MS&E349 HW 1

Tracy Cao, Sven Lerner, Jordan Rowley

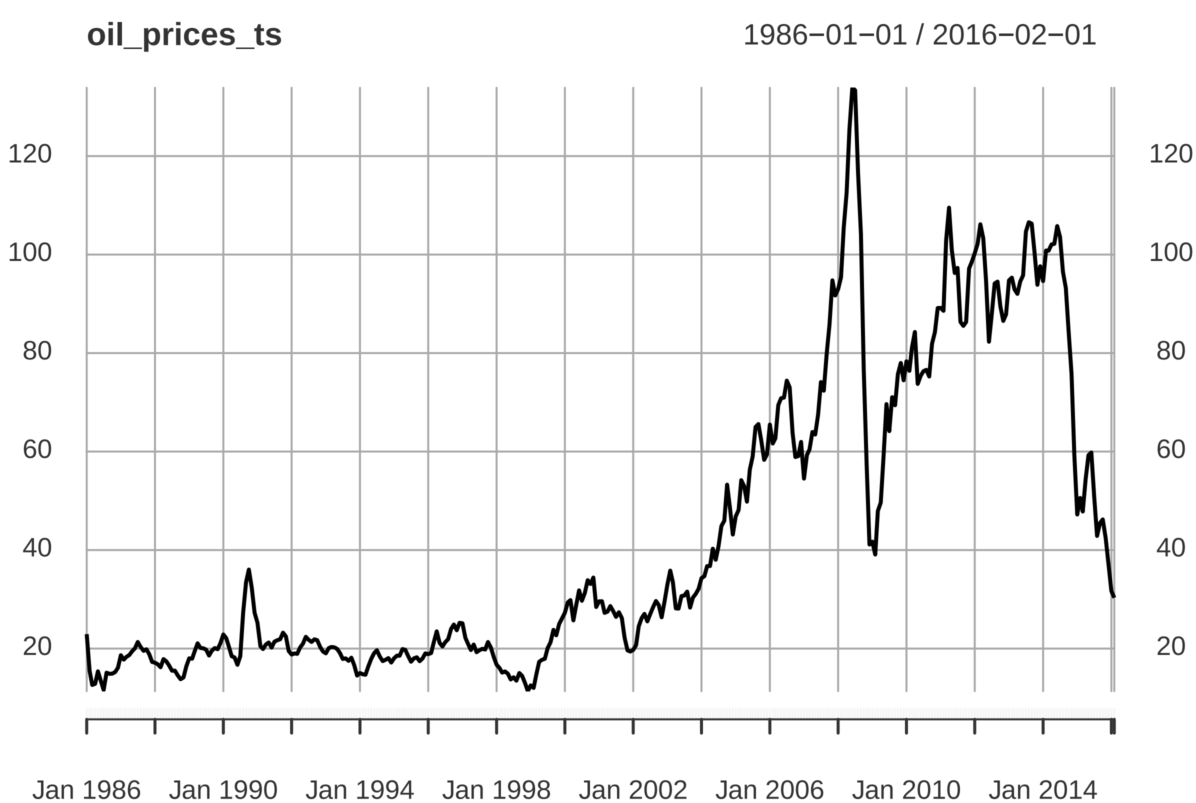
1/23/2021

Question 3: ARIMA

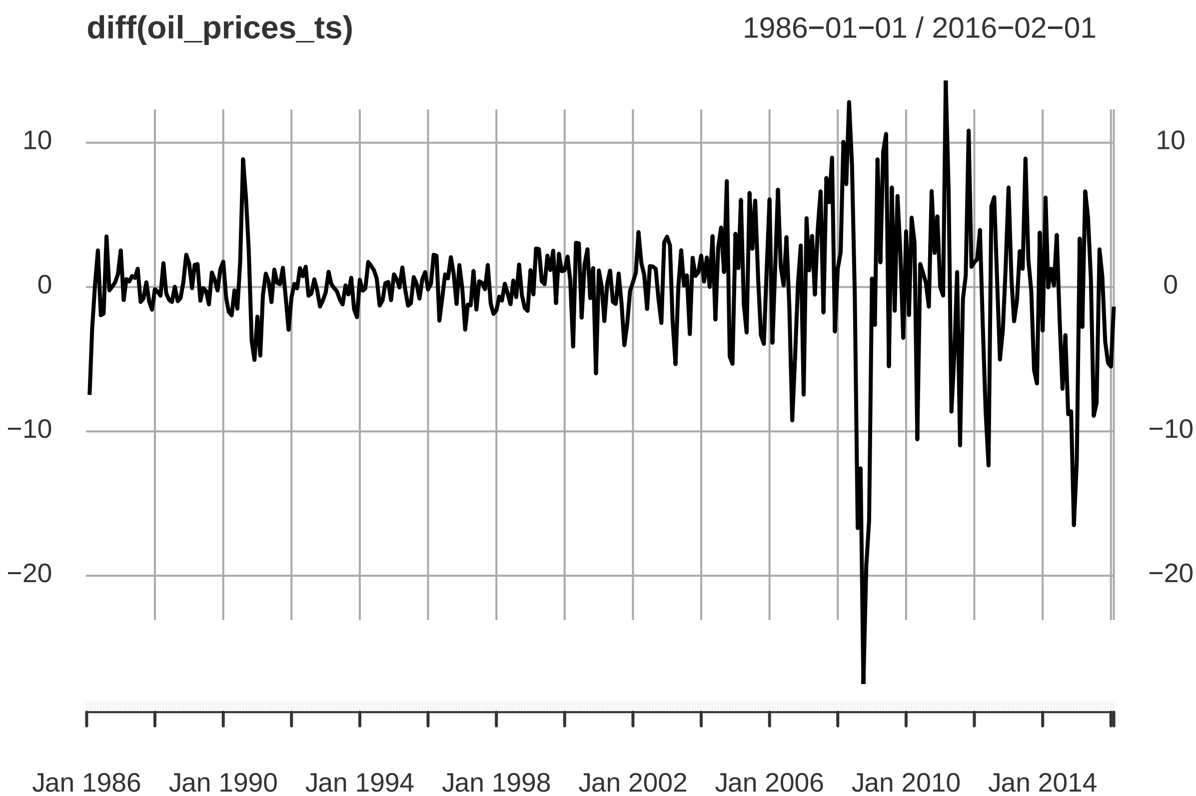
1.

First we'll import and plot the oil price data along with its first-difference.

```
oil_prices_raw <- read.table("m-COILWTICO.txt", header = TRUE)
oil_prices_ts <- xts(oil_prices_raw[,2],
                     order.by = as.Date(oil_prices_raw[,1]))

plot(oil_prices_ts)
```



```r
plot(diff(oil_prices_ts))
```



2.

While the first-differenced series seems to be centered about a mean of zero, we see clustering of volatility, which indicates that the series is not weakly stationary. While this series does not appear to be stationary, we will use the augmented Dickey-Fuller test to measure whether this series exhibits unit-root behavior.

3.

```r
adf.test(diff(oil_prices_ts)[-1])
```

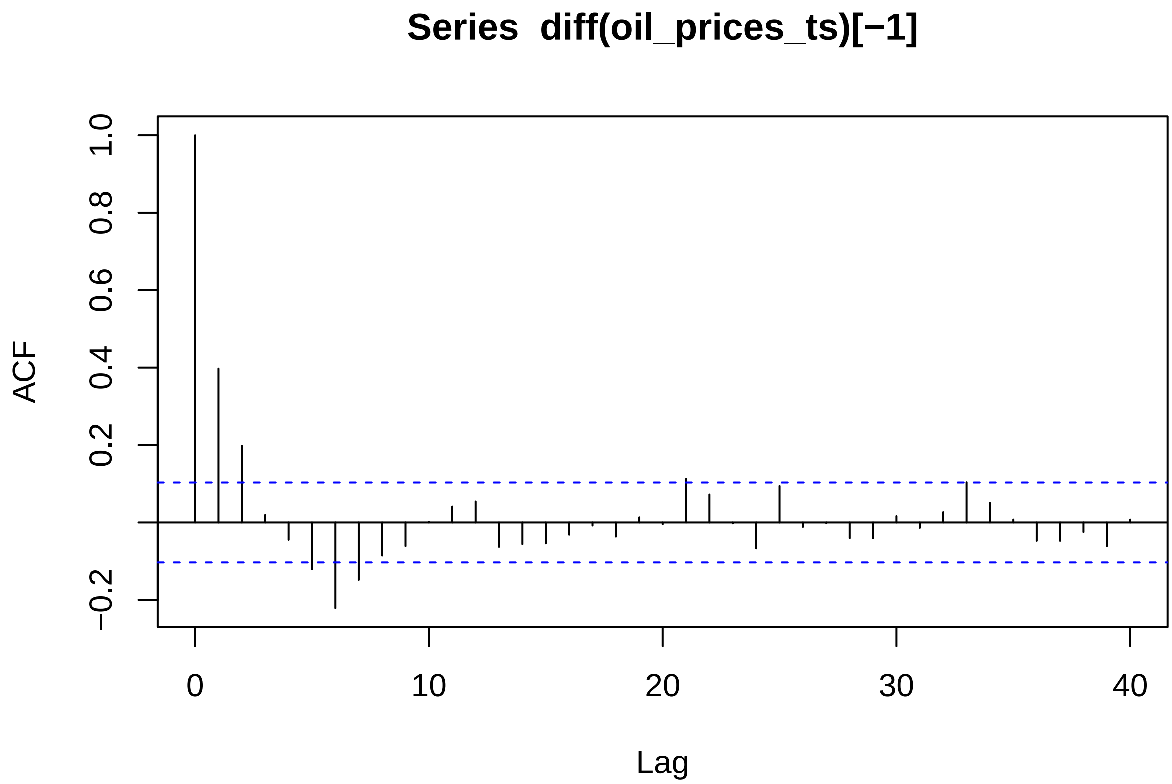
```
## Warning in adf.test(diff(oil_prices_ts)[-1]): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(oil_prices_ts)[-1]
## Dickey-Fuller = -7.3575, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

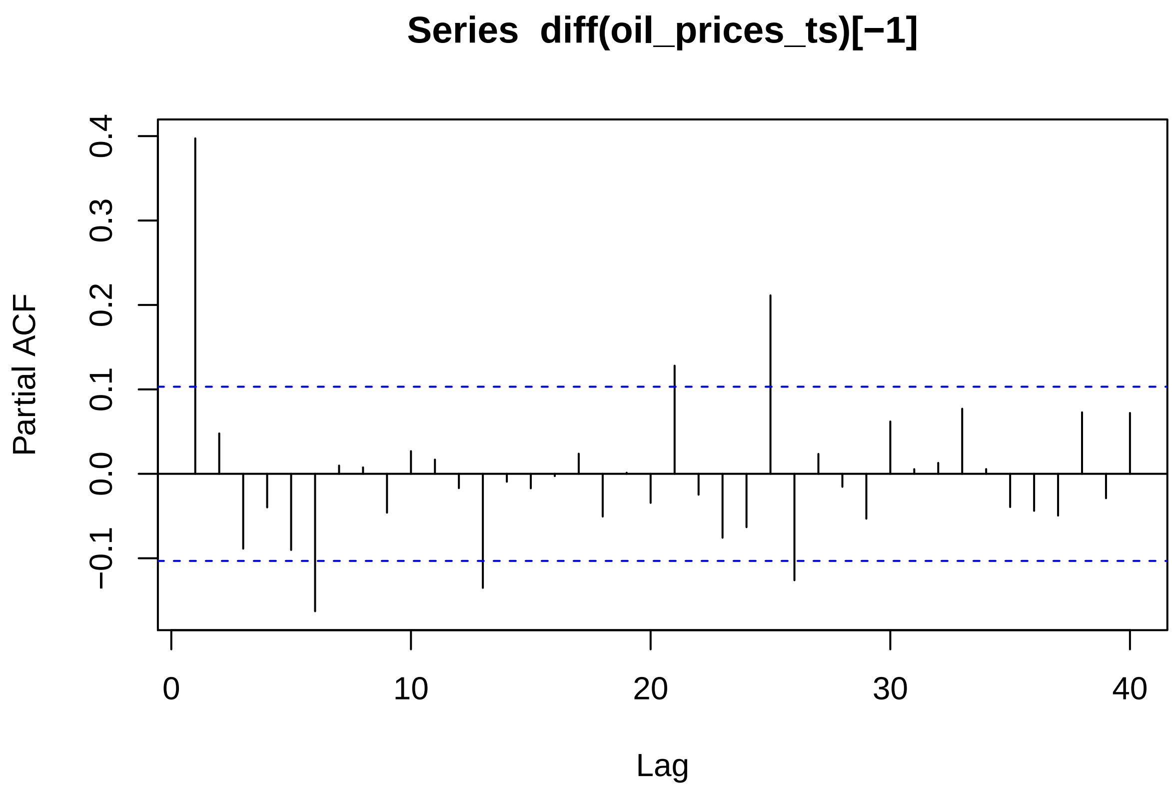
The ADF test gives us reason to believe that this series is indeed weakly stationary! We should therefore expect the ACF and PACF to be relatively tame.

4.

```r
acf(diff(oil_prices_ts)[-1], lag.max = 40)
```




```r
pacf(diff(oil_prices_ts)[-1], lag.max = 40)
```



We see oscillatory behavior in both the ACF and PACF. This indicates that there are both moving-average and autoregressive processes at work within this time series, but we do not see many spikes exceeding the blue 95% lines. Perhaps this series is well-approximated by white noise? We will use the Ljung-Box test to measure this.

5.

```r
Box.test(diff(oil_prices_ts)[-1], type = "Ljung-Box", lag = 12)
```

```
##
##  Box-Ljung test
##
## data:  diff(oil_prices_ts)[-1]
## X-squared = 110.11, df = 12, p-value < 2.2e-16
```

The small p-value indicates that there is indeed some structure in this time series that could be explained by moving-average or autoregressive processes.

6.

Let's fit an $AR(p)$ model to this time series. To estimate p, we notice that the PACF assumes relatively smaller values after lag 1, which indicates that an $AR(1)$ model may be appropriate. On the other hand, we also see spikes occur in intervals roughly equal to 6, indicating that an $AR(6)$ model may work well too, with $\phi_2 = ... = \phi_5 = 0$. We'll compare the AIC of these models.

```r
model1 <- arima(diff(oil_prices_ts)[-1], order = c(1, 0, 0))
model2 <- arima(diff(oil_prices_ts)[-1], order = c(6, 0, 0))

summary(model1)
```

```
##
## Call:
## arima(x = diff(oil_prices_ts)[-1], order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.3997     0.0036
## s.e.  0.0484     0.3466
##
## sigma^2 estimated as 15.68:  log likelihood = -1009.14,  aic = 2024.29
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.01032261 3.959947 2.642383 74.38655 241.1898 0.8096018
##                     ACF1
## Training set -0.02159609
```

```r
summary(model2)
```

```
##
## Call:
## arima(x = diff(oil_prices_ts)[-1], order = c(6, 0, 0))
##
## Coefficients:
##         ar1     ar2      ar3     ar4      ar5      ar6  intercept
##       0.362  0.0779  -0.0761  0.0059  -0.0291  -0.1626     0.0260
## s.e.  0.052  0.0554   0.0556  0.0555   0.0553   0.0520     0.2483
##
## sigma^2 estimated as 14.95:  log likelihood = -1000.7,  aic = 2017.4
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.001724805 3.867143 2.660276 101.1885 226.8743 0.8150842
##                       ACF1
## Training set -0.0002713045
```

We exclude the intercept/mean terms given its low t-score, in addition to the $\phi_2 \dots \phi_5$ terms. With these terms excluded, the aic of the $AR(6)$ model increases by roughly 10, while that of the $AR(1)$ model increases by about 2 (since the log likelihood will also change). Since the AIC of the $AR(6)$ model is slightly higher, we choose a simpler $AR(1)$ model of the form

$$Y_t = \phi_1 Y_{t-1} + \epsilon_t$$

with $\phi_1 \approx 0.3997$, $\epsilon_t \sim N(0, \sigma^2)$ where $\sigma^2 \approx 15.68$.

7.

Now we will fit an ARIMA model to our time series. To decide between models, we will again use the AIC.

```
min_aic <- Inf # keep a running min
best_order <- c(0, 0, 0)
  for (p in 0:6) for (q in 0:6){
    if ( p==0 && q == 0){
      next
    }
    arimaFit <- tryCatch(arima(diff(oil_prices_ts)[-1], order = c(p, 0, q)),
                         error = function(err) FALSE,  # turn failures to converge into logical output
                         warning = function(err) FALSE)

    if (!is.logical(arimaFit)){  # if convergence took place
      current_aic <- AIC(arimaFit)
      if(current_aic < min_aic){
        min_aic <- current_aic
        best_order <- c(p, 0, q)
        final.arima <- arima(diff(oil_prices_ts)[-1], order=best_order)
      }
    }
    else {
      next
    }
  }
arima_model <- arima(diff(oil_prices_ts)[-1], order = best_order, include.mean = TRUE)
print(arima_model$coef / sqrt(diag(arima_model$var.coef)))  #obtain t-ratios for each estimate
```

```
##        ar1        ar2        ar3        ar4        ar5        ma1        ma2
##  7.0954919 -7.2770703  7.9364692 -2.7626845 -3.0540338 -3.7334908  8.4205928
##        ma3  intercept
## -8.9626956  0.2637549
```

```
print(arima_model$sigma2) #esimate of noise variance
```

```
## [1] 14.62861
```

An ARIMA(5, 0, 3) model has the minimal AIC. The intercept term is excluded given its low t-ratio. The fitted model is therefore of the form $\Phi(B)Y_t = \Theta(B)\epsilon_t$, where $\Phi(B) = 1-\phi_1-\cdots-\phi_5$, $\Theta(B) = 1-\theta_1-\theta_2-\theta_3$, and $\epsilon_i \sim N(0, \sigma^2)$, $\sigma^2 \approx 14.6$

8

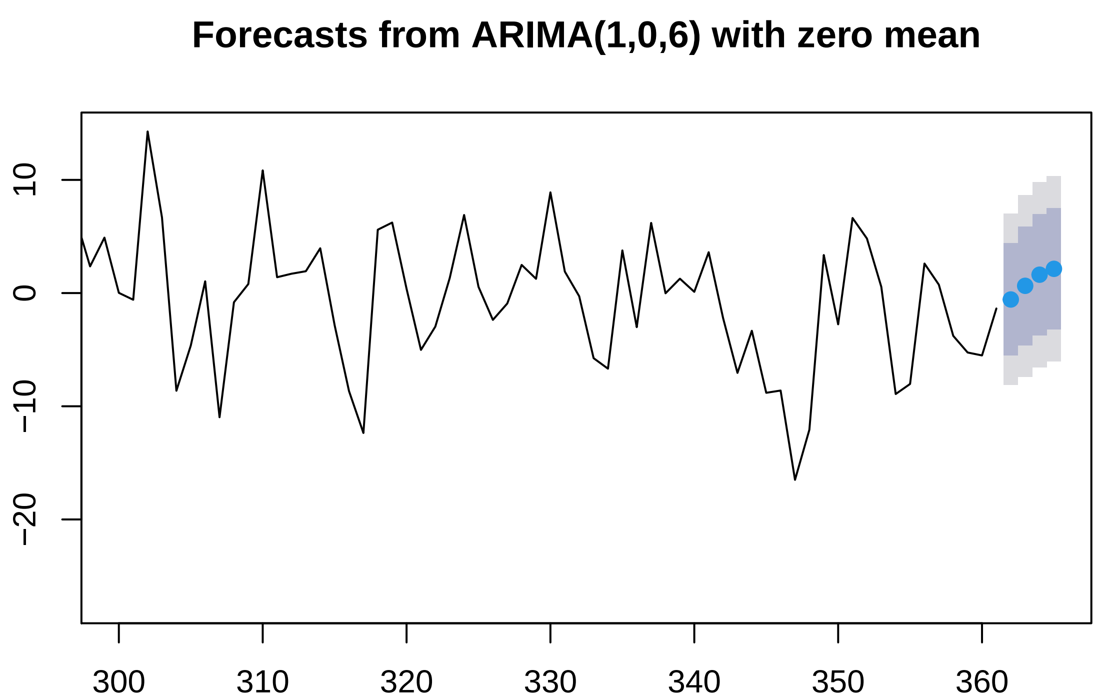
Now we will fit and ARIMA(1, 0, 6) model, and give a 4-step forecast along with 95% intervals. First, we determine whether we should include the mean in our model.

```
arima_model <- arima(diff(oil_prices_ts)[-1], order = c(1, 0, 6), include.mean = TRUE)
print(arima_model$coef / sqrt(diag(arima_model$var.coef)))  #obtain t-ratios for each estimate
```

```
##        ar1        ma1        ma2        ma3        ma4        ma5        ma6
##  5.5562268 -2.6145374 -0.5710060 -2.2037311 -0.2743020 -0.5706089 -3.1173202
##  intercept
##  0.3156258
```

The now t-ratio for the intercept terms warrants its exclusion in our model. We now create a forecast without it.

```r
arima_model <- arima(diff(oil_prices_ts)[-1], order = c(1, 0, 6), include.mean = FALSE)

four_step_forecast <- forecast(arima_model, h = 4)
plot(four_step_forecast, xlim = c(300, 365))  # zoom in
```




```r
four_step_forecast
```

```
##     Point Forecast     Lo 80    Hi 80     Lo 95     Hi 95
## 362     -0.5657690 -5.514318 4.382780 -8.133921  7.002383
## 363      0.6424893 -4.610869 5.895848 -7.391829  8.676808
## 364      1.6254031 -3.727451 6.978258 -6.561081  9.811888
## 365      2.1435675 -3.209773 7.496908 -6.043661 10.330796
```

Question 4. GARCH

##1.

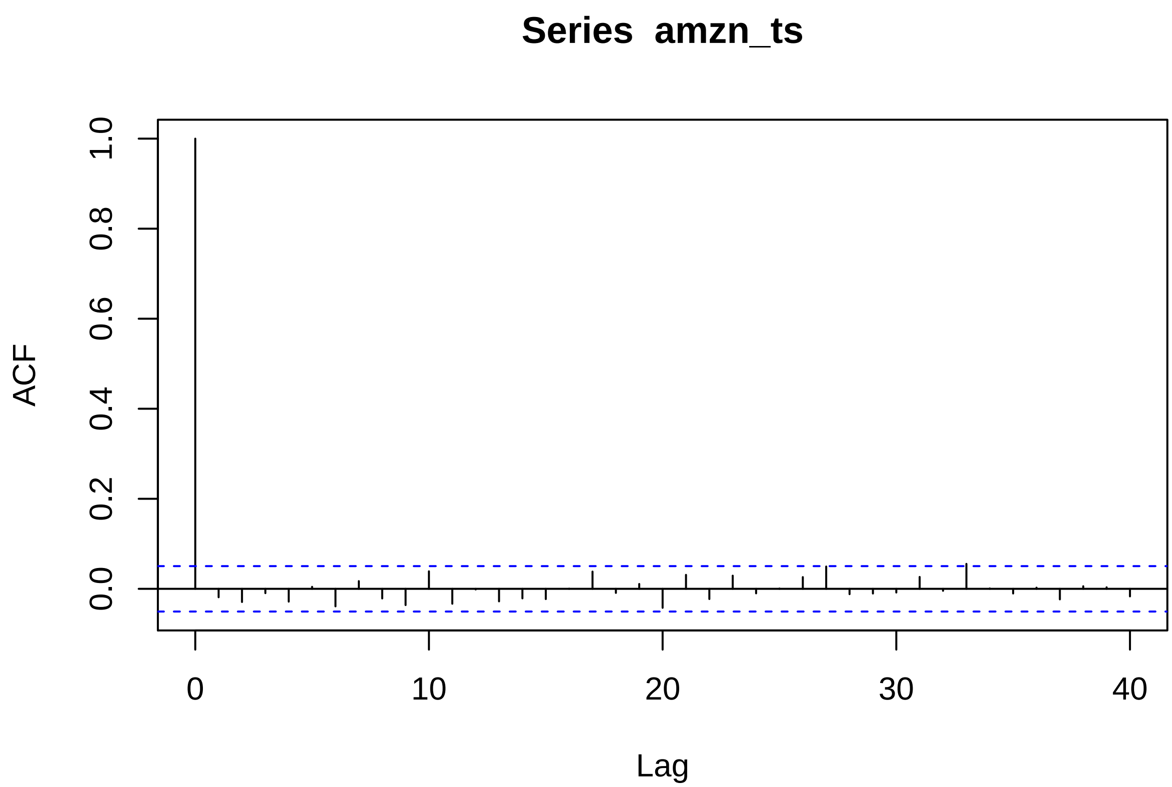
```
stocks_daily_ret <- read.table("d-amzn3dx0914.txt", header = TRUE)

amzn_ts <- log(1 + xts(stocks_daily_ret$amzn, order.by = as.Date(strsplit(toString(stocks_daily_ret$dat
                                                                     split = ", ")[[1]], "%Y%m%d")))*100
mean(amzn_ts)
```

```
## [1] 0.1192311
```

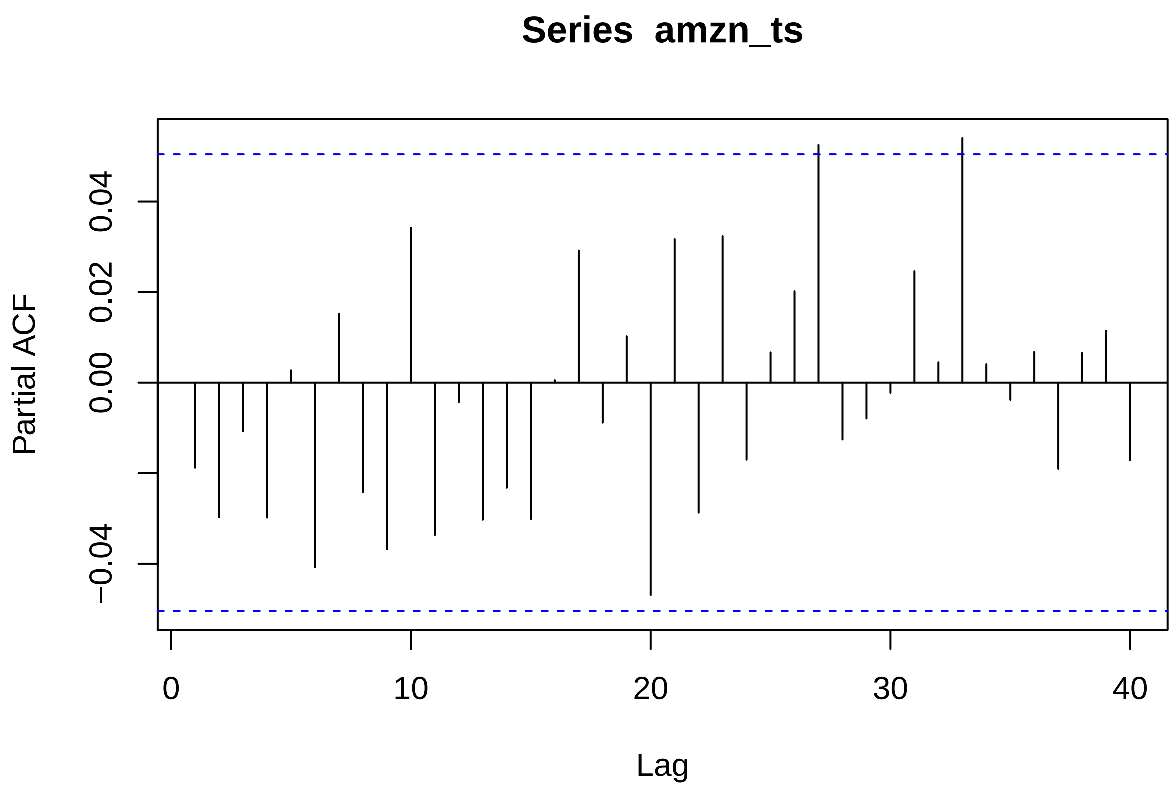
The expected value of the percentage log returns is not equal to zero. The value of this stock increased by a factor of 6 over this time interval, so this is understandable.

##2.

```
acf(amzn_ts, lag.max = 40)
```




```
pacf(amzn_ts, lag.max = 40)
```



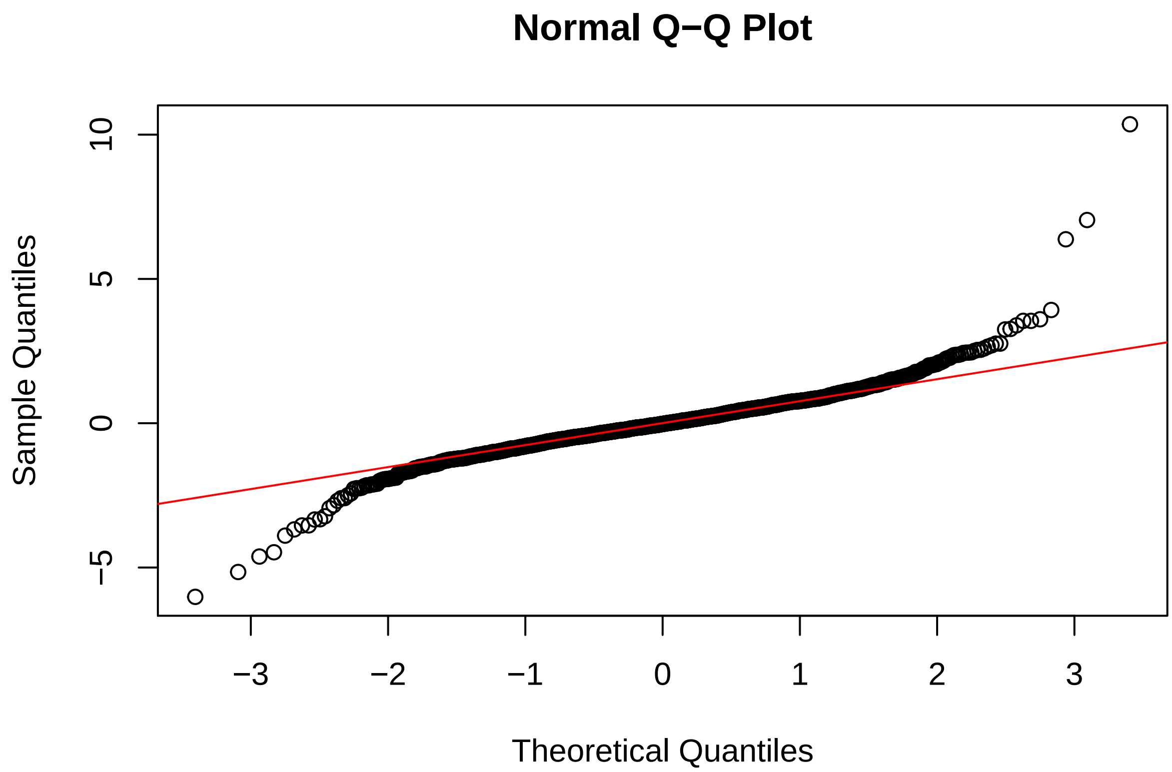
We do not see significant serial correlations in r_t.

##3.

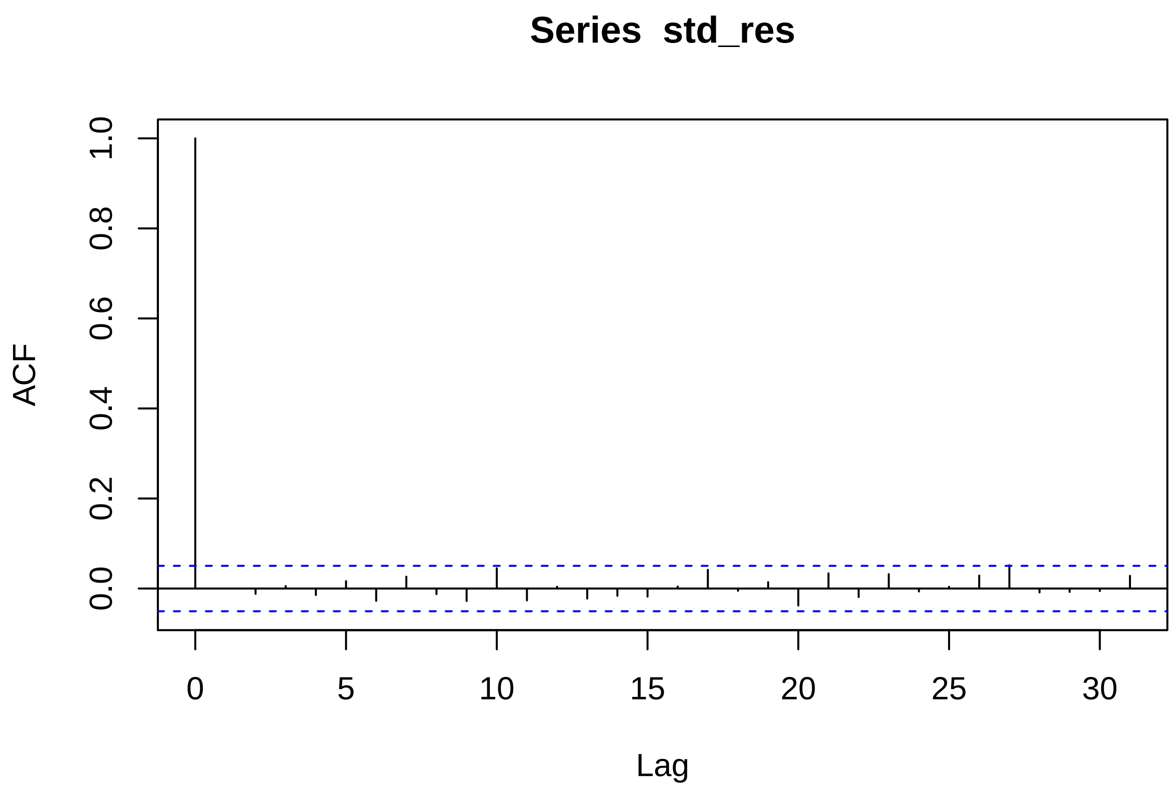
Now we will fit an ARMA-GARCH model to the returns. Using the AIC once more, we will determine proper values for the p q parameters of the component ARMA process. Then we use the same method to determine the proper parameters for the GARCH component.

```
#ARMA fit  (fix this part)(!)
min_aic <- Inf # keep a running min
best_order <- c(0, 0, 0)
  for (p in 0:6) for (q in 0:6){
    if ( p==0 && q == 0){
      next
    }
    arimaFit <- tryCatch(arima(amzn_ts, order = c(p, 0, q)),
                         error = function(err) FALSE,  # turn failures to converge into logical output
                         warning = function(err) FALSE)

    if (!is.logical(arimaFit)){  # if convergence took place
      current_aic <- AIC(arimaFit)
      if(current_aic < min_aic){
        min_aic <- current_aic
        best_order <- c(p, 0, q)
        final.arima <- arima(amzn_ts, order=best_order)
      }
    }
```

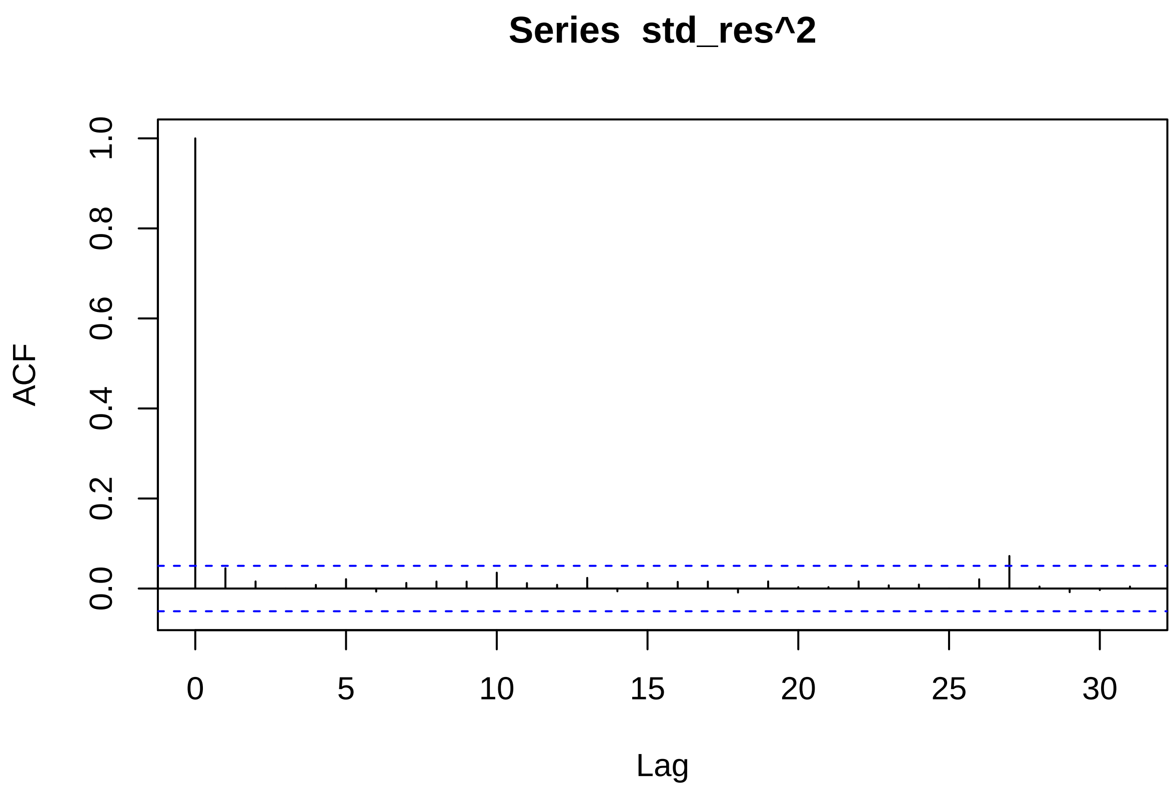
```r
    else {
      next
    }
  }

#GARCH fit

min_aic <- Inf
best_garch_order <- c(0, 0)

for (p in 0:5) for (q in 0:5){
    if ( p==0 && q == 0){
      next
    }
    garch_spec <- ugarchspec(variance.model=list(garchOrder=c(p,q)),
                      mean.model=list(armaOrder=c(best_order[1], best_order[3]),
                                      include.mean = T),
                                      distribution.model = "norm")

    garch_model <- tryCatch(ugarchfit(garch_spec, amzn_ts, solver = "hybrid"),
                            error = function(err) FALSE,
                            warning = function(err) FALSE)


    if (!is.logical(garch_model)){  # if convergence took place
      current_aic <- infocriteria(garch_model)[1]
      if(current_aic < min_aic){
        min_aic <- current_aic
        best_garch_order <- c(p, q)
      }
    }
    else {
      next
    }
  }

garch_spec <- ugarchspec(variance.model=list(garchOrder=best_garch_order),
                      mean.model=list(armaOrder=c(best_order[1], best_order[3]),
                                      include.mean = T),
                                      distribution.model = "norm")

garch_model <- ugarchfit(garch_spec, amzn_ts, solver = "hybrid")

res <- garch_model@fit[["residuals"]]

std_res <- (res - mean(res)) / sd(res)
qqnorm(std_res)  ##QQ plot of standardize residuals
qqline(std_res, col = "red")
```




```
acf(std_res)  #looks good
```



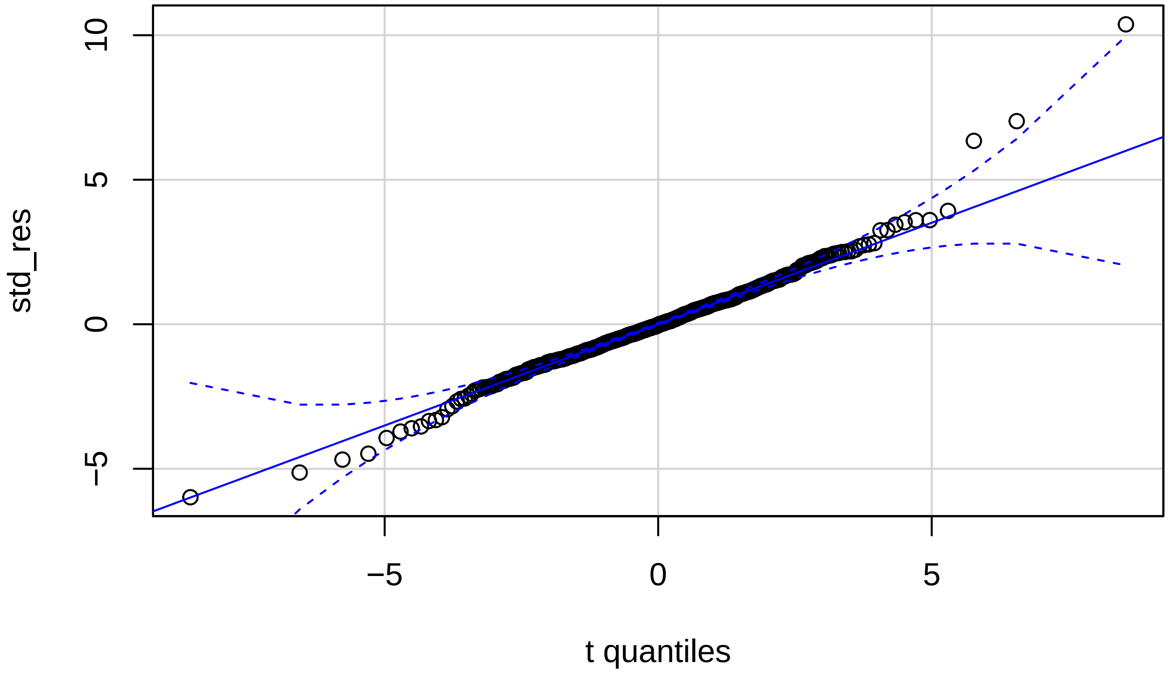

```r
acf(std_res^2)  # looks good too:  only once exceedance
```



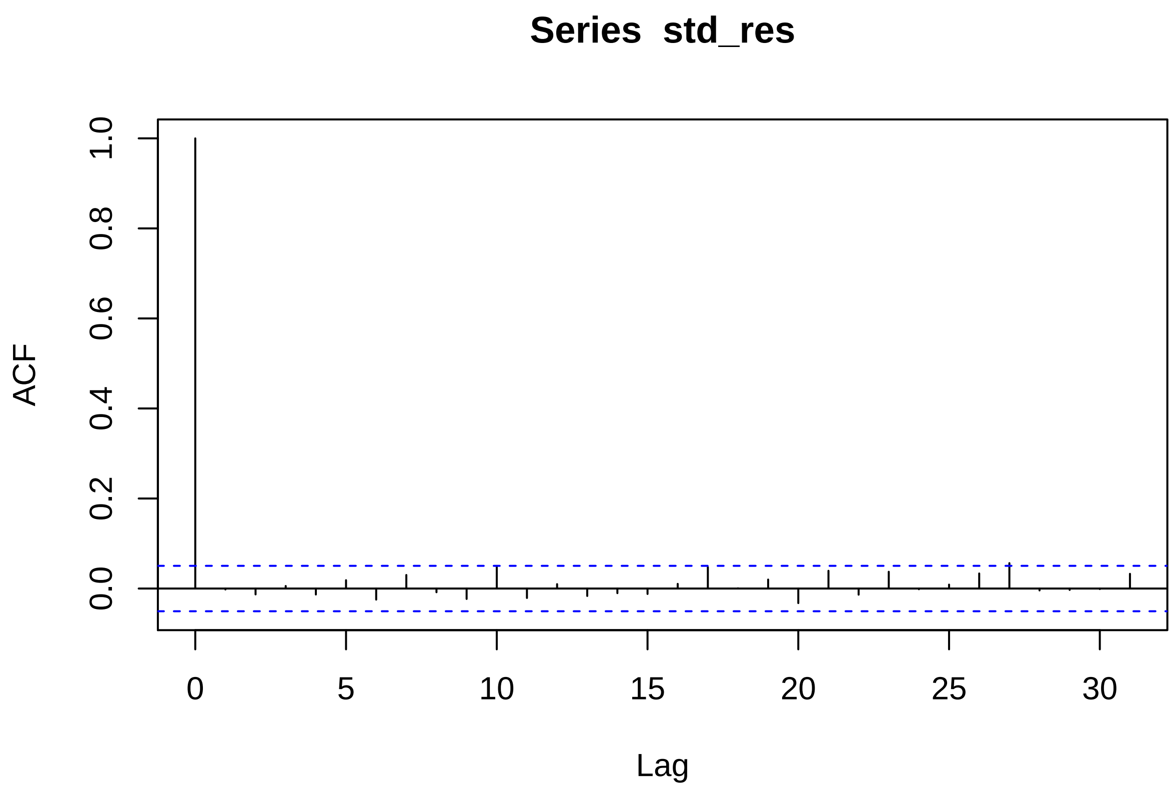
##4.

From our QQplot, We see fat-tailedness of the data. We will use a t-distribution to remedy this. Both the ACF and PACF indicate that the standardized residuals of our ARMA(1, 1)-GARCH(2, 3) have no leftover structure.

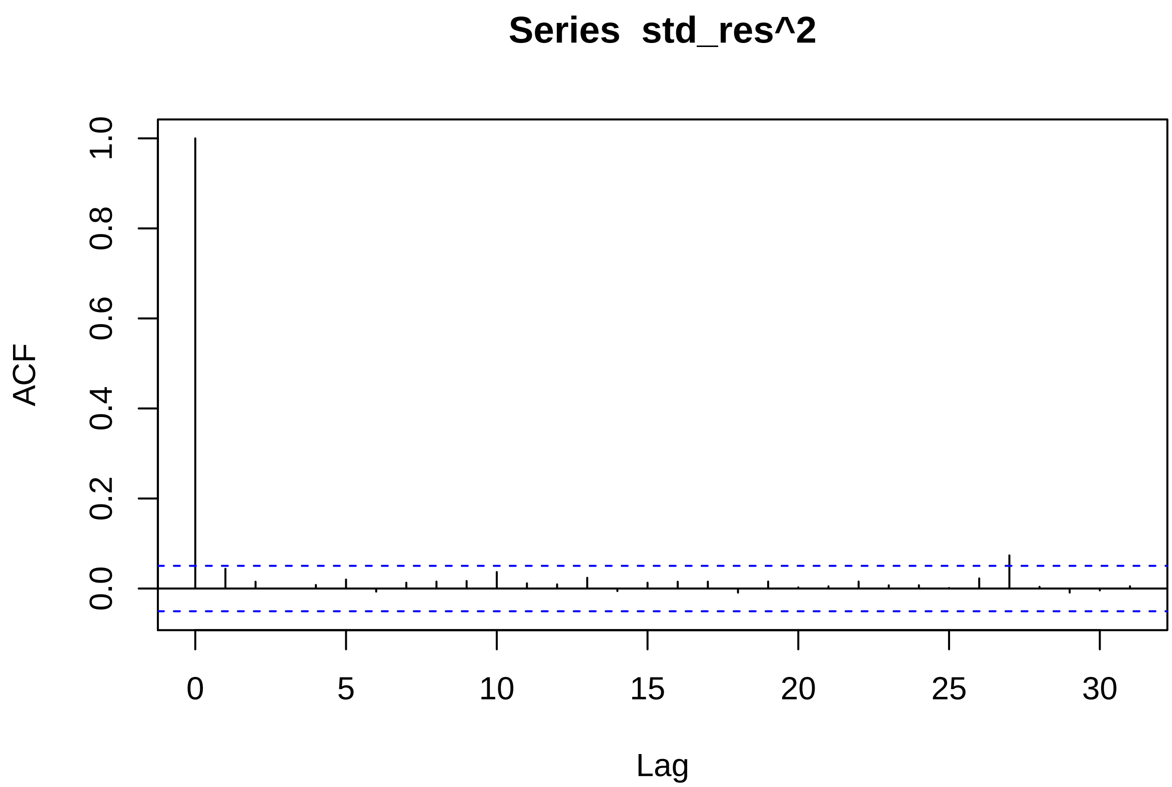
```
garch_spec <- ugarchspec(variance.model=list(garchOrder=best_garch_order),
                      mean.model=list(armaOrder=c(best_order[1],best_order[3]),
                                              include.mean = T),
                                    distribution.model = "std")

garch_model <- ugarchfit(garch_spec, amzn_ts, solver = "hybrid")
res <- garch_model@fit[["residuals"]]

std_res <- (res - mean(res)) / sd(res)
qqPlot(std_res, dist = "t", df = garch_model@fit[["coef"]][["shape"]], id = FALSE, lwd = 1)  ##QQ plot
```




```
acf(std_res)  #looks good
```




```
acf(std_res^2)  # looks good too:  only once exceedance
```



Our QQ plot looks much more in-line: most of the datapoints lie within the 95% confidence envelopes.

5.

Our fitted model is of the form

$$Y_t = \phi_1 Y_{t-1} + \theta_1 \epsilon_{t-1} + \epsilon_t$$

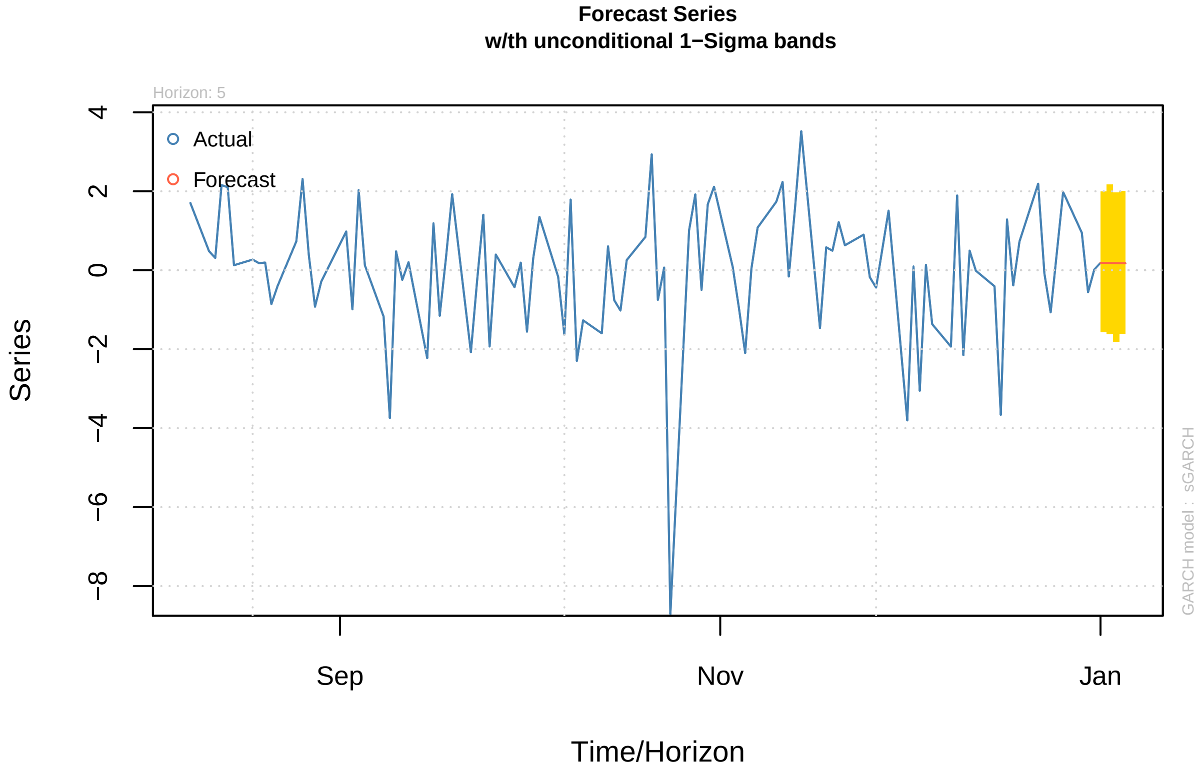
$$\epsilon_t \sim \sigma_t \cdot t_\nu \ ; \ \nu \approx 4.41$$

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1} + \sum_{i=1}^{3} \beta_i \sigma_{t-i}^2$$

6

Now we will use our garch model to forecast future values of Amazon return series.

```
garch_forecast <- ugarchforecast(garch_model, n.ahead = 5)
plot(garch_forecast, which = 1)
```



While the exact values returns are not predicable, we can predict how the distribution of returns evolves over time, assuming that an ARMA-GARCH model well-approximates the underlying processes that gave rise to our data in the first place.