Introduction

Overview

This report is concerned with performing exploratory analysis and exploring feasible time-series models for a dataset describing monthly prices for cocoa beans for the time period between the years 1980 to 2016. The dataset was obtained from Kaggle and was originally sourced from U.S. Government Works (Kaggle, 2024). The data provided contains price data for cocoa beans in an unknown currency.

Objectives

- Perform descriptive analysis of the dataset.
- Shortlist potential models based on suitability for this application and analyse the models for selection using ACF-PACF, EACF, BIC etc.
- Fit the selected models and find parameter estimates
- Utilize goodness-of-fit metrics such as AIC, BIC, MSE etc. to select the optimal model.

Method and Results

Visualising the Series

In the first step, the dataset is loaded into the environment and the 17th column of the csv file is obtained. This is because the complete dataset contains price data for multiple commodities and the series of interest for this report is the price of cocoa beans. The summary shows a mean price of 1900, a max of 3522.1 and a min of 860.7. The 1st quartile for anomalies is 1410.7 and the 3rd quartile is 2304.8. The median anomaly value is 1720.6. In total, there are 434 price instances in the dataset corresponding to the monthly prices between 1980 and 2016.

```
cocoa = read.csv("commodity-prices-2016.csv")[,c(1,17)]
summary(cocoa)
```

```
##      Date            Cocoa.beans
##  Length:434         Min.   : 860.7
##  Class :character   1st Qu.:1410.7
##  Mode  :character   Median :1720.6
##                     Mean   :1900.0
##                     3rd Qu.:2304.8
##                     Max.   :3522.1
```

The price column is converted to a time-series object with a frequency of 12 since it is monthly data. The series is then plotted to observe any trends or seasonality in Figure 1. The initial plot demonstrates trend that changes over time, decreasing overall till 2000 and increasing again. There does not seem to be obvious seasonality when overlaying the season labels on the plot. The series seems to have both AR and MA components since there are also fluctuations in the series throughout. Changing variance is observed as well.

```
cocoa.ts = ts(cocoa[,c(2)], frequency = 12, start = 1980, end = 2016)

plot(cocoa.ts, type='o', ylab='Monthly price of coca beans', xlab='Year', main = "Time series plot of monthly
cocoa beans price")
points(y=cocoa.ts,x=time(cocoa.ts), pch=as.vector(season(cocoa.ts)), col=c(1:12))
```

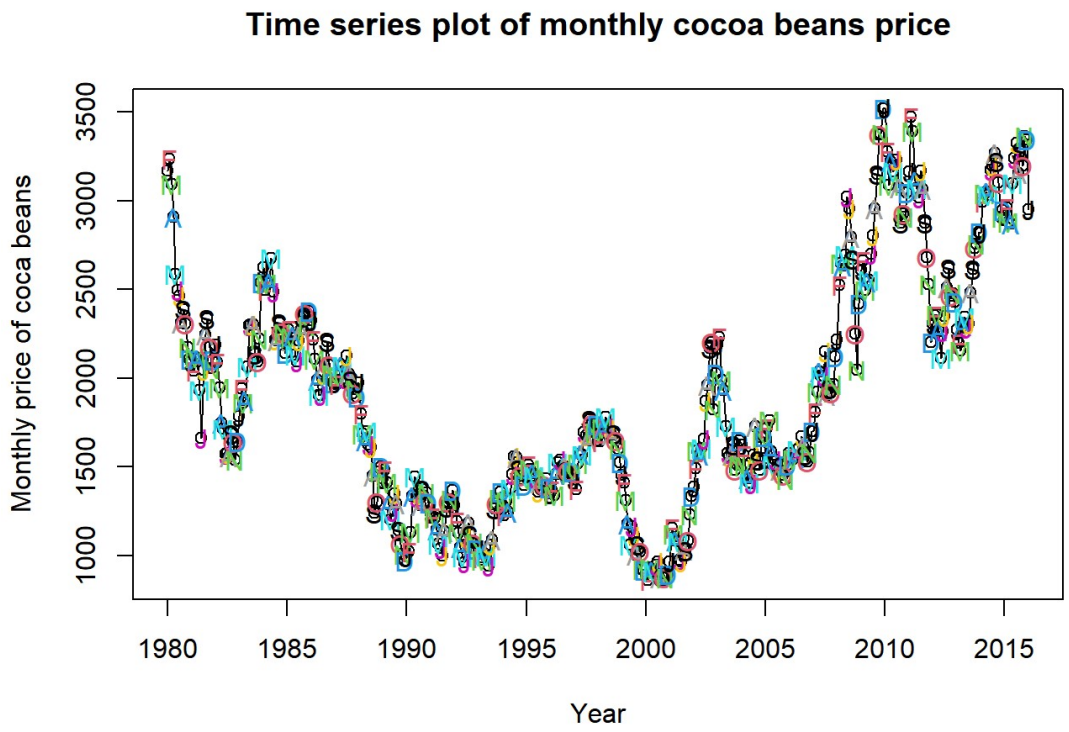

Figure 1: Plot of the monthly cocoa beans price data

Verifying Stationarity and Normality

The ACF and PACF plots for the series are shown in Figure 2. The ACF plot returns a slowly decaying plot with most peaks beyond the confidence intervals and the PACF plot produces three significant peaks beyond the confidence intervals at the first, second and third lags. Both of these suggest non-stationarity in the series due to the autocorrelations observed at multiple lags.

```
par(mfrow=c(2,1), mar=c(3,3,3,1))
acf(cocoa.ts, main="ACF plot", lag.max = 40)
pacf(cocoa.ts, main="PACF plot", lag.max = 40)
```

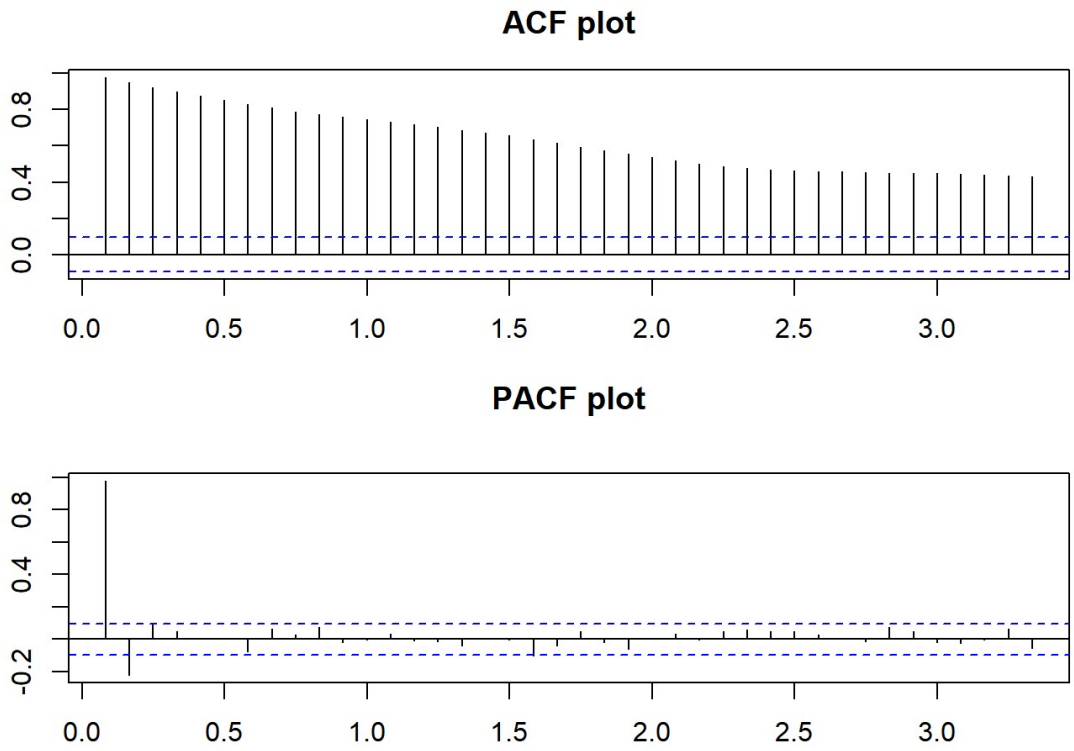

Figure 2: ACF and PACF plots of the monthly cocoa beans price data

```
par(mfrow=c(1,1))
```

Checking the stationarity using an Augmented Dickey-Fuller (ADF) test reveals a p-value of 0.4719, affirming the null hypothesis at the 5% significance level. This confirms that the series is not stationary. A Phillips-Perron test supports this with a p-value of 0.4538 which affirms the null hypothesis of non-stationarity once again.

The QQ plot of the series in Figure 3 shows both tails deviating from the reference line suggesting visually that the data points are not normally distributed. A Shapiro-Wilk test confirms this producing a p-value of 5.329e-11 which is less than the significance level of 0.05, affirming the alternate hypothesis that the series is not normally distributed.

```
adf.test(cocoa.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cocoa.ts
## Dickey-Fuller = -2.2501, Lag order = 7, p-value = 0.4719
## alternative hypothesis: stationary
```

```
pp.test(cocoa.ts)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  cocoa.ts
## Dickey-Fuller Z(alpha) = -11.742, Truncation lag parameter = 5, p-value
## = 0.4538
## alternative hypothesis: stationary
```

```
qqnorm(cocoa.ts, ylab="Monthly cocoa beans price", xlab="Normal Scores")
qqline(cocoa.ts)
```

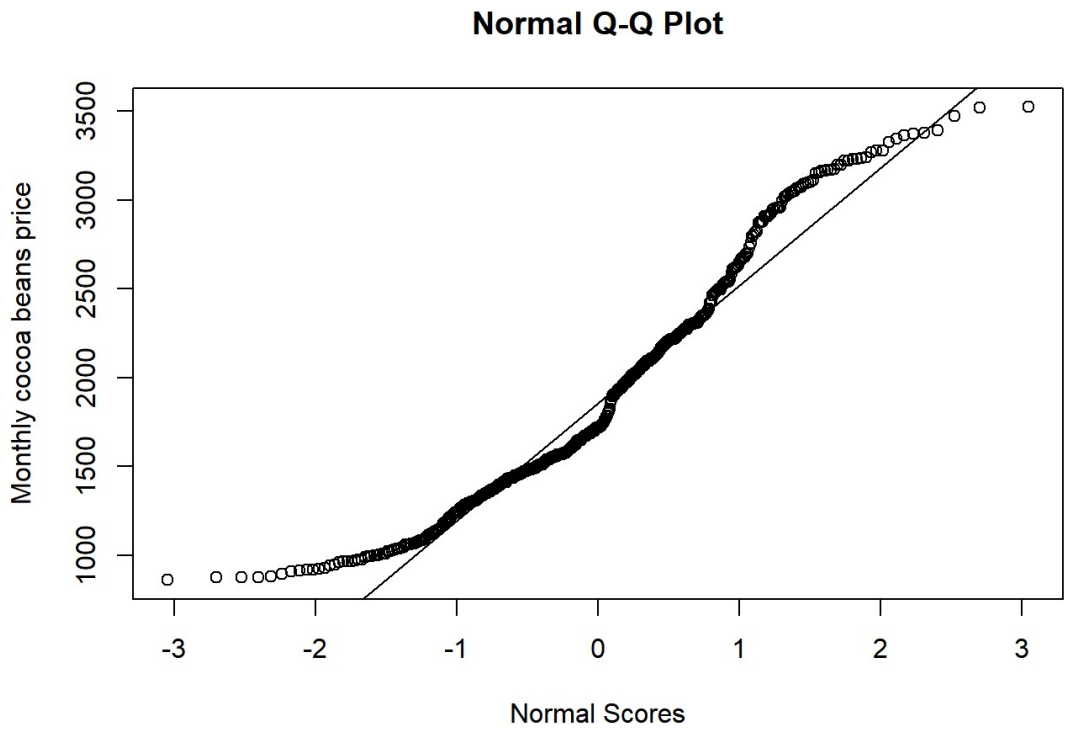

Figure 3: Q-Q plot of the time series anomalies data

```
shapiro.test(cocoa.ts)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cocoa.ts
## W = 0.94946, p-value = 5.329e-11
```

Linear Trend Models

A linear trend model is fitted to the tie series below:

```
# Fit linear model
linear.cocoa.ts = lm(cocoa.ts~time(cocoa.ts))
summary(linear.cocoa.ts)
```

```
##
## Call:
## lm(formula = cocoa.ts ~ time(cocoa.ts))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1089.3  -482.7  -150.7   545.9  1754.2
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -44388.241   5711.253  -7.772 5.72e-14 ***
## time(cocoa.ts)     23.166      2.858   8.104 5.53e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 619.6 on 431 degrees of freedom
## Multiple R-squared:  0.1322, Adjusted R-squared:  0.1302
## F-statistic: 65.68 on 1 and 431 DF,  p-value: 5.526e-15
```

```
plot(cocoa.ts, xaxt = "n", type='o', xlab = 'Days', ylab='Price of Cocoa Beans', main = "Fitted linear line t
o cocoa beans prices")
abline(linear.cocoa.ts, lty=2,col="red")
```

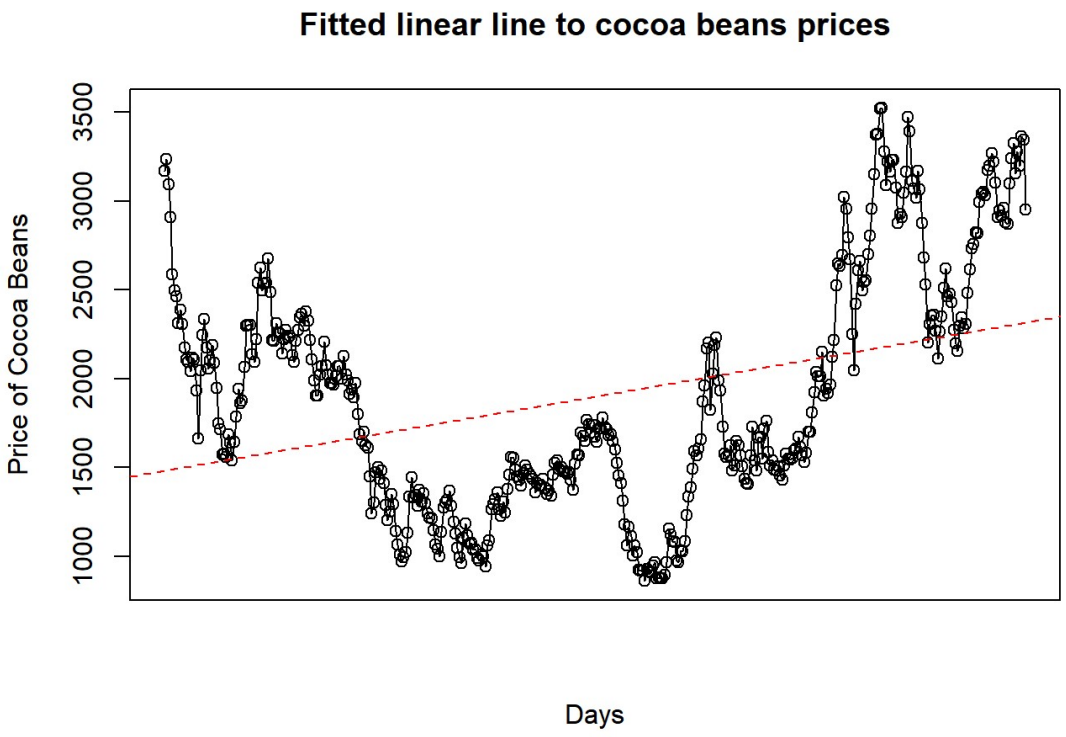

Figure 4: Plot of the time series cocoa beans prices with fitted linear line

Based on the above graph, the linear trend line is seen increasing from ~USD 1,900 to ~USD 2,400. It can be concluded that there is a misalignment between the linear trend and the model. However, it seems that the linear model correctly captures the upward trend in data.

However, the model exhibits an R squared value of 0.1302, indicating that only 13.02% of the data is explained by the model and the p value is 5.526e-15(>0.05), suggesting that the model is not statistically significant.

Residual Analysis

```
res.linear.cocoa.ts = rstudent(linear.cocoa.ts)

plot_residuals = function(res, time.returns) {
  par(mfrow=c(3,2))
  par(mar=c(4,4,3,1.5))
  plot(y = res, xaxt = "n", x = as.vector(time.returns), xlab = 'Days', ylab='Standardized Residuals',type
='l', main = "Standardised residuals from model.")
  hist(res,xlab='Standardized Residuals', main = "Histogram of standardised residuals.")
  qqnorm(y=res, main = "QQ plot of standardised residuals.")
  qqline(y=res, col = 2, lwd = 1, lty = 2)
  acf(res, main = "ACF of standardized residuals.")
  pacf(res, main = "PACF of standardized residuals.")
  par(mfrow=c(1,1))
}

plot_residuals(res.linear.cocoa.ts, time(cocoa.ts))
```

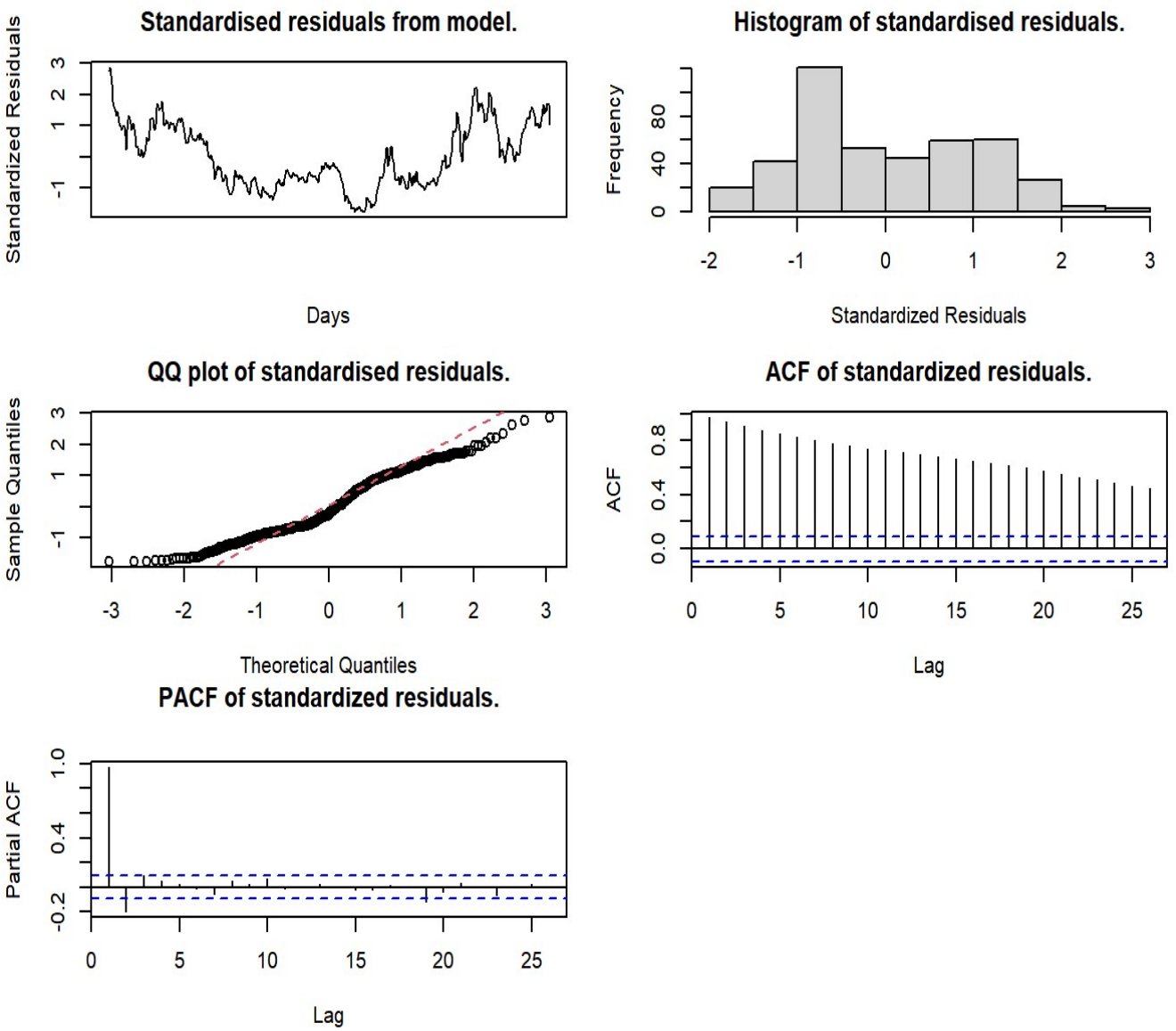

Figure 5: Plot of the residuals along with their histogram, QQ plot, ACF plot, and PACF plot

```
shapiro.test(res.linear.cocoa.ts)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.linear.cocoa.ts
## W = 0.95998, p-value = 1.814e-09
```

Residual Analysis: The residuals do not resemble white noise since a significant component of the trend and fluctuations remain in the residuals.

Histogram: The histogram here appears slightly skewed, suggesting that the residuals are not normally distributed and are significant.

QQ plot: Departures from the diagonal line indicate deviations from normality, particularly at the extremes, suggesting the stochastic component of the model does not follow a normal distribution.

ACF: The ACF here shows some potentially significant autocorrelations at various lags, suggesting the residuals may not be independent.

PACF: The PACF here also shows some significant spikes, further indicating potential issues with residual autocorrelation.

The Shapiro-Wilk test yields a p-value less than 0.05, confirming that the residuals are not normally distributed, as the null hypothesis of normality is rejected.

Quadratic Trend Model

```
# Fit quadratic model
t = time(cocoa.ts)
t2 = t^2
quad.cocoa.ts = lm(cocoa.ts ~ t + t2)
summary(quad.cocoa.ts)
```

```
##
## Call:
## lm(formula = cocoa.ts ~ t + t2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -717.70 -286.96  -46.16  266.90 1172.34
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.039e+07  7.298e+05   27.94   <2e-16 ***
## t           -2.043e+04  7.305e+02  -27.97   <2e-16 ***
## t2           5.119e+00  1.828e-01   28.00   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 369.2 on 430 degrees of freedom
## Multiple R-squared:  0.6926, Adjusted R-squared:  0.6912
## F-statistic: 484.5 on 2 and 430 DF,  p-value: < 2.2e-16
```

```
plot(ts(fitted(quad.cocoa.ts)), ylim = c(min(c(fitted(quad.cocoa.ts),
    as.vector(cocoa.ts))), max(c(fitted(quad.cocoa.ts),as.vector(cocoa.ts)))),
    xlab = 'Days', ylab='Price of cocoa beans', main = "Fitted quadratic curve to price of cocoa beans", type
="l",lty=2,col="red")
# axis(1, at=seq(1, 14, by=3), labels=seq(1, 179, by=36))
lines(as.vector(cocoa.ts),type="o")
```

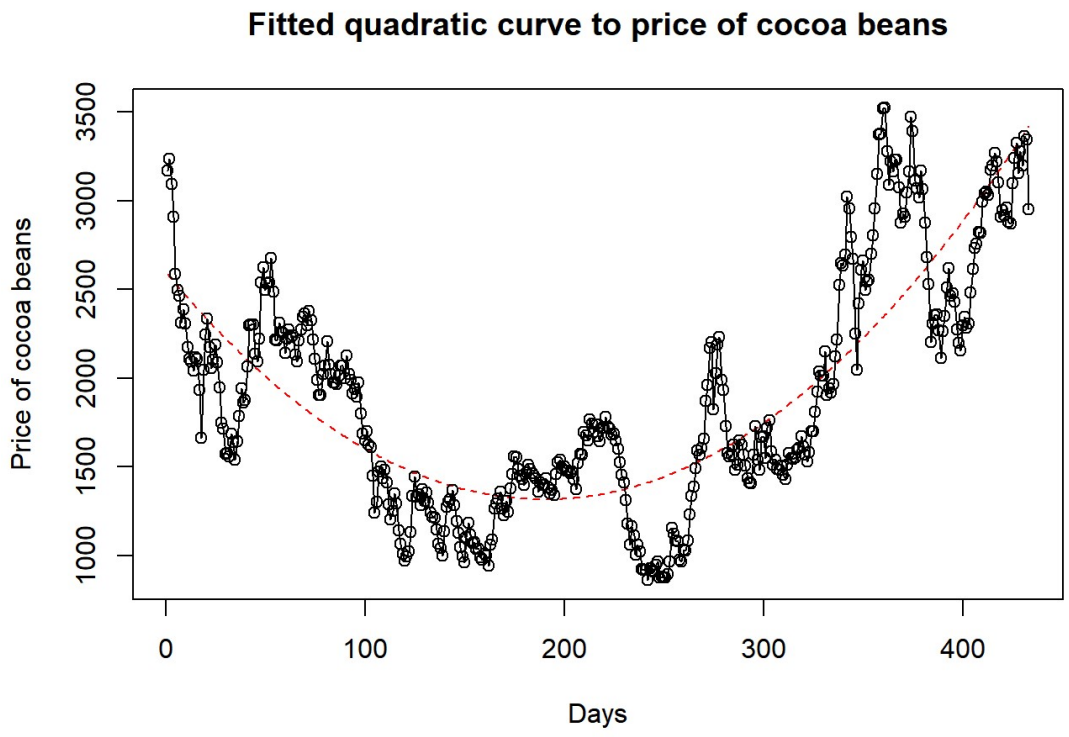

Figure 6: Plot of the time series of cocoa beans prices with fitted quadratic curve

Based on the above plot, the quadratic curve captures the overall trend and curvature of the data, allowing for a better fit. However, there is still significant variation and scatter in the individual data points around the curve.

The model exhibits an R squared value of 0.6912, indicating that 69.12% of the data is explained by the model but the p value is 2.2e-16(<0.05), suggesting that the model is statistically significant.

```
res.quad.cocoa.ts = rstudent(quad.cocoa.ts)

plot_residuals(res.quad.cocoa.ts, time(cocoa.ts))
```

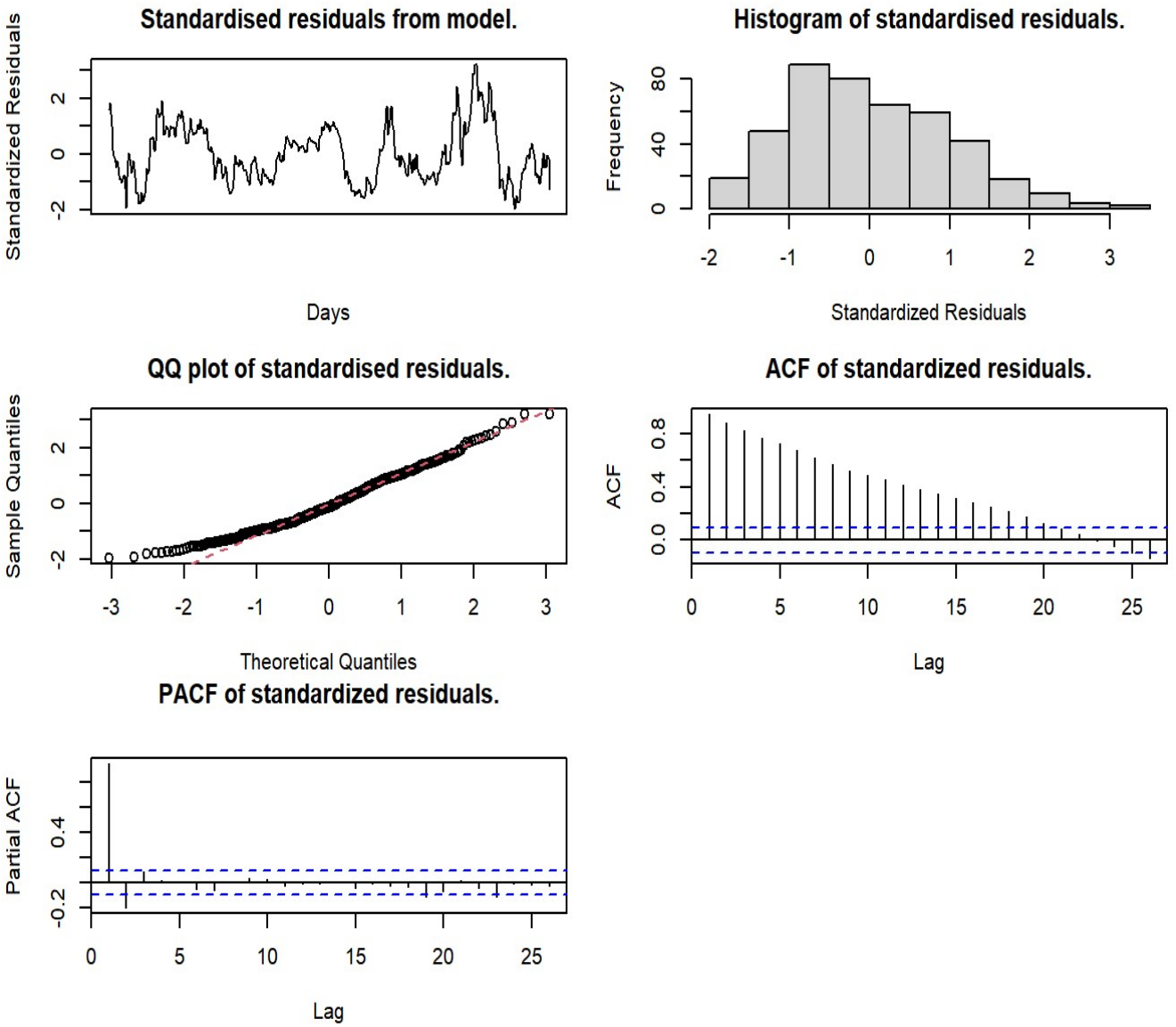

Figure 7: Plot of the residuals along with their histogram, QQ plot, ACF plot, and PACF plot

```
shapiro.test(res.quad.cocoa.ts)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.quad.cocoa.ts
## W = 0.97655, p-value = 1.887e-06
```

Residual Analysis: The residuals appear to fluctuate around zero with trend components remaining.

Histogram: The distribution appears approximately bell-shaped and symmetric around zero with a slight skew on the right. However, it is not severe which is consistent with the assumption of normality.

QQ plot: The points closely follow the diagonal line, suggesting the stochastic component of the model follow a normal distribution.

ACF Plot: The ACF plot exhibits a slow decaying pattern, suggesting non-stationarity and violates the assumption that the residuals are independent.

PACF Plot: Similar to the ACF, the PACF shows no significant spikes beyond the confidence limits, further supporting the assumption of independence of residuals.

In summary, while the quadratic model does a better job of explaining the variance compared to the linear model, the overall R squared value remains relatively low. Therefore, these models are not considered effective and will not be used for further analysis or predictions.

Transforming the Series

A function called summarise_series() is defined that creates Q-Q, ACF, PACF and McLeod Li plots for any transformed series that is provided.

A Box-Cox transformation may help with achieving normality in the data. A Box-Cox transformation is applied to the series, producing a lambda value of 0. A lambda value of 0 indicates a log transformation, suggesting that a log transformation should be carried out on the series.

A log transformation is then applied. The time series plot for the log transformed series is shown in Figure 4, its Q-Q plot vs the original Q-Q plot is shown in Figure 5, and ACF and PACF plots for the log transformed series are shown in Figure 6. The plot of the log transformed series does not look much more stabilized compared to the original plot. The QQ plots compared side by side show that the Log transformed series deviates less from the reference line compared to the non-transformed series, indicating that it is more normally distributed. However overall, the transformed series is still not normally distributed. The plot of McLeod-Li tests performed at multiple lags shows all the p-values lying below the significance threshold, implying that there is evidence of changing variance across all lags. A Shapiro-Wilk test performed on the transformed series still shows a very small p-value (2.635e-06), confirming non-normal distribution. Performing an ADF test reveals that the series is still not stationary due to a high p-value (0.502).

```
summarise_series = function(series, transform) {
  plot(series, type='o',ylab = "Cocoa beans price", main=paste('Time series plot of ', transform, ' of cocoa
bean prices'))
  par(mfrow=c(1,2), mar=c(3,4,3,1))
  qqnorm(series, main=paste('Q-Q plot of ', transform, ' series'), ylab=paste(transform, ' cocoa bean price
s'), xlab="Normal Scores")
  qqline(series)
  qqnorm(cocoa.ts, main="Original series Q-Q plot", ylab="Cocoa bean prices", xlab="Normal Scores")
  qqline(cocoa.ts)
  par(mfrow=c(1,1))

  par(mfrow=c(1,2), mar=c(3,3,5,1))
  acf(series, main=paste('ACF of the ', transform, ' series.'))
  pacf(series, main=paste('PACF of the ', transform, ' series.'))
  par(mfrow=c(1,1))

  McLeod.Li.test(y=series,main=paste("McLeod-Li Test Statistics for", transform, " of cocoa bean prices"))

  print(shapiro.test(series))
  print(adf.test(series))
}

BC = BoxCox.ar(cocoa.ts, lambda = seq(-2, 2 ) )
```

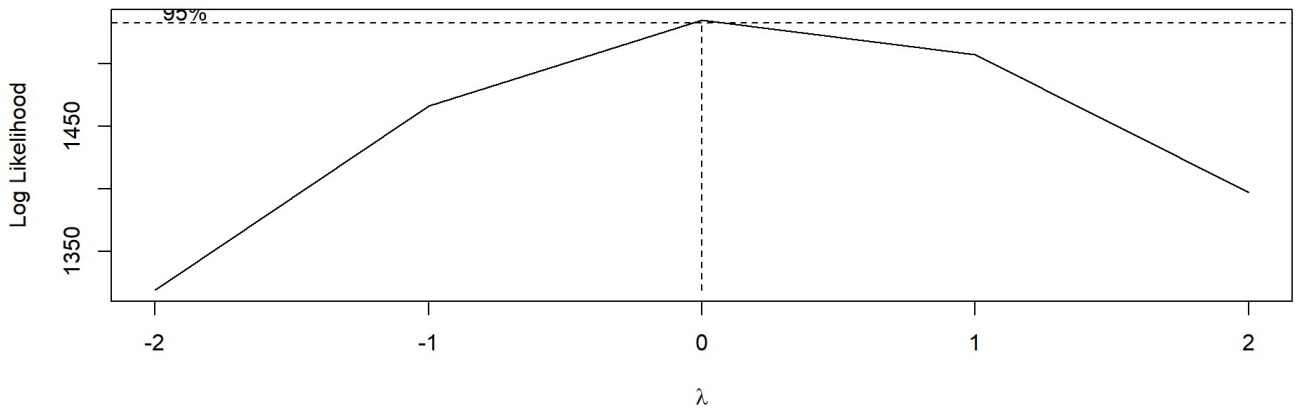

Figure 8: Log Likelihood plot for Box-Cox transformation

```
BC$ci
```

```
## [1] 0 0
```

```
lambda = BC$lambda[which(max(BC$loglike) == BC$loglike)]
lambda
```

```
## [1] 0
```

```
log.cocoa = log(cocoa.ts)
summarise_series(log.cocoa, "Log")
```

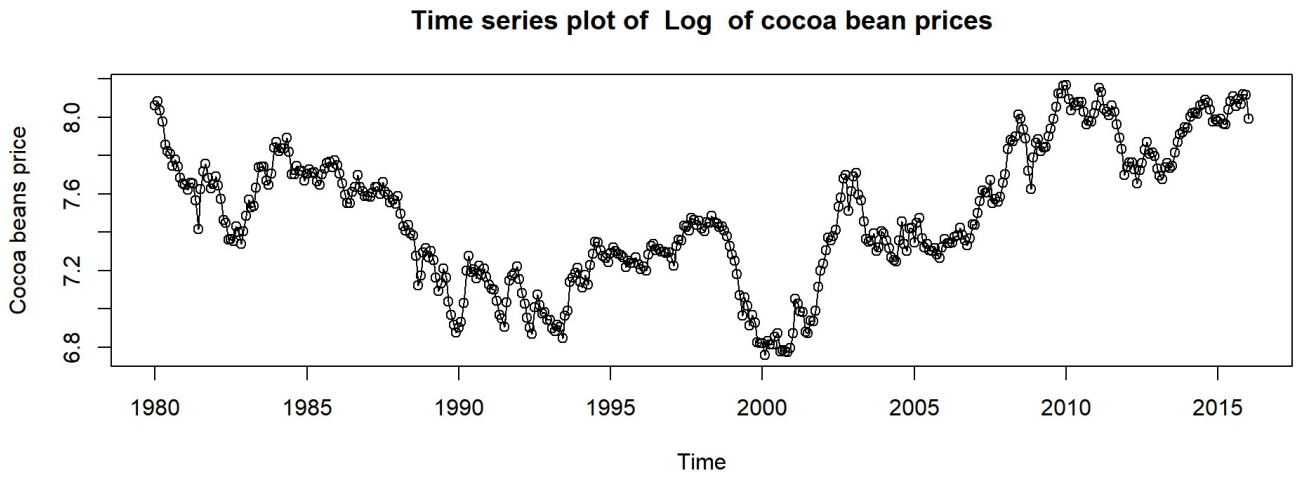

Figure 9: Time series plot of log transformed cocoa beans price series

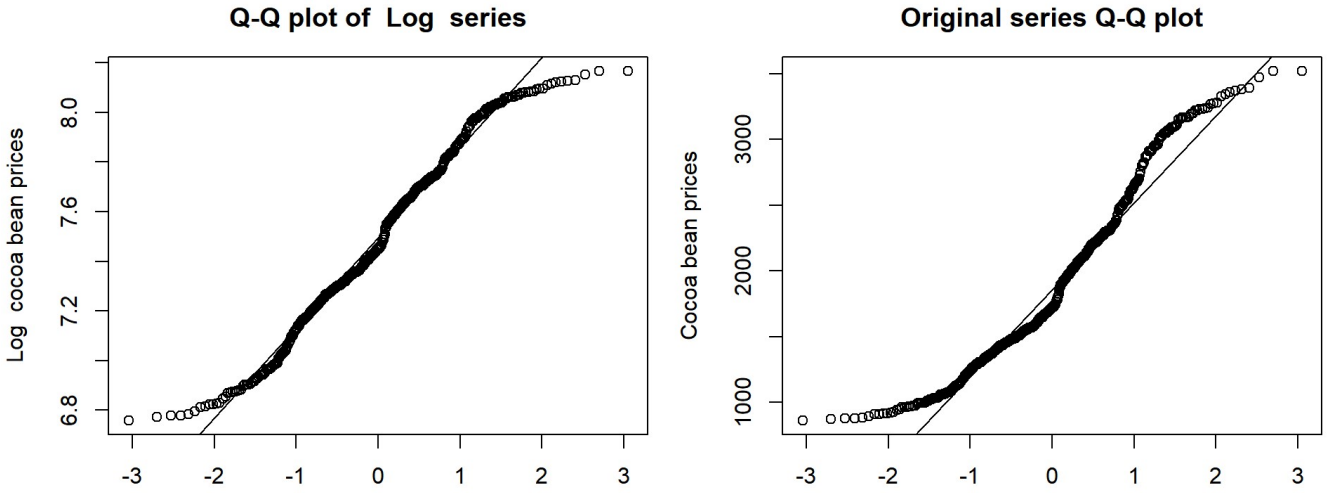

Figure 10: Q-Q plots for log transformed cocoa beans price series vs original series

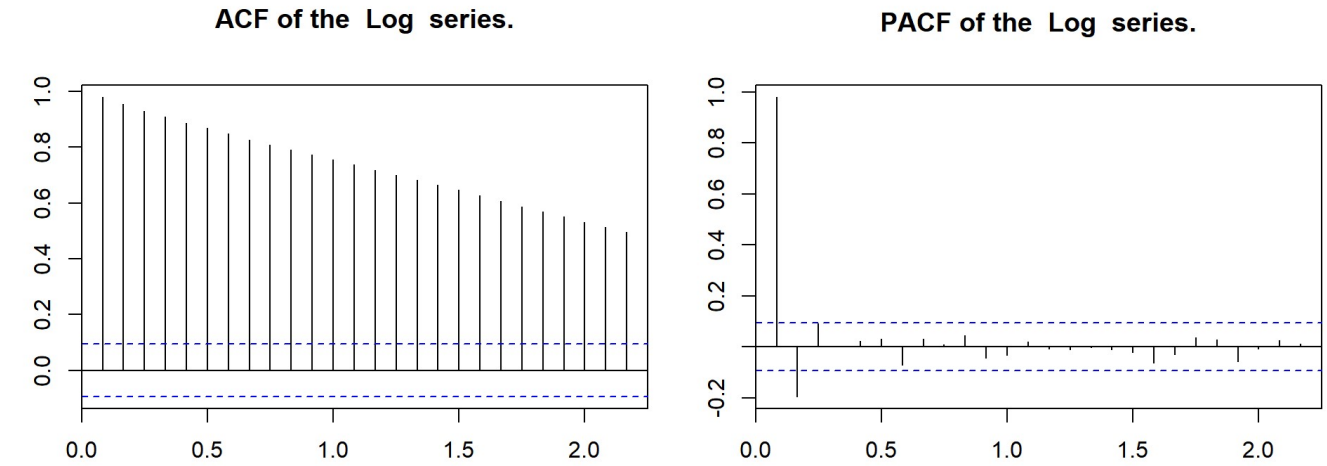

Figure 11: ACF and PACF plots for log transformed cocoa beans price series

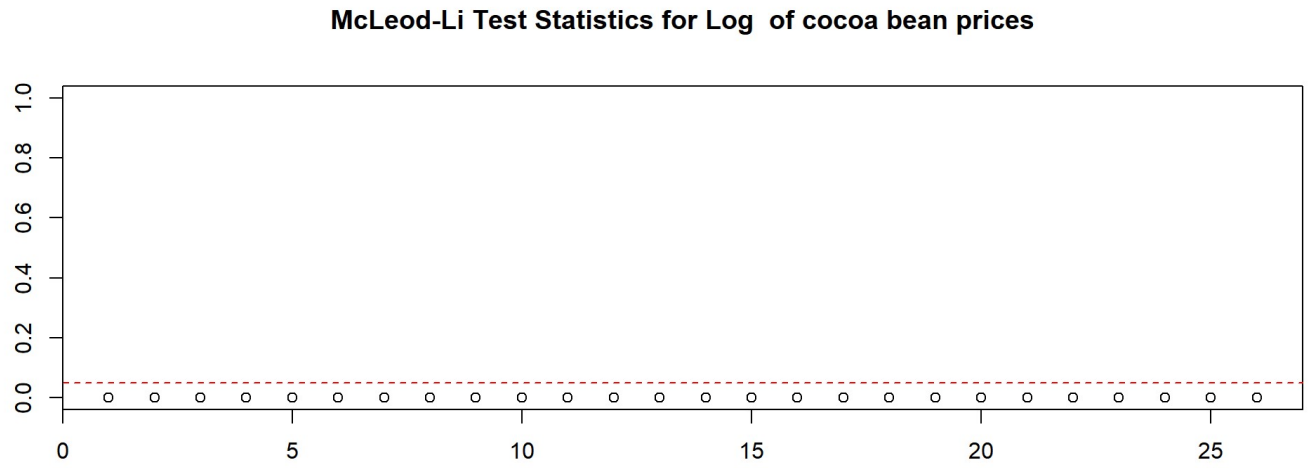

Figure 12: McLeod-Li plot for log-transformed series at multiple lags

```
##
##  Shapiro-Wilk normality test
##
## data:  series
## W = 0.97723, p-value = 2.635e-06
##
##
##  Augmented Dickey-Fuller Test
##
## data:  series
## Dickey-Fuller = -2.1788, Lag order = 7, p-value = 0.502
## alternative hypothesis: stationary
```

Differencing the Series

The existence of changing variance across multiple lags, MA and AR components suggests that this series would benefit from ARIMA-GARCH modelling. Obtaining the first lag of the log transformed series would provide the returns series for the cocoa bean prices series. Therefore, the first difference of the log series is obtained to attempt at making the series stationary. The time series plot for the returns series is shown in Figure 7, its Q-Q plot vs the original Q-Q plot is shown in Figure 8, and ACF and PACF plots for the returns series are shown in Figure 9. The returns of the cocoa beans price series is plotted and is visually observed to have a constant mean and not much changing variance. This is confirmed with the results of the ADF test producing a p-value of 0.01, indicating that the null hypothesis is rejected and the returns series is stationary. The Q-Q plot of the returns series deviates less from the reference line compared to the original series and can be said to be more normal. The results of the Shapiro-Wilk test confirm normality, with a high p-value of 0.1975. The ACF plot for the returns series shows a significant peak at the first lag. The PACF plot is very similar to the ACF plot, with a significant peak at the first and second lags. Therefore, q can be assumed to be 1 and p can be taken to be 2 based on the ACF and PACF plots respectively. These p and q values suggest the modeling of ARIMA(1,1,1) and ARIMA(2,1,1)

Since the returns series provided satisfactory results in terms of normality and stationarity, no more differencing is performed in order to avoid over-differencing.

```
r.cocoa = diff(log.cocoa, differences = 1)*100
summarise_series(r.cocoa, "returns")
```

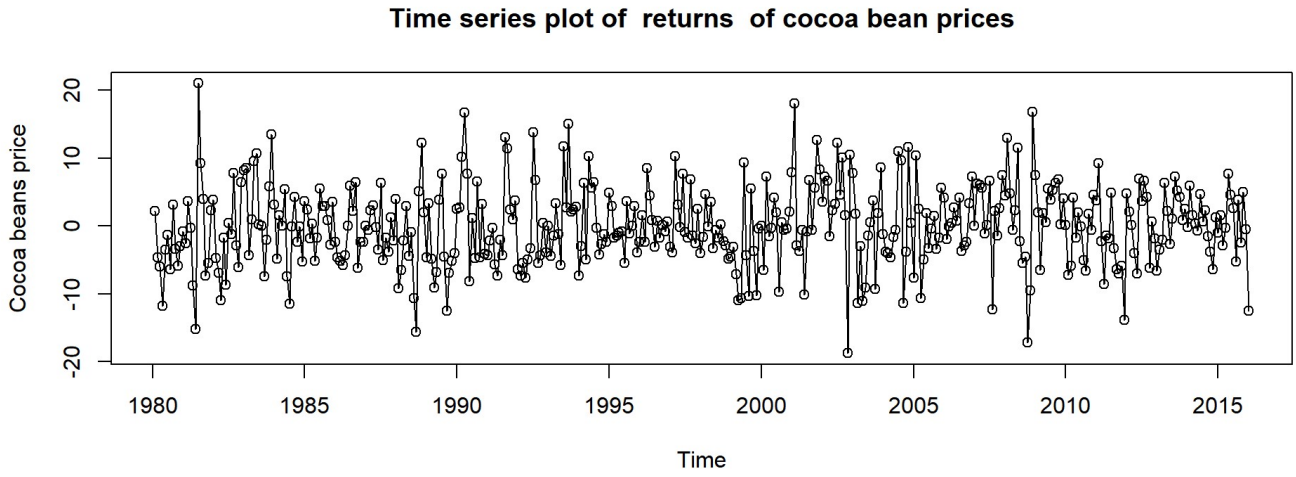

Figure 13: Time series plot of returns series

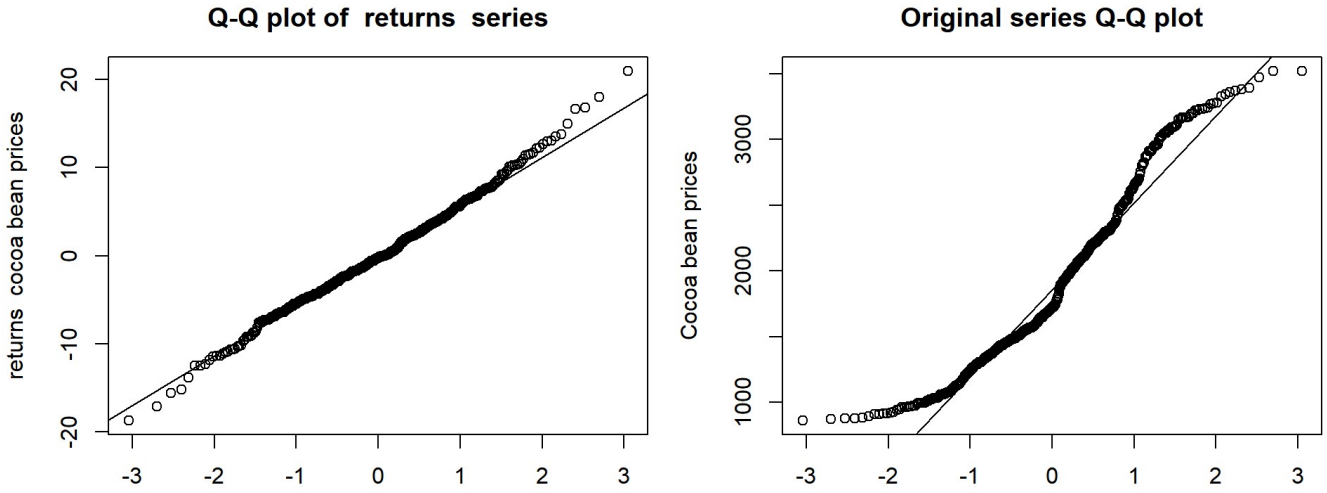

Figure 14: Q-Q plots for returns series vs original series

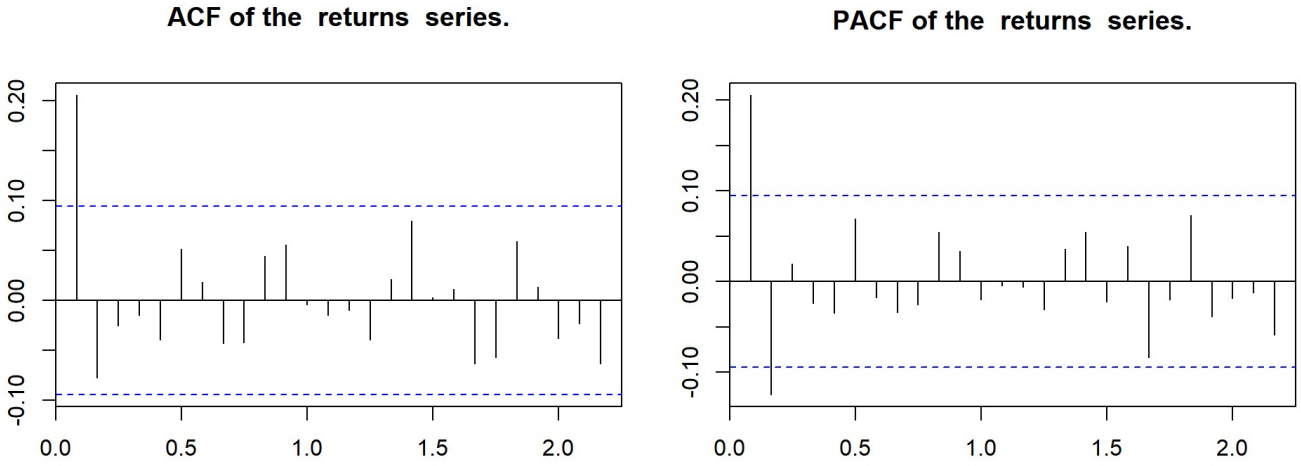

Figure 15: ACF and PACF plots for returns series

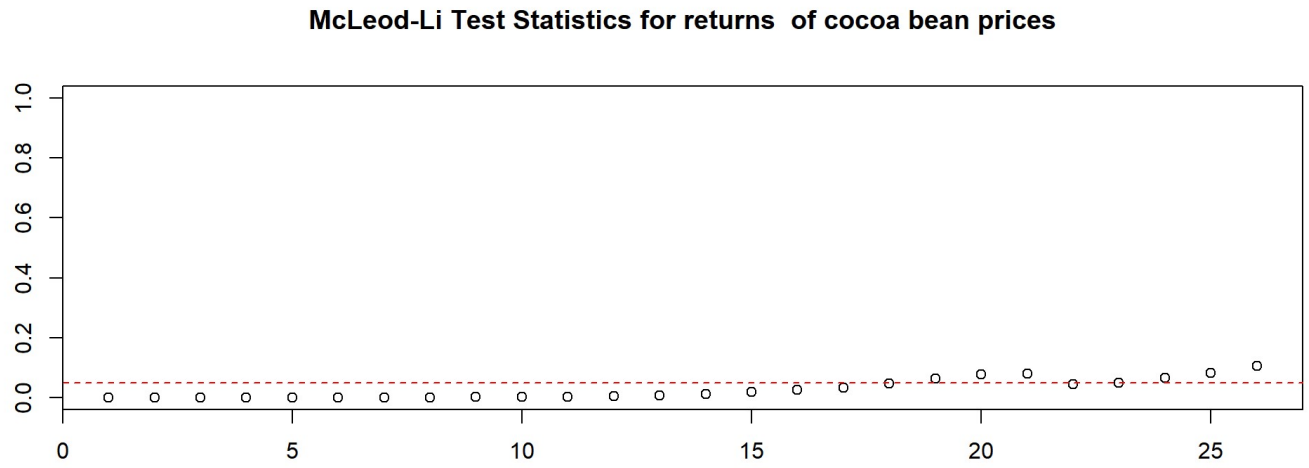

Figure 16: McLeod-Li plot for returns series at multiple lags

```
##
##  Shapiro-Wilk normality test
##
## data:  series
## W = 0.99516, p-value = 0.1975
##
##
##  Augmented Dickey-Fuller Test
##
## data:  series
## Dickey-Fuller = -7.5347, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
# p = 2, q = 1
```

ARIMA Modelling

EACF Table

The EACF table for the returns series is shown below. The top-left ‘o’ is obtained from the EACF table and this is observed to occur at AR = 0 and MA = 1. This results in the possible models being ARIMA(0,1,1), ARIMA(0,1,2), and ARIMA(1,1,2) based on the best model and its neighbors.

```
eacf(r.cocoa)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o o o o o o o o o o  o  o  o
## 1 x x o o o o o o o o o  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x x x o o o o o o o o  o  o  o
## 4 x x o x o o o o o o o  o  o  o
## 5 x x x o x o o o o o o  o  o  o
## 6 x x x o x o o o o o o  o  o  o
## 7 x o x o o x o o o o o  o  o  o
```

```
# ARMA(0,1,1) ARMA(0,1,2) ARMA(1,1,2)
```

The BIC table for the returns series is shown in Figure 10. Plotting the BIC table for the returns series shows that the shortlisted models are ARIMA(0,1,3), ARIMA(1,1,4).

```
res = armasubsets(y=r.cocoa, nar=4, nma=4, y.name='p', ar.method='ols')
```

```
## Reordering variables and trying again:
```

```
plot(res)
```

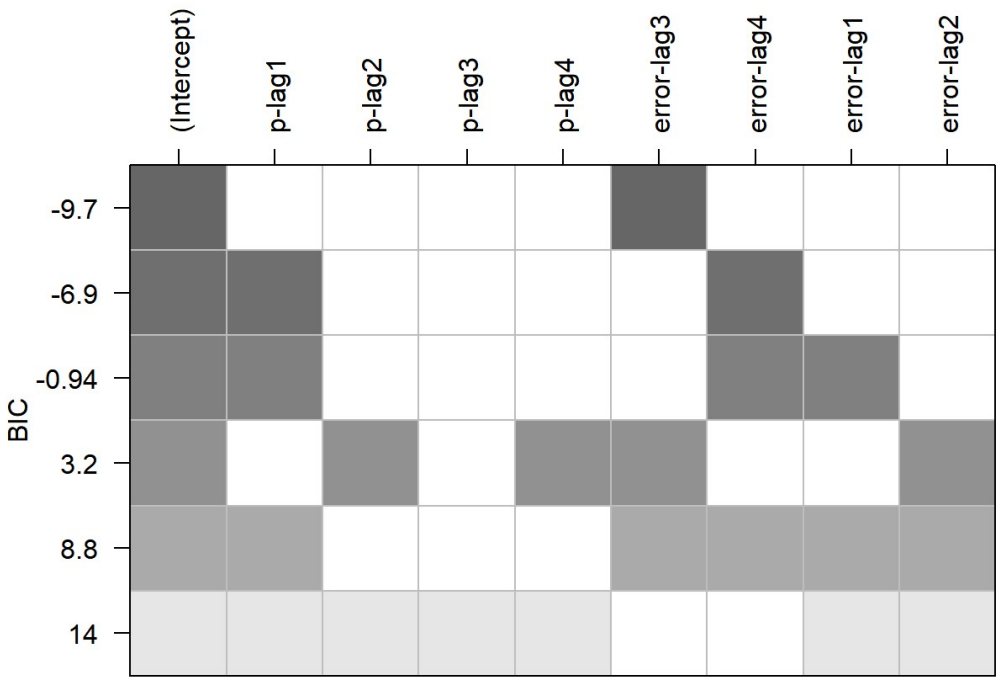

Figure 17: BIC table for returns series

```
#ARIMA(0,1,3)
#ARIMA(1,1,4)
```

ARIMA Parameter Estimation

Based on the above model selection, the final models for modeling are listed below:

- ARIMA(1,1,1)
- ARIMA(2,1,1)
- ARIMA(0,1,1)
- ARIMA(0,1,2)
- ARIMA(1,1,2)
- ARIMA(0,1,3)
- ARIMA(1,1,4)

A function named arima_fit() is defined to fit ARIMA models based on the orders required, print the coefficients and return the models obtained using both CSS and ML estimators. The models listed above are created using arima_fit(). All the models except ARIMA(1,1,4) have similar significance levels for the coefficients between CSS and ML estimators. Since the significance levels differed between CSS and ML for ARIMA(1,1,4), CSS-ML was tested as well. This produced significance levels similar to the ML estimator for ARIMA(1,1,4).

Performing residual analysis on the models using the residual.analysis() function provided on Canvas (Canvas, 2024) produces white noise residuals for all the models tested. Shapiro-Wilk tests performed on all the model residuals produced significant p-values each time, suggesting that there is non-normality in the residuals. However, the histograms of residuals are very symmetric for all models and Q-Q plots for all models are very close to the reference line. None of the ACF plots of standardised residuals contain significant peaks. All the plots from the Ljung-Box tests have points above the significance levels, indicating no autocorrelations are observed in the residuals at any of the lags. All of this suggests that there are no seasonal components remaining in the residuals, which implies that there are no seasonal components in the series.

```
arima_fit = function(data, order) {

  arima_ml = Arima(data, order = order, method = 'ML')
  print(paste("Model Fitting Using ML for ARIMA", paste(order, collapse = ",")))
  print(coeftest(arima_ml))

  arima_css = Arima(data, order = order, method = 'CSS')
  print(paste("Model Fitting Using CSS for ARIMA", paste(order, collapse = ",")))
  print(coeftest(arima_css))

  return(list(ML = arima_ml, CSS = arima_css))
}

models = list()

model_111 = arima_fit(log.cocoa, c(1, 1, 1))
```

```
## [1] "Model Fitting Using ML for ARIMA 1,1,1"
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -0.17853    0.14736 -1.2115 0.225701
## ma1  0.41924    0.13312  3.1494 0.001636 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "Model Fitting Using CSS for ARIMA 1,1,1"
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -0.18386    0.14665 -1.2537  0.20994
## ma1  0.42500    0.13188  3.2226  0.00127 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
models[[1]] = model_111_ml = model_111$ML

model_211 = arima_fit(log.cocoa, c(2, 1, 1))
```

```
## [1] "Model Fitting Using ML for ARIMA 2,1,1"
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1  0.068590   0.339802  0.2019   0.8400
## ar2 -0.092028   0.090608 -1.0157   0.3098
## ma1  0.167934   0.340241  0.4936   0.6216
##
## [1] "Model Fitting Using CSS for ARIMA 2,1,1"
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.11628    0.45861  0.2535   0.7998
## ar2 -0.10271    0.10958 -0.9373   0.3486
## ma1  0.12082    0.46317  0.2608   0.7942
```

```
models[[2]] = model_211_ml = model_211$ML

model_011 = arima_fit(log.cocoa, c(0, 1, 1))
```

```
## [1] "Model Fitting Using ML for ARIMA 0,1,1"
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1 0.255608   0.049588  5.1546 2.542e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "Model Fitting Using CSS for ARIMA 0,1,1"
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1  0.25600    0.04959  5.1625 2.437e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
models[[3]] = model_011_ml = model_011$ML

model_012 = arima_fit(log.cocoa, c(0, 1, 2))
```

```
## [1] "Model Fitting Using ML for ARIMA 0,1,2"
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1  0.233201   0.048785  4.7802 1.752e-06 ***
## ma2 -0.064914   0.047390 -1.3698    0.1708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "Model Fitting Using CSS for ARIMA 0,1,2"
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1  0.233516   0.048823  4.7829 1.728e-06 ***
## ma2 -0.064811   0.047434 -1.3663    0.1718
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
models[[4]] = model_012_ml = model_012$ML

model_112 = arima_fit(log.cocoa, c(1, 1, 2))
```

```
## [1] "Model Fitting Using ML for ARIMA 1,1,2"
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.40573    0.49242  0.8240   0.4100
## ma1 -0.17067    0.48681 -0.3506   0.7259
## ma2 -0.17008    0.11810 -1.4400   0.1499
##
## [1] "Model Fitting Using CSS for ARIMA 1,1,2"
##
## z test of coefficients:
##
##       Estimate Std. Error z value Pr(>|z|)
## ar1  0.2299777  0.4883447  0.4709   0.6377
## ma1  0.0047782  0.4867830  0.0098   0.9922
## ma2 -0.1272123  0.1321688 -0.9625   0.3358
```

```
models[[5]] = model_112_ml = model_112$ML

model_013 = arima_fit(log.cocoa, c(0, 1, 3))
```

```
## [1] "Model Fitting Using ML for ARIMA 0,1,3"
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1  0.237447   0.048521  4.8937 9.896e-07 ***
## ma2 -0.078162   0.049301 -1.5854    0.1129
## ma3 -0.039131   0.048154 -0.8126    0.4164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "Model Fitting Using CSS for ARIMA 0,1,3"
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1  0.237808   0.048563  4.8969 9.736e-07 ***
## ma2 -0.078082   0.049325 -1.5830    0.1134
## ma3 -0.039264   0.048273 -0.8134    0.4160
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
models[[6]] = model_013_ml = model_013$ML

model_114 = arima_fit(log.cocoa, c(1, 1, 4))
```

```
## [1] "Model Fitting Using ML for ARIMA 1,1,4"
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.706998   0.315941 -2.2378 0.025237 *
## ma1  0.947558   0.317528  2.9842 0.002843 **
## ma2  0.089445   0.102791  0.8702 0.384212
## ma3 -0.087309   0.073721 -1.1843 0.236286
## ma4  0.010476   0.054541  0.1921 0.847679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## [1] "Model Fitting Using CSS for ARIMA 1,1,4"
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.675391   0.251134 -2.6894 0.007159 **
## ma1  0.918073   0.254180  3.6119 0.000304 ***
## ma2  0.083019   0.091334  0.9090 0.363369
## ma3 -0.083984   0.070749 -1.1871 0.235198
## ma4  0.015407   0.050566  0.3047 0.760598
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model_114_ml = model_114$ML
model_114_css = model_114$CSS
model_114_cssml = Arima(log.cocoa, order = c(1, 1, 4), method = 'CSS-ML')
print(paste("Model Fitting Using CSS-ML for ARIMA", paste(c(1, 1, 4), collapse = ",")))
```

```
## [1] "Model Fitting Using CSS-ML for ARIMA 1,1,4"
```

```
print(coeftest(model_114_cssml))
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.706276   0.317632 -2.2236 0.026178 *
## ma1  0.946766   0.319185  2.9662 0.003015 **
## ma2  0.089194   0.103073  0.8653 0.386848
## ma3 -0.087221   0.073735 -1.1829 0.236851
## ma4  0.010559   0.054550  0.1936 0.846508
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
models[[7]] = model_114_ml

for (m in models) {
  print(arimaorder(m))
  residual.analysis(model = m, std=TRUE, start=2, class="ARIMA")
}
```

```
## p d q
## 1 1 1
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99151, p-value = 0.01396
```

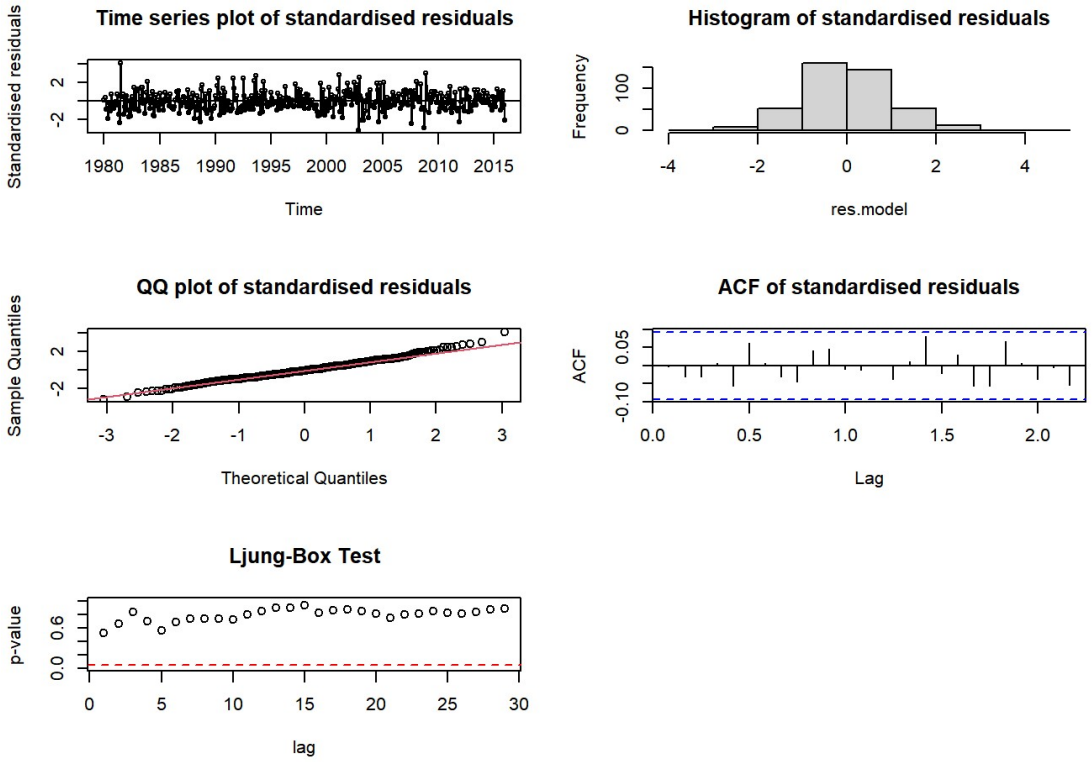

Figure 18: Results of residual analysis for ARIMA(1,1,1)

```
## p d q
## 2 1 1
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99178, p-value = 0.01705
```

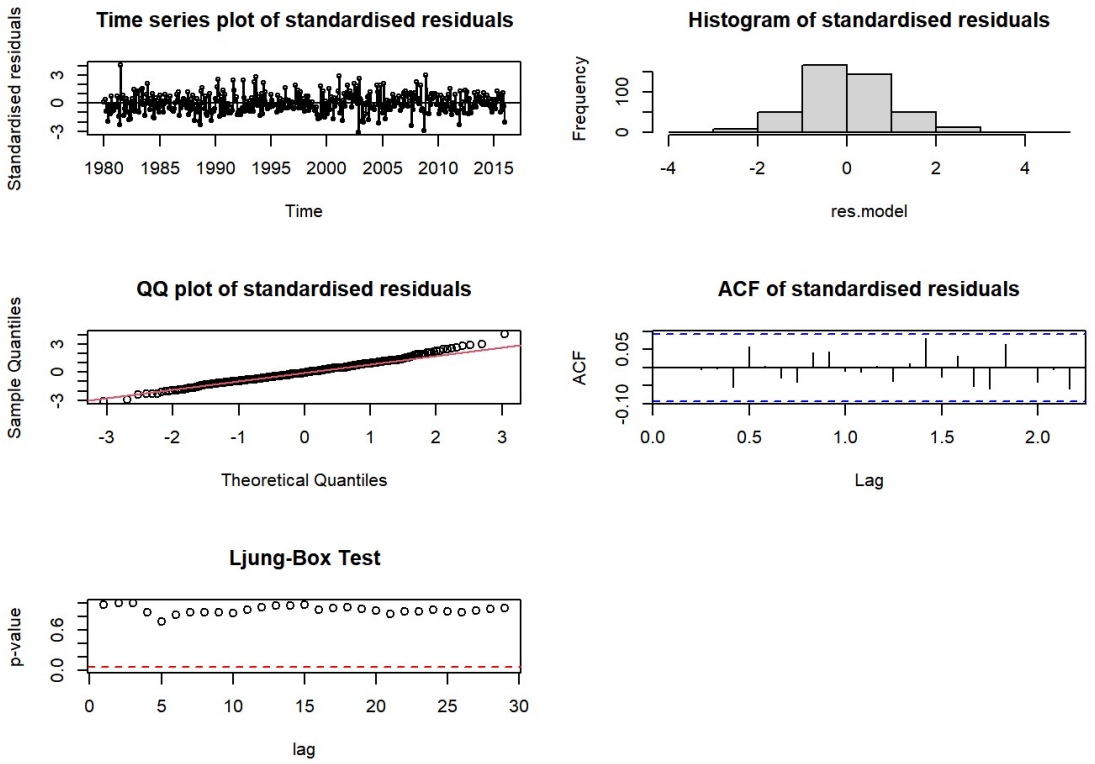

Figure 19: Results of residual analysis for ARIMA(2,1,1)

```
## p d q
## 0 1 1
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99091, p-value = 0.009134
```

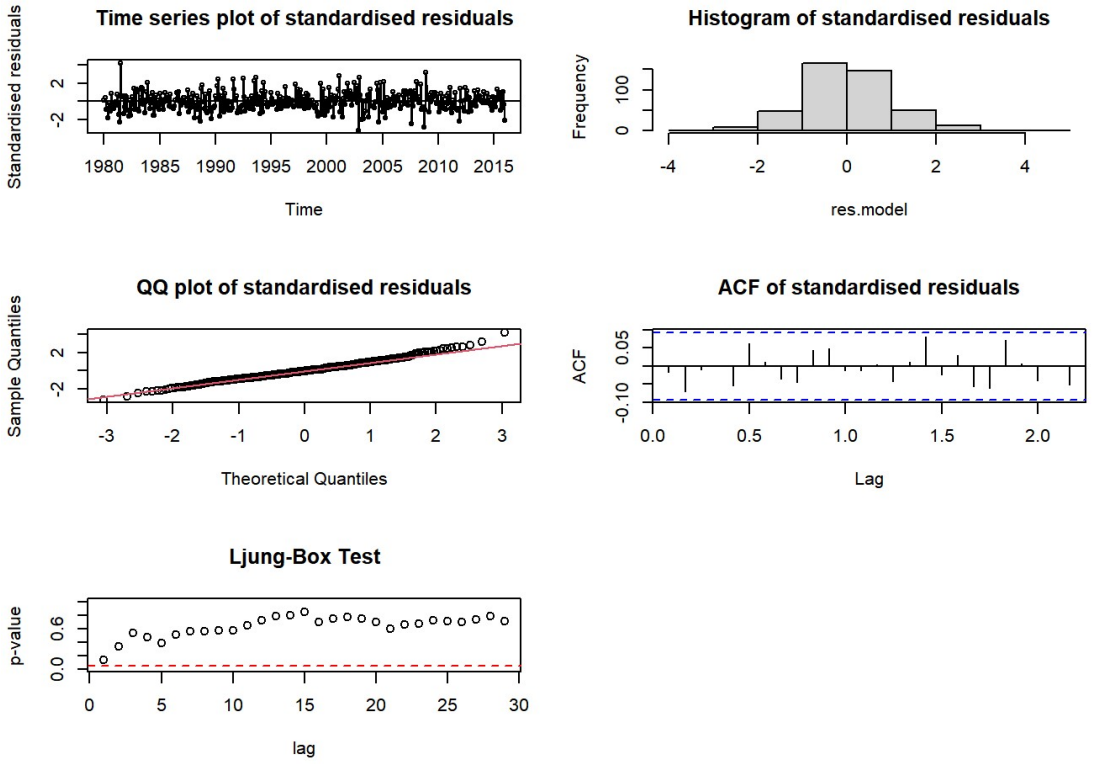

Figure 20: Results of residual analysis for ARIMA(0,1,1)

```
## p d q
## 0 1 2
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99178, p-value = 0.01702
```

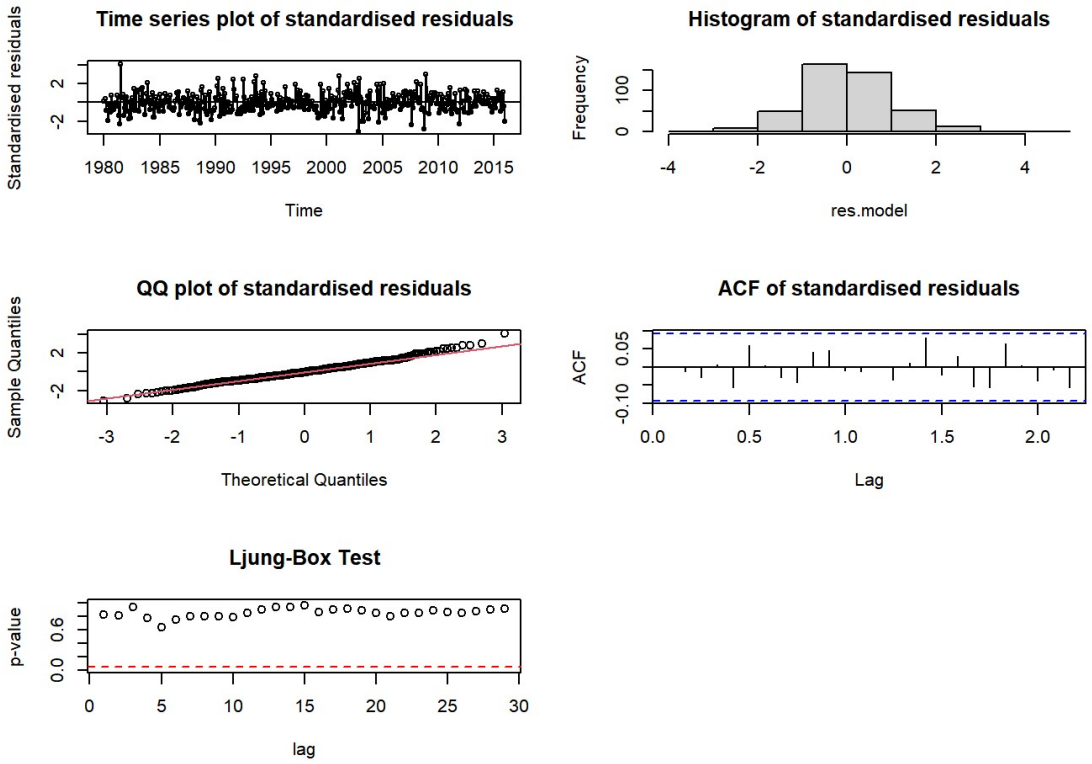

Figure 21: Results of residual analysis for ARIMA(0,1,2)

```
## p d q
## 1 1 2
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99192, p-value = 0.0188
```

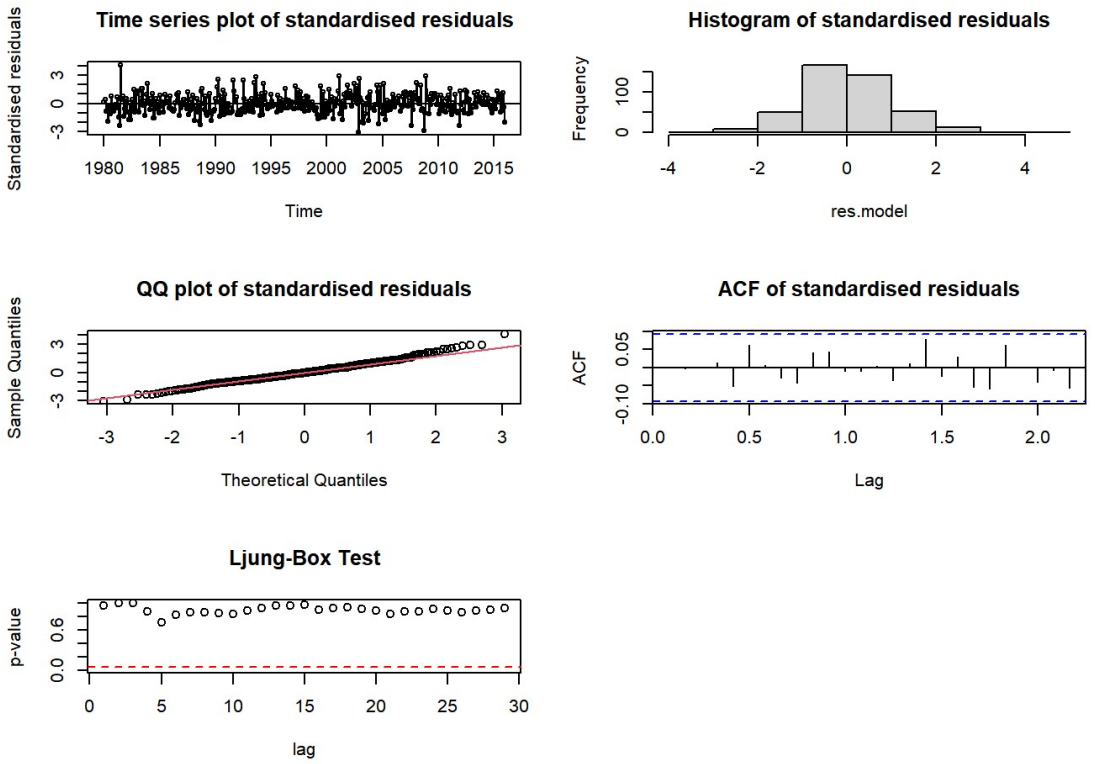

Figure 22: Results of residual analysis for ARIMA(1,1,2)

```
## p d q
## 0 1 3
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99184, p-value = 0.0178
```

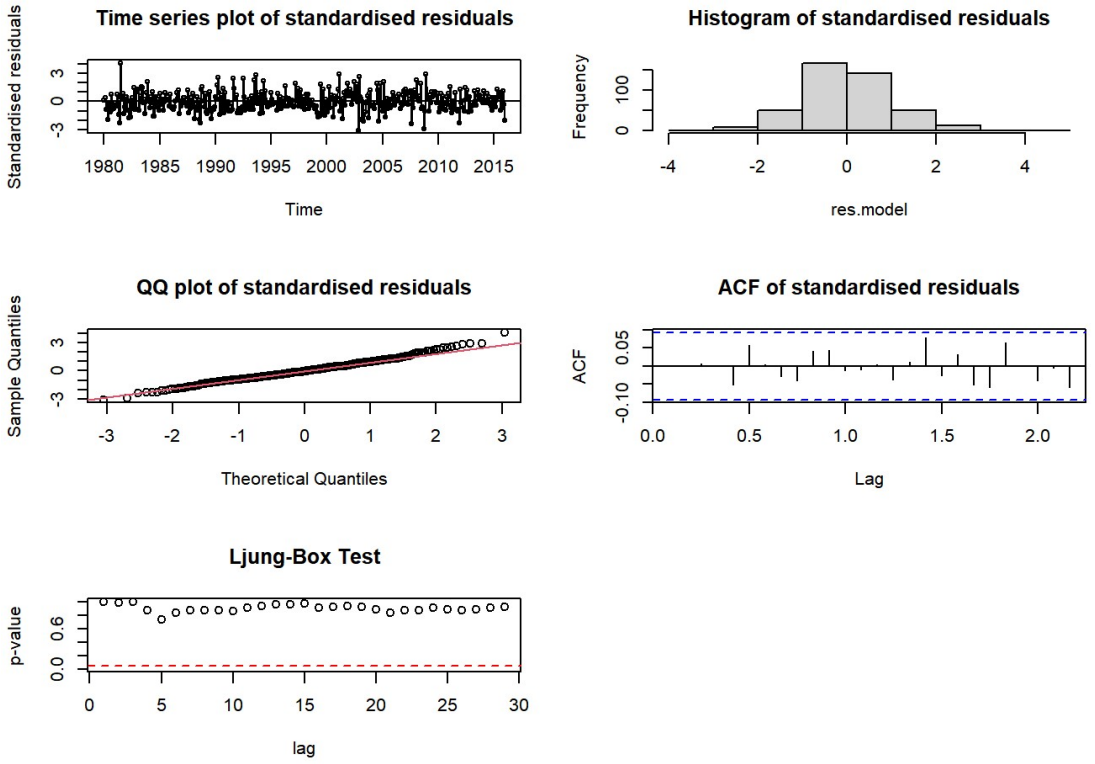

Figure 23: Results of residual analysis for ARIMA(0,1,3)

```
## p d q
## 1 1 4
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99161, p-value = 0.01506
```

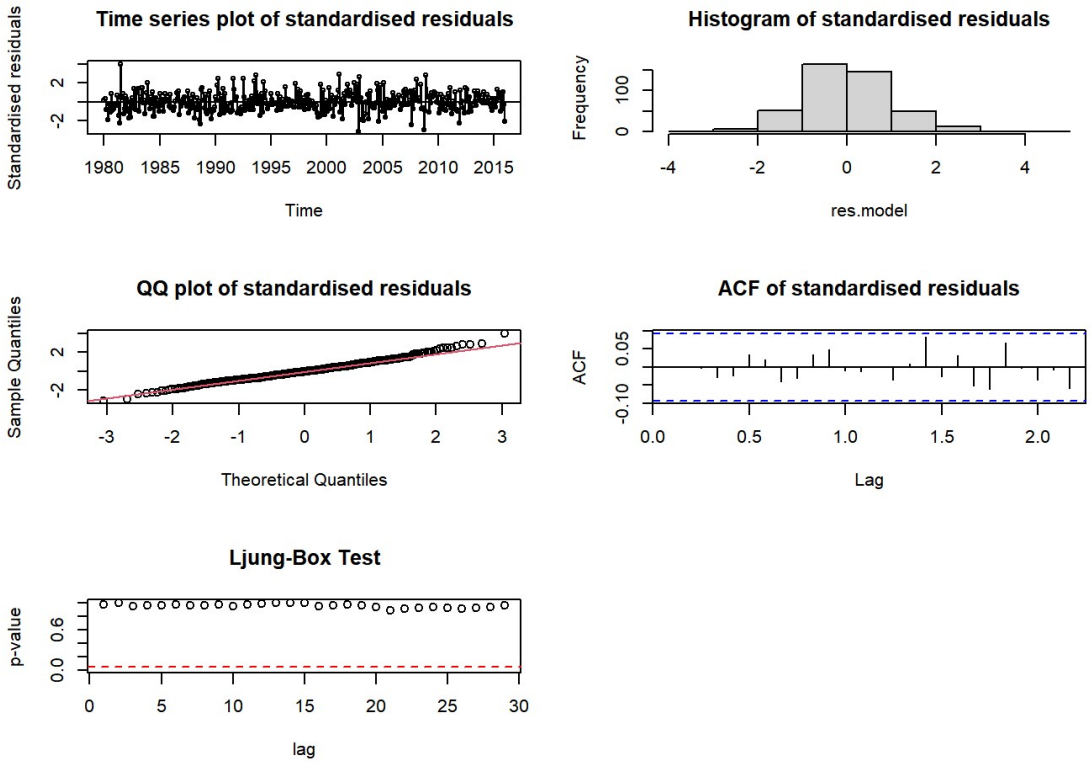

Figure 24: Results of residual analysis for ARIMA(1,1,4)

AIC and BIC Analysis

The AIC() and BIC() functions are used to determine the AIC and BIC values for each model. The sort.score() function developed by Yong Kai Wong is utilized to sort the AIC and BIC scores in ascending order in order to aid selection of the model with the least AIC and BIC (Canvas, 2024). Model ARIMA(0,1,1) has the lowest scores for both AIC and BIC.

```
sort.score(AIC(model_111$ML, model_211$ML, model_011$ML, model_012$ML, model_112$ML, model_013$ML, model_114
$ML),
           score = "aic")
```

```
##              df       AIC
## model_011$ML  2 -1240.096
## model_012$ML  3 -1239.963
## model_111$ML  3 -1239.465
## model_013$ML  4 -1238.624
## model_112$ML  4 -1238.492
## model_211$ML  4 -1238.468
## model_114$ML  6 -1235.931
```

```
sort.score(BIC(model_111$ML, model_211$ML, model_011$ML, model_012$ML, model_112$ML, model_013$ML, model_114
$ML),
           score = "bic")
```

```
##              df       BIC
## model_011$ML  2 -1231.959
## model_012$ML  3 -1227.758
## model_111$ML  3 -1227.260
## model_013$ML  4 -1222.351
## model_112$ML  4 -1222.219
## model_211$ML  4 -1222.194
## model_114$ML  6 -1211.520
```

The error metrics for all the ARIMA models including ME, RMSE, MAE, MPE, MAPE, MASE, and ACF1 are calculated and displayed below. The top four models for each metric are also displayed, demonstrating that ARIMA(0,1,1) has the lowest error for ME, MAPE, MASE, and ACF1. ARIMA(1,1,4) is the best model for RMSE, MAE, and MPE, and is in the top four models for the other error metrics as well. Despite this, considering the lower AIC and BIC scores obtained for ARIMA(0,1,1) makes it possible to conclude that ARIMA(0,1,1) is the best ARIMA model with regards to performance on AIC, BIC and error metrics.

```
Smodel_111 <- forecast::accuracy(summary(model_111_ml))
Smodel_211 <- forecast::accuracy(summary(model_211_ml))
Smodel_011 <- forecast::accuracy(summary(model_011_ml))
Smodel_012 <- forecast::accuracy(summary(model_012_ml))
Smodel_112 <- forecast::accuracy(summary(model_112_ml))
Smodel_013 <- forecast::accuracy(summary(model_013_ml))
Smodel_114 <- forecast::accuracy(summary(model_114_ml))

df.Smodels <- data.frame(
  rbind(
    Smodel_111,
    Smodel_211,
    Smodel_011,
    Smodel_012,
    Smodel_112,
    Smodel_013,
    Smodel_114
  ),
  row.names = c("ARIMA(1,1,1)", "ARIMA(2,1,1)", "ARIMA(0,1,1)", "ARIMA(0,1,2)", "ARIMA(1,1,2)", "ARIMA
(0,1,3)", "ARIMA(1,1,4)")
)

colnames(df.Smodels) <- c("ME", "RMSE", "MAE", "MPE", "MAPE", "MASE", "ACF1")

df.Smodels
```

```
##                         ME       RMSE        MAE          MPE      MAPE
## ARIMA(1,1,1) -0.0001596422 0.05717357 0.04449922 -0.004356008 0.5972674
## ARIMA(2,1,1) -0.0001551430 0.05710680 0.04454158 -0.004480686 0.5979194
## ARIMA(0,1,1) -0.0001677220 0.05726442 0.04429363 -0.004313633 0.5946356
## ARIMA(0,1,2) -0.0001558581 0.05714046 0.04457997 -0.004412885 0.5983407
## ARIMA(1,1,2) -0.0001530510 0.05710517 0.04452421 -0.004548479 0.5976773
## ARIMA(0,1,3) -0.0001543947 0.05709628 0.04451436 -0.004532487 0.5975939
## ARIMA(1,1,4) -0.0001605515 0.05700868 0.04418082 -0.004517245 0.5930187
##                    MASE          ACF1
## ARIMA(1,1,1) 0.2592086 -0.0037560402
## ARIMA(2,1,1) 0.2594554  0.0004526505
## ARIMA(0,1,1) 0.2580111 -0.0164435231
## ARIMA(0,1,2) 0.2596790  0.0030177851
## ARIMA(1,1,2) 0.2593542  0.0011648691
## ARIMA(0,1,3) 0.2592968 -0.0003381985
## ARIMA(1,1,4) 0.2573540  0.0001732062
```

```
setDT(df.Smodels, keep.rownames = "Models")[]
```

```
##          Models            ME       RMSE        MAE          MPE      MAPE
## 1: ARIMA(1,1,1) -0.0001596422 0.05717357 0.04449922 -0.004356008 0.5972674
## 2: ARIMA(2,1,1) -0.0001551430 0.05710680 0.04454158 -0.004480686 0.5979194
## 3: ARIMA(0,1,1) -0.0001677220 0.05726442 0.04429363 -0.004313633 0.5946356
## 4: ARIMA(0,1,2) -0.0001558581 0.05714046 0.04457997 -0.004412885 0.5983407
## 5: ARIMA(1,1,2) -0.0001530510 0.05710517 0.04452421 -0.004548479 0.5976773
## 6: ARIMA(0,1,3) -0.0001543947 0.05709628 0.04451436 -0.004532487 0.5975939
## 7: ARIMA(1,1,4) -0.0001605515 0.05700868 0.04418082 -0.004517245 0.5930187
##         MASE          ACF1
## 1: 0.2592086 -0.0037560402
## 2: 0.2594554  0.0004526505
## 3: 0.2580111 -0.0164435231
## 4: 0.2596790  0.0030177851
## 5: 0.2593542  0.0011648691
## 6: 0.2592968 -0.0003381985
## 7: 0.2573540  0.0001732062
```

```
df.Smodels[order(df.Smodels$ME),][1:4,c(1,2)]
```

```
##          Models            ME
## 1: ARIMA(0,1,1) -0.0001677220
## 2: ARIMA(1,1,4) -0.0001605515
## 3: ARIMA(1,1,1) -0.0001596422
## 4: ARIMA(0,1,2) -0.0001558581
```

```
df.Smodels[order(df.Smodels$RMSE),][1:4,c(1,3)]
```

```
##          Models       RMSE
## 1: ARIMA(1,1,4) 0.05700868
## 2: ARIMA(0,1,3) 0.05709628
## 3: ARIMA(1,1,2) 0.05710517
## 4: ARIMA(2,1,1) 0.05710680
```

```
df.Smodels[order(df.Smodels$MAE),][1:4,c(1,4)]
```

```
##          Models        MAE
## 1: ARIMA(1,1,4) 0.04418082
## 2: ARIMA(0,1,1) 0.04429363
## 3: ARIMA(1,1,1) 0.04449922
## 4: ARIMA(0,1,3) 0.04451436
```

```
df.Smodels[order(df.Smodels$MASE),][1:4,c(1,5)]
```

```
##          Models          MPE
## 1: ARIMA(1,1,4) -0.004517245
## 2: ARIMA(0,1,1) -0.004313633
## 3: ARIMA(1,1,1) -0.004356008
## 4: ARIMA(0,1,3) -0.004532487
```

```
df.Smodels[order(df.Smodels$ACF1),][1:4,c(1,6)]
```

```
##          Models      MAPE
## 1: ARIMA(0,1,1) 0.5946356
## 2: ARIMA(1,1,1) 0.5972674
## 3: ARIMA(0,1,3) 0.5975939
## 4: ARIMA(1,1,4) 0.5930187
```

```
df.Smodels[order(df.Smodels$ACF1),][1:4,c(1,7)]
```

```
##          Models      MASE
## 1: ARIMA(0,1,1) 0.2580111
## 2: ARIMA(1,1,1) 0.2592086
## 3: ARIMA(0,1,3) 0.2592968
## 4: ARIMA(1,1,4) 0.2573540
```

```
df.Smodels[order(df.Smodels$ACF1),][1:4,c(1,8)]
```

```
##          Models          ACF1
## 1: ARIMA(0,1,1) -0.0164435231
## 2: ARIMA(1,1,1) -0.0037560402
## 3: ARIMA(0,1,3) -0.0003381985
## 4: ARIMA(1,1,4)  0.0001732062
```

ARIMA-GARCH Modelling

The fact that the cocoa beans price time series could be made normal and stationary using its returns series suggests that ARIMA-GARCH modelling could be a suitable model for this series. The ARIMA component of the modelling has already been performed in the previous section, and now the GARCH orders can be explored.

GARCH Orders for Squared Series

The GARCH orders are examined by obtaining the absolute standardardised residuals from the selected ARIMA model which is ARIMA(0,1,1). The ACF plot of the absolute standardardised residuals shows two significant early lags, suggesting that q=2 and the PACF plot shows two significant early lags as well, suggesting that max(p,q)=2. This provides the range of possible p values to be p=0,1,2 and the resulting GARCH orders would be GARCH(0,2), GARCH(1,2) and GARCH(2,2).

The EACF table for the absolute standardardised residuals shows the top-left corner model to be at MA=2 and AR=2. The GARCH models for this combination are GARCH(0,2), GARCH(1,2), GARCH(2,2) since max(p,q)=2 and q=2. For the neighboring model where max(p,q)=2 and q=3, no models can be obtained as max(p,q)=2 is mathematically not possible when q=3. For the neighboring model where max(p,q)=3 and q=2, the possible value for p is 3, resulting in a GARCH order of GARCH(3,2). For the neighboring model where max(p,q)=3 and q=3, the possible values for p are p=0,1,2,3 which results in the GARCH orders GARCH(0,3), GARCH(1,3), GARCH(2,3), and GARCH(3,3).

```
abs.r.res.cocoa = abs(rstandard(model_011_ml))

par(mfrow=c(2,1), mar=c(3,3,3,1))
acf(abs.r.res.cocoa, main="ACF plot for absolute returns series", lag.max = 40)
pacf(abs.r.res.cocoa, main="PACF plot for absolute returns series", lag.max = 40)
```

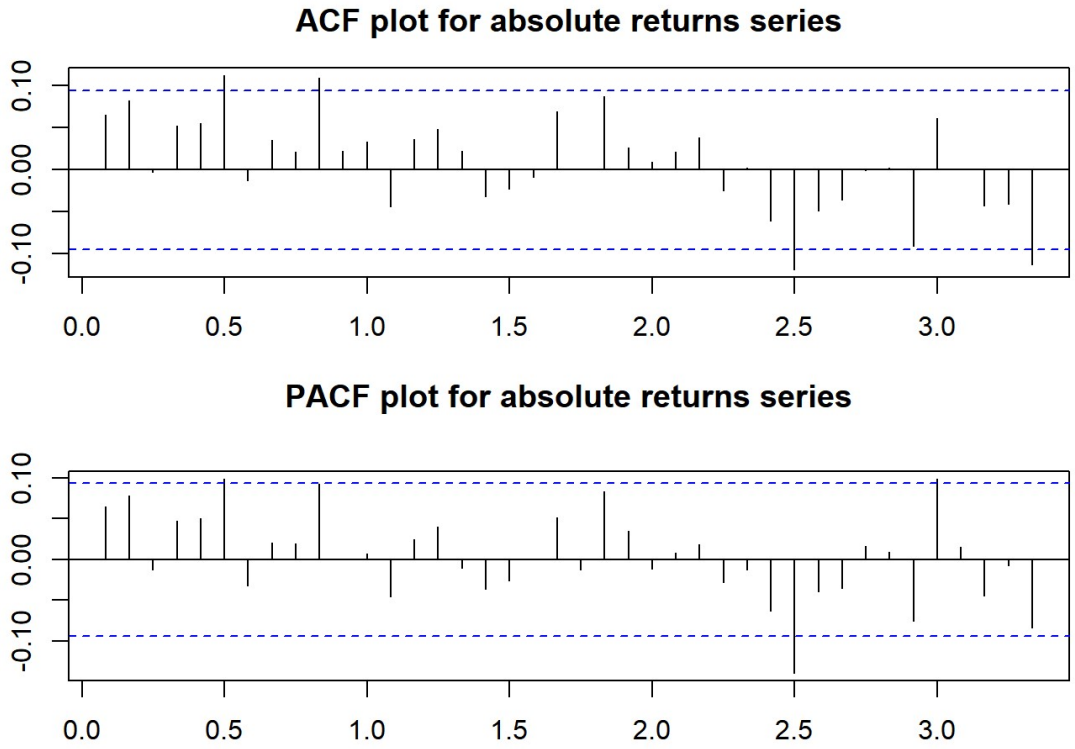

Figure 25: ACF and PACF plots for absolute returns series

```
par(mfrow=c(1,1))

#max(p,q)=2 q=2 therefore p=0,1,2 therefore GARCH(0,2), GARCH(1,2), GARCH(2,2)

eacf(abs.r.res.cocoa)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o o o o x o o o x o  o  o  o
## 1 x o o o o x o o o x o  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x x o o o x o o o o o  o  o  o
## 4 x x o x o o o o o o o  o  o  o
## 5 x x x x o o o o o o o  o  o  o
## 6 x o x x x x o o o o o  o  o  o
## 7 x x o x o x o o o o o  o  o  o
```

```
#max(p,q)=2 q=2 therefore p=0,1,2 therefore GARCH(0,2), GARCH(1,2), GARCH(2,2)
#max(p,q)=2 q=3 therefore no models
#max(p,q)=3 q=2 therefore p=3 therefore GARCH(3,2)
#max(p,q)=3 q=3 therefore p=0,1,2,3 therefore GARCH(0,3), GARCH(1,3), GARCH(2,3), GARCH(3,3)
```

Now, the squared standardardised residuals are obtained and the ACF and PACF of the squared residuals are plotted. The ACF plot of the squared standardardised residuals shows two significant early lags, suggesting that q=2 and the PACF plot shows two significant early lags as well, suggesting that max(p,q)=2. This provides the range of possible p values to be p=0,1,2 and the resulting GARCH orders would be GARCH(0,2), GARCH(1,2) and GARCH(2,2).

The EACF table for the squared standardardised residuals shows the top-left corner model to be at MA=2 and AR=0. No GARCH models for this model or its neighbors can be obtained since these result in the following mathematical contradictions:

- max(p,q)=0 q=2 therefore no possible models from this combination

- max(p,q)=0 q=3 therefore no possible models from this combination
- max(p,q)=1 q=2 therefore no possible models from this combination
- max(p,q)=1 q=3 therefore no possible models from this combination

```
sq.r.res.cocoa = rstandard(model_011_ml)^2

par(mfrow=c(2,1), mar=c(3,3,3,1))
acf(sq.r.res.cocoa, main="ACF plot for squared returns series", lag.max = 40)
pacf(sq.r.res.cocoa, main="PACF plot for squared returns series", lag.max = 40)
```

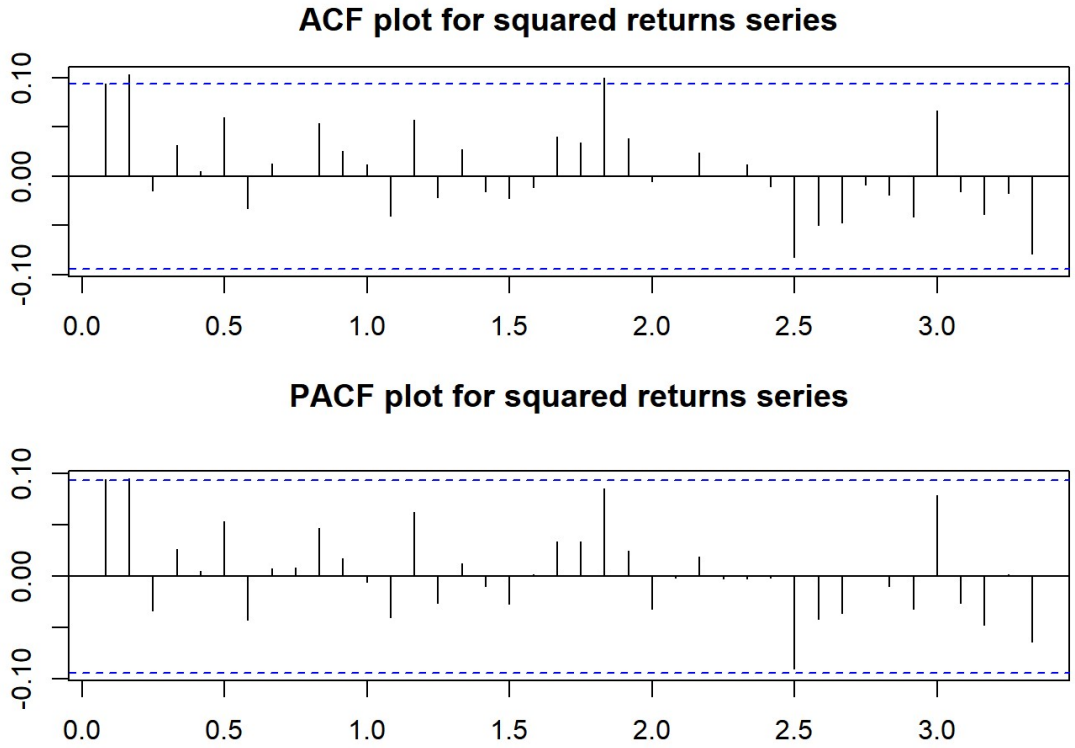

Figure 26: ACF and PACF plots for squared returns series

```
par(mfrow=c(1,1))

#max(p,q)=2 q=2 therefore p=0,1,2 therefore GARCH(0,2), GARCH(1,2), GARCH(2,2)

eacf(sq.r.res.cocoa)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o x o o o o o o o o o  o  o  o
## 1 x x o o o o o o o o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x x o o o o o o o o o  o  o  o
## 4 x x o x o o o o o o o  o  o  o
## 5 o x x x x o o o o o o  o  o  o
## 6 x x o x o x o o o o o  o  o  o
## 7 x x o x x x x o o o o  o  o  o
```

```
#max(p,q)=0 q=2 therefore no models
#max(p,q)=0 q=3 therefore no models
#max(p,q)=1 q=2 therefore no models
#max(p,q)=1 q=3 therefore no models
```

Fitting ARMA-GARCH Models

The resulting GARCH orders from the examination above are listed below;

- GARCH(0,2)
- GARCH(1,2)
- GARCH(2,2)
- GARCH(3,2)
- GARCH(0,3)
- GARCH(1,3)
- GARCH(2,3)
- GARCH(3,3)

The garchFit() function from the fGarch library is used to fit the ARIMA-GARCH models with the GARCH order listed above and an ARIMA order of ARIMA(0,1,1). The residuals from the resulting models are plotted along with their Q-Q plots, histograms, ACF plots and Ljung-Box plots. For each set of residuals, Shapiro-Wilk tests are also conducted to determine normality. The resulting models all produce significant p-values for the Shapiro-Wilk test, suggesting that the residuals are not normally distributed. However, the histograms of residuals are very symmetric for all models and Q-Q plots for all models are very close to the reference line. None of the ACF plots of standardised residuals contain significant peaks. All the plots from the Ljung-Box tests have points above the significance levels, indicating no autocorrelations are observed in the residuals at any of the lags. In fact, the residuals for all the models seem very similar to each other, indicating similar performance. The white noise residuals also suggest that these models explain the series well.

```
# GARCH(0,2), GARCH(1,2), GARCH(2,2) GARCH(3,2) GARCH(0,3), GARCH(1,3), GARCH(2,3), GARCH(3,3)
model_011_02 = fGarch::garchFit(~ arma(0,1)+garch(2,0), data=r.cocoa, trace=F)
model_011_12 = fGarch::garchFit(~ arma(0,1)+garch(2,1), data=r.cocoa, trace=F)
model_011_22 = fGarch::garchFit(~ arma(0,1)+garch(2,2), data=r.cocoa, trace=F)
model_011_32 = fGarch::garchFit(~ arma(0,1)+garch(2,3), data=r.cocoa, trace=F)
model_011_03 = fGarch::garchFit(~ arma(0,1)+garch(3,0), data=r.cocoa, trace=F)
model_011_13 = fGarch::garchFit(~ arma(0,1)+garch(3,1), data=r.cocoa, trace=F)
model_011_23 = fGarch::garchFit(~ arma(0,1)+garch(3,2), data=r.cocoa, trace=F)
model_011_33 = fGarch::garchFit(~ arma(0,1)+garch(3,3), data=r.cocoa, trace=F)

summary(model_011_02)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_02, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99118, p-value = 0.01123
```

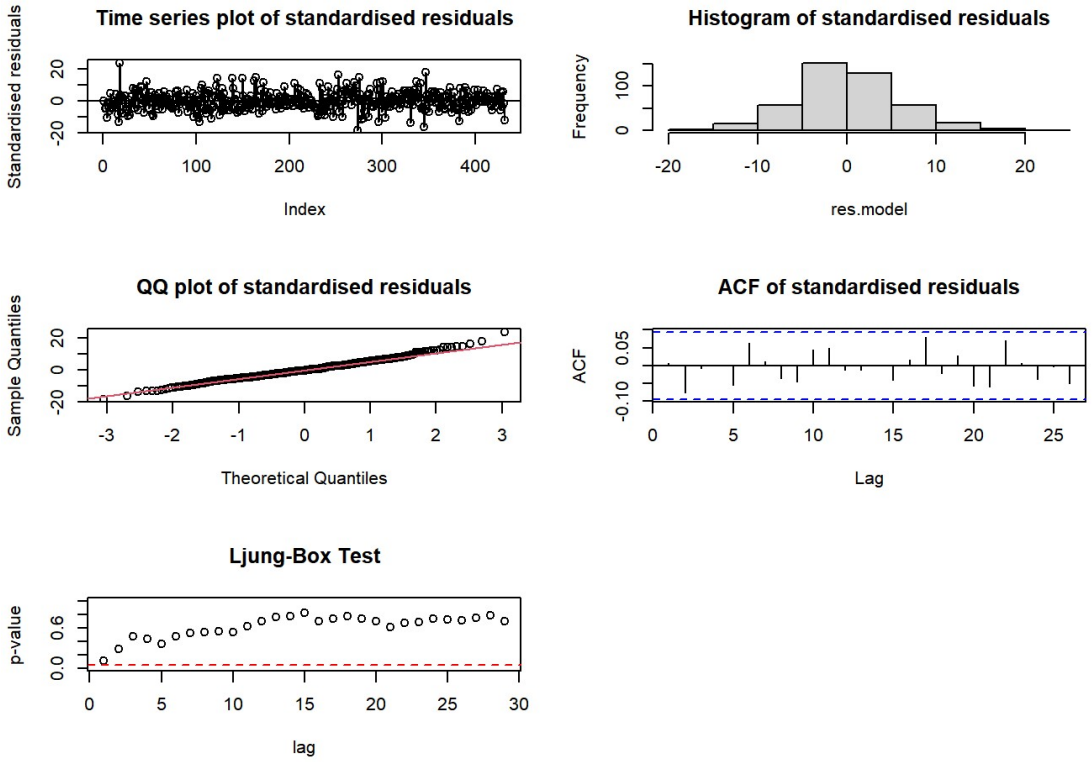

Figure 27: Results of residual analysis for ARMA(0,1)+GARCH(0,2)

```
summary(model_011_12)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_12, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99122, p-value = 0.01157
```

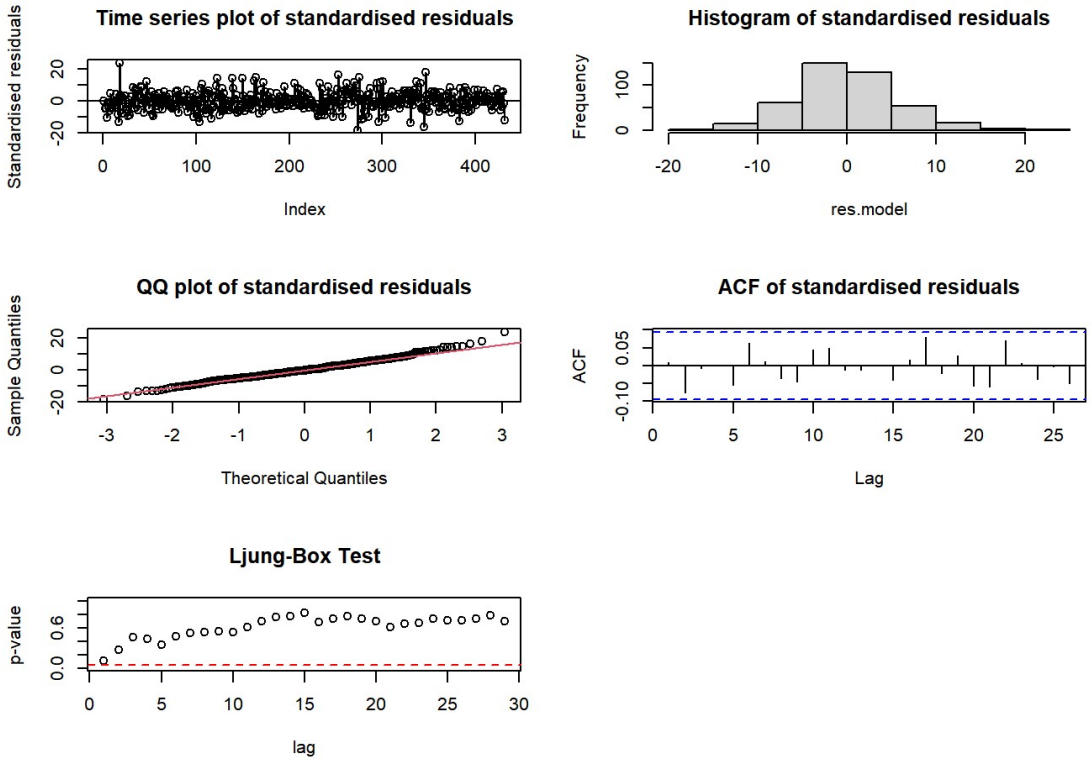

Figure 28: Results of residual analysis for ARMA(0,1)+GARCH(1,2)

```
summary(model_011_22)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_22, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99112, p-value = 0.01074
```

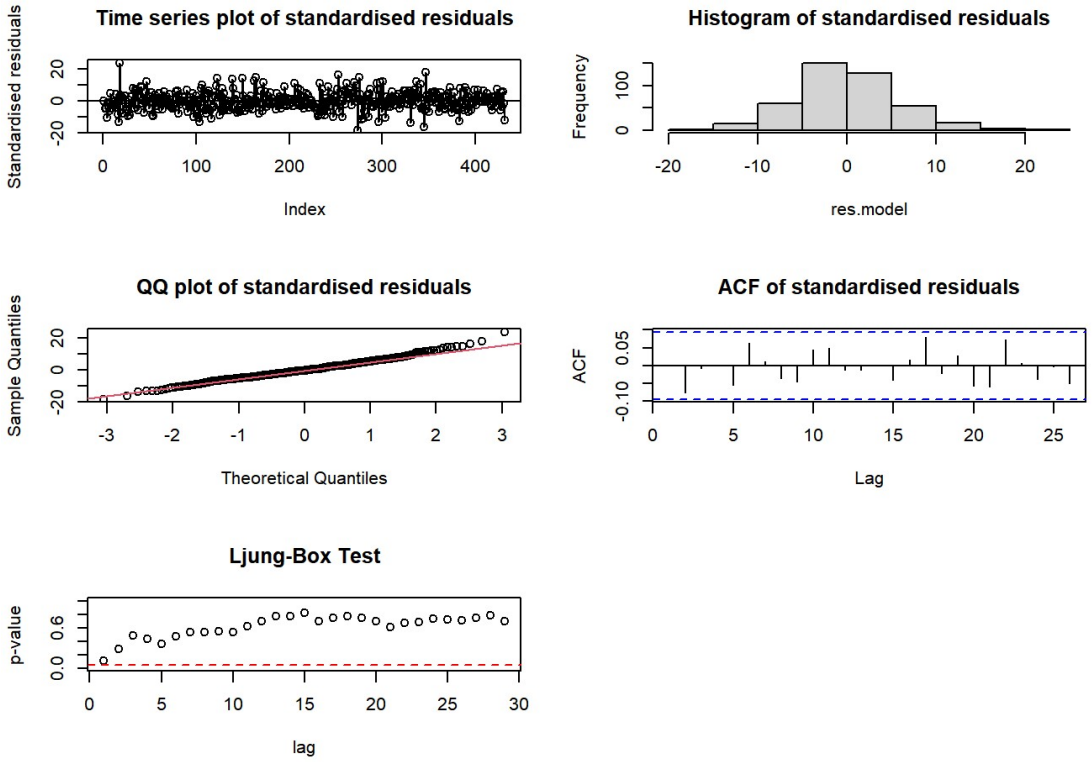

Figure 29: Results of residual analysis for ARMA(0,1)+GARCH(2,2)

```
summary(model_011_32)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_32, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99132, p-value = 0.01236
```

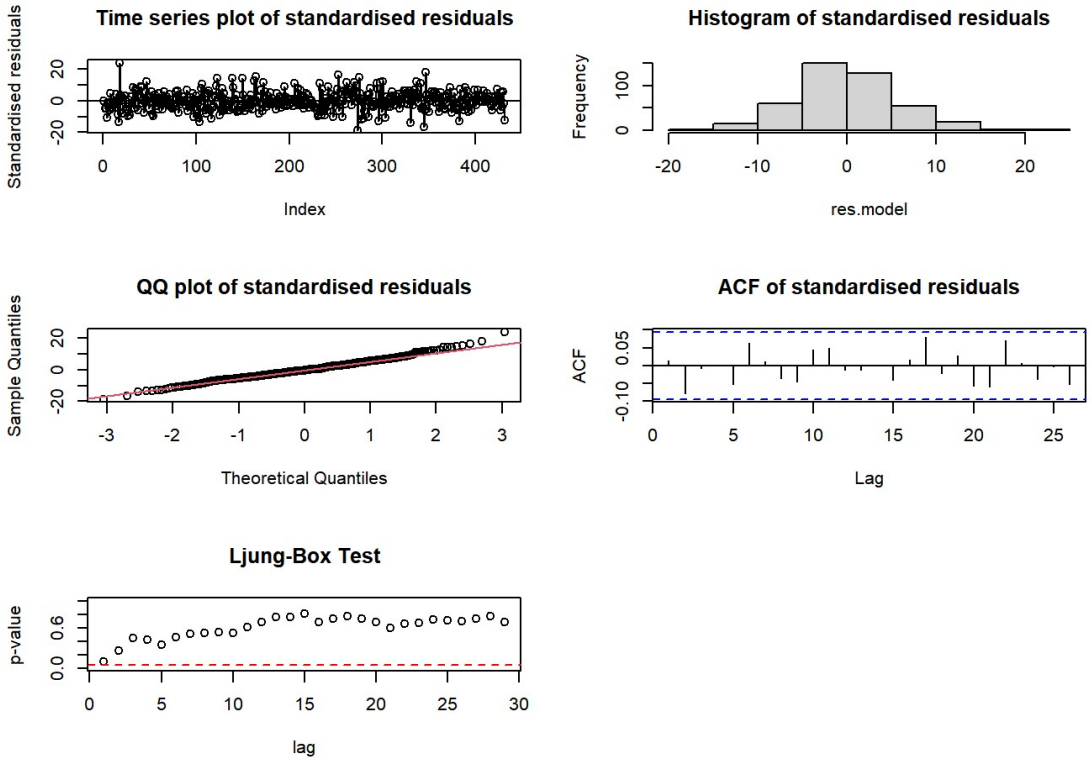

Figure 30: Results of residual analysis for ARMA(0,1)+GARCH(3,2)

```
summary(model_011_03)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_03, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99116, p-value = 0.01104
```

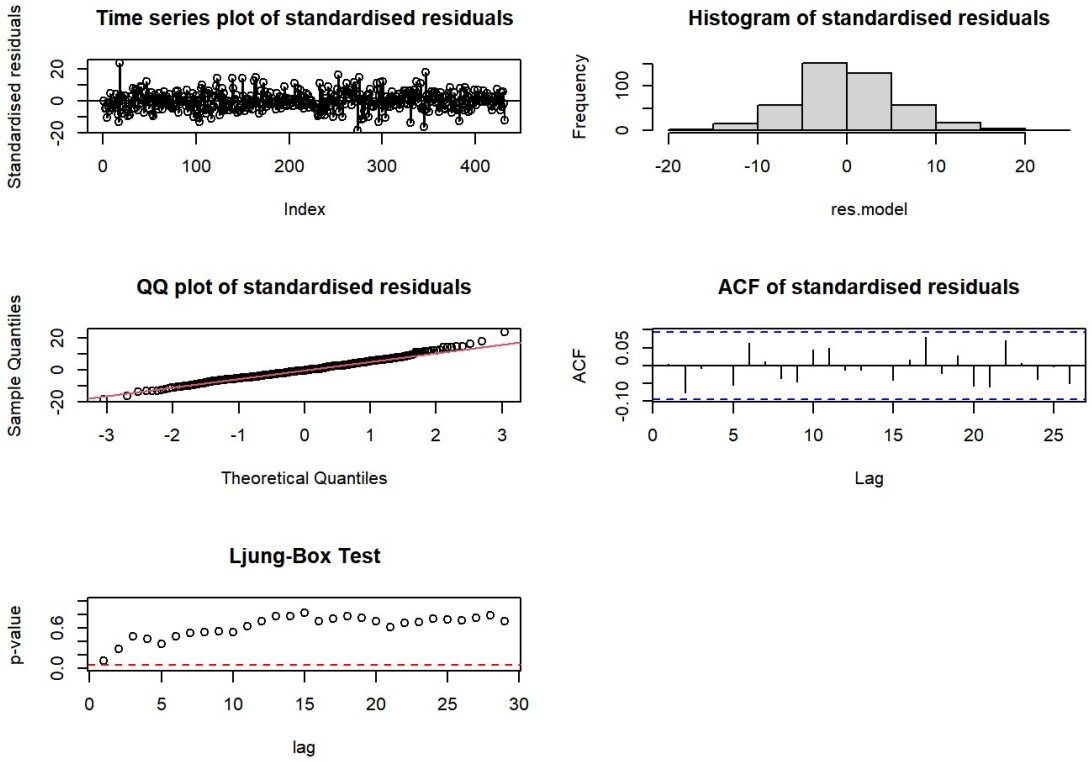

Figure 31: Results of residual analysis for ARMA(0,1)+GARCH(0,3)

```
summary(model_011_13)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_13, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99123, p-value = 0.0116
```

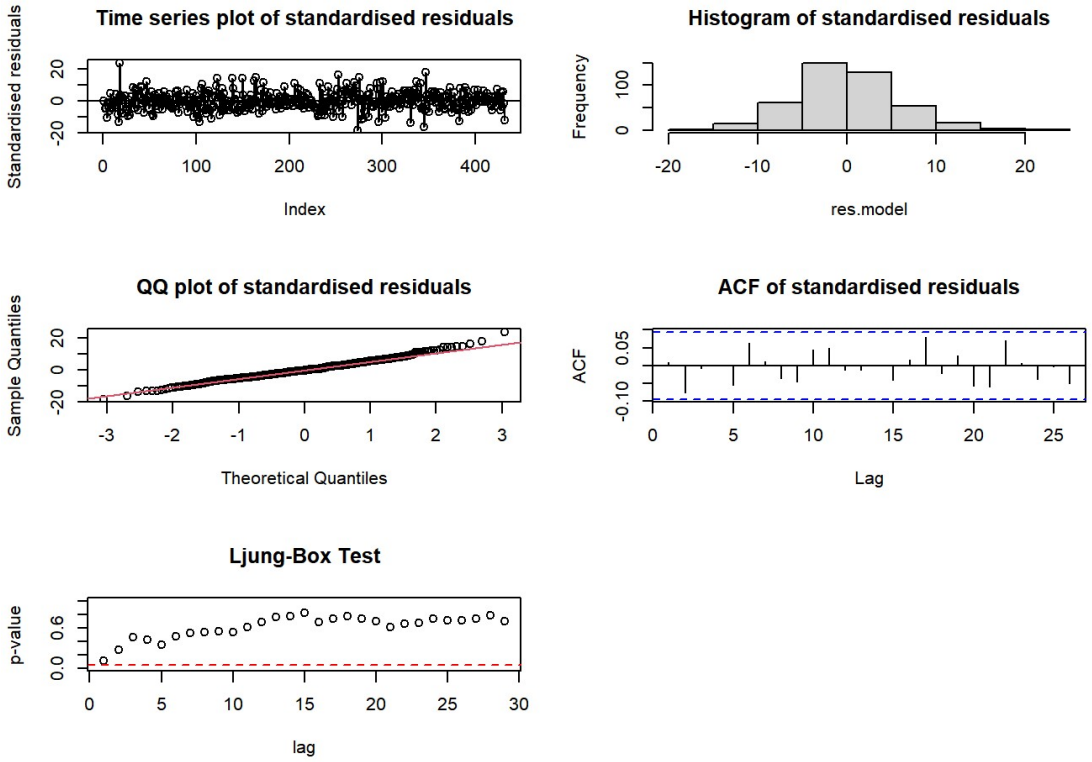

Figure 32: Results of residual analysis for ARMA(0,1)+GARCH(1,3)

```
summary(model_011_23)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_23, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99113, p-value = 0.01079
```

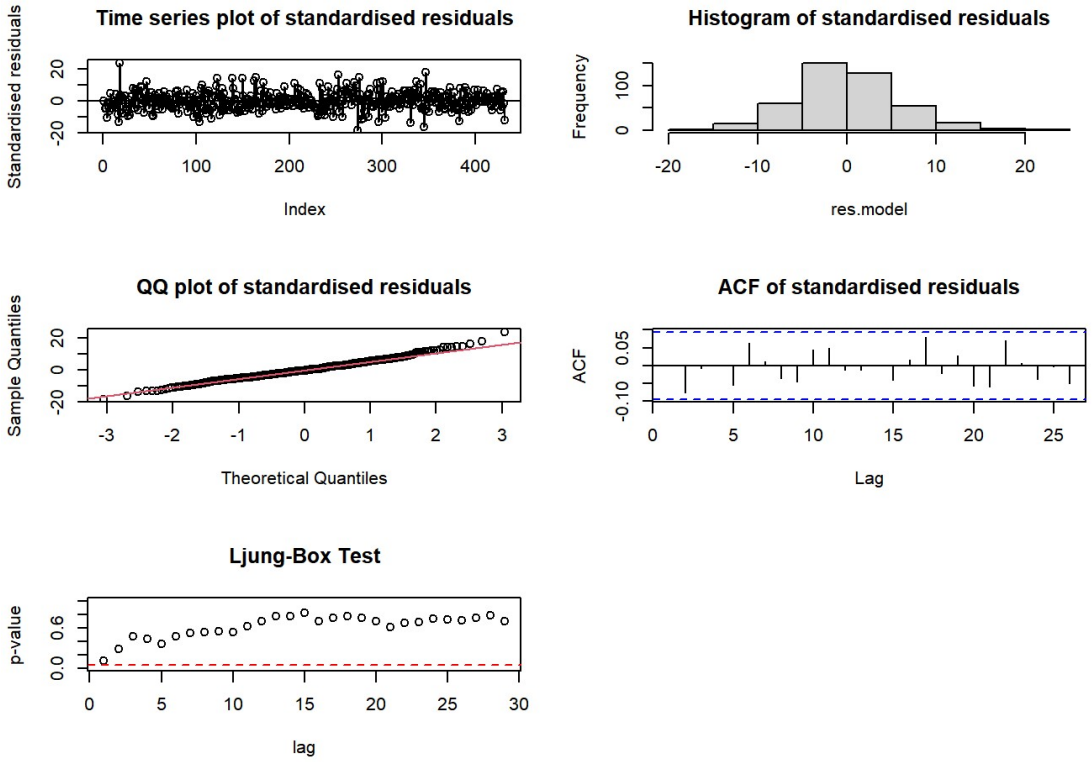

Figure 33: Results of residual analysis for ARMA(0,1)+GARCH(2,3)

```
summary(model_011_33)
```

```
## Length  Class   Mode
##      1 fGARCH     S4
```

```
residual.analysis(model=model_011_33, std=TRUE, start=2, class="fGARCH")
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.99132, p-value = 0.01236
```

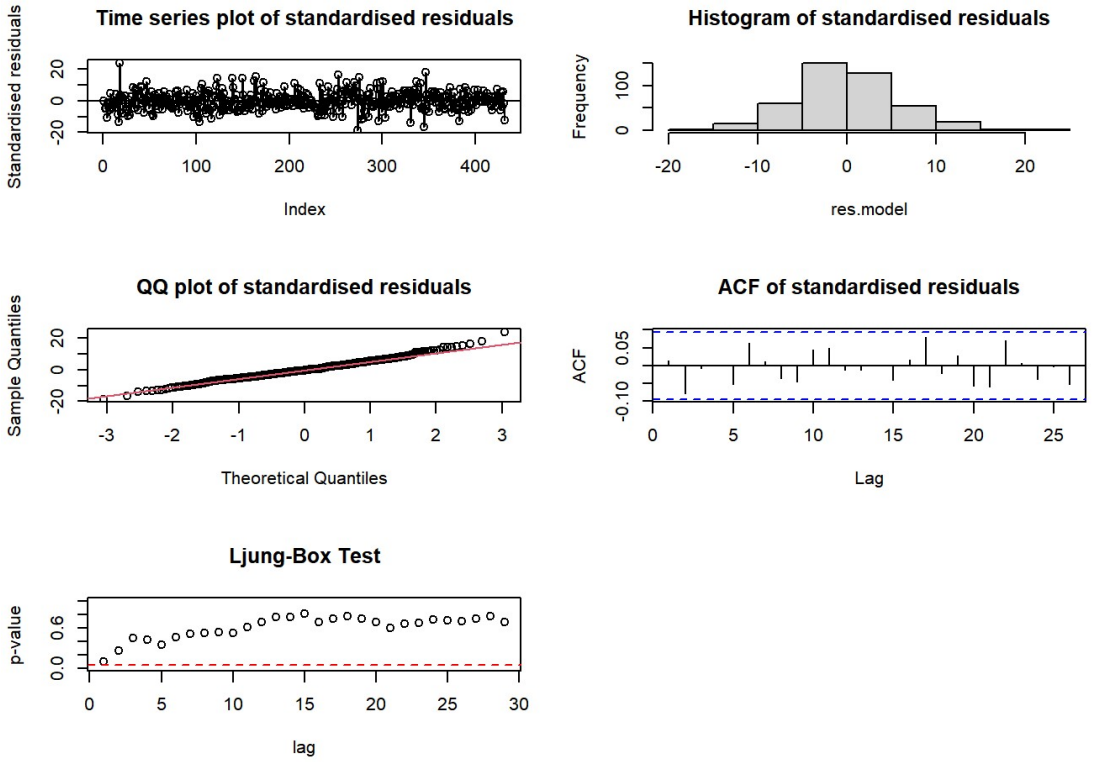

Figure 34: Results of residual analysis for ARMA(0,1)+GARCH(3,3)

The AIC and BIC scores for each model are obtained and displayed below. The model with the lowest AIC and BIC scores is ARMA(0,1)+GARCH(0,2), making it the best ARMA-GARCH model tested.

Overall, the satisfactory residual performance and the fact that the original series had evidence of changing variance prompted **ARMA(0,1)+GARCH(0,2)** to be selected as the best model tested in this report.

```
# AIC BIC
df = data.frame(AIC = c(model_011_02@fit$ics[1], model_011_12@fit$ics[1], model_011_22@fit$ics[1], model_011_
32@fit$ics[1], model_011_03@fit$ics[1], model_011_13@fit$ics[1], model_011_23@fit$ics[1], model_011_33@fit$ic
s[1]),
                 BIC = c(model_011_02@fit$ics[2], model_011_12@fit$ics[2], model_011_22@fit$ics[2], model_011
_32@fit$ics[2], model_011_03@fit$ics[2], model_011_13@fit$ics[2], model_011_23@fit$ics[2], model_011_33@fit$i
cs[2]))
rownames(df) = c("ARMA(0,1)+GARCH(0,2)", "ARMA(0,1)+GARCH(1,2)", "ARMA(0,1)+GARCH(2,2)", "ARMA(0,1)+GARCH
(3,2)", "ARMA(0,1)+GARCH(0,3)", "ARMA(0,1)+GARCH(1,3)", "ARMA(0,1)+GARCH(2,3)", "ARMA(0,1)+GARCH(3,3)")

print("AIC and BIC values sorted by AIC")
```

```
## [1] "AIC and BIC values sorted by AIC"
```

```
df[order(df$AIC),]
```

```
##                           AIC      BIC
## ARMA(0,1)+GARCH(0,2) 6.322364 6.369452
## ARMA(0,1)+GARCH(2,2) 6.324818 6.390741
## ARMA(0,1)+GARCH(1,2) 6.325174 6.381680
## ARMA(0,1)+GARCH(0,3) 6.327019 6.383525
## ARMA(0,1)+GARCH(3,2) 6.328617 6.403958
## ARMA(0,1)+GARCH(2,3) 6.329701 6.405042
## ARMA(0,1)+GARCH(1,3) 6.329710 6.395634
## ARMA(0,1)+GARCH(3,3) 6.333247 6.418006
```

```
print("AIC and BIC values sorted by BIC")
```

```
## [1] "AIC and BIC values sorted by BIC"
```

```
df[order(df$BIC),]
```

```
##                            AIC      BIC
## ARMA(0,1)+GARCH(0,2) 6.322364 6.369452
## ARMA(0,1)+GARCH(1,2) 6.325174 6.381680
## ARMA(0,1)+GARCH(0,3) 6.327019 6.383525
## ARMA(0,1)+GARCH(2,2) 6.324818 6.390741
## ARMA(0,1)+GARCH(1,3) 6.329710 6.395634
## ARMA(0,1)+GARCH(3,2) 6.328617 6.403958
## ARMA(0,1)+GARCH(2,3) 6.329701 6.405042
## ARMA(0,1)+GARCH(3,3) 6.333247 6.418006
```

The ARMA(0,1)+GARCH(0,2) model is redefined using ugarchspec in order to create forecasts. From the resulting individual statistics, only MA1 is significant with a p-value of 0.02214.

```
spec = ugarchspec(variance.model = list(model = 'sGARCH',
                                        garchOrder = c(0,2)
                                        ),
                                      mean.model = list(armaOrder = c(0,1)))
model_011_02_2 = ugarchfit(spec = spec, data = r.cocoa,
                           solver = "hybrid",
                           solver.control = list(trace=0))
model_011_02_2
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(0,2)
## Mean Model   : ARFIMA(0,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##        Estimate  Std. Error     t value Pr(>|t|)
## mu    -0.013751    0.346353 -3.9701e-02 0.968331
## ma1    0.255863    0.049654  5.1530e+00 0.000000
## omega  0.023547    0.005746  4.0983e+00 0.000042
## beta1  0.005396    0.000146  3.6907e+01 0.000000
## beta2  0.993604    0.000006  1.6042e+05 0.000000
##
## Robust Standard Errors:
##        Estimate  Std. Error     t value Pr(>|t|)
## mu    -0.013751    0.325151 -4.2290e-02  0.96627
## ma1    0.255863    0.045183  5.6628e+00  0.00000
## omega  0.023547    0.002773  8.4903e+00  0.00000
## beta1  0.005396    0.000774  6.9712e+00  0.00000
## beta2  0.993604    0.000006  1.7677e+05  0.00000
##
## LogLikelihood : -1367.247
##
## Information Criteria
## ------------------------------------
##
## Akaike       6.3530
## Bayes        6.4001
## Shibata      6.3527
## Hannan-Quinn 6.3716
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1272  0.7214
## Lag[2*(p+q)+(p+q)-1][2]    1.1994  0.6167
## Lag[4*(p+q)+(p+q)-1][5]    2.1334  0.6726
## d.o.f=1
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      3.568 0.05889
## Lag[2*(p+q)+(p+q)-1][5]     7.749 0.03400
## Lag[4*(p+q)+(p+q)-1][9]     9.278 0.07122
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1095 0.500 2.000  0.7407
## ARCH Lag[5]    0.5548 1.440 1.667  0.8673
## ARCH Lag[7]    1.7327 2.315 1.543  0.7736
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.8964
```

```
## Individual Statistics:
## mu    0.17615
## ma1   0.02214
## omega 0.19815
## beta1 0.19829
## beta2 0.19829
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                      t-value     prob sig
## Sign Bias           0.716897 0.473830
## Negative Sign Bias  3.131847 0.001857 ***
## Positive Sign Bias  0.007224 0.994239
## Joint Effect       12.745043 0.005222 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## -----------------------------------
##   group statistic p-value(g-1)
## 1    20     19.02       0.4556
## 2    30     35.92       0.1760
## 3    40     39.48       0.4484
## 4    50     53.19       0.3162
##
##
## Elapsed time : 0.08617997
```

The 10-month forecast for the returns series is obtained using ugarchforecast() and the forecast for the returns series is first plotted. The plot shows the forecast continuing with its upward projection till it hits 0, and then remains constant for the remainder of the forecast. The confidence interval is about 10 units wide in the flat region and the lower confidence interval is seen to be wider during the period where the forecast increases.

The forecast of unconditional variance is also plotted and is seen to remain relatively constant during the forecast period, showing just a slight linear decrease.

```
frc = ugarchforecast(model_011_02_2, n.ahead = 10, data=r.cocoa)
frc
```

```
##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: sGARCH
## Horizon: 10
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=Jan 2016]:
##        Series Sigma
## T+1  -3.08044 5.572
## T+2  -0.01375 5.572
## T+3  -0.01375 5.572
## T+4  -0.01375 5.572
## T+5  -0.01375 5.571
## T+6  -0.01375 5.571
## T+7  -0.01375 5.570
## T+8  -0.01375 5.570
## T+9  -0.01375 5.570
## T+10 -0.01375 5.569
```

```
plot(frc, which = 1)
```

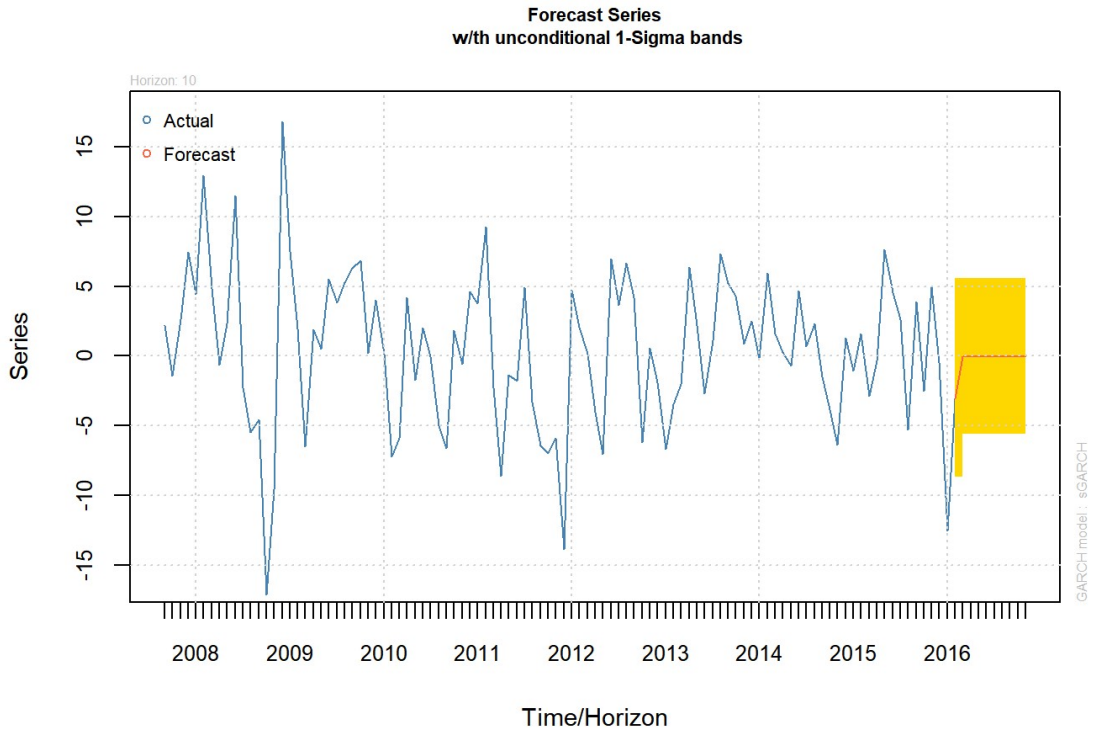

Figure 35: 10-month forecast for returns series using ARMA(0,1)+GARCH(0,2)

```
plot(frc, which = 3) # plot of unconditional variance
```

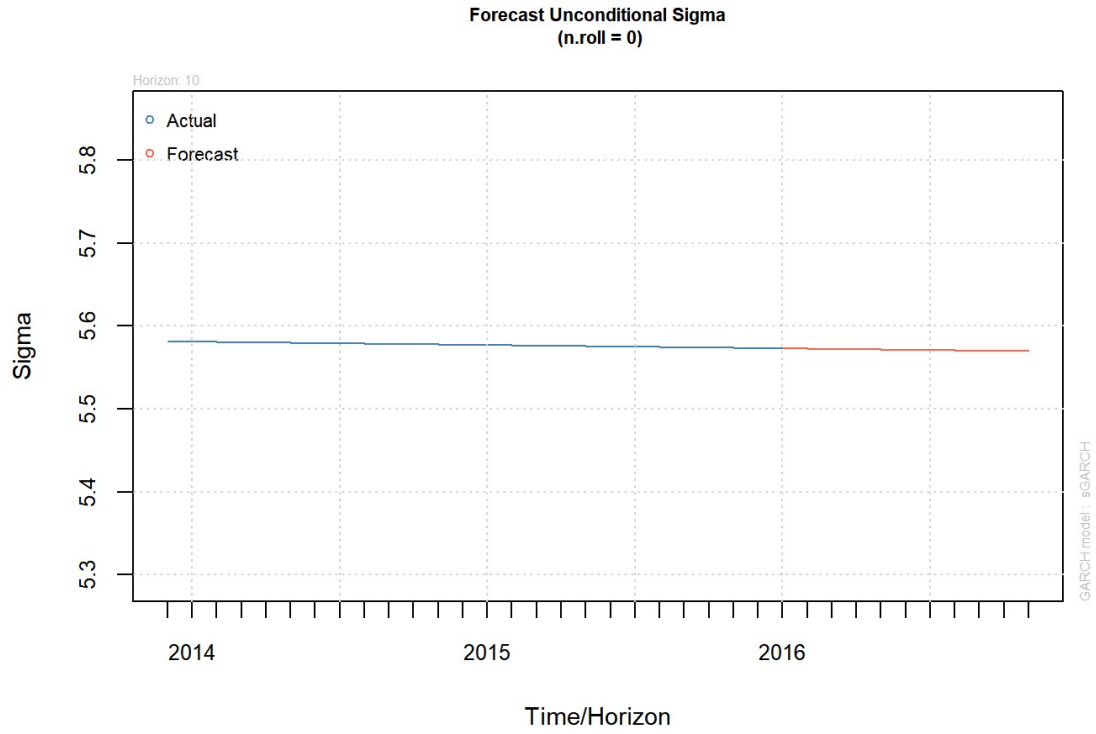

Figure 36: Plot of unconditional variance of forecast

The forecast is then transformed back into the original value range by reversing the log and differencing transformations applied. The final forecast is then plotted along with the tail end of the original series. The forecast is seen to dip slightly between the first and second months and then remain relatively stable across the rest of the forecast period.

```
frcMean = as.vector(frc@forecast$seriesFor/100)
frcRaw = exp(diffinv(frcMean, xi = log.cocoa[length(log.cocoa)]))

plot(cocoa.ts, , type="o", xlim = c(2015,2017),
     ylab = "Cocoa price series",
     main = "Cocoa price series and 10-month forecast")
lines(ts(as.vector(frcRaw), start = c(2016,3), frequency = 12), col="red", type="o")
```

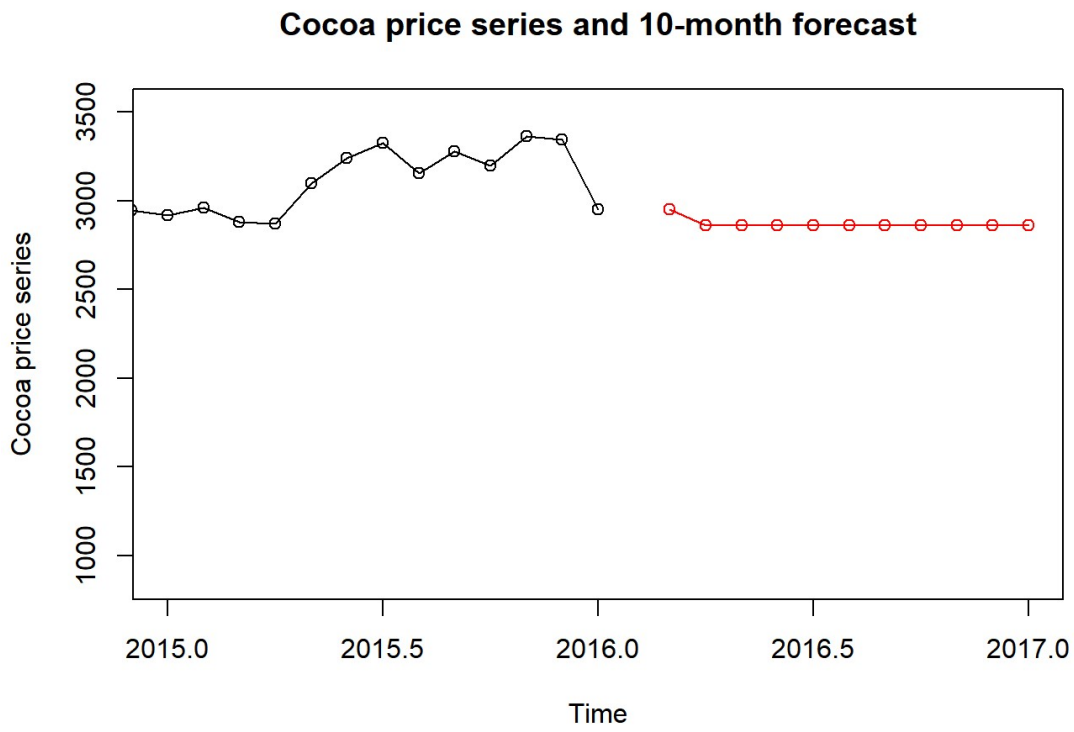

Figure 37: 10-month forecast for original series using ARMA(0,1)+GARCH(0,2)

Conclusion

During the investigation, a total of two trend models, seven ARIMA models, and eight ARIMA-GARCH models were fitted and tested for the time series data describing cocoa beans prices for the period 1980-2016. Visual analysis of the series confirmed a possible combination of AR and MA components due to the trend and fluctuations seen in the series. ADF testing confirmed non-stationarity in the series and Shapiro-Wilk testing confirmed non-normality in the data. This was further supported by the ACF and PACF plots which demonstrated autocorrelations at multiple lags. Applying linear and quadratic trend models to the series produced models that approximated only the trend but did not describe much of the data. Applying the Box-Cox transformation to the series provided a lambda value of 0, indicating a log transformation. Log transforming the series provided visual improvements but further testing confirmed that the log-transformed series was still not stationary and had non-normal distribution. Obtaining the returns series by differencing the log series, however, produced a stationary series with normal distribution and was chosen as the suitable transformation for the series.

The ACF and PACF plots of the returns series, in addition to the EACF table and neighboring model selection, provided a list of seven ARIMA models for fitting. These models were fitted and their parameter coefficients were discussed. Afterwards, obtaining the AIC and BIC scores, along with the ME, RMSE, MAE, MASE, and ACF1 error metrics shortlisted the two best models to be ARIMA(1,1,4) and ARIMA(0,0,1). None of the ARIMA models had normal residuals based on Shapiro-Wilk testing, however. However, the superior performance of ARIMA(0,0,1) when considering AIC and BIC scores and its overall satisfactory performance with the other error metrics justified its selection as the most suitable model to describe the coca beans price series.

The first difference of the log series achieving stationarity and normality suggested the use of ARMA-GARCH modelling, which was attempted by finding the GARCH orders using the absolute residuals and squared residuals from the ARIMA(0,1,1) model. This resulted in a total of eight GARCH models, each of which produced good residuals. However, none of the ARIMA-GARCH models had normal residuals based on Shapiro-Wilk testing. The residual analysis also suggested no seasonal components remaining in the residuals, eliminating the need for seasonal models. Based on the residuals, AIC scores, and BIC scores, ARMA(0,1)+GARCH(0,2) was chosen as the best model. Performing a 10-month forecast on the returns series produced a slight uptick and subsequently constant forecast. Reversing the log and differencing transformations on the forecasts to obtain the original series forecast produced a slight downtick and subsequently constant forecast for the 10-month period.

Further investigation for time series modelling for the cocoa beans price series could be performed by implementing alternate models that eliminate non-normality in the residuals, since this was not achieved in this investigation.

References

- Canvas (2024) sort.score.R. Available at: https://rmit.instructure.com/courses/124176/files/36179115?module_item_id=5935464 (https://rmit.instructure.com/courses/124176/files/36179115?module_item_id=5935464) (accessed 1st June 2024).
- Canvas (2024) residual.analysis.R. Available at: https://rmit.instructure.com/courses/124176/files/38344705?module_item_id=5935463 (https://rmit.instructure.com/courses/124176/files/38344705?module_item_id=5935463) (accessed 1st June 2024).
- Kaggle (2020) ‘Worldwide Commodity Prices’ 2024 Available at: https://www.kaggle.com/datasets/vagifa/usa-commodity-prices?resource=download (https://www.kaggle.com/datasets/vagifa/usa-commodity-prices?resource=download) (accessed 15th May 2024).