ECOM90024 - Assignment 3 - Question 1

Josh Copeland

2024-05-22

```
# Packages used for this assignment

# library(tidyverse)
# library(lubridate)
# library(forecast)
# library(urca)
# library(tseries)
# library(rugarch)
# library(janitor)
```

a) Using the Case Shiller housing price data from Assignment 2, test for the presence of a unit root using an appropriate Augmented Dicky Fuller and determine the order of integration.

Before testing for a unit root, I first need to generate the demeaned, detrended and seasonally adjusted series. We can infer if its reasonable to assume any OLS deterministic trends by looking at Chart 1. It tells me that it is not appropriate to apply any OLS deterministic trends or seasonality to this data:

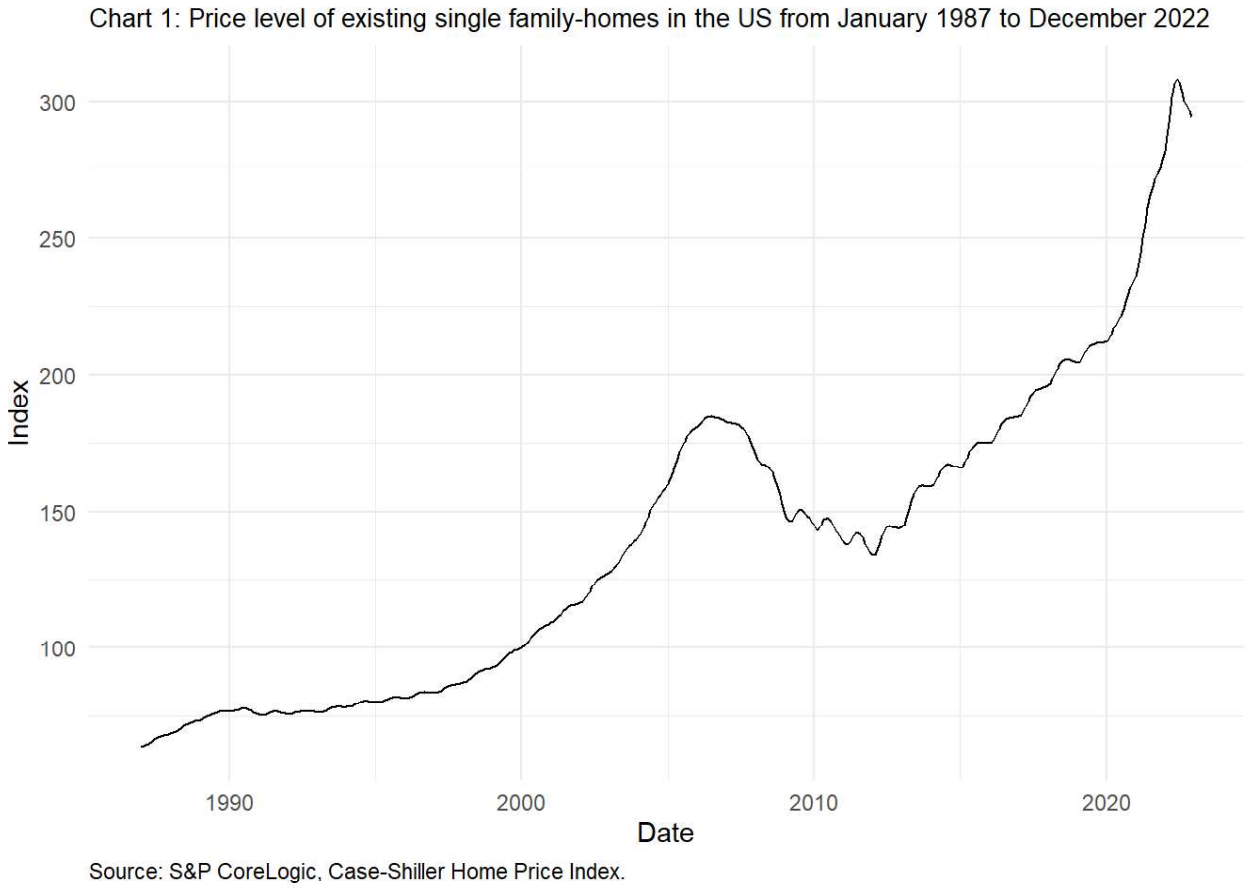
- Although there is an upwards trend to the data, the clear cyclical fluctuations in this series are so intense that it makes it difficult to fit any functional form to it. Furthermore, we are only estimating our time series model up to the end of 2018, and we can see there is a very significant acceleration in the growth of this series from 2020 onwards. It would therefore not make sense to apply any such trend, as its clear the data generating process changed significantly after this period, which would make whatever deterministic trend a very poor out of sample fit.
- I do not see any clear evidence of persistent seasonal effects in this data. There are some short-term fluctuations at different parts of the history, but these are inconsistent in both their timing and magnitude. Therefore, an OLS method could not effectively deal with these fluctuations, even if they might be seasonal in nature.

Therefore, given we are not amending this data for seasonal or trend effects, simply demeaning it will give us the cyclical series we will be creating a time series model for.

```
data <- read_csv("csindex.csv") %>%
  select(date = DATE, price_level = CSUSHPISA) %>%
  mutate(date = dmy(date)) %>%
  na.omit()
```

```
## Rows: 433 Columns: 2
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): DATE
## dbl (1): CSUSHPISA
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
ggplot(data, aes(x = date, y = price_level)) +
  geom_line() +
  ggtitle("Chart 1: Price level of existing single family-homes in the US from January 1987 t
o December 2022") +
  labs( x = "Date",
        y = "Index",
        caption = "Source: S&P CoreLogic, Case-Shiller Home Price Index.") +
  theme(plot.title = element_text(size = 10),
        plot.caption = element_text(hjust = 0))
```




```
#################### GENERATING  CYCLICAL SERIES ################################

data <- data %>%
  mutate(mean = mean(price_level)) %>%
  mutate(demeaned = price_level - mean) %>%
  select(date, price_level, mean, cyclical = demeaned) %>%
  filter(date < "2019-01-01")
```

Now we are ready to conduct the Augmented Dicky-Fuller (ADF) test:

- First, I set the maximum number of potential lags for our ADF test (kmax) using the Ng & Perron (1995) testing procedure. For our data, this is equal to 17 lags.
- I then perform the ADF test iteratively, starting with the numbers of lags suggested by kmax. If the t-statistic associated with the coefficient estimated on the last lagged difference is greater than 1.6 in absolute value then we use this number of lags. If not, then we continue reducing the number of lags by one until this occurs. I have used a loop to automate this process.
- As this cyclical series reflects the demeaned, detrended and seasonally adjusted series, we use the "none" option (Model 1) or our ADF test, as we've already accounted for the trend and drift term within the price level.
- Using 13 lags we conduct the ADF test on the level of our cyclical series and extract a test statistic of -1.1345. This is larger than all of the relevant critical values. Therefore fail to reject the null hypothesis of there being a unit root in this series, meaning we must difference the data and test again.
- Using 17 lags, we conduct the ADF test on the differenced cyclical series and extract a test statistic of -2.716. This is larger than all of the given critical values in our test. Therefore reject the null hypothesis of there being a unit root in the differenced series at any reasonable significance level.
- Therefore, we conclude this series is integrated order 1 (I(1)).

```
################################################################################
############################## ADF TEST ########################################
################################################################################

#################### SPECIFYING THE NUMBER OF ADF LAGS ##########################

# SETTING KMAX

kmax <- ceiling(12*(length(data$cyclical)/100)^(1/4))


############################### ADF TEST ON LEVEL SERIES ########################

for (i in 1:(kmax-1)){

  k = kmax+1-i

  df_data_level = ur.df(data$cyclical, type = "none", lags = k)

  tstat = df_data_level@testreg$coefficients[k+1,3]

  if(abs(tstat) >= 1.6){break}

}

k
```

```
## [1] 13
```

```
# SUMMARY OF ADF TEST

summary(df_data_level)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98034 -0.07340  0.00019  0.10372  1.62979
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## z.lag.1      -0.0003347  0.0002950  -1.134 0.257355
## z.diff.lag1   1.1355985  0.0487062  23.315  < 2e-16 ***
## z.diff.lag2  -0.1640082  0.0763936  -2.147 0.032478 *
## z.diff.lag3  -0.2555920  0.0758312  -3.371 0.000832 ***
## z.diff.lag4   0.2695614  0.0770077   3.500 0.000523 ***
## z.diff.lag5  -0.0754508  0.0783192  -0.963 0.336013
## z.diff.lag6  -0.0997157  0.0784138  -1.272 0.204324
## z.diff.lag7   0.0893750  0.0785810   1.137 0.256152
## z.diff.lag8  -0.0385159  0.0786329  -0.490 0.624564
## z.diff.lag9   0.0362871  0.0785770   0.462 0.644504
## z.diff.lag10 -0.0270013  0.0774449  -0.349 0.727557
## z.diff.lag11  0.2496148  0.0762207   3.275 0.001161 **
## z.diff.lag12  0.2379591  0.0769939   3.091 0.002155 **
## z.diff.lag13 -0.3986258  0.0493993  -8.069 1.09e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2445 on 356 degrees of freedom
## Multiple R-squared:  0.9468, Adjusted R-squared:  0.9447
## F-statistic: 452.9 on 14 and 356 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -1.1345
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
############################## ADF TEST ON DIFF SERIES ########################

for (i in 1:(kmax-1)){

  k = kmax+1-i

  df_data_diff = ur.df(diff(data$cyclical), type = "none", lags = k)

  tstat = df_data_diff@testreg$coefficients[k+1,3]

  if(abs(tstat) >= 1.6){break}

}

k
```

```
## [1] 17
```

```
# SUMMARY OF ADF TEST

summary(df_data_diff)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96596 -0.07214  0.01242  0.10286  1.52686
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## z.lag.1      -0.047575   0.017517  -2.716 0.006939 **
## z.diff.lag1   0.170294   0.054036   3.151 0.001766 **
## z.diff.lag2   0.002473   0.054551   0.045 0.963870
## z.diff.lag3  -0.239218   0.054546  -4.386 1.54e-05 ***
## z.diff.lag4   0.060308   0.056022   1.077 0.282446
## z.diff.lag5  -0.057956   0.056197  -1.031 0.303115
## z.diff.lag6  -0.152703   0.053113  -2.875 0.004289 **
## z.diff.lag7  -0.033949   0.053291  -0.637 0.524507
## z.diff.lag8  -0.082384   0.053023  -1.554 0.121153
## z.diff.lag9  -0.045358   0.053029  -0.855 0.392948
## z.diff.lag10 -0.069176   0.052764  -1.311 0.190708
## z.diff.lag11  0.183902   0.052666   3.492 0.000542 ***
## z.diff.lag12  0.406965   0.052630   7.733 1.15e-13 ***
## z.diff.lag13  0.036841   0.056762   0.649 0.516748
## z.diff.lag14  0.041176   0.056844   0.724 0.469334
## z.diff.lag15  0.036271   0.054604   0.664 0.506968
## z.diff.lag16 -0.077215   0.054603  -1.414 0.158228
## z.diff.lag17  0.094010   0.054383   1.729 0.084759 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2459 on 347 degrees of freedom
## Multiple R-squared:  0.6774, Adjusted R-squared:  0.6607
## F-statistic: 40.48 on 18 and 347 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -2.716
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

b) Using the data from January 1987 to December 2018, identify an estimate an approprite time series model. Make sure to report all relevant estimation

results, plots, statistical tests and information criteria that you are relying on in determining your preferred model. (2 Marks)

As we have already determined the OLS components of our forecast (i.e. none) we first need to confirm our cyclical series is not white noise. If it is, then we cannot use an ARMA process to usefully model its conditional mean.

After conducting the portmanteau Box-Pierce and Ljung-Box tests, we can verify this differenced demeaned data is not white noise. This is because both tests return a very small p-value, rejecting the null hypothesis of both tests that there is no autocorrelation in the time series up to lag m (32). This implies dependence in this time series, suitable for ARMA modelling.

```
############################## PORTMANTEAU TESTS ##################################

m = ceiling((sqrt(length(data$price_level))))

m
```

```
## [1] 20
```

```
Box.test(diff(data$price_level), lag = m, type = "Box-Pierce")
```

```
##
##  Box-Pierce test
##
## data:  diff(data$price_level)
## X-squared = 1576.3, df = 20, p-value < 2.2e-16
```

```
Box.test(diff(data$price_level), lag = m, type = "Ljung-Box")
```

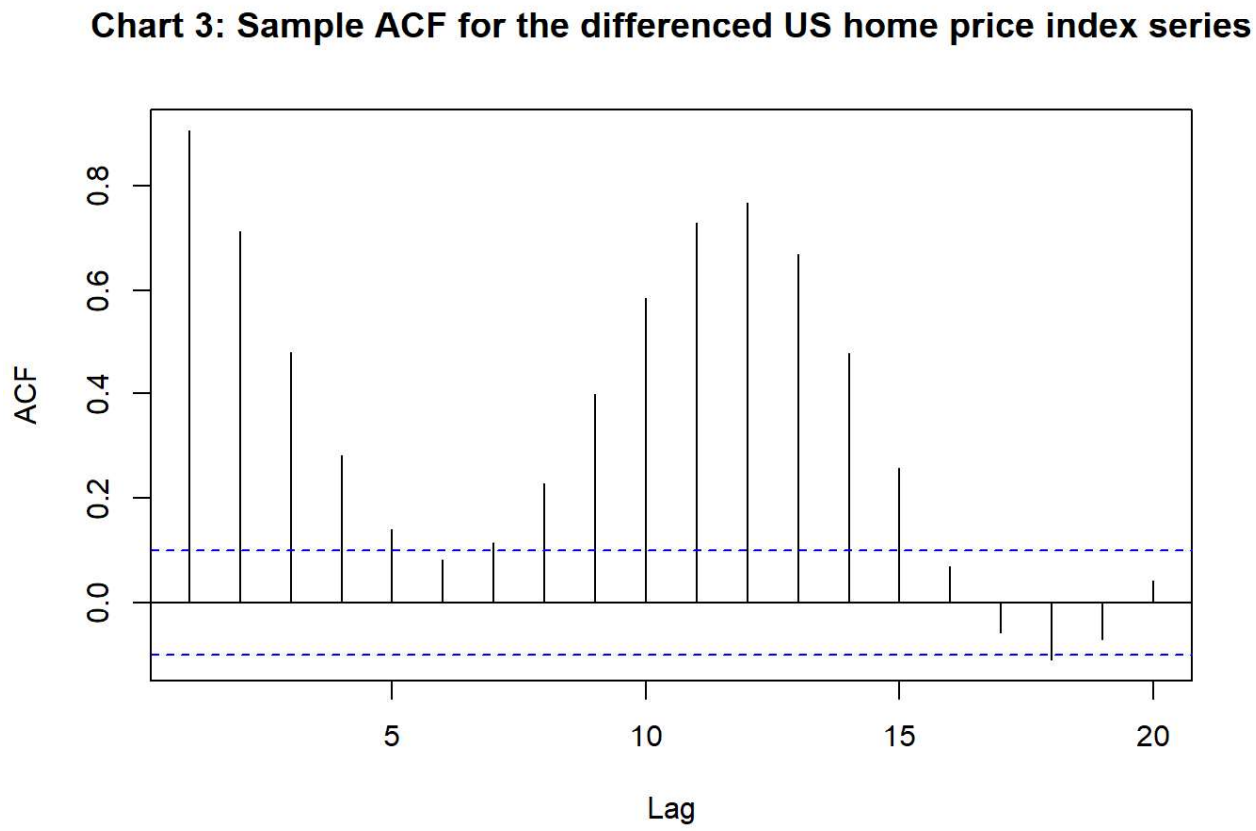
```
##
##  Box-Ljung test
##
## data:  diff(data$price_level)
## X-squared = 1617.4, df = 20, p-value < 2.2e-16
```

We then need to generate the ACF and PACF of our cyclical series to get a sense of their dependence. When doing this, we need to use the differenced series as it is I(1), and it is the differenced data we are creating a time series model for.

- Chart 3 shows the sample autocorrelation function (ACF) for the differenced demeaned US home price index for the 1st to the 20th lag. It shows there are significant lags up to lag 18.
- Chart 4 shows the sample partial autocorrelation function (PACF) for the differenced demeaned US home price index from the 1st to the 20th lag.There are significant lags up until lag 13.

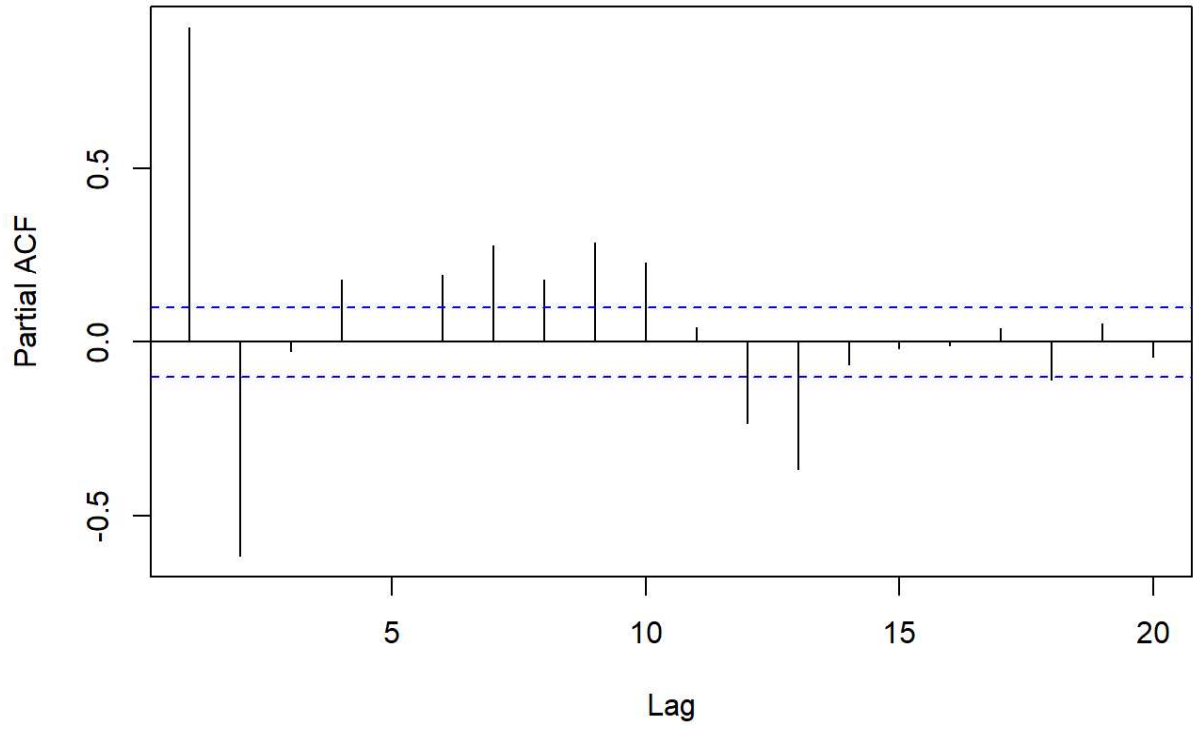
- Based of this information, this tells us that when choosing an ARIMA(p,d,q), we should search for a model with a p parameter up to order 18. We need a minimum value of 1 because we need to account for dependency in our model (ie. we know this isn't a white noise process). We will search for a model with a q parameter up to 4 given it's not advisable we have any more than 4 MA terms in our ARIMA model.

```
############ ANALYSING DEPENDENCE STRUCTURE OF DATA ############################

acf <- acf(diff(data$cyclical), plot = FALSE)
pacf <- pacf(diff(data$cyclical), plot = FALSE)

plot(acf[1:20], main = "Chart 3: Sample ACF for the differenced US home price index series")
```




```
plot(pacf[1:20], main = "Chart 4: Sample PACF for the differenced US home price index serie
s")
```

Chart 4: Sample PACF for the differenced US home price index series



Now we need to generate a range of ARMA(p,d,q) models and choose the preferred model using AIC/BIC.

- Using our observations from above, we will search for models with parameters ranges of AR(1) - AR(15) and MA(0) - MA(4). We need to use at least 1 AR term to account for the persistence we know exists in our conditional mean equation.
- We then use the auto.arima() function to search for the best ARIMA model available to us for this data, using our ARIMA(p,d,q) parameter ranges above to constrain its search. We also complete two calls of auto.arima() to generate a preference from both AIC and BIC.
- AIC and BIC prefer different models: ARIMA(3,1,2) and ARIMA(1,1,2) respectively. Given this conflict, we will take the more parsimonious option given by BIC as our preferred model: ARIMA(1,1,2).
- After generating this preferred model, we do some basic diagnostics on it:
    - Chart 5 indicates the roots of the lag polynominal of this model are all well within the unit circle, suggesting we have accounted for any units roots in the time series effectively.
    - Chart 6 shows the residuals. Unfortunately these do not look like white noise. It may be possible ARCH effects exist in this process.

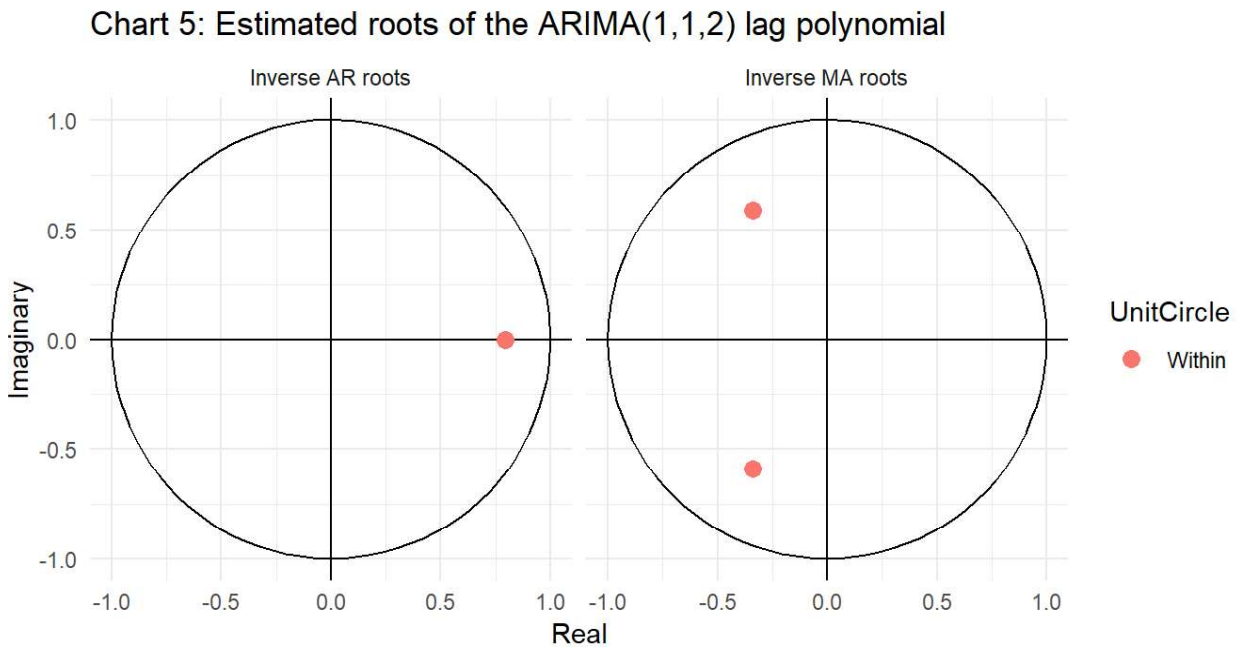
```
############ ANALYSING DEPENDENCE STRUCTURE OF DATA ############################

aic_best_model <- auto.arima(data$price_level,

                         #Parameter values

                         start.p = 1,
                         max.p = 18,
                         max.d = 1,
                         max.q = 4,

                         #Other options
                         stepwise = FALSE, #To ensure an exhaustive search
                         ic = "aic"

                         )

summary(aic_best_model)
```

```
## Series: data$price_level
## ARIMA(3,1,2) with drift
##
## Coefficients:
##          ar1     ar2      ar3     ma1     ma2   drift
##       0.8564  0.1222  -0.2381  0.6221  0.2815  0.3628
## s.e.  0.1437  0.2120   0.1117  0.1410  0.0816  0.1153
##
## sigma^2 = 0.09719:  log likelihood = -95.45
## AIC=204.89   AICc=205.19   BIC=232.53
##
## Training set error measures:
##                         ME      RMSE      MAE          MPE      MAPE      MASE
## Training set 0.0001039785 0.3088984 0.187775 -0.002275276 0.1404094 0.2521955
##                      ACF1
## Training set 0.0006532528
```

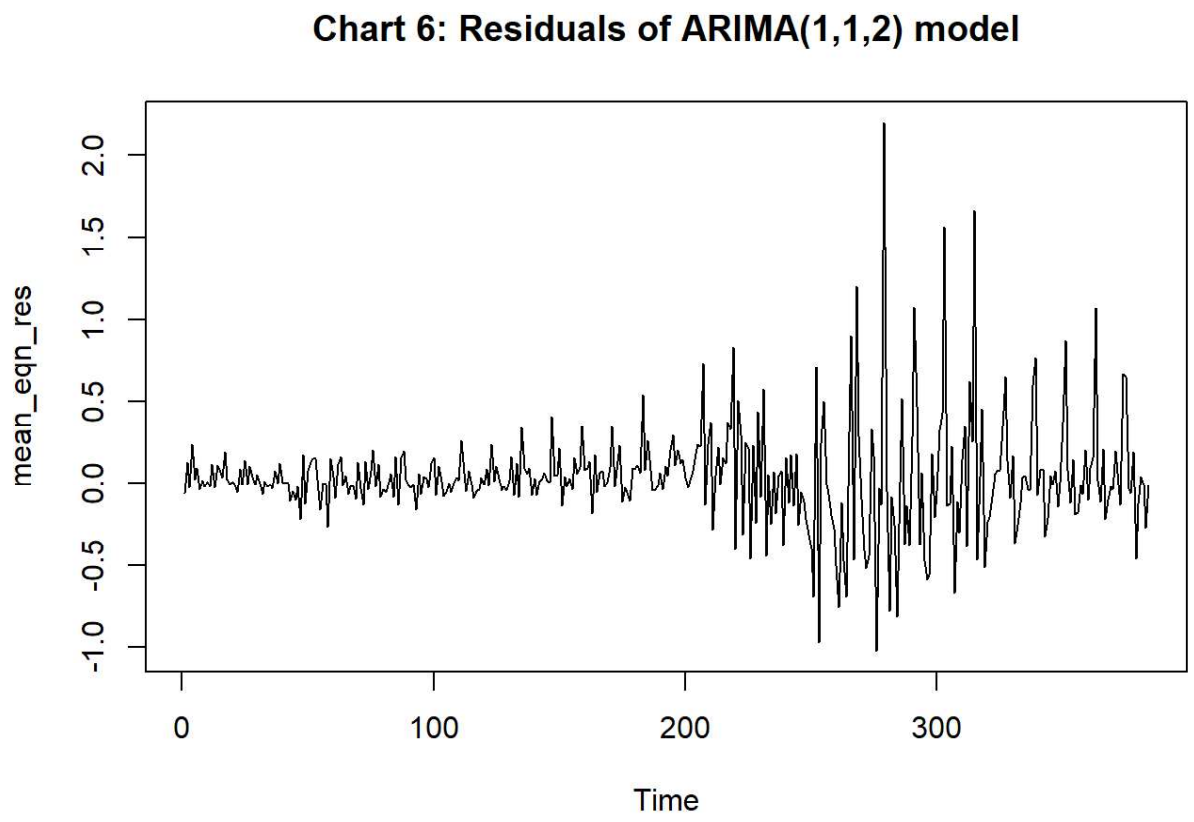
```
bic_best_model <- auto.arima(data$price_level,

                         #Parameter values

                         start.p = 1,
                         max.p = 15,
                         max.d = 1,
                         max.q = 4,

                         #Other options
                         stepwise = FALSE, #To ensure an exhaustive search
                         ic = "bic"

                         )

summary(bic_best_model)
```

```
## Series: data$price_level
## ARIMA(1,1,2)
##
## Coefficients:
##          ar1     ma1     ma2
##       0.7964  0.6760  0.4618
## s.e.  0.0335  0.0481  0.0447
##
## sigma^2 = 0.1005:  log likelihood = -103.37
## AIC=214.73   AICc=214.84   BIC=230.52
##
## Training set error measures:
##                       ME      RMSE       MAE       MPE      MAPE      MASE
## Training set 0.03431676 0.3153293 0.1903392 0.0306196 0.1424586 0.2556394
##                     ACF1
## Training set 0.03043132
```

```
# AIC chooses ARIMA(3,1,2) whereas BIC chooses ARIMA(1,1,2). We will use the more parsimoniou
s model, as specified by BIC.

mean_eqn <- Arima(data$price_level,
                  order = c(1,1,2),
                  include.mean = FALSE,
                  method = "CSS-ML",
                  xreg = data$mean)



####################### TESTING GENERAL MODEL DIAGNOSTICS ######################

autoplot(mean_eqn, main = "Chart 5: Estimated roots of the ARIMA(1,1,2) lag polynomial")
```



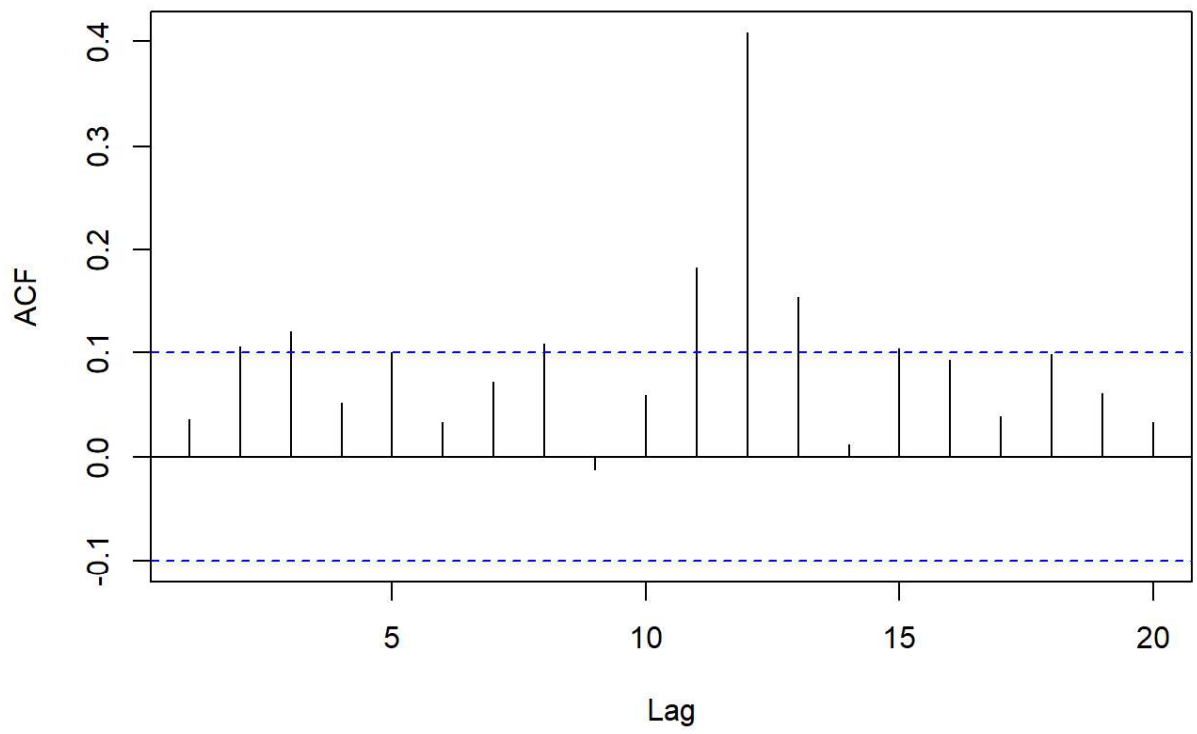

```
mean_eqn_res <- mean_eqn$resid

plot(mean_eqn_res, main = "Chart 6: Residuals of ARIMA(1,1,2) model")
```



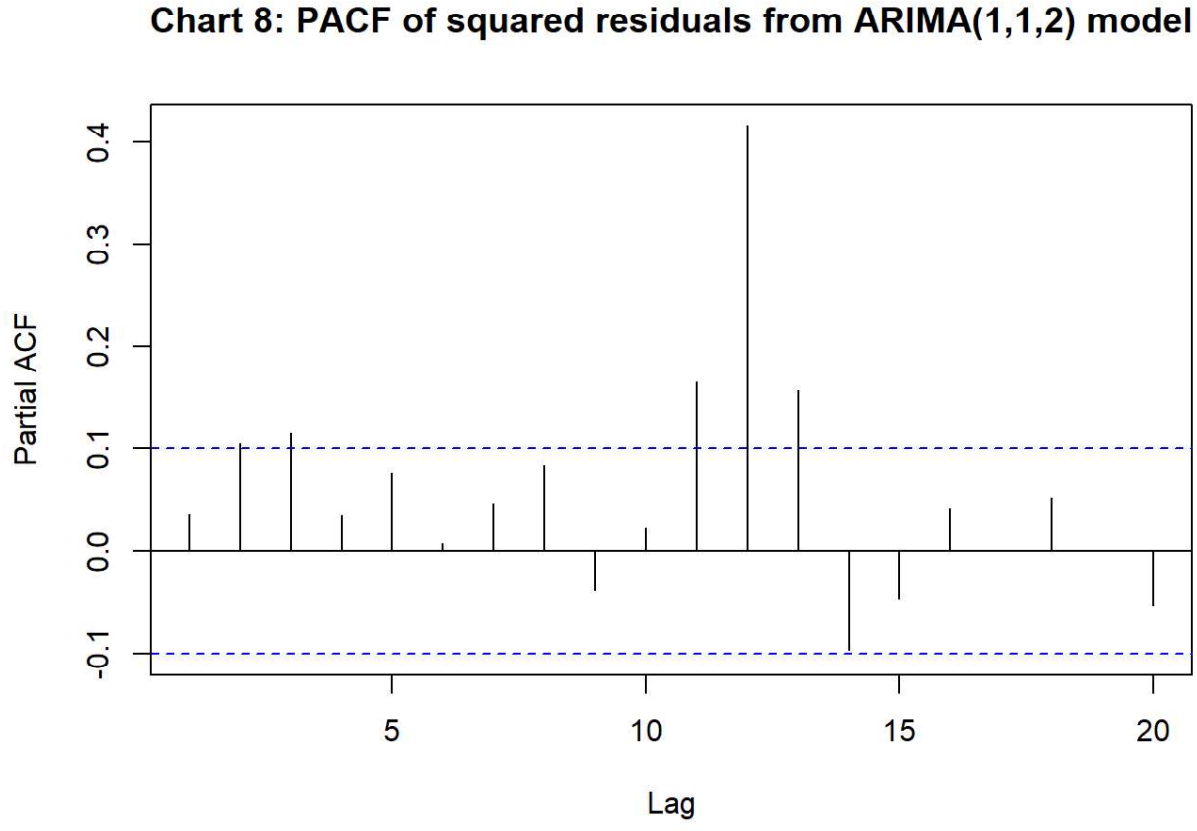
Given this observation of the residuals, we now need to test for ARCH effects in these residuals to assess if its appropriate to create a model for the conditional variance of this series.

- Chart 7 and Chart 8 show the ACF and PACF of the squared residuals from our conditional mean equation respectively. They show the presence of several signficant lags, suggesting ARCH effects are present.
- This is confirmed by our portmanteau tests on these squared residuals. Their very small p-values means we reject the null hypothesis that these squared residuals are white noise, telling us there is dependence in them.
- This means these residuals are suitable for ARCH modelling.

```
############################## TESTING FOR ARCH EFFECTS ##########################

mean_eqn_res_sq <- mean_eqn_res^2

acf.res <- acf(mean_eqn_res_sq, plot = FALSE)
pacf.res <- pacf(mean_eqn_res_sq, plot = FALSE)

plot(acf.res[1:20], main = "Chart 7: ACF of squared residuals from ARIMA(1,1,2) model")
```




```
plot(pacf.res[1:20], main = "Chart 8: PACF of squared residuals from ARIMA(1,1,2) model")
```



```
# There are many significant lags in the ACF and PACF, therefore adding an ARCH variance to t
his model would likely improve model fit.

# This is confirmed by our portmanteau tests below, which tell us the squared residuals are i
ndeed not white noise.


Box.test(mean_eqn_res_sq, type = "Box-Pierce", l = m)
```

```
##
##  Box-Pierce test
##
## data:  mean_eqn_res_sq
## X-squared = 123.5, df = 20, p-value < 2.2e-16
```

```
Box.test(mean_eqn_res_sq, type = "Ljung-Box", l = m)
```

```
##
##  Box-Ljung test
##
## data:  mean_eqn_res_sq
## X-squared = 127.86, df = 20, p-value < 2.2e-16
```

Now we need to estimate our ARCH model:

- From Chart 7, we can see there are significant ACF lags in the squared residuals up to lag 15. Therefore, we will test for models with ARCH terms up to order 15. Please note, estimating all of these models added dozens of pages to the printout for my submission. I have removed most of them from this pdf for brevity.
- Using AIC/BIC we determine our preferred model, where both measures give different results: ARCH(13) and ARCH(12) respectively. We will go with the more parsimonious ARCH(12) from the BIC.
- After estimating this model, we assess if our estimation is sufficient. We conclude it is because:
- The residuals now look much more like white noise (Chart 9)
- Our portmanteau tests both return large p-values (both > 0.05), meaning we cannot reject the null hypothesis the standardised squared residuals from this ARCH(12) model are white noise. This confirms the ARCH effects have been dealt with in this model.
- However, for completeness, we check if a more parsimonious GARCH(1,1) sufficiently arrives at the same solution. We don't find this is the case, as the portmanteau tests on those standardised square residuals both return very small p-values. This means we reject the null hypothesis that this series is white noise, and conclude it has not dealt effectively with the ARCH effects present in the squared residuals. Therefore, we stay with our ARCH(13) model.

Ultimately, after all this analysis, we settle on an ARIMA(1,1,2)-ARCH(12) model as our preferred model for forecasting the cyclical components of our time series.

```
############################ ESTIMATING ARCH MODELS ##############################

z = 15

prefix.arch <- "arch"
suffix.arch <- seq(1,z)

arch.label <- paste(prefix.arch, suffix.arch, sep = "" )

arch.info <- data.frame(row.names = arch.label, matrix(NA,z,2))

colnames(arch.info) <- c("aic","bic")

T <- length(mean_eqn_res)

for(i in 1:z){
archmod <- garch(mean_eqn_res, order = c(0,i))
arch.info[i,1] <- AIC(archmod)
arch.info[i,2] <- AIC(archmod, k = log(T-i-1))
}
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     9.360644e-02     1.000e+00
##      2     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1 -2.539e+02
##      1    4 -2.547e+02  3.03e-03  3.92e-03  6.3e-02  6.0e+03  1.3e-02  1.18e+01
##      2    5 -2.554e+02  2.70e-03  2.95e-03  6.4e-02  2.0e+00  1.3e-02  7.15e+01
##      3    6 -2.571e+02  6.64e-03  6.95e-03  1.3e-01  2.0e+00  2.6e-02  6.25e+01
##      4    9 -2.744e+02  6.30e-02  9.26e-02  6.6e-01  2.0e+00  3.8e-01  4.12e+01
##      5   13 -2.754e+02  3.60e-03  8.73e-03  8.8e-03  1.2e+01  8.5e-03  5.07e-01
##      6   17 -2.871e+02  4.09e-02  5.13e-02  3.5e-01  8.5e-01  5.1e-01  9.49e-02
##      7   18 -2.922e+02  1.74e-02  1.14e-02  1.5e-01  0.0e+00  3.5e-01  1.14e-02
##      8   19 -2.958e+02  1.22e-02  8.80e-03  1.4e-01  0.0e+00  4.3e-01  8.80e-03
##      9   20 -2.967e+02  3.06e-03  3.72e-03  1.1e-01  2.5e-03  4.3e-01  3.72e-03
##     10   21 -2.972e+02  1.55e-03  4.25e-03  6.3e-02  0.0e+00  3.0e-01  4.25e-03
##     11   22 -2.975e+02  8.83e-04  7.32e-04  1.1e-02  0.0e+00  5.7e-02  7.32e-04
##     12   23 -2.975e+02  2.51e-05  2.87e-05  1.3e-03  0.0e+00  6.5e-03  2.87e-05
##     13   24 -2.975e+02  4.63e-07  4.73e-07  2.0e-04  0.0e+00  9.7e-04  4.73e-07
##     14   25 -2.975e+02  3.42e-10  3.22e-10  1.3e-05  0.0e+00  6.5e-05  3.22e-10
##     15   26 -2.975e+02  4.74e-12  4.31e-12  3.8e-06  0.0e+00  1.9e-05  4.31e-12
##
##  ***** RELATIVE FUNCTION CONVERGENCE *****
##
##  FUNCTION    -2.974818e+02   RELDX        3.828e-06
##  FUNC. EVALS      26         GRAD. EVALS      16
##  PRELDF       4.309e-12      NPRELDF      4.309e-12
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    2.272723e-02     1.000e+00    -8.318e-04
##      2    2.438815e+00     1.000e+00    -2.044e-05
```

```
## Warning in sqrt(pred$e): NaNs produced
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     8.867979e-02     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1 -2.598e+02
##      1    4 -2.611e+02  4.71e-03  6.39e-03  4.7e-02  1.7e+04  1.0e-02  5.33e+01
##      2    7 -2.685e+02  2.78e-02  3.55e-02  4.2e-01  3.1e+00  1.0e-01  5.78e+01
##      3    9 -2.708e+02  8.33e-03  1.51e-02  7.3e-02  2.6e+00  2.0e-02  2.27e+01
##      4   10 -2.724e+02  6.00e-03  7.26e-03  5.9e-02  2.0e+00  2.0e-02  9.48e+00
##      5   12 -2.783e+02  2.11e-02  2.72e-02  2.6e-01  2.0e+00  1.0e-01  8.80e+00
##      6   13 -2.811e+02  9.79e-03  1.83e-02  1.8e-01  2.0e+00  1.0e-01  8.43e-01
##      7   14 -2.836e+02  9.00e-03  1.19e-02  1.3e-01  1.9e+00  1.0e-01  3.47e-01
##      8   15 -2.860e+02  8.34e-03  1.33e-02  1.2e-01  1.9e+00  1.0e-01  3.41e-01
##      9   18 -2.861e+02  4.26e-04  1.30e-03  4.0e-03  1.1e+01  3.5e-03  1.90e-01
##     10   19 -2.862e+02  3.90e-04  3.76e-04  3.7e-03  2.0e+00  3.5e-03  1.36e-01
##     11   20 -2.864e+02  5.37e-04  5.45e-04  8.0e-03  2.0e+00  7.0e-03  1.34e-01
##     12   24 -2.949e+02  2.88e-02  4.71e-02  4.2e-01  9.9e-01  6.2e-01  1.24e-01
##     13   25 -3.007e+02  1.92e-02  1.54e-02  1.2e-01  0.0e+00  2.9e-01  1.54e-02
##     14   27 -3.044e+02  1.22e-02  1.22e-02  2.1e-01  0.0e+00  7.2e-01  1.22e-02
##     15   28 -3.058e+02  4.48e-03  5.36e-03  1.5e-02  0.0e+00  7.7e-02  5.36e-03
##     16   30 -3.062e+02  1.33e-03  9.90e-04  4.9e-02  0.0e+00  2.2e-01  9.90e-04
##     17   31 -3.063e+02  2.35e-04  2.44e-04  2.5e-02  0.0e+00  1.2e-01  2.44e-04
##     18   32 -3.063e+02  4.12e-06  4.05e-06  5.4e-04  0.0e+00  3.3e-03  4.05e-06
##     19   33 -3.063e+02  2.16e-07  2.15e-07  3.0e-04  0.0e+00  1.5e-03  2.15e-07
##     20   34 -3.063e+02  5.25e-09  5.69e-09  1.4e-04  0.0e+00  6.8e-04  5.69e-09
##     21   35 -3.063e+02  3.36e-11  3.36e-11  9.9e-06  0.0e+00  4.9e-05  3.36e-11
##
##  ***** RELATIVE FUNCTION CONVERGENCE *****
##
##  FUNCTION    -3.062615e+02   RELDX        9.943e-06
##  FUNC. EVALS      35         GRAD. EVALS      22
##  PRELDF       3.355e-11      NPRELDF      3.355e-11
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    1.669451e-02     1.000e+00     3.803e-05
##      2    2.448084e+00     1.000e+00     9.047e-07
##      3    9.179436e-02     1.000e+00     1.851e-05
```

```
## Warning in sqrt(pred$e): NaNs produced
```

```
##     55  120 -4.061e+02 -2.46e+07  4.90e-15  1.9e-14  2.0e+00  6.8e-15 -4.58e-01
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION    -4.061245e+02   RELDX        1.858e-14
##  FUNC. EVALS     120         GRAD. EVALS      55
##  PRELDF       4.897e-15      NPRELDF     -4.576e-01
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    3.587529e-03     1.000e+00    -7.761e+01
##      2    6.908477e-02     1.000e+00    -5.470e+01
##      3    6.342174e-02     1.000e+00    -3.703e+01
##      4    4.798667e-02     1.000e+00    -9.993e+00
##      5    4.470904e-02     1.000e+00    -4.129e+00
##      6    2.589806e-02     1.000e+00     1.065e+01
##      7    7.137878e-16     1.000e+00     1.011e+02
##      8    3.358726e-03     1.000e+00     7.490e+01
##      9    4.235512e-02     1.000e+00    -1.469e+01
##     10    1.381420e-02     1.000e+00     3.112e+01
##     11    3.941420e-02     1.000e+00    -3.898e+00
##     12    8.220428e-02     1.000e+00    -7.161e+01
##     13    1.382064e-01     1.000e+00    -2.234e+02
##     14    5.701691e-02     1.000e+00    -3.435e+01
##     15    1.775536e-02     1.000e+00     5.171e+01
##     16    3.405055e-02     1.000e+00     1.503e+01
```

```
akaike.choice <- rownames(arch.info)[which.min(arch.info[,1])]
bayes.choice <- rownames(arch.info)[which.min(arch.info[,2])]

akaike.choice
```

```
## [1] "arch13"
```

```
bayes.choice
```

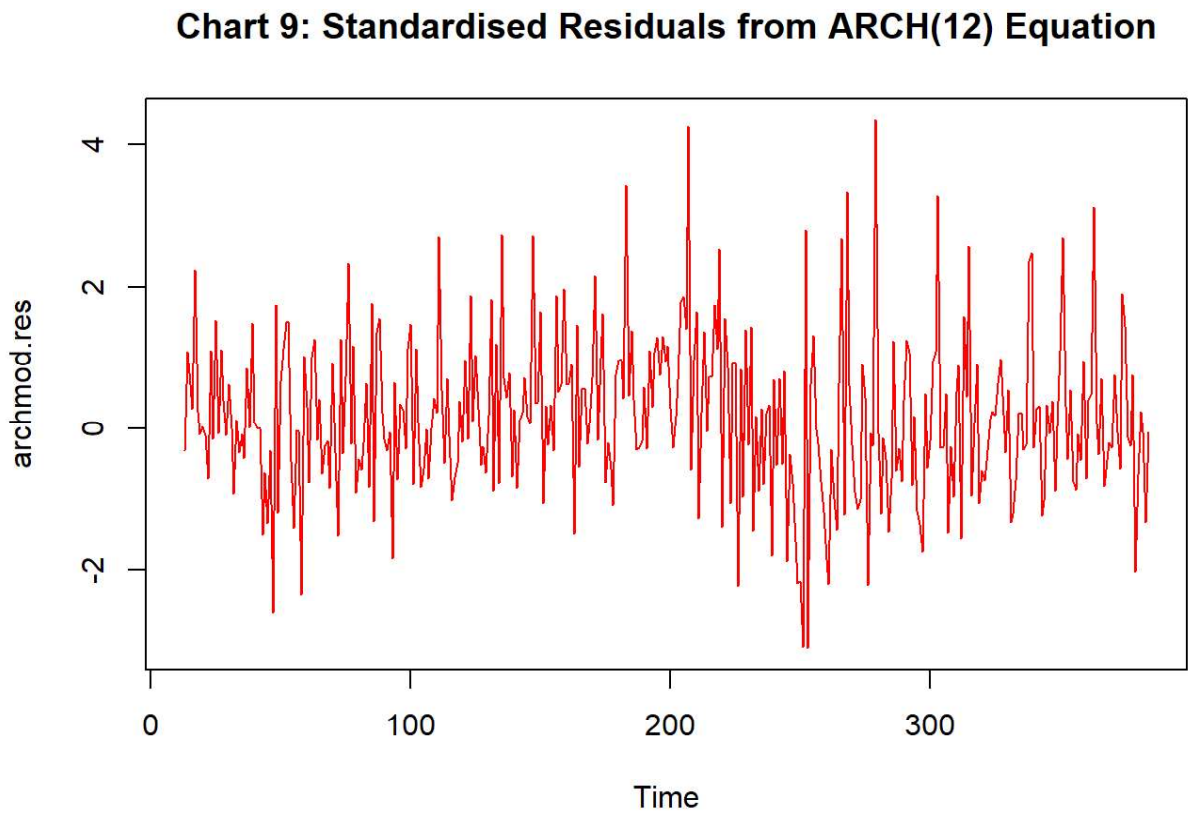
```
## [1] "arch12"
```

```
########################## REXAMINING RESIDUALS ################################


archmod <- garch(mean_eqn_res, order = c(0,12))
```

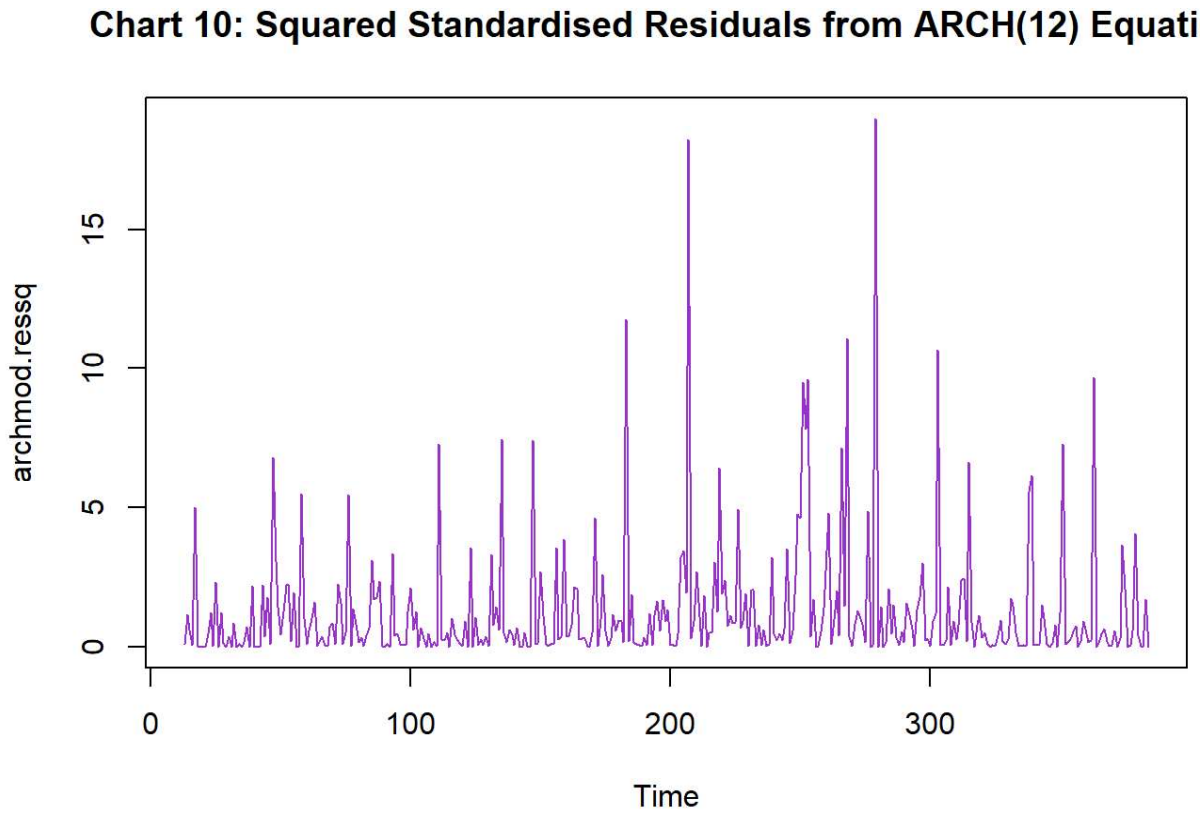
```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     3.941324e-02     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##      4     5.000000e-02     1.000e+00
##      5     5.000000e-02     1.000e+00
##      6     5.000000e-02     1.000e+00
##      7     5.000000e-02     1.000e+00
##      8     5.000000e-02     1.000e+00
##      9     5.000000e-02     1.000e+00
##     10     5.000000e-02     1.000e+00
##     11     5.000000e-02     1.000e+00
##     12     5.000000e-02     1.000e+00
##     13     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1 -3.271e+02
##      1    4 -3.401e+02  3.81e-02  3.69e-02  9.5e-02  1.3e+05  1.0e-02  2.32e+03
##      2    6 -3.649e+02  6.80e-02  7.31e-02  1.8e-01  2.5e+05  2.0e-02  9.53e+04
##      3    8 -3.663e+02  4.05e-03  4.47e-03  1.3e-02  4.7e+01  2.0e-03  7.57e+02
##      4   10 -3.686e+02  6.25e-03  6.66e-03  2.7e-02  2.6e+00  4.0e-03  2.67e+02
##      5   11 -3.720e+02  8.89e-03  1.07e-02  5.4e-02  3.1e+00  8.0e-03  2.07e+02
##      6   12 -3.774e+02  1.45e-02  2.30e-02  8.2e-02  2.0e+00  1.6e-02  1.25e+02
##      7   13 -3.845e+02  1.83e-02  2.82e-02  5.1e-02  2.0e+00  1.6e-02  5.86e+01
##      8   16 -4.020e+02  4.36e-02  4.56e-02  1.5e-01  2.0e+00  6.4e-02  4.35e+01
##      9   18 -4.027e+02  1.66e-03  9.68e-03  4.5e-02  3.6e+00  1.3e-02  5.73e+01
##     10   20 -4.045e+02  4.39e-03  4.08e-03  5.0e-03  4.4e+00  1.3e-03  6.42e+01
##     11   21 -4.065e+02  4.91e-03  6.19e-03  1.0e-02  3.4e+00  2.6e-03  7.11e+01
##     12   23 -4.081e+02  4.13e-03  6.72e-03  3.3e-02  3.9e+00  1.0e-02  6.03e+01
##     13   25 -4.094e+02  3.17e-03  3.75e-03  3.8e-03  2.8e+00  1.0e-03  4.96e+01
##     14   26 -4.098e+02  8.44e-04  7.15e-04  2.2e-03  2.0e+00  1.0e-03  5.05e+01
##     15   28 -4.106e+02  1.96e-03  1.70e-03  1.2e-02  2.1e+00  4.1e-03  5.03e+01
##     16   30 -4.110e+02  1.02e-03  9.89e-04  2.8e-03  2.6e+00  8.2e-04  4.86e+01
##     17   31 -4.112e+02  5.30e-04  7.53e-04  4.1e-03  2.2e+00  1.6e-03  4.96e+01
##     18   32 -4.120e+02  1.81e-03  1.86e-03  8.2e-03  2.0e+00  3.3e-03  4.87e+01
##     19   34 -4.122e+02  4.16e-04  3.98e-04  1.7e-03  3.4e+00  6.6e-04  5.53e+01
##     20   36 -4.122e+02  9.42e-05  9.44e-05  3.6e-04  2.1e+01  1.3e-04  6.01e+01
##     21   38 -4.123e+02  1.81e-04  1.81e-04  7.5e-04  6.4e+00  2.6e-04  6.27e+01
##     22   41 -4.123e+02  3.59e-06  3.59e-06  1.5e-05  2.1e+04  5.2e-06  6.40e+01
##     23   43 -4.123e+02  7.19e-06  7.19e-06  3.0e-05  2.0e+03  1.0e-05  1.17e+02
##     24   45 -4.123e+02  1.44e-06  1.44e-06  6.1e-06  4.2e+04  2.1e-06  1.18e+02
##     25   47 -4.123e+02  2.87e-06  2.87e-06  1.2e-05  5.3e+03  4.2e-06  1.19e+02
##     26   49 -4.123e+02  5.75e-07  5.75e-07  2.4e-06  1.1e+05  8.4e-07  1.20e+02
##     27   51 -4.123e+02  1.15e-07  1.15e-07  4.9e-07  5.4e+05  1.7e-07  1.21e+02
##     28   53 -4.123e+02  2.30e-07  2.30e-07  9.7e-07  6.7e+04  3.4e-07  1.21e+02
##     29   55 -4.123e+02  4.60e-08  4.60e-08  1.9e-07  1.3e+06  6.7e-08  1.21e+02
##     30   57 -4.123e+02  9.20e-08  9.20e-08  3.9e-07  1.7e+05  1.3e-07  1.21e+02
##     31   59 -4.123e+02  1.84e-08  1.84e-08  7.8e-08  3.4e+06  2.7e-08  1.21e+02
##     32   61 -4.123e+02  3.68e-08  3.68e-08  1.6e-07  4.2e+05  5.4e-08  1.21e+02
##     33   64 -4.123e+02  7.36e-10  7.36e-10  3.1e-09  8.4e+07  1.1e-09  1.21e+02
```

```
##     34   67 -4.123e+02  5.89e-09  5.89e-09  2.5e-08  2.6e+06  8.6e-09  1.21e+02
##     35   70 -4.123e+02  1.18e-10  1.18e-10  5.0e-10  5.3e+08  1.7e-10  1.21e+02
##     36   72 -4.123e+02  2.35e-10  2.35e-10  1.0e-09  6.6e+07  3.4e-10  1.21e+02
##     37   74 -4.123e+02  4.71e-11  4.71e-11  2.0e-10  2.0e+00  6.9e-11 -4.86e-01
##     38   76 -4.123e+02  9.42e-11  9.42e-11  4.0e-10  2.0e+00  1.4e-10 -4.86e-01
##     39   78 -4.123e+02  1.88e-11  1.88e-11  8.0e-11  2.0e+00  2.7e-11 -4.86e-01
##     40   80 -4.123e+02  3.77e-11  3.77e-11  1.6e-10  2.0e+00  5.5e-11 -4.86e-01
##     41   82 -4.123e+02  7.54e-12  7.53e-12  3.2e-11  2.0e+00  1.1e-11 -4.86e-01
##     42   84 -4.123e+02  1.51e-11  1.51e-11  6.4e-11  2.0e+00  2.2e-11 -4.86e-01
##     43   86 -4.123e+02  3.01e-11  3.01e-11  1.3e-10  2.0e+00  4.4e-11 -4.86e-01
##     44   88 -4.123e+02  6.03e-12  6.03e-12  2.6e-11  2.0e+00  8.8e-12 -4.86e-01
##     45   90 -4.123e+02  1.21e-12  1.21e-12  5.1e-12  2.0e+00  1.8e-12 -4.86e-01
##     46   92 -4.123e+02  2.41e-12  2.41e-12  1.0e-11  2.0e+00  3.5e-12 -4.86e-01
##     47   94 -4.123e+02  4.81e-13  4.82e-13  2.0e-12  2.0e+00  7.0e-13 -4.86e-01
##     48   96 -4.123e+02  9.65e-13  9.64e-13  4.1e-12  2.0e+00  1.4e-12 -4.86e-01
##     49   98 -4.123e+02  1.93e-12  1.93e-12  8.2e-12  2.0e+00  2.8e-12 -4.86e-01
##     50  100 -4.123e+02  3.86e-13  3.86e-13  1.6e-12  2.0e+00  5.6e-13 -4.86e-01
##     51  102 -4.123e+02  7.69e-14  7.72e-14  3.3e-13  2.0e+00  1.1e-13 -4.86e-01
##     52  104 -4.123e+02  1.55e-13  1.54e-13  6.5e-13  2.0e+00  2.3e-13 -4.86e-01
##     53  106 -4.123e+02  2.96e-14  3.09e-14  1.3e-13  2.0e+00  4.5e-14 -4.86e-01
##     54  108 -4.123e+02  6.38e-14  6.17e-14  2.6e-13  2.0e+00  9.0e-14 -4.86e-01
##     55  110 -4.123e+02  1.16e-14  1.23e-14  5.2e-14  2.0e+00  1.8e-14 -4.86e-01
##     56  112 -4.123e+02  2.43e-14  2.47e-14  1.0e-13  2.0e+00  3.6e-14 -4.86e-01
##     57  114 -4.123e+02  5.79e-15  4.94e-15  2.1e-14  2.0e+00  7.2e-15 -4.85e-01
##     58  117 -4.123e+02  3.82e-14  3.95e-14  1.7e-13  2.0e+00  5.8e-14 -4.86e-01
##     59  120 -4.123e+02  5.52e-16  7.90e-16  3.4e-15  2.0e+00  1.2e-15 -4.86e-01
##     60  122 -4.123e+02  2.48e-15  1.58e-15  6.7e-15  2.0e+00  2.3e-15 -4.86e-01
##     61  123 -4.123e+02 -2.43e+07  3.16e-15  1.3e-14  2.0e+00  4.6e-15 -4.85e-01
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION    -4.122720e+02   RELDX        1.341e-14
##  FUNC. EVALS     123         GRAD. EVALS      61
##  PRELDF       3.160e-15      NPRELDF     -4.852e-01
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    4.395211e-03     1.000e+00    -6.236e+00
##      2    6.896346e-02     1.000e+00    -5.874e+01
##      3    7.339746e-02     1.000e+00    -5.004e+01
##      4    6.155829e-02     1.000e+00    -2.444e+01
##      5    4.571945e-02     1.000e+00    -1.040e+01
##      6    3.432144e-02     1.000e+00    -7.553e+00
##      7    8.435738e-16     1.000e+00     9.401e+01
##      8    7.110923e-03     1.000e+00     5.999e+01
##      9    4.343837e-02     1.000e+00    -2.364e+01
##     10    3.424507e-03     1.000e+00     2.154e+01
##     11    3.759013e-02     1.000e+00    -9.032e+00
##     12    9.470333e-02     1.000e+00    -7.897e+01
##     13    1.406921e-01     1.000e+00    -2.308e+02
```

```
archmod.res <- archmod$residuals

archmod.res <- na.omit(archmod.res)

plot(archmod.res, col = "red", main = "Chart 9: Standardised Residuals from ARCH(12) Equatio
n", type = "l")
```




```
archmod.ressq <- archmod.res^2

plot(archmod.ressq, col = "darkorchid", main = "Chart 10: Squared Standardised Residuals from
ARCH(12) Equation", type = "l")
```




```
######################### VERIFYING IF ARCH EFFECTS ARE GONE ####################

Box.test(archmod.ressq, lag = m, type = "Box-Pierce")
```

```
##
##  Box-Pierce test
##
## data:  archmod.ressq
## X-squared = 28.859, df = 20, p-value = 0.09058
```

```
Box.test(archmod.ressq, lag = m, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  archmod.ressq
## X-squared = 29.887, df = 20, p-value = 0.0717
```

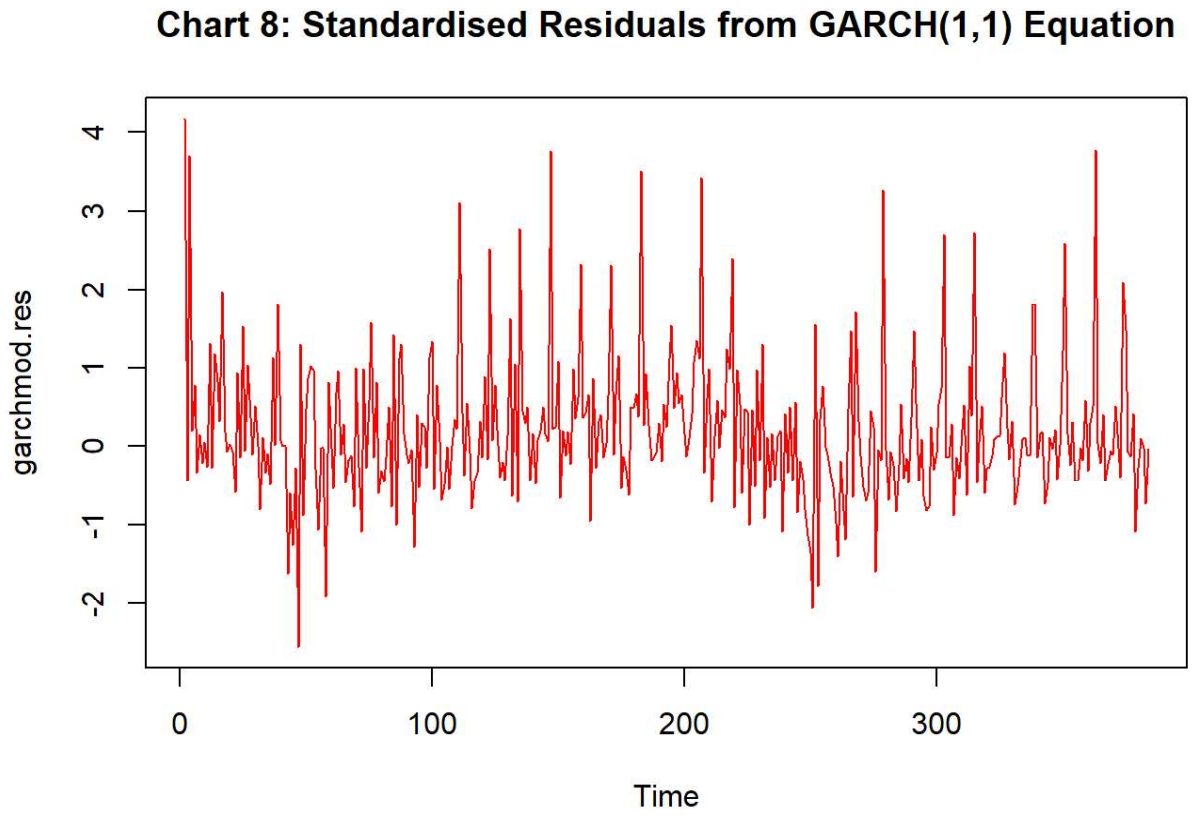
```
############ TESTING IF A GARCH(1,1) MODEL GIVES SIMILAR RESULTS #################

garchmod <- garch(mean_eqn_res, order = c(1,1))
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     8.867979e-02     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1 -2.546e+02
##      1    4 -2.554e+02  2.97e-03  3.74e-03  4.8e-02  9.5e+03  1.0e-02  1.78e+01
##      2    5 -2.560e+02  2.63e-03  2.69e-03  5.2e-02  2.5e+00  1.0e-02  8.27e+01
##      3    6 -2.574e+02  5.44e-03  5.62e-03  1.0e-01  2.0e+00  2.0e-02  7.98e+01
##      4    9 -2.772e+02  7.13e-02  1.40e-01  7.2e-01  2.0e+00  4.5e-01  6.72e+01
##      5   11 -2.888e+02  4.00e-02  2.25e-01  3.4e-02  2.0e+00  4.5e-02  4.49e+00
##      6   14 -3.224e+02  1.04e-01  6.86e-02  1.6e-01  2.0e+00  1.8e-01  1.95e+01
##      7   16 -3.361e+02  4.08e-02  3.24e-02  3.0e-02  2.0e+00  3.6e-02  2.67e+03
##      8   18 -3.362e+02  1.79e-04  1.87e-02  6.3e-03  3.4e+00  7.3e-03  1.42e+03
##      9   19 -3.388e+02  7.79e-03  1.38e-02  3.1e-03  2.3e+00  3.6e-03  3.68e+02
##     10   23 -3.487e+02  2.83e-02  5.19e-02  1.3e-01  2.0e+00  1.6e-01  4.66e+02
##     11   27 -3.487e+02  1.37e-04  2.32e-04  2.0e-04  2.6e+02  2.4e-04  5.23e-02
##     12   28 -3.487e+02  1.03e-05  1.21e-05  1.6e-04  2.0e+00  2.4e-04  1.85e-02
##     13   35 -3.536e+02  1.36e-02  9.46e-03  3.2e-01  0.0e+00  5.0e-01  1.81e-02
##     14   37 -3.542e+02  1.73e-03  9.28e-02  1.0e-01  1.4e+00  1.7e-01  9.77e-02
##     15   39 -3.588e+02  1.30e-02  1.23e-02  4.8e-02  1.2e-02  8.4e-02  2.71e-02
##     16   41 -3.649e+02  1.67e-02  1.65e-02  3.9e-02  1.0e+00  8.4e-02  7.08e-02
##     17   46 -3.650e+02  1.30e-04  1.42e-04  9.7e-06  6.8e+01  1.7e-05  3.14e-02
##     18   48 -3.651e+02  1.91e-04  1.99e-04  1.9e-05  2.2e+00  3.4e-05  2.62e-02
##     19   50 -3.651e+02  2.65e-05  2.77e-05  3.8e-06  8.9e+00  6.7e-06  2.54e-02
##     20   52 -3.651e+02  7.00e-05  7.78e-05  1.9e-05  2.3e+00  3.3e-05  2.51e-02
##     21   55 -3.651e+02  5.14e-07  5.79e-07  2.4e-07  3.8e+01  6.7e-07  2.48e-02
##     22   57 -3.651e+02  9.88e-07  1.11e-06  5.0e-07  6.0e+00  1.3e-06  2.48e-02
##     23   59 -3.651e+02  1.94e-07  2.16e-07  1.0e-07  1.1e+02  2.7e-07  2.48e-02
##     24   61 -3.651e+02  3.84e-07  4.26e-07  2.1e-07  1.6e+01  5.3e-07  2.48e-02
##     25   63 -3.651e+02  7.64e-08  8.43e-08  4.3e-08  3.3e+02  1.1e-07  2.48e-02
##     26   66 -3.651e+02  1.53e-09  1.68e-09  8.6e-10  1.7e+04  2.1e-09  2.48e-02
##     27   68 -3.651e+02  3.06e-09  3.37e-09  1.7e-09  2.1e+03  4.3e-09  2.48e-02
##     28   70 -3.651e+02  6.11e-10  6.73e-10  3.4e-10  4.1e+04  8.5e-10  2.48e-02
##     29   72 -3.651e+02  1.22e-09  1.35e-09  6.9e-10  5.2e+03  1.7e-09  2.48e-02
##     30   74 -3.651e+02  2.44e-09  2.69e-09  1.4e-09  2.6e+03  3.4e-09  2.48e-02
##     31   76 -3.651e+02  4.89e-10  5.39e-10  2.8e-10  5.2e+04  6.8e-10  2.48e-02
##     32   78 -3.651e+02  9.77e-11  1.08e-10  5.5e-11  2.6e+05  1.4e-10  2.48e-02
##     33   80 -3.651e+02  1.95e-10  2.15e-10  1.1e-10  3.2e+04  2.7e-10  2.48e-02
##     34   82 -3.651e+02  3.91e-10  4.31e-10  2.2e-10  1.6e+04  5.5e-10  2.48e-02
##     35   85 -3.651e+02  7.82e-12  8.62e-12  4.4e-12  3.2e+06  1.1e-11  2.48e-02
##     36   87 -3.651e+02  1.56e-11  1.72e-11  8.8e-12  4.0e+05  2.2e-11  2.48e-02
##     37   89 -3.651e+02  3.13e-12  3.45e-12  1.8e-12  8.1e+06  4.4e-12  2.48e-02
##     38   91 -3.651e+02  6.26e-12  6.89e-12  3.5e-12  1.0e+06  8.7e-12  2.48e-02
##     39   93 -3.651e+02  1.25e-12  1.38e-12  7.1e-13  2.0e+07  1.7e-12  2.48e-02
##     40   95 -3.651e+02  2.50e-12  2.76e-12  1.4e-12  2.5e+06  3.5e-12  2.48e-02
##     41   97 -3.651e+02  4.99e-13  5.52e-13  2.8e-13  5.1e+07  7.0e-13  2.48e-02
##     42   99 -3.651e+02  1.00e-12  1.10e-12  5.6e-13  6.3e+06  1.4e-12  2.48e-02
##     43  101 -3.651e+02  2.00e-12  2.21e-12  1.1e-12  3.2e+06  2.8e-12  2.48e-02
```

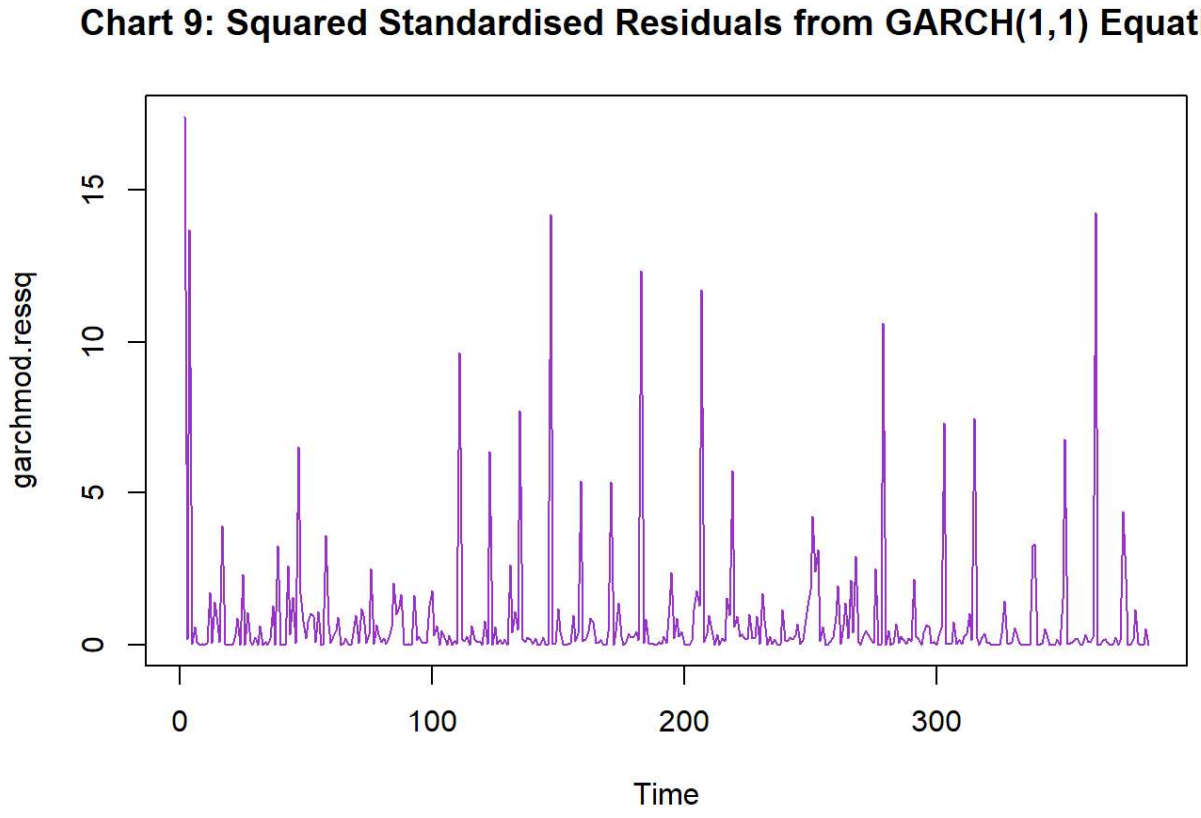
```
##     44  104 -3.651e+02  3.88e-14  4.41e-14  2.3e-14  6.3e+08  5.6e-14  2.48e-02
##     45  106 -3.651e+02  8.22e-14  8.82e-14  4.5e-14  7.9e+07  1.1e-13  2.48e-02
##     46  108 -3.651e+02  1.58e-13  1.76e-13  9.0e-14  4.0e+07  2.2e-13  2.48e-02
##     47  110 -3.651e+02  3.08e-14  3.53e-14  1.8e-14  7.9e+08  4.5e-14  2.48e-02
##     48  112 -3.651e+02  7.78e-15  7.06e-15  3.6e-15  4.0e+09  8.9e-15  2.48e-02
##     49  114 -3.651e+02  1.31e-14  1.41e-14  7.2e-15  4.9e+08  1.8e-14  2.48e-02
##     50  115 -3.651e+02 -2.74e+07  2.82e-14  1.4e-14  9.9e+08  3.6e-14  2.48e-02
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION    -3.650940e+02   RELDX        1.443e-14
##  FUNC. EVALS     115         GRAD. EVALS      50
##  PRELDF       2.824e-14      NPRELDF      2.480e-02
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    1.247719e-14     1.000e+00     1.297e+02
##      2    2.345661e-01     1.000e+00     1.601e+02
##      3    8.653133e-01     1.000e+00     2.015e+02
```

```
## Warning in sqrt(pred$e): NaNs produced
```

```
garchmod.res <- garchmod$residuals

garchmod.res <- na.omit(garchmod.res)

garchmod.ressq <- garchmod.res^2

plot(garchmod.res, col = "red", main = "Chart 8: Standardised Residuals from GARCH(1,1) Equat
ion", type = "l")
```




```
plot(garchmod.ressq, col ="darkorchid", main = "Chart 9: Squared Standardised Residuals from
GARCH(1,1) Equation", type = "l")
```



```
Box.test(garchmod.ressq, lag = m, type = "Box-Pierce")
```

```
##
##  Box-Pierce test
##
## data:  garchmod.ressq
## X-squared = 78.895, df = 20, p-value = 6.041e-09
```

```
Box.test(garchmod.ressq, lag = m, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  garchmod.ressq
## X-squared = 81.69, df = 20, p-value = 2.024e-09
```

```
# The GARCH(1,1) model fails to remove ARCH effects from the squared residuals!


################################## CREATING FINAL MODEL ##########################


# My chosen model for this time series if an ARIMA(1,1,2)-ARCH(12) model.

# rugarch cannot handle differencing the data so we need to do this ourselves and then re-int
egrate after deriving forecasts

cyclical_diff <- diff(data$cyclical)

final_model <- ugarchspec(

  variance.model = list(model ="sGARCH", garchOrder = c(12,0)),

  mean.model = list(armaOrder = c(1,2), include.mean = TRUE),

  distribution.model = "norm"

)
```

c) Using your estimation results from part b, as well as the forecast developed for Assignment 2, compute and plot appropriate point and interval forecasts for the period spanning January 2019 to January 2022. Do the forecasts perform well when

compared to the actual realisations? Why or why not? Compare the forecasts using RMSE and MAE metrics.

The code snippet below extracts a our ARIMA(1,1,2)-ARCH(12) forecast from our preferred model, re-integrates the data and also adds back in the mean term to produce our final time series point estimate and interval forecast. It also charts the forecast we produced in Assignment 2 using our ARMA(13,3) model with a deterministic trend. Chart 10 shows the forecasts of both models and their intervals relative to actuals.

Before commenting on the overall efficacy of these forecasts, first some observations on comparing the two forecasts:

- Firstly, the old forecast is wavy rather than straight. This suggests there is some kind of complex root in the estimated process (ARMA(13,3)) we used to forecast this series in Assignment 2. The fact this is no longer the case in our new forecasts suggest we have appropriately dealt with the unit roots present in this time series.
- Secondly, the prediction intervals for the new model we have generated are notably wider than the ARMA model. This is due to the fact we have modelled a conditional variance, where its time varying nature means prediction intervals must accomodate the possibility of future higher volatility, and are hence wider then a typical ARMA model.
- Thirdly, by looking at our RMSE and MAE metrics, we can see that overall the out of sample fit for the old model is actually better as the old model produces lower values for both to these metrics. While this certainly suggests the old model is preferable, given what we know about the prior model (i.e. not accounting for a unit root), we should treat such perceived preference with skepticism. The COVID-19 pandemic clearly had a massive impact on the underlying data generating process, which no covariance stationary modelling framework could effectively capture. I don't think it is reasonable to say that a certain model is "better" using empirical metrics when there are clearly major theoretical flaws underpinning its construction, as is the case with the old model.

Finally, it is important to point out both of these models perform very poorly relative to the actuals from 2019-01-01 to 2022-01-01. I think there are two key reasons for this:

- The first is the incidence of the COVID-19 pandemic, and what limitation of using the exclusively the past history of a variable to forecast its future. The COVID-19 pandemic represented a once in a century level of disruption to the global economy, from which it is still recovering from. This is a problem because of how we use covariance stationary assumptions to forecast time series data. We use this assumption because conceptually, you can only effectively forecast a time series if its history is well behaved and predictable. The COVID-19 pandemic was such an enormous shock, that it would be unreasonable to expect that the time series forecasting techniques we have learnt thus far could be used to forecast its impact. The pandemic experience was so far removed from historical conditions that it's unsurprising the model performs poorly.
- Somewhat related to this point, another limitation is the fact that both of these models are univariate. Modelling house prices is inherently complex as they are asymmetrically impacted by a wide range of different economic variables (such as the monetary policy stance, growth in household wealth or sentiment, etc). By limiting our forecasting model of house prices to use exclusively the past history of house prices, we are unable to account for the impact of any significant changes in economic variables that are know to have a strong impact on house prices.

```
################################################################################
############################### FORECASTING DATA ###############################
################################################################################

############################## CREATING FORECASTING DATA ########################

data_fc <- read_csv("csindex.csv") %>%
  select(date = DATE, price_level = CSUSHPISA) %>%
  mutate(date = dmy(date)) %>%
  na.omit()
```

```
## Rows: 433 Columns: 2
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): DATE
## dbl (1): CSUSHPISA
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
data_fc <- data_fc  %>%
  filter(date >= as.Date("2019-01-01") & date < as.Date("2022-02-01"))



#Set number of forecast intervals based in forecast data

forecast_intervals <- length(data_fc$date)




############################## FORECASTING VALUES ################################

final_model_forecast_fit_diff <- ugarchfit(spec = final_model, data = cyclical_diff)

final_model_forecast_diff <- ugarchforecast(final_model_forecast_fit_diff, n.ahead = forecast
_intervals)



############################## EXTRACTING FORECASTS ################################



#CYCLICAL

cond_mean_forecast <- data.frame(final_model_forecast_diff@forecast$seriesFor) %>%  clean_nam
es() %>%
  select(cond_mean = x1971_01_19) %>%
  mutate(cond_mean_cumsum = cumsum(cond_mean)) %>%
  select(cond_mean = cond_mean_cumsum)

cond_var_forecast <- data.frame(final_model_forecast_diff@forecast$sigmaFor) %>%
  clean_names() %>%
  select(cond_var = x1971_01_19) %>%
    mutate(upper_interval = cond_var * 1.96) %>%
  mutate(lower_interval = cond_var * -1.96) %>%
  mutate(

    upper_cumsum = cumsum(upper_interval),
    lower_cumsum = cumsum(lower_interval),

    ) %>%
  select(upper_forecast_interval = upper_cumsum, lower_forecast_interval = lower_cumsum)

garch_forecast <- cond_mean_forecast %>%
  cbind(cond_var_forecast)


# Re-integrate cyclical forecasts into final time series forecast


mean <- mean(data$mean) # mean series to create final time series
```

```
last_value <- tail(data,1) %>%
  pull(cyclical)

garch_forecast <- garch_forecast %>%
  mutate(cond_mean = cond_mean + last_value) %>%
  mutate(

    upper_forecast_interval = upper_forecast_interval + last_value,
    lower_forecast_interval = lower_forecast_interval + last_value,

         ) %>%

  select(new_point_estimate = cond_mean, new_high_prediction_interval = upper_forecast_interv
al,
         new_low_prediction_interval = lower_forecast_interval) %>%
  mutate_all(~ . + mean)


######################### START COMPILING FINAL DATA FRAME ######################


data_fc <- data_fc %>%
  bind_cols(garch_forecast)



############################### IMPORTING ASSIGNEMNT 2 FORECAST #################


old_forecast <- read_csv("data_final.csv") %>%
  mutate(date = dmy(date)) %>%
  select(-price_level)
```

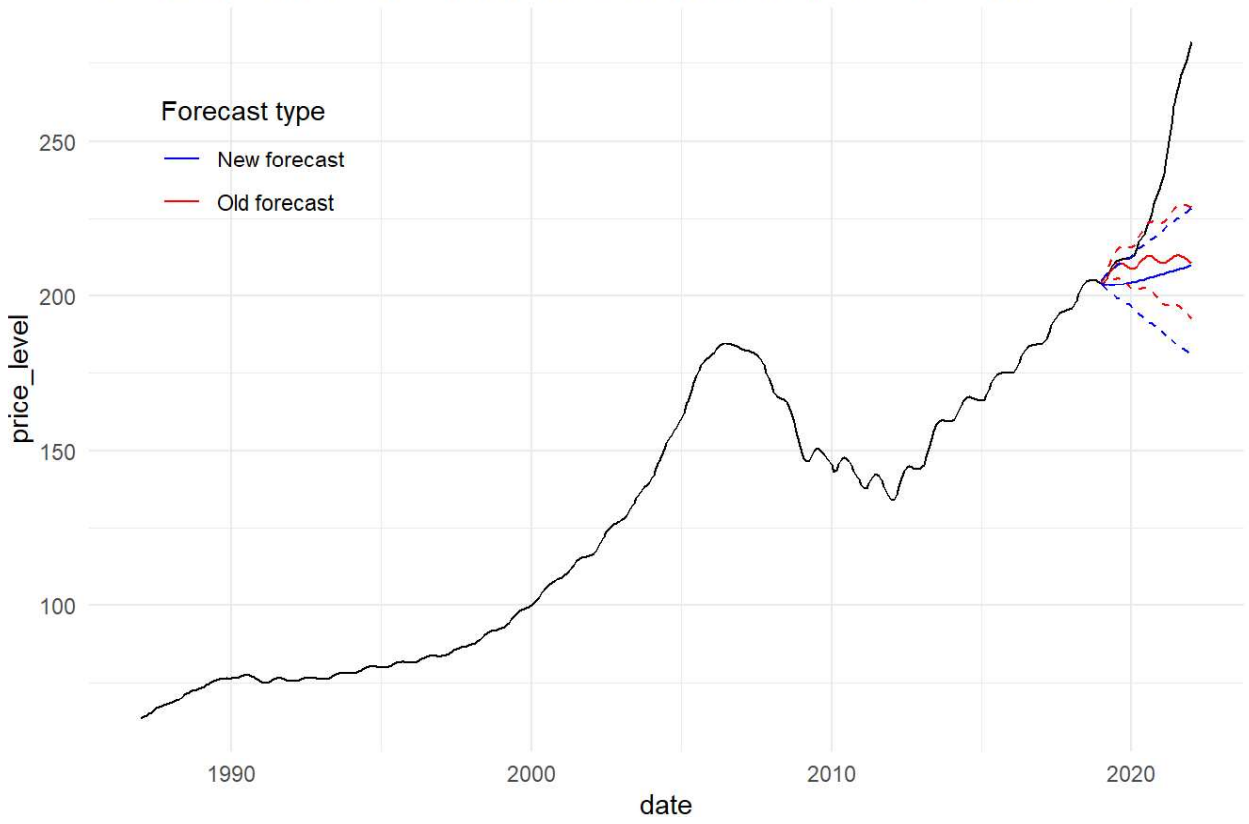
```
## Rows: 421 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): date
## dbl (4): price_level, old_point_estimate, old_high_prediction_interval, old_...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
############################## FINAL DATAFRAME ###################################

data_final <- data_fc %>%
  left_join(old_forecast, by = "date") %>%
  bind_rows(data)


############################## CHARTING RESULTS ##################################

ggplot(data_final, aes(x = date)) +
  geom_line(aes(y = price_level)) +

  # Old forecast

  geom_line(aes(y = old_point_estimate, colour = "Old forecast")) +
  geom_line(aes(y = old_high_prediction_interval, colour = "Old forecast"), linetype = "dashe
d") +
  geom_line(aes(y = old_low_prediction_interval, colour = "Old forecast"), linetype = "dashe
d") +

  # New forecast

    geom_line(aes(y = new_point_estimate, colour = "New forecast")) +
  geom_line(aes(y = new_high_prediction_interval, colour = "New forecast"), linetype = "dashe
d") +
  geom_line(aes(y = new_low_prediction_interval, colour = "New forecast"), linetype = "dashe
d") +

  scale_colour_manual(name = "Forecast type",
                      values = c("Old forecast" = "red", "New forecast" = "blue")) +

  ggtitle("Chart 10: Price level existing family homes in the USE from January 1987 to Januar
y 2022 \n with old (ARMA(13,3) & deterministic trend) and new (ARIMA(1,1,2)-ARCH(12) and mea
n)") +

    theme(plot.title = element_text(size = 10),
        plot.caption = element_text(hjust = 0),
        legend.position = c(0.05, 0.90),
        legend.justification = c(0, 1))
```

```
## Warning: A numeric `legend.position` argument in `theme()` was deprecated in ggplot2
## 3.5.0.
## ℹ Please use the `legend.position.inside` argument of `theme()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 384 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 384 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 384 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 384 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 384 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 384 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

Chart 10: Price level existing family homes in the USE from January 1987 to January 2022 with old (ARMA(13,3) & deterministic trend) and new (ARIMA(1,1,2)-ARCH(12) and mean)



```
############################## FORMALLY COMPARING FORECASTS ####################

forecast_metric_data <- data_final %>%
  select(date,price_level, old_point_estimate, new_point_estimate) %>%
  na.omit()


# CALCULATE RMSE

rmse_old <- sqrt(mean((forecast_metric_data$price_level - forecast_metric_data$old_point_esti
mate)^2))

rmse_new <- sqrt(mean((forecast_metric_data$price_level - forecast_metric_data$new_point_esti
mate)^2))

rmse_old
```

```
## [1] 31.59915
```

```
rmse_new
```

```
## [1] 34.28731
```

```
# CALCULATE MAE

mae_old <- mean(abs(forecast_metric_data$price_level - forecast_metric_data$old_point_estimat
e))

mae_new <- mean(abs(forecast_metric_data$price_level - forecast_metric_data$new_point_estimat
e))

mae_old
```

```
## [1] 21.40834
```

```
mae_new
```

```
## [1] 25.8235
```