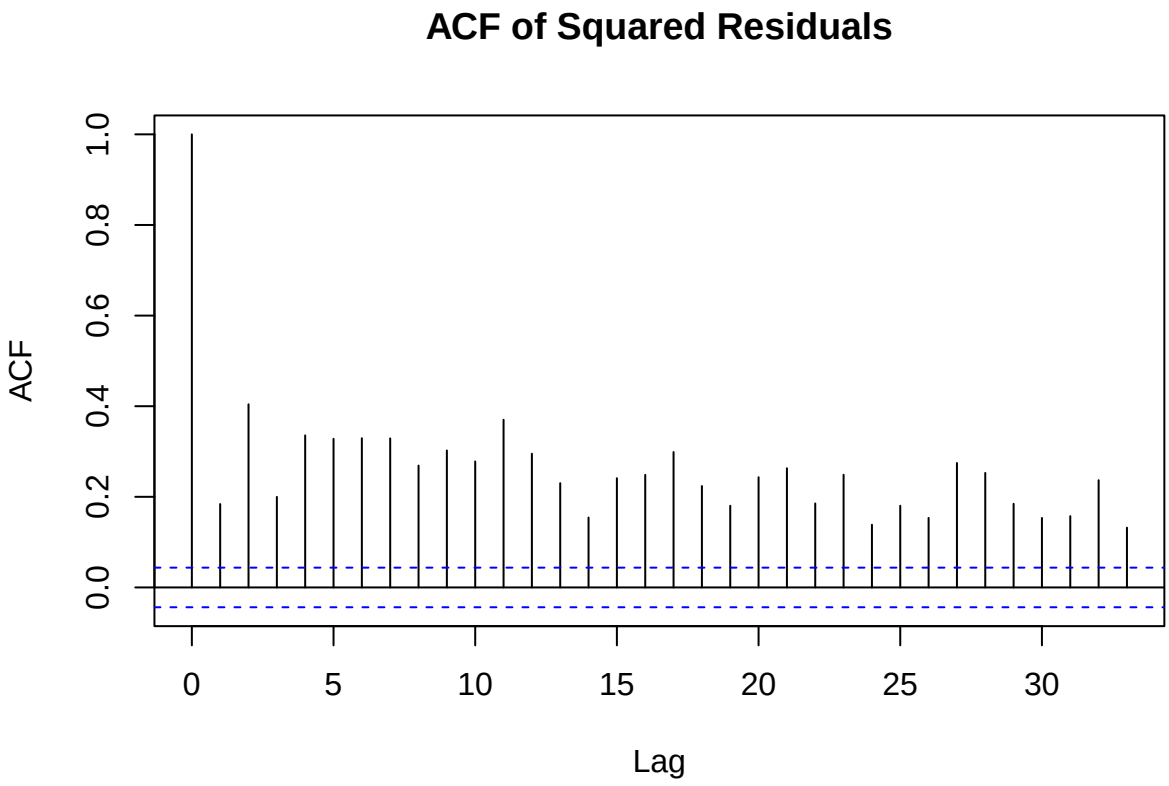
```
# Fit the ARMA(3,0,3) model to the LR data
arma <- arima(LR, order = c(3, 0, 3))

#Obtain the residuals
residuals <- arma$residuals

# We plot the residuals
hist(residuals, breaks = "FD", col = "lightblue", main = "Histogram of Residuals")
```




```r
# ACF plot of residuals
acf(residuals, main = "ACF of Residuals")
```




```r
# ACF plot of squared residuals
residualssq <- residuals^2
acf(residualssq, main = "ACF of Squared Residuals")
```



The histogram seems to follow a similar structure to a Gaussian process (normal distribution) which would indicate that the model is a good fit. Looking at the ACF of the residuals this assumption would stand since no significant autocorrelation seems to be present. However the squared residuals show high autocorrelation which means that there is a GARCH effect. Therefore we should look into using a ARMA-GARCH model for this data set.

Exercise 4

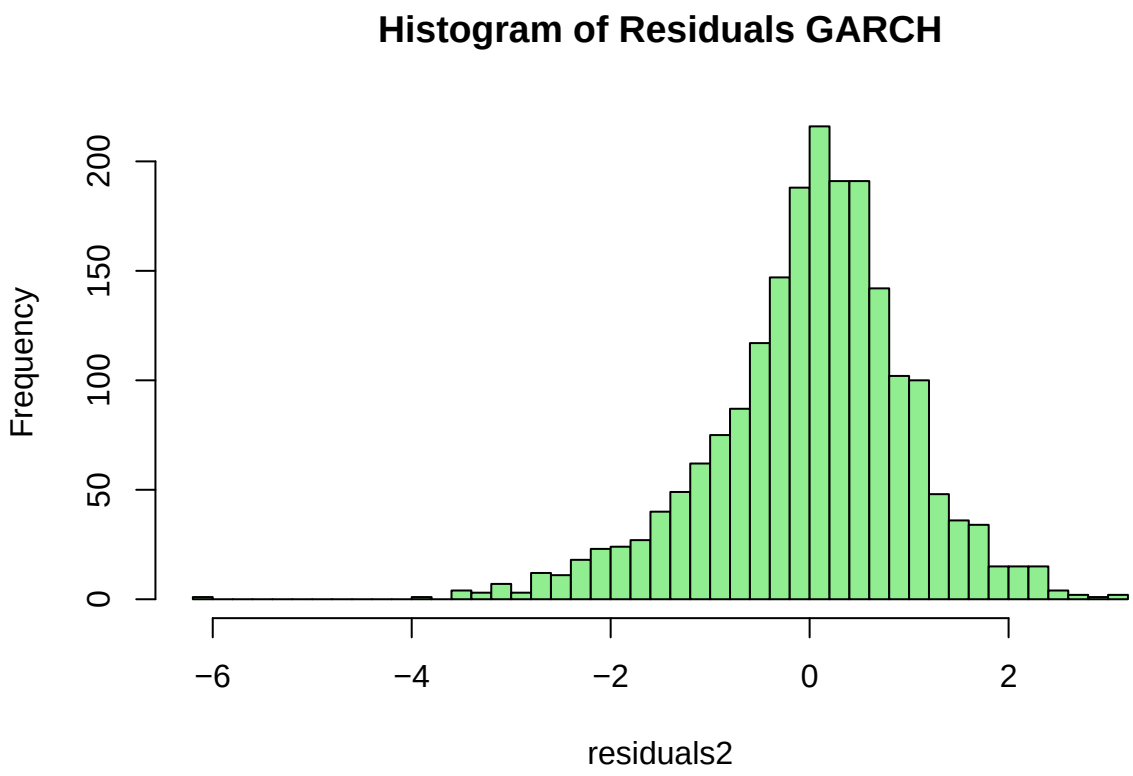
```
# Specify the GARCH(p, q) model orders
p_values <- 0:5
q_values <- 1:5

# Function to fit GARCH model and calculate AIC
fit_garch_model <- function(p, q) {
  garch_spec <- ugarchspec(mean.model = list(armaOrder = c(0, 0), include.mean = FALSE), variance.model
  garch_fit <- ugarchfit(spec = garch_spec, data = residuals)
  return(infocriteria(garch_fit)[1])
}

# Fit GARCH models and calculate AIC for different p and q values
aic_matrix <- matrix(nrow = length(p_values), ncol = length(q_values))
for (i in 1:length(p_values)) {
  for (j in 1:length(q_values)) {
    p <- p_values[i]
```

```
    q <- q_values[j]
    aic_matrix[i, j] <- fit_garch_model(p, q)
  }
}

# Find the minimum AIC value and corresponding p, q values
min_aic <- min(aic_matrix, na.rm = TRUE)
min_aic_indices <- which(aic_matrix == min_aic, arr.ind = TRUE)
best_p <- p_values[min_aic_indices[,1]]
best_q <- q_values[min_aic_indices[,2]]

# Print the results
cat("Best GARCH(p, q) model with minimum AIC:", "p =", best_p, "q =", best_q, "\n")
```

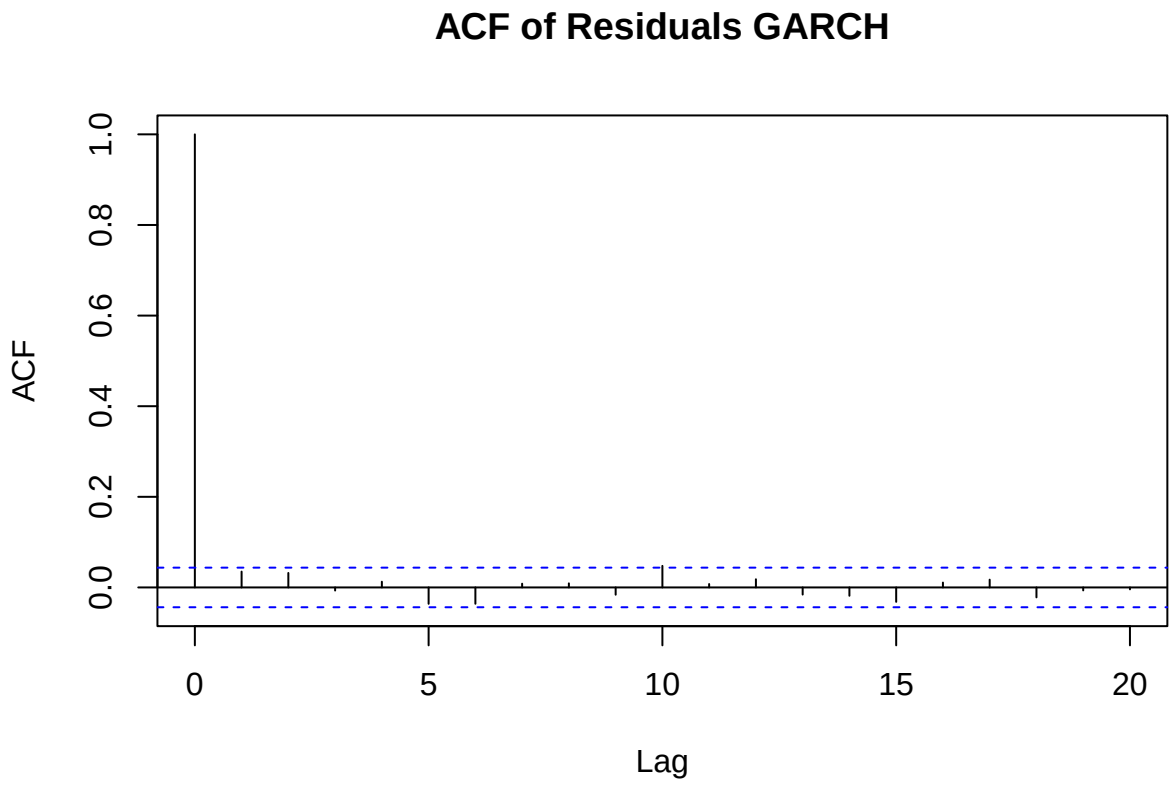
```
## Best GARCH(p, q) model with minimum AIC: p = 2 q = 1
```

Exercise 5

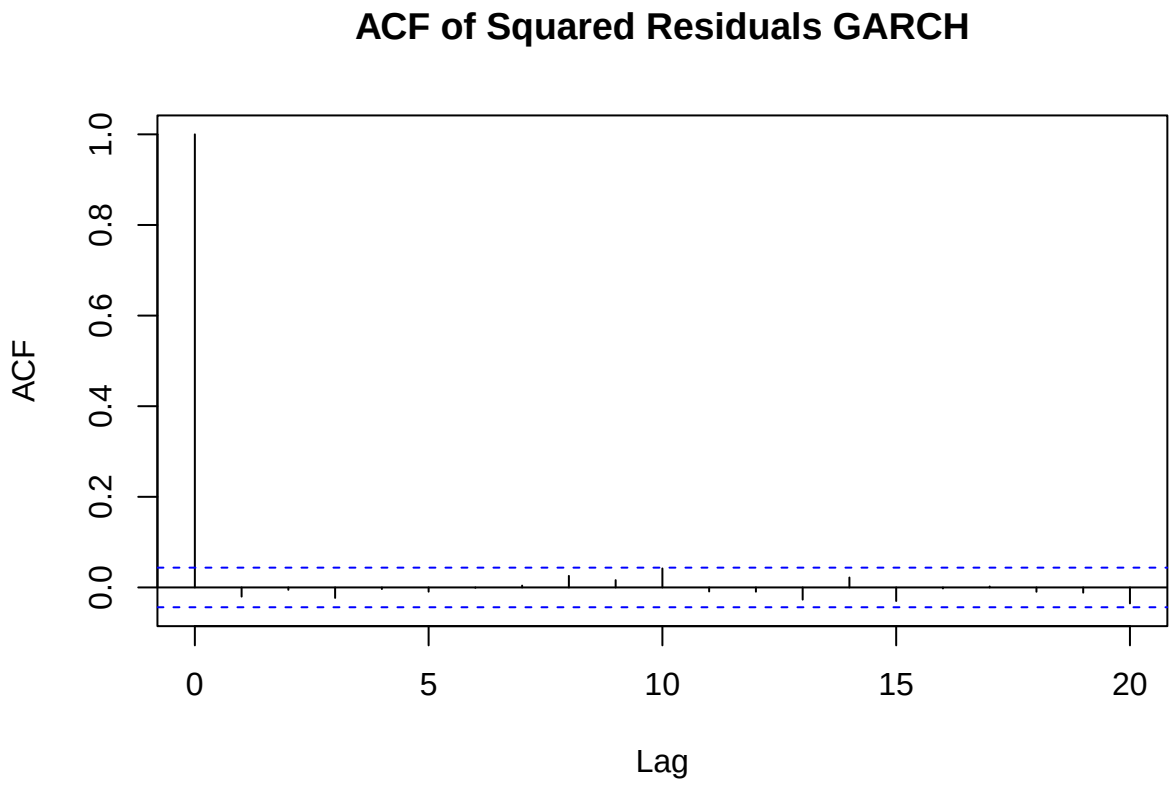
```
# Fit GARCH(2, 1) model to residuals
garch_spec2 <- ugarchspec(mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
                          variance.model = list(model = "sGARCH", garchOrder = c(2, 1)))
garch_fit2 <- ugarchfit(spec = garch_spec2, data = residuals)
residuals2 <- residuals(garch_fit2, standardize = TRUE)

# Distribution analysis
hist(residuals2, breaks = "FD", col = "lightgreen", main = "Histogram of Residuals GARCH")
```




```r
# Autocorrelation plot of residuals
acf(residuals2, lag.max = 20, main = "ACF of Residuals GARCH")
```




```r
# Autocorrelation plot of squared residuals
residualssq2 <- residuals2^2
acf(residualssq2, lag.max = 20, main = "ACF of Squared Residuals GARCH")
```



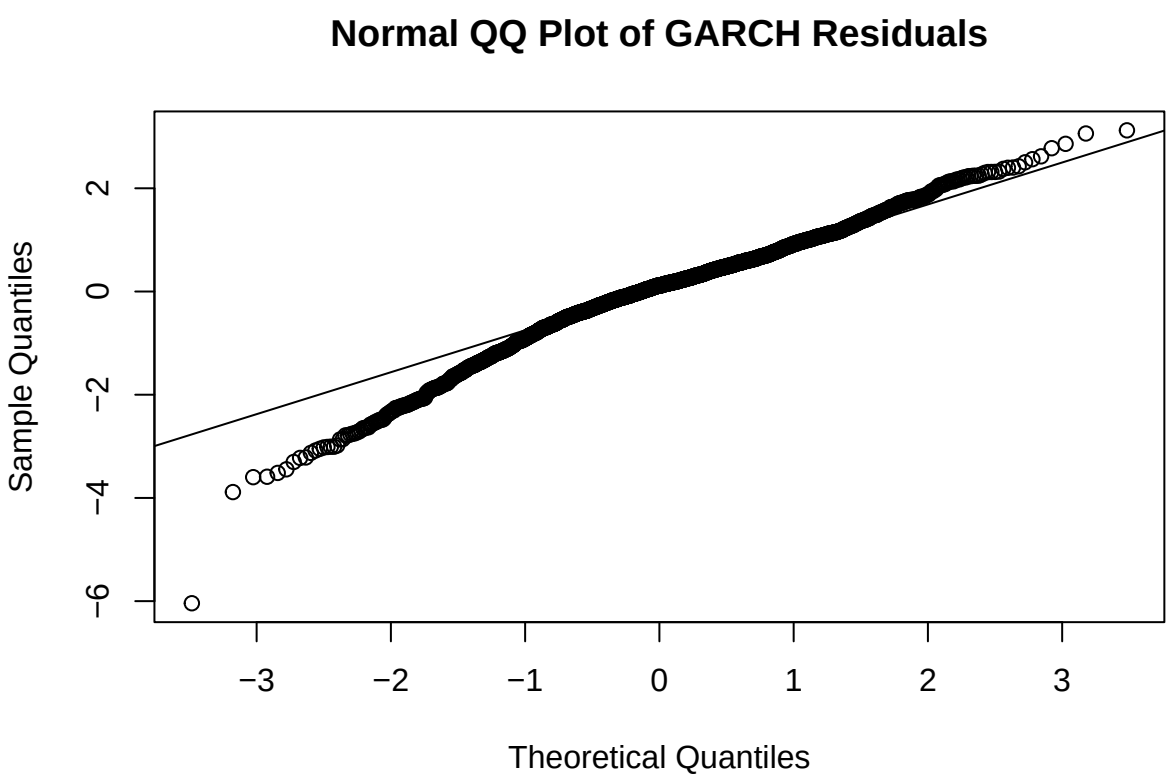
The histogram of the residuals of a GARCH model is similar to a normal distribution but exhibits a heavier or fatter tail on the right side, it suggests that the residuals have a distribution with greater positive skewness. This means that there is a higher probability of extreme positive values in the residuals compared to a normal distribution.

The presence of fat tails in the residuals is often associated with the phenomenon of extreme events or outliers occurring more frequently than what would be expected under a normal distribution. Fat tails indicate that the GARCH model may not fully capture the extreme behavior or volatility clustering observed in the data.

Other than the skewness of the histogram of the residuals, the ACFs seem to show no significant autocorrelation between the residuals.

Exercise 6

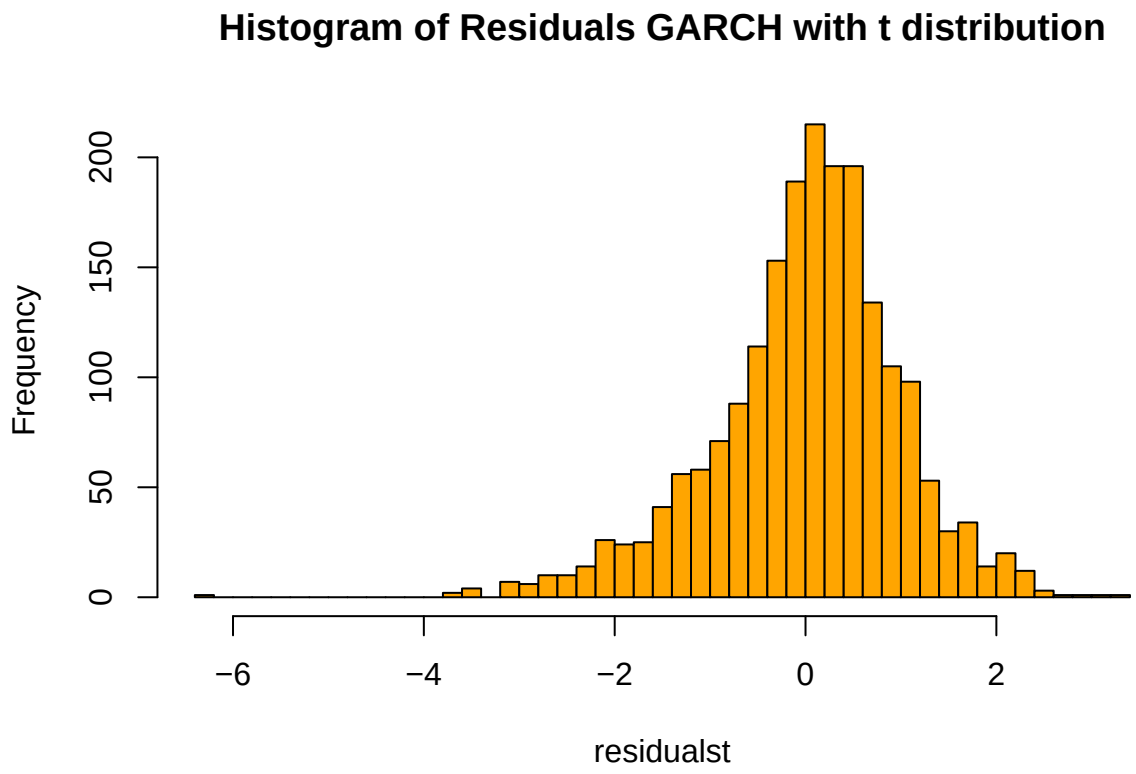
```
# Plot normal QQ plot of GARCH residuals
qqnorm(residuals2, main = "Normal QQ Plot of GARCH Residuals")
qqline(residuals2)
```



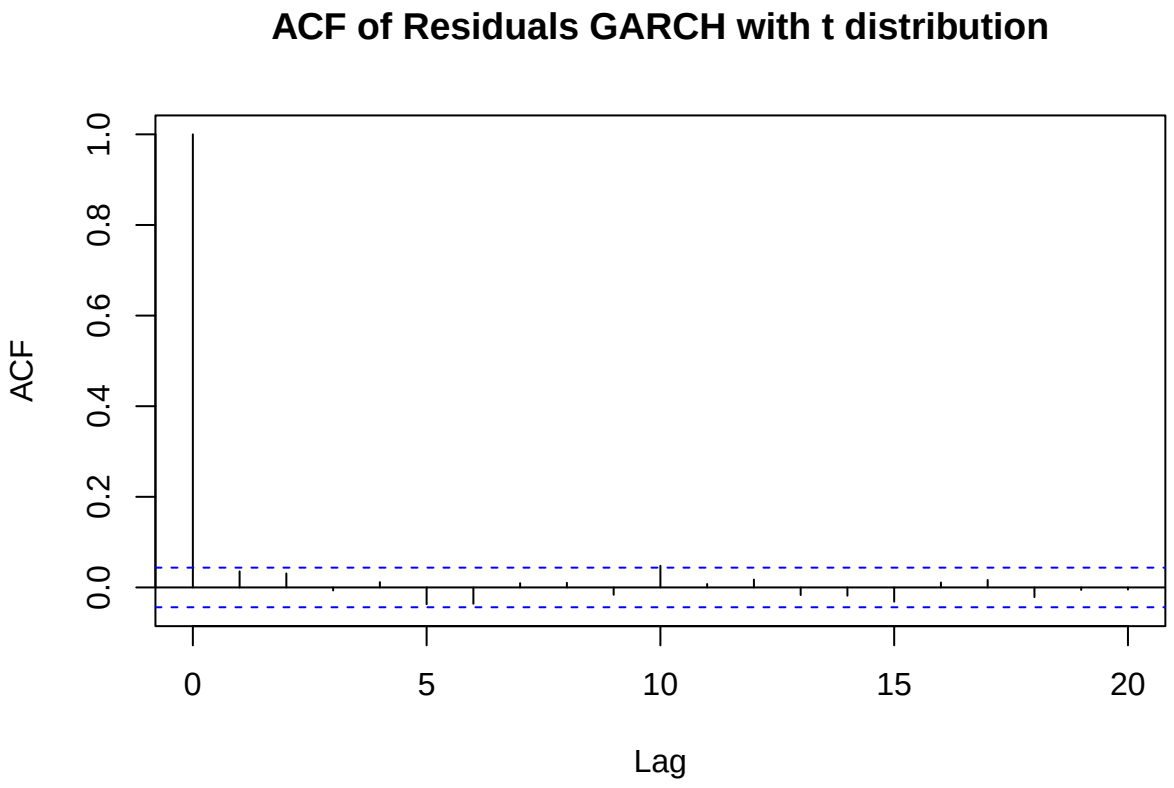
In a QQ plot we would expect for all data points to follow a similar path than the straight line, which would mean that it follow a Gaussian or normal distribution. we can see that this is not true for the above plot. In this plot we can see that the lower quantiles of the data have values which are much lower than expected and therefore it explains the light left tale on the histogram of the residuals. While in the higher quatiles we see that the data points are over the lines, this explains the fat right tale of the histogram. This plot tells us that we should make adjustments to the model.

Exercise 7

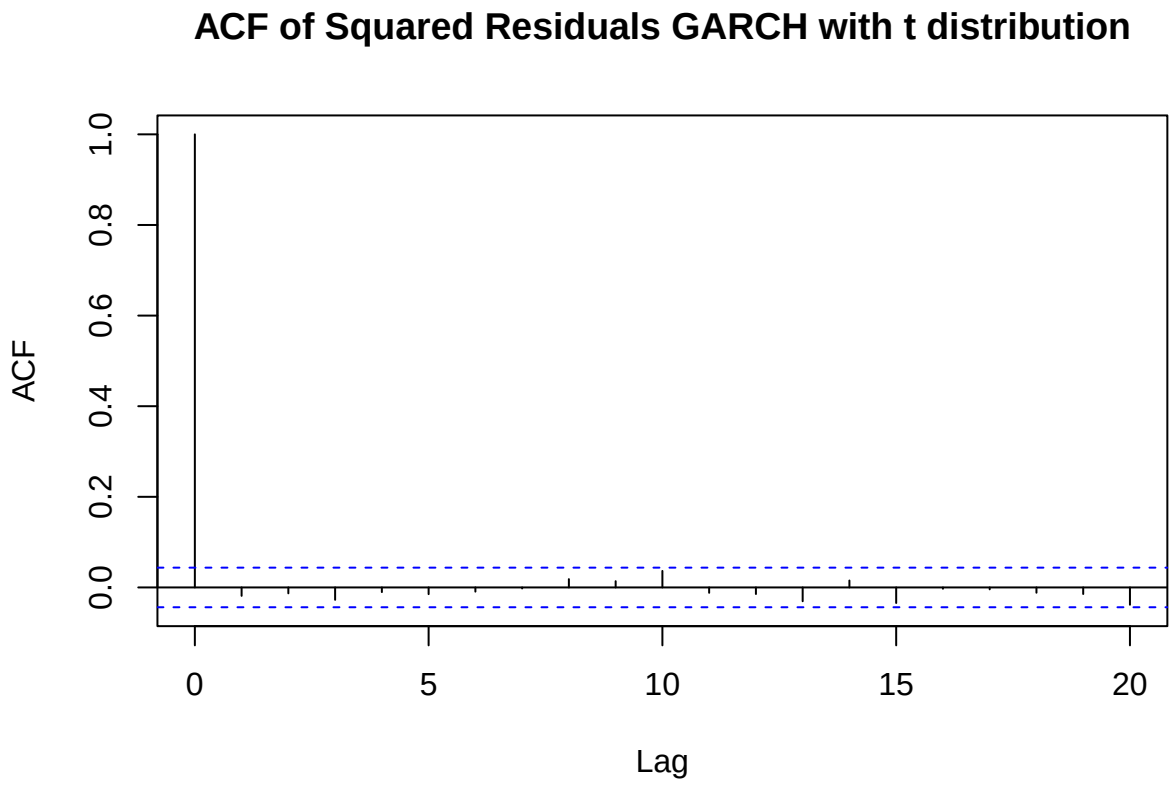
```
# Fit GARCH(2, 1) model to residuals using a t distribution
garch_spect <- ugarchspec(mean.model = list(armaOrder = c(0, 0), include.mean = FALSE),
                          variance.model = list(model = "sGARCH", garchOrder = c(2, 1)), distribution.m
garch_fitt <- ugarchfit(spec = garch_spect, data = residuals)
residualst <- residuals(garch_fitt, standardize = TRUE)
# Distribution analysis
hist(residualst, breaks = "FD", col = "orange", main = "Histogram of Residuals GARCH with t distributio
```



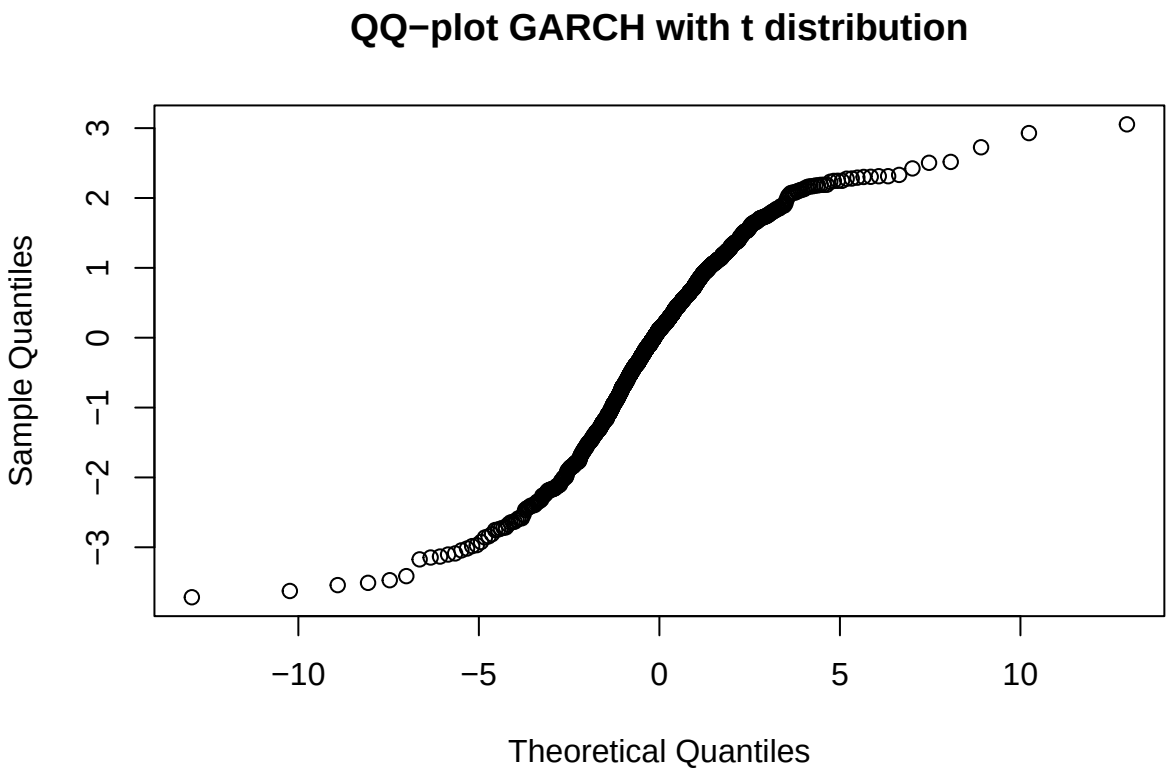

```
# Autocorrelation plot of residuals
acf(residualst, lag.max = 20, main = "ACF of Residuals GARCH with t distribution")
```




```r
# Autocorrelation plot of squared residuals
residualssqt <- residualst^2
acf(residualssqt, lag.max = 20, main = "ACF of Squared Residuals GARCH with t distribution")
```




```r
# qqplot function for t distribution
# Define a function named qqplotfunc that takes three arguments: Xdat, df, and name
qqplotfunc <- function(Xdat, df, name) {

  # Calculate the number of data points in the dataset Xdat
  npts <- length(Xdat)

  # Generate a sequence of evenly spaced values from 0 to 1, excluding the extreme values
  qvec <- seq(from = 0, to = 1, length.out = npts)[-c(1, npts)]

  # Generate the QQ plot
  # Plot the quantiles of the t distribution against the quantiles of the dataset Xdat
  plot(qt(qvec, df), quantile(Xdat, qvec),
       main = paste0(name),
       xlab = "Theoretical Quantiles",
       ylab = "Sample Quantiles")
}

#executind the qqplot
qqplotfunc(residualst, 3, "QQ-plot GARCH with t distribution")
```



As we saw before the residual data has fat tails which indicates that a t distribution might be better for the model. When running the test diagnostics for this new model we would expect to see both on the qq plot and the histogram fatter tails and we see them in both. The t distribution allows us to give more weight to outliers on the data and have a more robust model. JUdging the ACF and PACF we see that no autocorrelation occurs. This model is much better suitted for the data that we were given.

Exercise 8

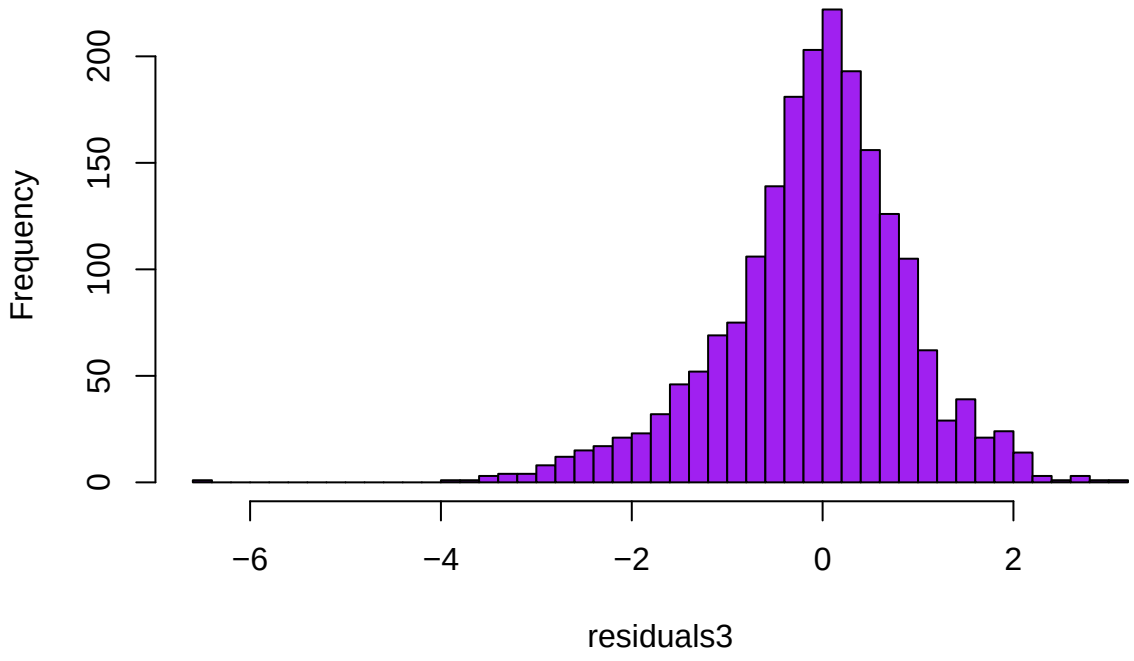
```
#For estimating the coefficients of the whole ARMA-GARCH model.
garch_spec3 <- ugarchspec(mean.model = list(armaOrder = c(3, 3), include.mean = TRUE),
                          variance.model = list(model = "sGARCH", garchOrder = c(2, 1)), distribution.m
garch_fit3 <- ugarchfit(spec = garch_spec3, data = LR)
residuals3 <- residuals(garch_fit3, standardize = TRUE)
garch_fit3
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
```

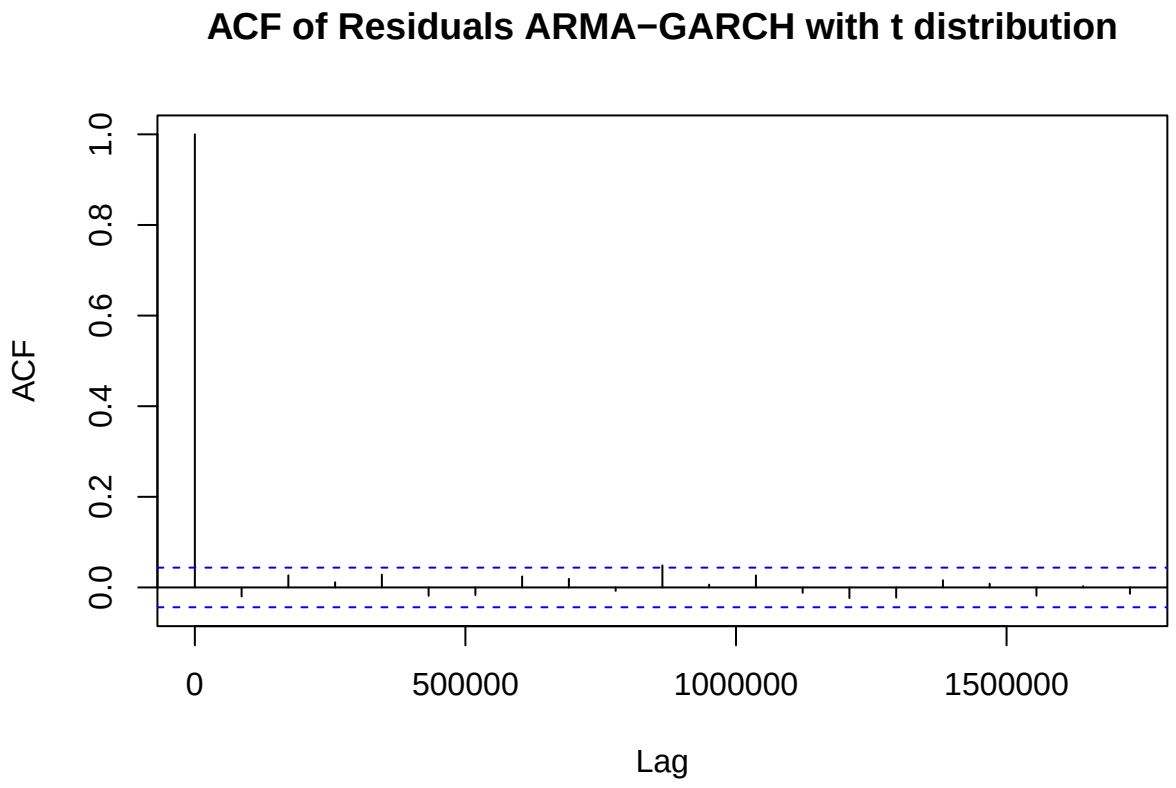
```
## Mean Model   : ARFIMA(3,0,3)
## Distribution : std
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000989    0.000124     7.9831 0.000000
## ar1     0.228394    0.017728    12.8831 0.000000
## ar2    -0.477117    0.015027   -31.7504 0.000000
## ar3     0.817287    0.012109    67.4941 0.000000
## ma1    -0.285819    0.014113   -20.2523 0.000000
## ma2     0.448792    0.017524    25.6107 0.000000
## ma3    -0.863691    0.000440 -1961.5983 0.000000
## omega   0.000004    0.000002     1.8515 0.064094
## alpha1  0.000000    0.024910     0.0000 1.000000
## alpha2  0.181463    0.037058     4.8967 0.000001
## beta1   0.813122    0.026237    30.9916 0.000000
## shape   4.994844    0.652860     7.6507 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.000989    0.000126     7.86700 0.000000
## ar1     0.228394    0.017672    12.92439 0.000000
## ar2    -0.477117    0.013632   -34.99997 0.000000
## ar3     0.817287    0.011971    68.27390 0.000000
## ma1    -0.285819    0.013400   -21.32966 0.000000
## ma2     0.448792    0.016215    27.67685 0.000000
## ma3    -0.863691    0.000563 -1533.04208 0.000000
## omega   0.000004    0.000004     0.83466 0.403912
## alpha1  0.000000    0.029834     0.00000 1.000000
## alpha2  0.181463    0.052383     3.46418 0.000532
## beta1   0.813122    0.042326    19.21086 0.000000
## shape   4.994844    0.613167     8.14597 0.000000
##
## LogLikelihood : 6376.949
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.3238
## Bayes        -6.2904
## Shibata      -6.3239
## Hannan-Quinn -6.3116
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                          statistic p-value
## Lag[1]                       0.795  0.3726
## Lag[2*(p+q)+(p+q)-1][17]     9.251  0.3296
## Lag[4*(p+q)+(p+q)-1][29]    13.346  0.7020
## d.o.f=6
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
```

```
## ------------------------------------
##                           statistic p-value
## Lag[1]                       0.5872  0.4435
## Lag[2*(p+q)+(p+q)-1][8]      2.9626  0.6924
## Lag[4*(p+q)+(p+q)-1][14]     5.6419  0.6868
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]   0.05091 0.500 2.000  0.8215
## ARCH Lag[6]   1.57568 1.461 1.711  0.5913
## ARCH Lag[8]   1.86037 2.368 1.583  0.7696
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  8.1802
## Individual Statistics:
## mu     0.05527
## ar1    0.04753
## ar2    0.04701
## ar3    0.02095
## ma1    0.04234
## ma2    0.06930
## ma3    0.02093
## omega  0.25031
## alpha1 0.12808
## alpha2 0.70138
## beta1  0.77873
## shape  0.29893
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.69 2.96 3.51
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value     prob sig
## Sign Bias           1.6447 0.100180
## Negative Sign Bias  0.1152 0.908280
## Positive Sign Bias  1.8367 0.066397   *
## Joint Effect       13.4415 0.003773 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     63.00    1.287e-06
## 2    30     79.27    1.483e-06
## 3    40     95.36    1.272e-06
## 4    50    114.65    3.519e-07
##
##
## Elapsed time : 1.524686
```

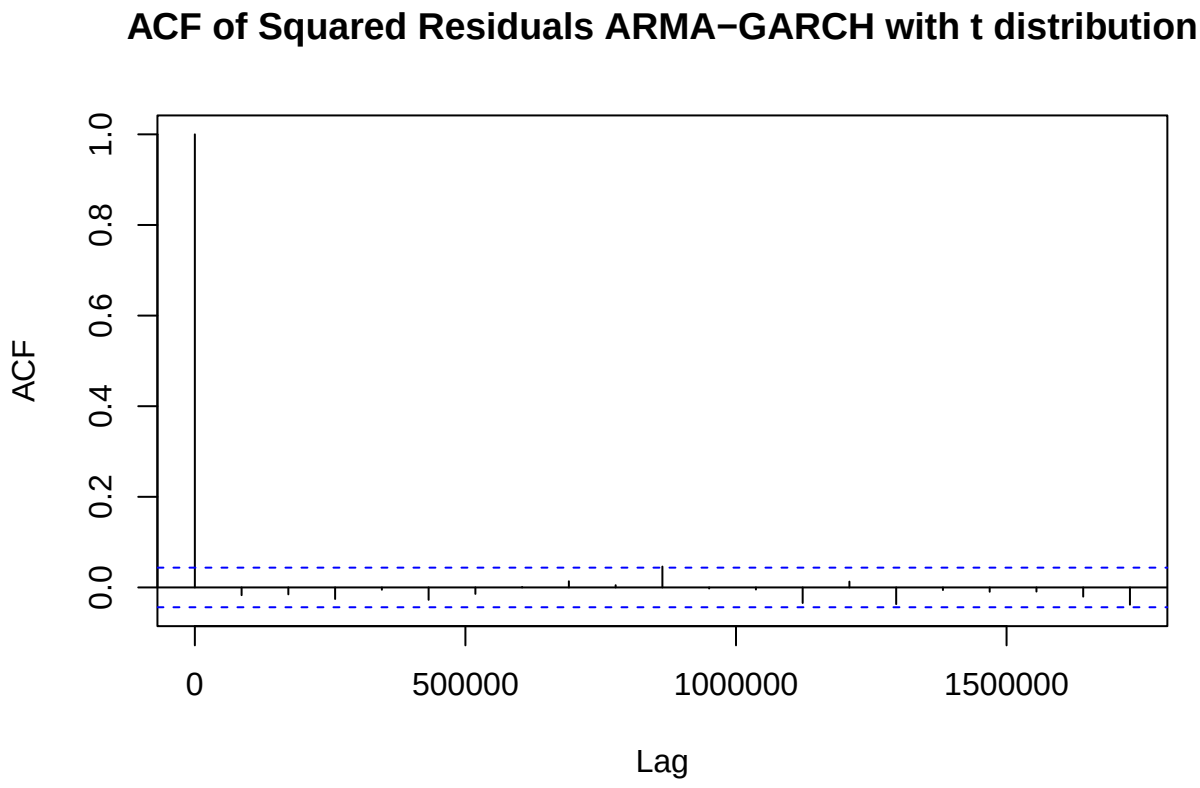
```r
# Distribution analysis
hist(residuals3, breaks = "FD", col = "purple", main = "Histogram of Residuals ARMA-GARCH with t distri
```



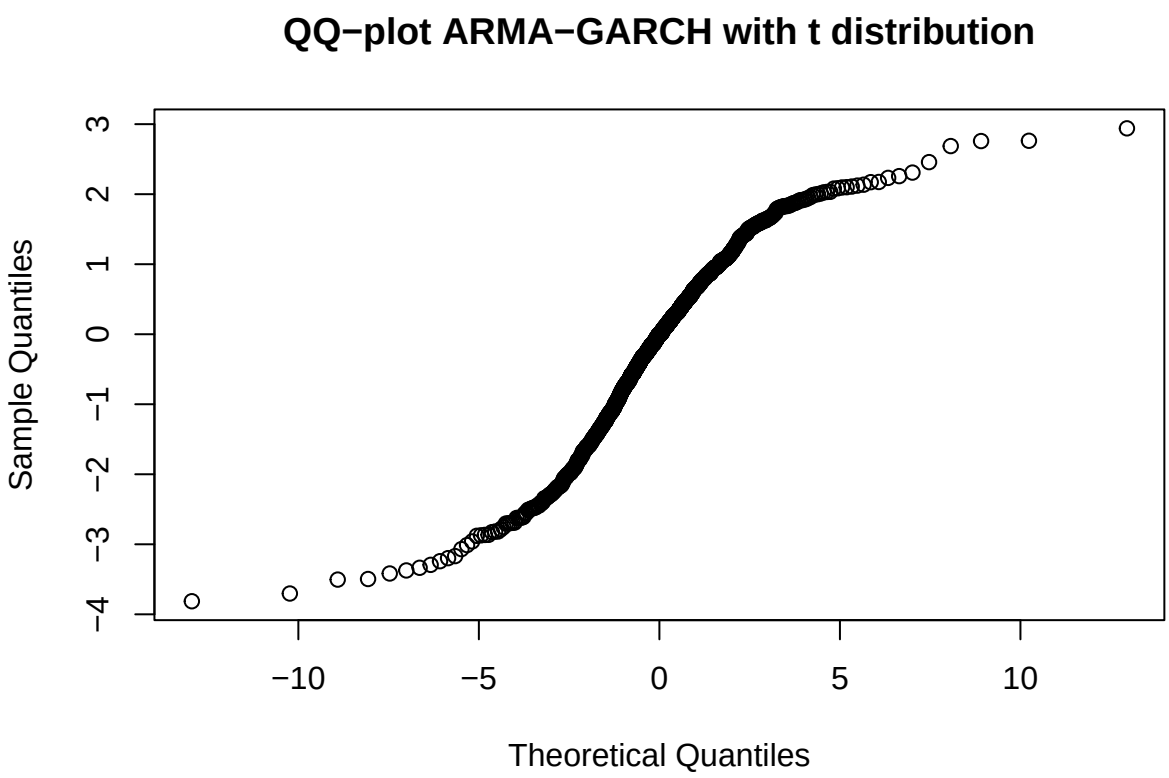

```r
# Autocorrelation plot of residuals
acf(residuals3, lag.max = 20, main = "ACF of Residuals ARMA-GARCH with t distribution")
```




```
# Autocorrelation plot of squared residuals
residualssq3 <- residuals3^2
acf(residualssq3, lag.max = 20, main = "ACF of Squared Residuals ARMA-GARCH with t distribution")
```



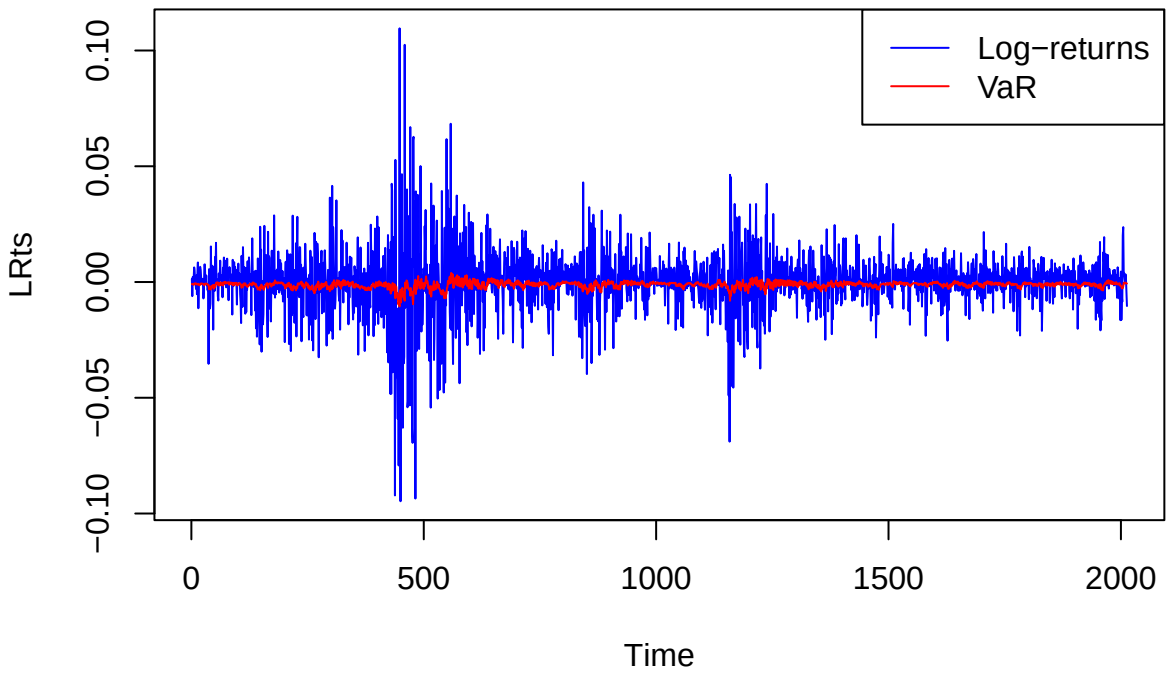

```
#executind the qqplot
qqplotfunc(residuals3, 3, "QQ-plot ARMA-GARCH with t distribution")
```



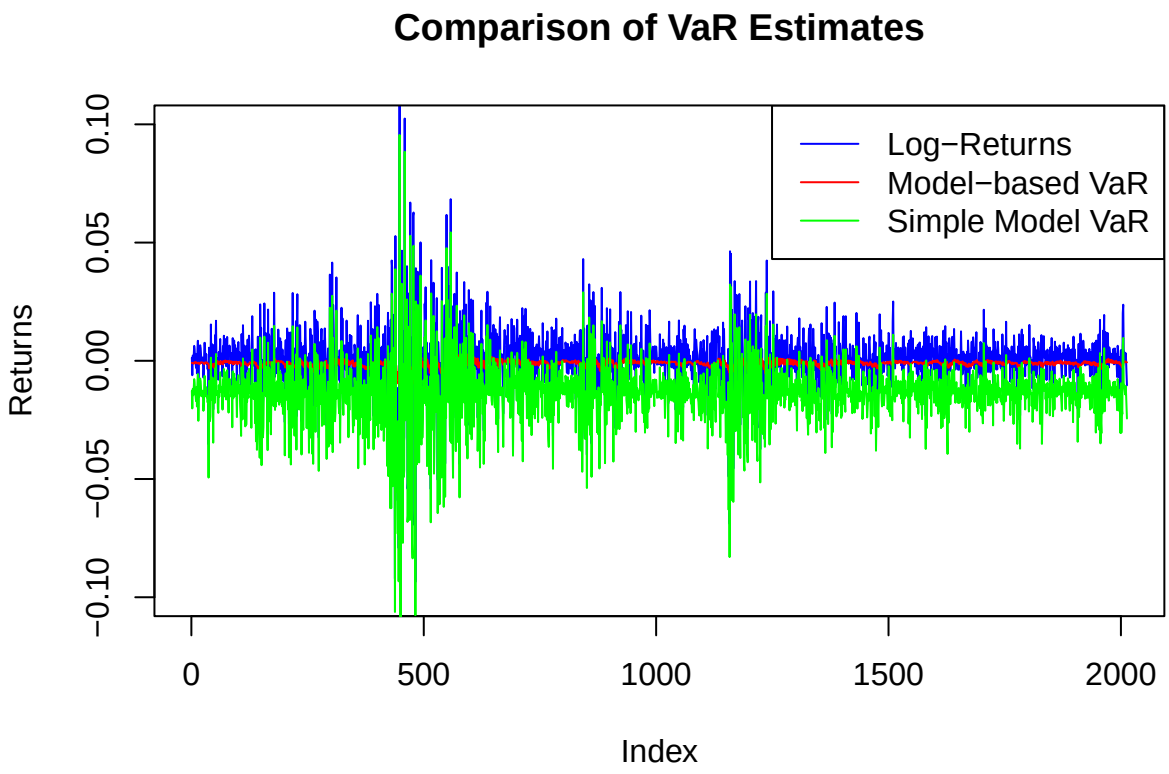
All diagnostics seem to check. The histogram and qq-plot show fat tails and assign more weight to the outliers while the ACF and PACF show no autocorrelation of the residuals.

Exercise 9

```
#Extract the fitted values
fitted_values <- fitted(garch_fit3)

#Calculate the inverse of the cumulative distribution function (CDF) of the standardized residuals at a
alpha <- 0.05
quantile <- qt(alpha, 3)

#Compute the VaR for each time period t using the formula:
VaR <- fitted_values * (exp(quantile) - 1)

# Convert log_returns and VaR to time series
LRts <- ts(LR, start = start(LR), frequency = frequency(LR))
VaR_ts <- ts(VaR, start = start(LR), frequency = frequency(LR))

#Plot
plot(LRts, type = "l", col = "blue", ylim = c(min(LRts), max(LRts)))
lines(VaR_ts, col = "red")
legend("topright", legend = c("Log-returns", "VaR"), col = c("blue", "red"), lty = 1)
```




```r
# Calculate sample variance
sample_var <- var(LR)

# Calculate VaR based on the simple model
VaR_simple <- LR - sqrt(sample_var)

# Plot log-returns, model-based VaR, and simple model VaR
plot(LR, type = "l", col = "blue", ylim = c(-0.1, 0.1), main = "Comparison of VaR Estimates", ylab = "R
lines(VaR_ts, col = "red")
lines(VaR_simple, col = "green")
legend("topright", legend = c("Log-Returns", "Model-based VaR", "Simple Model VaR"), col = c("blue", "r
```



The second estimation seems to follow a much better path since it is more similar to the actual Log-returns of the S&P 500, while the first estimation gives a much lower value at risk that does not overlap the Log retuns at any point as we can see on the graph.

Exercise 10

```
# Number of exceedances for VaR_simple
Exsimple <- sum(LR < VaR_simple)

# Number of exceedances for VaR_model
Exmodel <- sum(LR < VaR_ts)

cat("Number of exceedances for VaR_simple:", Exsimple, "\n")
```

```
## Number of exceedances for VaR_simple: 0
```

```
cat("Number of exceedances for VaR_ts:", Exmodel, "\n")
```

```
## Number of exceedances for VaR_ts: 831
```

A

A lower number of exceedances indicates that the VaR estimate is more conservative and captures a larger proportion of extreme events. It suggests that the estimated VaR provides a more accurate and reliable measure of downside risk.

On the other hand, a higher number of exceedances implies that the VaR estimate is not capturing enough extreme events, and there is a higher frequency of returns exceeding the VaR threshold.

This is consistent with the process for creating a model since we had to account for more outliers and extreme events by using a t-dsitribution.

B

Under the simplifying assumption that the event of observing an exceedance on a given day is independent of observing exceedances on all other days and has probability α, the number of exceedances follows a binomial distribution. The expected number of exceedances can be calculated with $n*\alpha$, which represents the average number of exceedances we would expect to observe based on the assumed probability of exceedance on each day.

These assumptions do not seem to be realistic as they assume independence and a constant probability of exceedance for each day.

C

```
# Expected number of exceedances
Exexp <- length(LR) * alpha

# Compare the number of exceedances with the expected number of observations
if ( Exsimple > Exexp) {
  message("VaR_simple overestimates the number of exceedances.")
} else if (Exsimple < Exexp) {
  message("VaR_simple underestimates the number of exceedances.")
} else {
  message("VaR_simple accurately estimates the number of exceedances.")
}
```

```
## VaR_simple underestimates the number of exceedances.
```

```
if (Exmodel > Exexp) {
  message("VaR_ts overestimates the number of exceedances.")
} else if (Exmodel < Exexp) {
  message("VaR_ts underestimates the number of exceedances.")
} else {
  message("VaR_ts accurately estimates the number of exceedances.")
}
```

```
## VaR_ts overestimates the number of exceedances.
```

For $\alpha = 0.05$ the simple model Values at Risk underestimate the exceedances, while the ts model overestimates the exceedances. Ideally we would want the models to be as a close to the expected values as possible

but if we have to choose between underestimating or overestimatig it would be preferable to overestimate the values at risk since we may loose on potential profits but the trades that we do are safer and lead to profits overtime, while understimating would mean that we are taking on trades that are riskier than assumed and can lead to potential loses. On finance we always search for lower risk.