ECOM90024 - Assignment 2 - Question 2

Josh Copeland

2024-04-22

```
# Packages used for this assignment

# library(tidyverse)
# library(lubridate)
# library(forecast)
```

Question 2

You are an analyst working for a real estate investment fund and are tasked with monitoring and forecasting house prices in the United States. The file csindex.csv contains monthly observations of the Case-Shiller U.S. National Home Price Index from January 1987 to December 2022. You are required to compute all your estimations and plots in R.

a) Generate a plot of the data and provide a brief description of the observed time series. (1 Mark)

Chart 1 below shows a plot of the data from the Case-Shiller US National Home Price Index, which seeks to measures the price level of existing single family-homes in the US. The data provided shows this price level from January 1987 to December 2022.

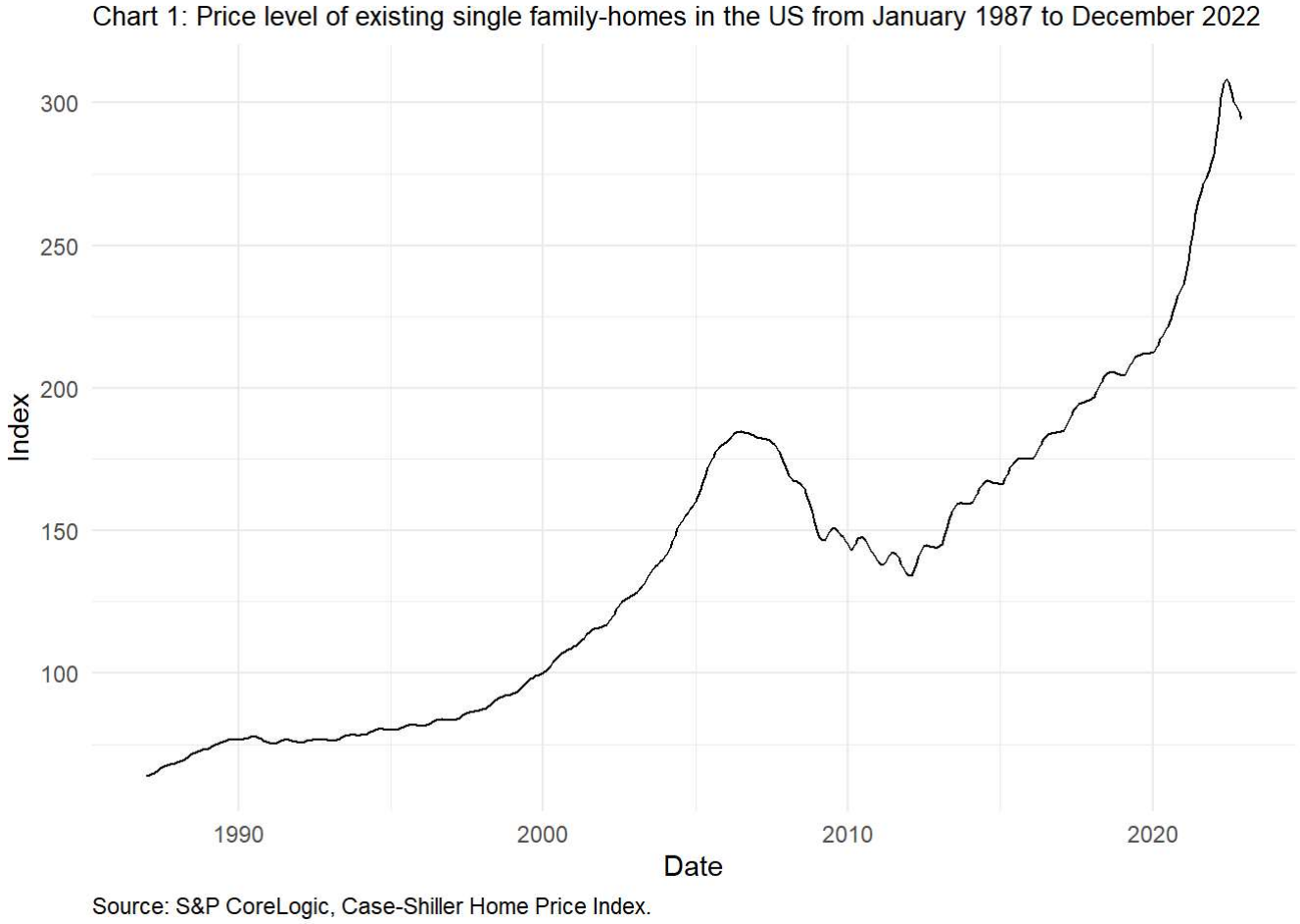
There is a clear upward trend in this series over time, but there is also clear evidence of cyclical fluctuations in the data. It is unclear to what extent the curvature of this series is attributable to the trend or the incidence of cyclical fluctuations in the series over time. It's also possible that some of the shorter-term fluctuations (that seem to increase over time) could be attributable to seasonal patterns in the data. Testing and accounting for both of these deterministic patterns in the data is required to isolate the cyclical fluctuations.

The clearest indicator of cyclical fluctuations in this data is the significant acceleration of growth in the series in the mid-2000s (and the decline which followed in 2007/2008), likely driven by the unsustainable household credit conditions that culminated in the US' sub-prime mortgage crisis that kicked off the Global Financial Crisis.

```
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 to D
ecember 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))
```



b) Using the data from January 1987 to December 2018, identify and estimate an appropriate time series model using the steps outlined in Lecture 6. Make sure to report all relevant estimation

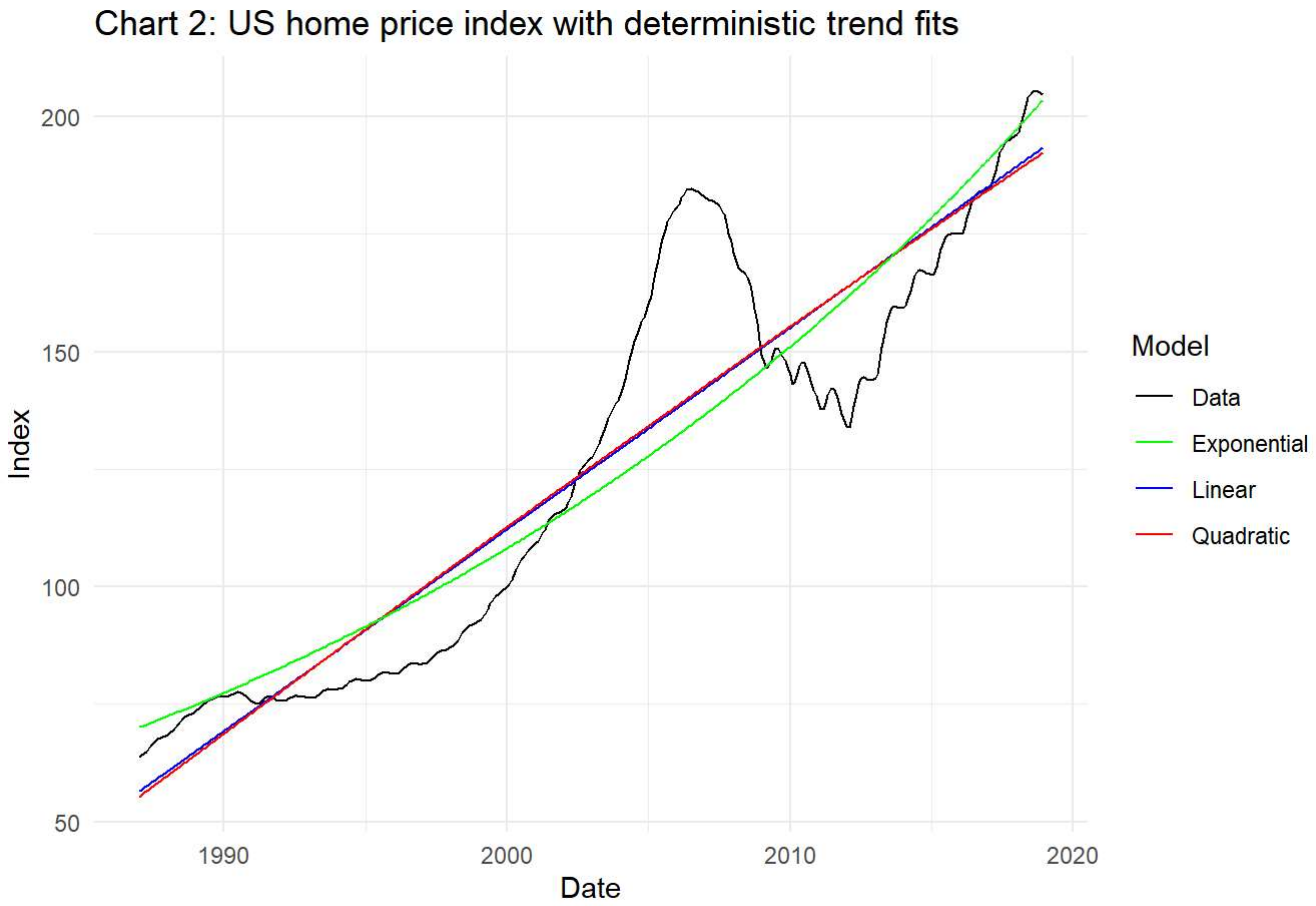
results, plots, statistical tests and information criteria that you are relying on in determining your preferred model. (2 Marks)

Each step of the general approach for estimating a time series model is available below in order.

- Specify and estimate deterministic components
    - As noted in part (a), there seems to be some long-run upwards trend to the series. Therefore, I have evaluated the fit of a collection of deterministic trend models below (linear, quadratic and exponential). Their fits have been illustrated in Chart 2. I evaluate the linear model as optimum model for decomposing the trend component of this time series, because it has the lowest AIC/BIC of the models evaluated (as seen in the model_selection_criteria table below). By taking the residuals of this model we can observe the detrended series, as seen in Chart 3.
    - Using the detrended series we are able to determine if there is any deterministic seasonality in the data: I conclude this is not the case. This is done by generating a collection of dummy variables for each month of the year, and then regressing the detrended series on these dummy variables (as seen in the summary of the seasonal_model object). None of the estimated ceofficients are statistically different from zero. Moreover, they are jointly equal to zero according to the reported overall F-statistic, which has a p-value greater than 0.05. Therefore, we can conclude that the seasonal means are not statistically different from zero and we do not have to de-season our data. This means the residuals of the linear model, or the detrended series, also represents the cyclical component of our time series.

```
################################################################################
################### SPECIFYING DETERMINISTIC TRENDS #############################
################################################################################

################## GENERATING TREND COMPONENT OF TIME SERIES ####################

data <- data %>%
  filter(date < as.Date("2019-01-01"))

data <- data %>%
  mutate(

    time = seq(1,length(data$price_level)),
    time_square = time^2,
    hits_log = log(price_level)

  )


# ESTIMATING MODELS


# Linear model

price_model_linear <- lm(price_level ~ time, data = data)

#Quadratic model

price_model_quadratic <- lm(price_level ~ time + time_square, data = data)

#Exponential model

price_model_exponential_descaled <- nls(price_level ~ a*exp(b*time), start = list(a=8000, b = 0.
01), data = data)


# EXTRACTING MODEL FITS

data <- data %>%
  mutate(

    linear_model = predict(price_model_linear),

    quadratic_model = predict(price_model_quadratic),

    exponential_model = predict(price_model_exponential_descaled)

  )


ggplot(data, aes(x = date)) +
  geom_line(aes(y = price_level, colour = "Data")) +
```

```
  geom_line(aes(y = linear_model, colour = "Linear")) +
  geom_line(aes(y = quadratic_model, colour = "Quadratic")) +
  geom_line(aes(y = exponential_model, colour = "Exponential")) +
  scale_color_manual(values = c("Data"= "black" ,"Linear" = "blue", "Quadratic" = "red", "Expone
ntial" = "green")) +
  labs(colour = "Model") +
  labs(x = "Date", y = "Index", caption = "Source: S&P CoreLogic, Case-Shiller Home Price Index
& author's calculations.") +
  ggtitle("Chart 2: US home price index with deterministic trend fits") +
    theme(plot.caption = element_text(hjust = 0))
```

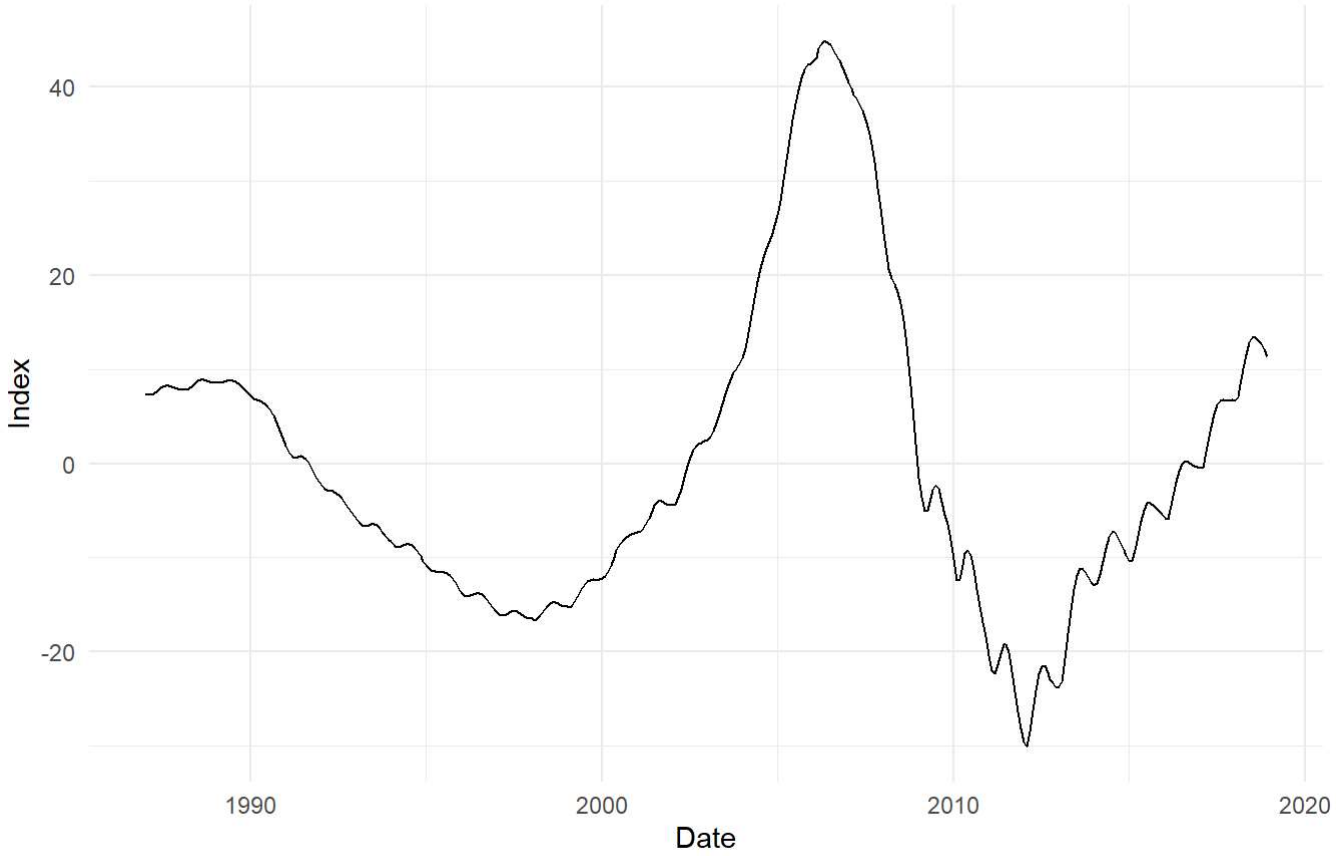


```
# COMPARING MODEL OUTPUTS

model_list <- list(price_model_linear, price_model_quadratic, price_model_exponential_descaled)

model_names <- c("price_model_linear", "price_model_quadratic", "price_model_exponential_descale
d")

extract_criteria <- function(model, model_name) {
  summary_data <- summary(model)
  aic <- AIC(model)
  bic <- BIC(model)
  return(data.frame(Model = model_name, AIC = aic, BIC = bic))
}

model_selection_criteria <- map2_df(model_list, model_names, extract_criteria) %>%
  arrange(AIC)

print(model_selection_criteria)
```

```
##                              Model      AIC      BIC
## 1               price_model_linear 3245.817 3257.669
## 2            price_model_quadratic 3247.453 3263.255
## 3 price_model_exponential_descaled 3297.471 3309.322
```

```
# Linear model returns the lowest AIC/BIC. Therefore use this as the trend series.

# Need to generate the detrended series to test for seasonal effects

data <- data %>%
  mutate(

    detrended = resid(price_model_linear)

  )

ggplot(data, aes(x = date)) +
  geom_line(aes(y = detrended)) +
  labs(x = "Date", y = "Index", caption = "Source: S&P CoreLogic, Case-Shiller Home Price Index
& author's calculations.") +
  ggtitle("Chart 3: Detrended series of US home price index") +
  theme(plot.caption = element_text(hjust = 0))
```

Chart 3: Detrended series of US home price index



Source: S&P CoreLogic, Case-Shiller Home Price Index & author's calculations.

```
################## GENERATING SEASONAL COMPONENT OF TIME SERIES ##################



data <- data %>%
  mutate(month = month(date)) %>%
  mutate(

    jan_dummy = case_when(month == 1 ~ 1, TRUE ~ 0),
    feb_dummy = case_when(month == 2 ~ 1, TRUE ~ 0),
    mar_dummy = case_when(month == 3 ~ 1, TRUE ~ 0),
    apr_dummy = case_when(month == 4 ~ 1, TRUE ~ 0),
    may_dummy = case_when(month == 5 ~ 1, TRUE ~ 0),
    jun_dummy = case_when(month == 6 ~ 1, TRUE ~ 0),
    jul_dummy = case_when(month == 7 ~ 1, TRUE ~ 0),
    aug_dummy = case_when(month == 8 ~ 1, TRUE ~ 0),
    sep_dummy = case_when(month == 9 ~ 1, TRUE ~ 0),
    oct_dummy = case_when(month == 10 ~ 1, TRUE ~ 0),
    nov_dummy = case_when(month == 11~ 1, TRUE ~ 0),
    dec_dummy = case_when(month == 12 ~ 1, TRUE ~ 0),


  )

seasonal_model <- lm(detrended ~ 0 + jan_dummy + feb_dummy + mar_dummy +
                       apr_dummy + may_dummy + jun_dummy +
                       jul_dummy + aug_dummy + sep_dummy +
                       oct_dummy + nov_dummy + dec_dummy,
                     data = data)


summary(seasonal_model)
```

```
##
## Call:
## lm(formula = detrended ~ 0 + jan_dummy + feb_dummy + mar_dummy +
##     apr_dummy + may_dummy + jun_dummy + jul_dummy + aug_dummy +
##     sep_dummy + oct_dummy + nov_dummy + dec_dummy, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.578 -11.730  -4.288   7.911  45.200
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## jan_dummy  -1.2419     2.9473  -0.421    0.674
## feb_dummy  -1.5055     2.9473  -0.511    0.610
## mar_dummy  -1.1474     2.9473  -0.389    0.697
## apr_dummy  -0.4607     2.9473  -0.156    0.876
## may_dummy   0.2968     2.9473   0.101    0.920
## jun_dummy   0.9454     2.9473   0.321    0.749
## jul_dummy   1.2736     2.9473   0.432    0.666
## aug_dummy   1.2584     2.9473   0.427    0.670
## sep_dummy   0.9063     2.9473   0.308    0.759
## oct_dummy   0.4243     2.9473   0.144    0.886
## nov_dummy  -0.1021     2.9473  -0.035    0.972
## dec_dummy  -0.6472     2.9473  -0.220    0.826
##
## Residual standard error: 16.67 on 372 degrees of freedom
## Multiple R-squared:  0.003379,   Adjusted R-squared:  -0.02877
## F-statistic: 0.1051 on 12 and 372 DF,  p-value: 0.9999
```

- Generate the residuals
    - Because we do not need to deseasonalise our data, the detrended series is sufficient for isolating the cyclical component of the time series. The final components that constitute this time series (observed, trend and cyclical) are avalable in the data_decomposed dataframe.
        - Before continuing, it is important to establish that this cyclical series is not white noise, such that it demonstrates some level of dependence which can be usefully modelled through an ARMA process. After conducting the portmanteau Box-Pierce and Ljung-Box tests, we can verify this cyclical 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. This implies dependence in this time series, suitable for ARMA modelling.

```
################################################################################
######################### GENERATING THE RESIDUALS #############################
################################################################################



data_decomposed <- data %>%
  select(date, price_level, trend = linear_model, cyclical = detrended)


# CONFIRMING THE CYCLICAL DATA IS NOT WHITE NOISE

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

Box.test(data_decomposed$cyclical, lag = m, type = "Box-Pierce")
```

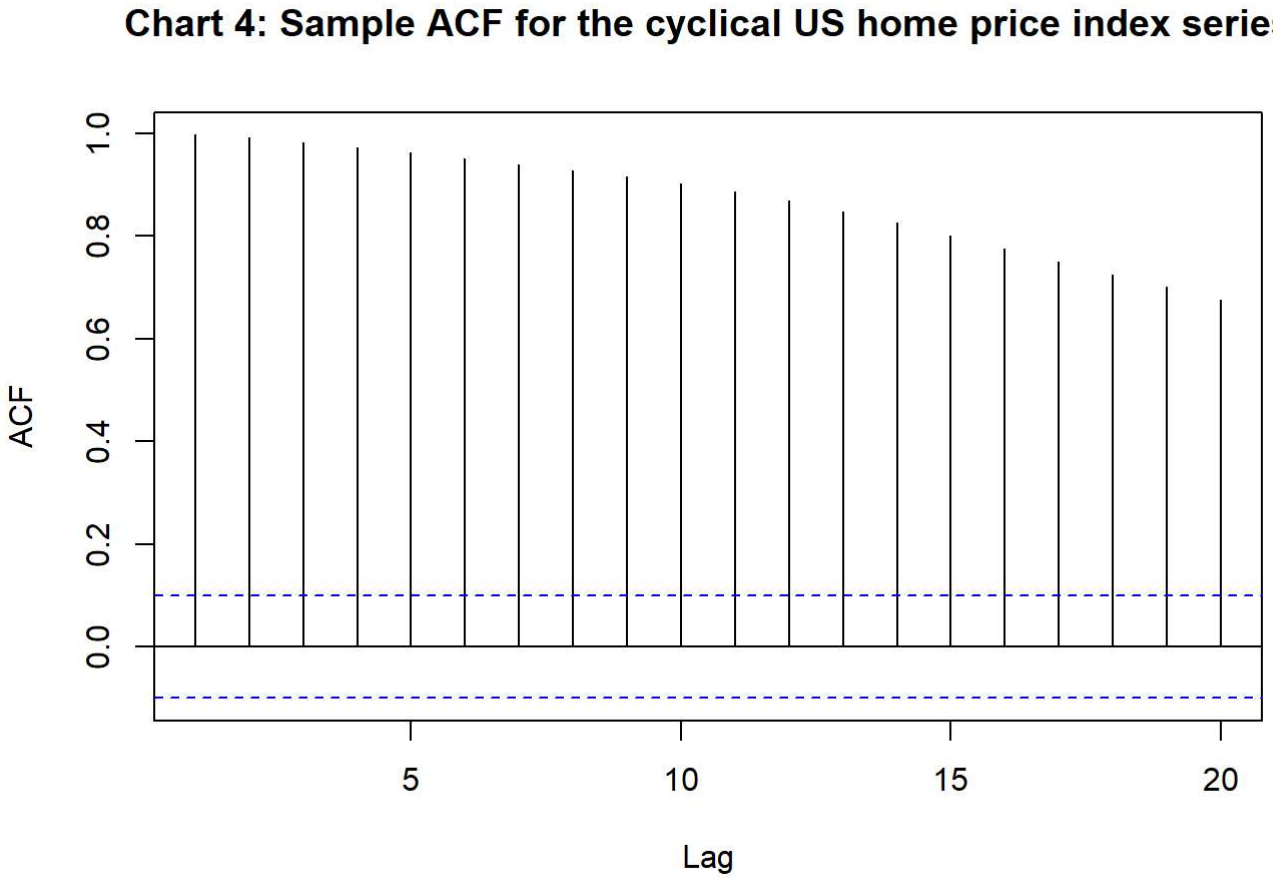
```
##
##  Box-Pierce test
##
## data:  data_decomposed$cyclical
## X-squared = 5893, df = 20, p-value < 2.2e-16
```

```
Box.test(data_decomposed$cyclical, lag = m, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  data_decomposed$cyclical
## X-squared = 6071.2, df = 20, p-value < 2.2e-16
```

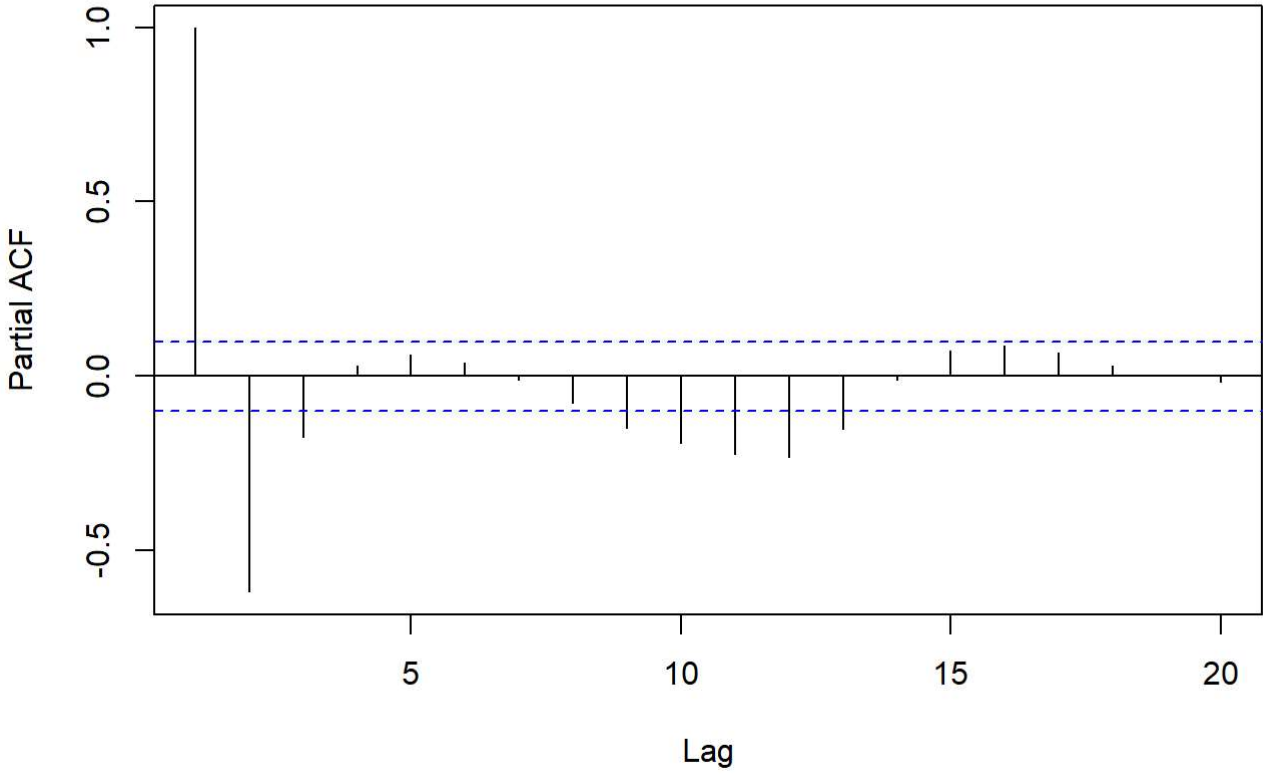
- Generate the ACF and PACF of the residuals to get a sense of their dependence structure.
    - Chart 4 shows the sample autocorrelation function (ACF) for the cyclical US home price index series for the 1st to 20th lag. It shows that these autocorrelations are decaying gradually over these lags.
    - Chart 5 shows the sample partial autocorrelation function (PACF) for the cyclical US home price index from the 1st to 20th lag. These lags cut off initially at lag 3, but then become significant again from lags 9 to 13.
    - Based off this information, this tells us that when choosing an ARMA(p,q) model, we should search for a model with a p parameter up to order 13. We also need a minimum value of 1 because we need to account for dependency in our model (i.e. we know this isn't a white noise process). Given its not advisable to have a very large number of MA terms in a model, we will only search for a model with a q parameter up to order 4.

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


# GENERATING ACF AND PACF

acf_cyclical <- acf(data_decomposed$cyclical, plot = FALSE)
pacf_cyclical <- pacf(data_decomposed$cyclical, plot = FALSE)

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




```
plot(pacf_cyclical[1:20], main = "Chart 5: Sample PACF for the cyclical US home price index seri
es")
```

Chart 5: Sample PACF for the cyclical US home price index series



- Estimate a range of ARMA(p,q) and choose your preferred model using AIC and BIC
    - By making these observations, we are able to search for suitable ARMA models. As mentioned above the previous code snippet, we will search models with parameter ranges of AR(1) - AR(13) and MA(0)-Ma(4).
    - Using these minimum and maximum parameters values as inputs into a loop which estimates an ARMA model for every possible parameter combination, we are able to automate the comparison of their AIC and BIC values to determine which is our preferred model.
    - As can be seen from the output of the bayes.choice and akaike.choice objects in the code snippet below, both information criteria are minimised for the same model: ARMA(12,3). Therefore, we choose this as our preferred model for forecasting the cyclical component of the US home price index.
    - Although this is our preferred model, and the best one using the tools currently available to us for this assignment, it is important to point out that this is not the perfect model.
    - By observing several diagnostics on the best_model object, it's clear there are several problems with this model. Firstly, Chart 6 shows us that some of the estimated roots of the lag polynomial for this model lie right on the unit circle boundary, which could be due to the cyclical series being non-stationary. Secondly, looking at the residuals of the model, it seems like they are heteroskedastic and therefore not covariance stationary, which is a problem because this likely means we have failed to incorporate some important information into the model. This hunch is confirmed by Charts 8 and 9 and the subsequent portmanteau tests, which tell us that the model residuals are indeed not white noise.

```
################################################################################
################### GENERATING AND EVALUATING ARMA MODELS ######################
################################################################################

######################### GENERATING BEST ARMA MODEL ###########################

# Specify the parameter values from the ACF and PACF

pstart = 1
pend = 13

qstart = 0
qend = 4

#Generate the dataframe to store different ARMA(p,q) model combinations in

prefix.ar <- "ar"
suffix.ar <- seq(from = pstart, to = pend)

prefix.ma <- "ma"
suffix.ma <- seq(from = qstart, to = qend)

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

akaike <- data.frame(row.names = ar.label, matrix(NA,(pend-pstart+1),qend+1))
colnames(akaike) <- ma.label

bayes <- data.frame(row.names = ar.label, matrix(NA,(pend-pstart+1),qend+1))
colnames(bayes) <- ma.label

# Generate the loop required to iterate over different ARMA(p,q) combinations and store them app
ropriate in the dataframe provided above.

for(i in pstart:pend){

for(j in (qstart+1):(qend+1)){

model <- try(Arima(data_decomposed$cyclical,
order = c(i,0,j-1),
include.mean = FALSE,
include.drift = FALSE,
method = "CSS-ML"))
if(!inherits(model, "try-error") && model$code == 0){
akaike[i,j] <- AIC(model)
bayes[i,j]<- BIC(model)}
}

}
```

```
## Error in stats::arima(x = x, order = order, seasonal = seasonal, include.mean = include.mean,
:
##   non-stationary AR part from CSS
## Error in stats::arima(x = x, order = order, seasonal = seasonal, include.mean = include.mean,
:
##   non-stationary AR part from CSS
```

```
# Code to try extract which model AIC and BIC prefers

akaike.row <- rownames(akaike)[which(akaike == min(akaike, na.rm = TRUE),arr.ind = TRUE)[ , 1]]
akaike.col <- colnames(akaike)[which(akaike == min(akaike, na.rm = TRUE),arr.ind = TRUE)[ , 2]]

akaike.choice <- c(akaike.row, akaike.col)

print(akaike.choice)
```

```
## [1] "ar13" "ma3"
```

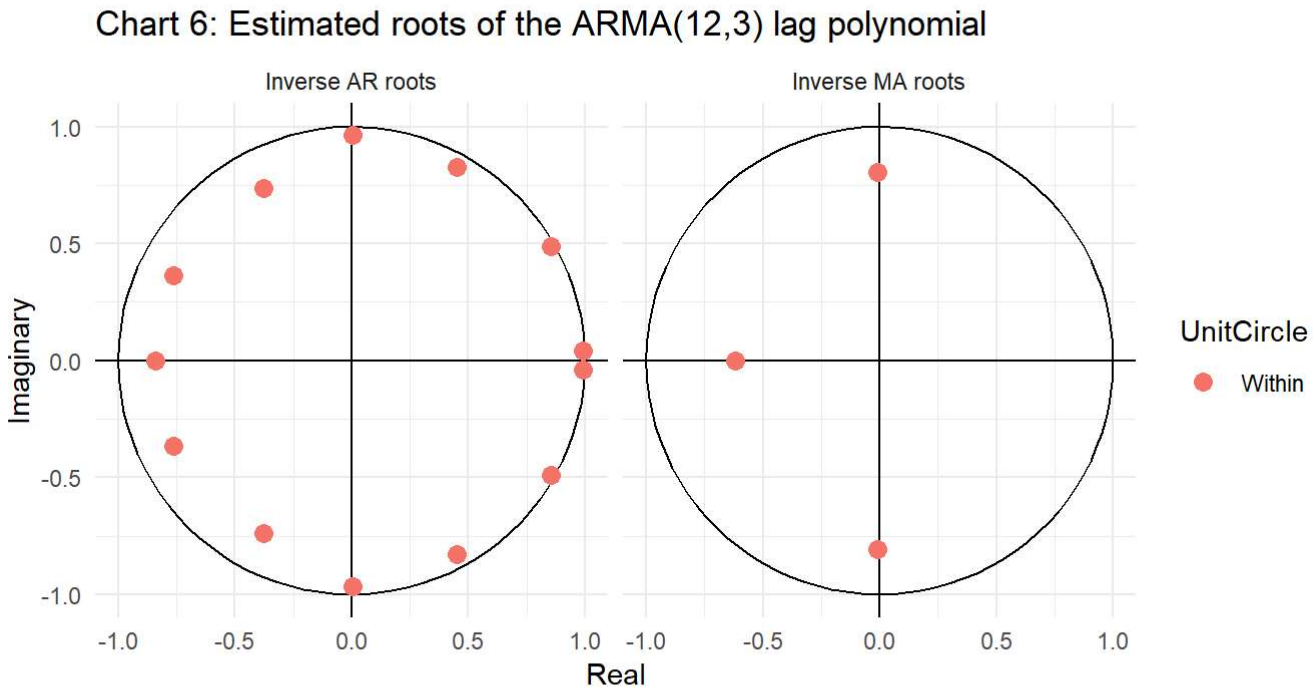
```
bayes.row <- rownames(bayes)[which(bayes == min(bayes, na.rm = TRUE),arr.ind = TRUE)[ , 1]]
bayes.col <- colnames(akaike)[which(bayes == min(bayes, na.rm = TRUE),arr.ind = TRUE)[ , 2]]

bayes.choice <- c(bayes.row, bayes.col)

print(bayes.choice)
```

```
## [1] "ar13" "ma3"
```

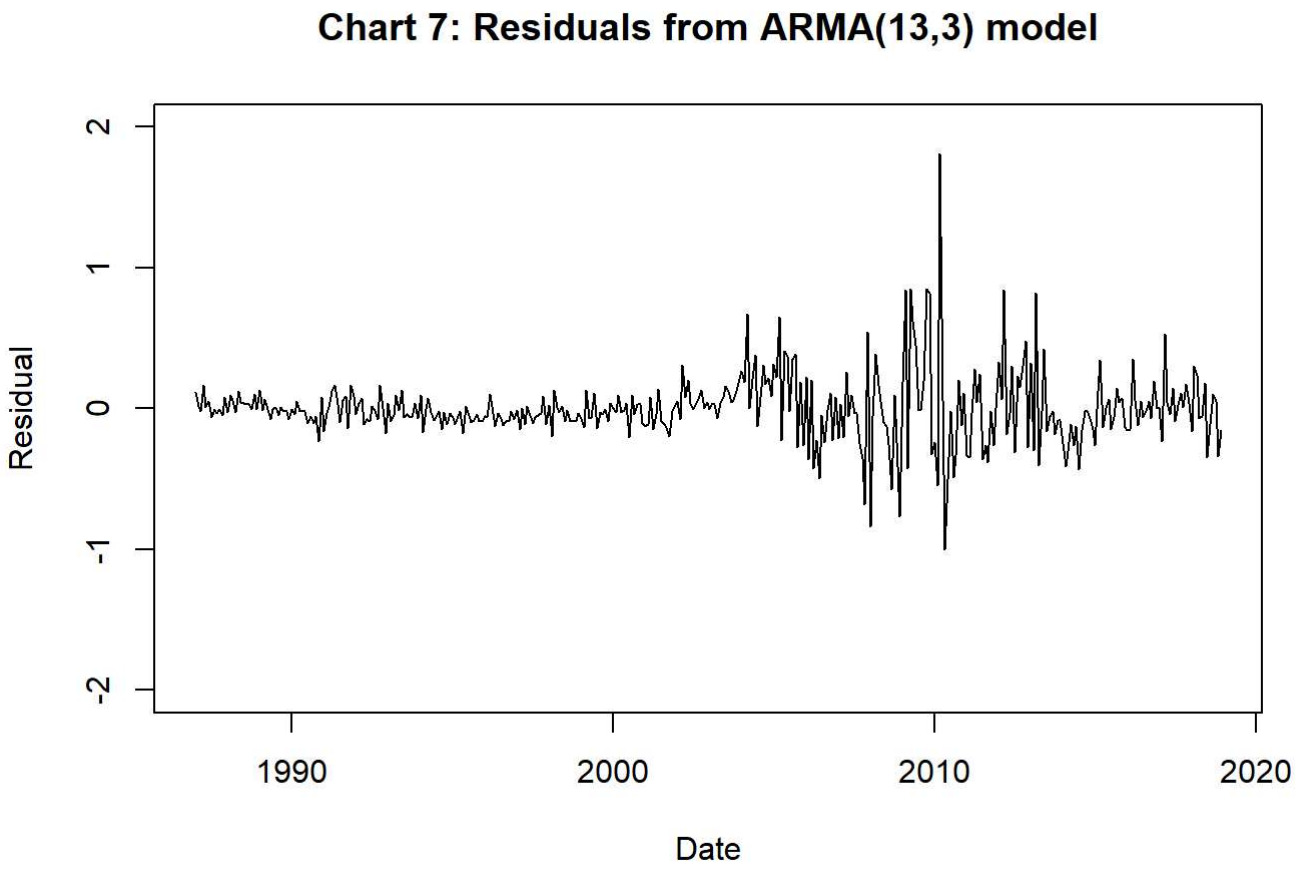
```
#Both AIC and BIC agree that ARMA(13,3) is the best model fit.



############################## TESTING THE BEST MODEL FIT ########################

 best_model <- Arima(data_decomposed$cyclical,
                     order = c(13,0,3),
                     include.mean = FALSE,
                     xreg = data_decomposed$trend,
                     method = "CSS-ML")

summary(best_model)
```

```
## Series: data_decomposed$cyclical
## Regression with ARIMA(13,0,3) errors
##
## Coefficients:
##          ar1      ar2     ar3     ar4      ar5     ar6     ar7      ar8     ar9
##       1.4932  -0.6578  0.0455  0.4579  -0.5326  0.1046  0.1763  -0.1686  0.1027
## s.e.  0.1735   0.4315  0.5378  0.3438   0.1392  0.1740  0.1333   0.1103  0.1367
##         ar10    ar11    ar12     ar13     ma1     ma2     ma3    xreg
##       0.0065  0.1654  0.1214  -0.3269  0.6337  0.6597  0.4001  0.0041
## s.e.  0.1077  0.1033  0.1241   0.0785  0.1723  0.2246  0.1163  0.0203
##
## sigma^2 = 0.06151:  log likelihood = -8.37
## AIC=52.73   AICc=54.6   BIC=123.84
##
## Training set error measures:
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.001469187 0.2424602 0.1520224 3.299717 5.869655 0.2288897
##                     ACF1
## Training set 0.02004152
```

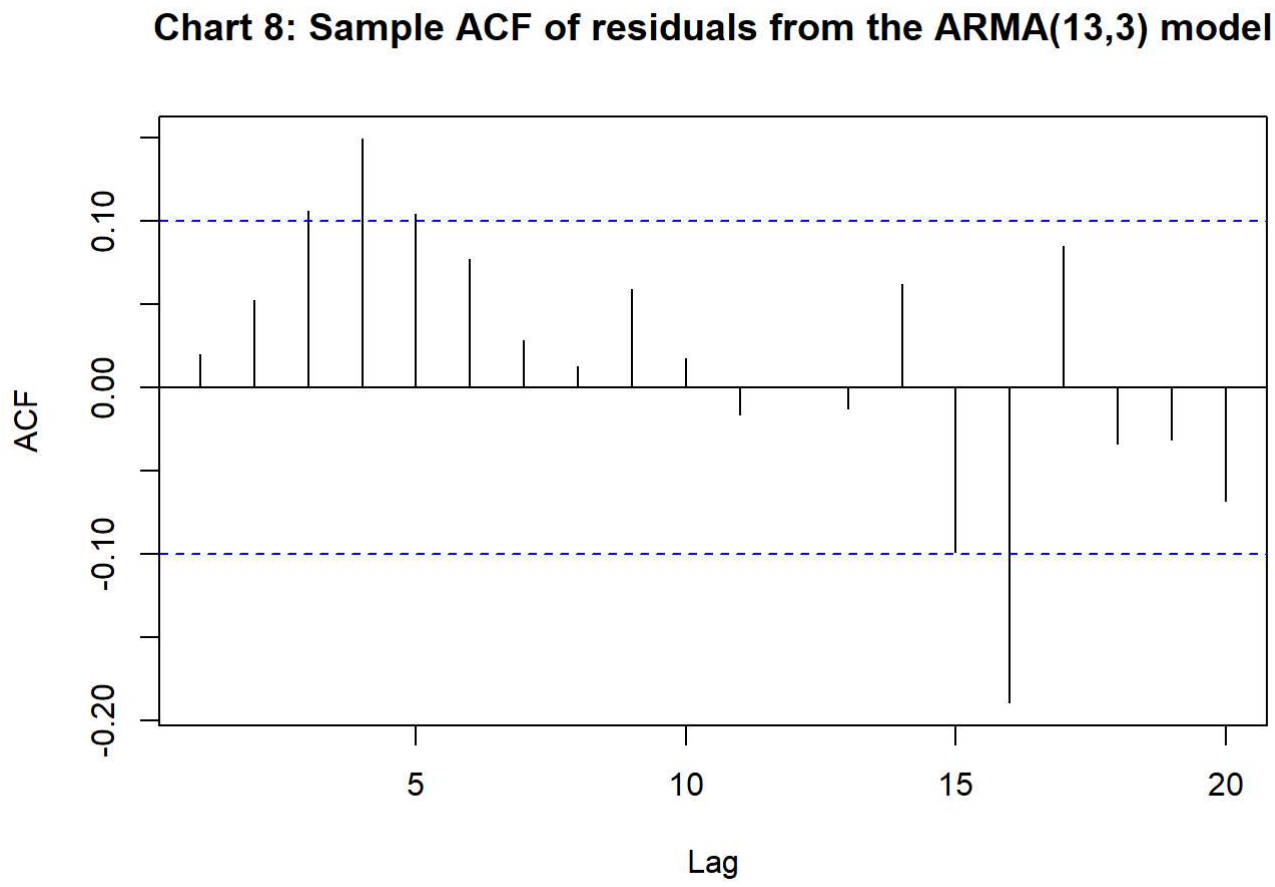
```
autoplot(best_model, main = "Chart 6: Estimated roots of the ARMA(12,3) lag polynomial")
```



```
plot(data_decomposed$date, best_model$residuals,
     main = "Chart 7: Residuals from ARMA(13,3) model",
     xlab = "Date",
     ylab = "Residual",
     type = "l",
     ylim = c(-2,2))
```

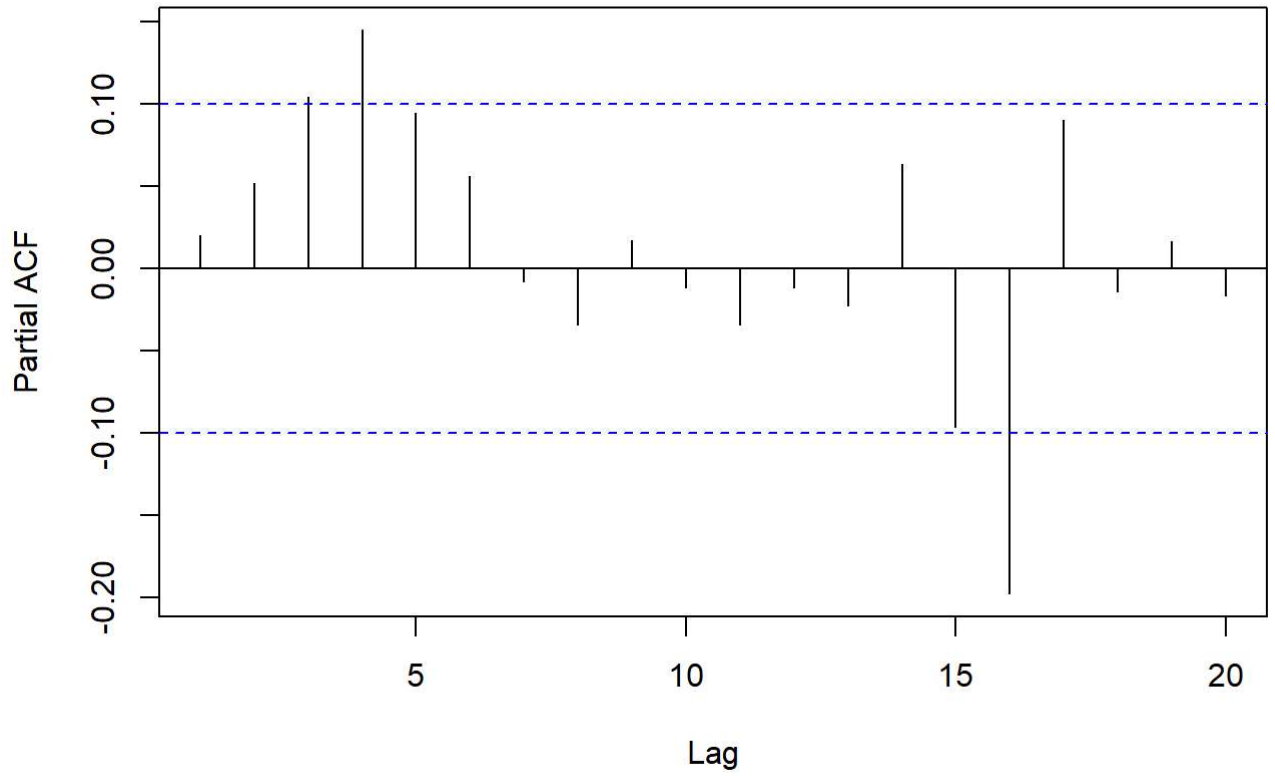



```
acf.resid <- acf(best_model$resid, plot = FALSE)
pacf.resid <- pacf(best_model$resid, plot = FALSE)

plot(acf.resid[1:20], main = "Chart 8: Sample ACF of residuals from the ARMA(13,3) model")
```




```
plot(pacf.resid[1:20], main = "Chart 9: Sample PACF of residuals from the ARMA(13,3) model")
```

Chart 9: Sample PACF of residuals from the ARMA(13,3) model




```
Box.test(best_model$resid, type = "Box-Pierce", l = m)
```

```
##
##  Box-Pierce test
##
## data:  best_model$resid
## X-squared = 46.741, df = 20, p-value = 0.0006372
```

```
Box.test(best_model$resid, type = "Ljung-Box", l = m)
```

```
##
##  Box-Ljung test
##
## data:  best_model$resid
## X-squared = 48.331, df = 20, p-value = 0.0003822
```

c) Using your estimation results from part b, 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? (2

Marks)

- The code snippet below uses the models defined in part (b) to construct a forecast for the price level of existing single family-homes in the US from January 2019 to January 2022. Chart 10 visualises the result of this exercise. As can be seen, the forecast seems to perform reasonably well up until 2020, after which performance deteriorates significantly. Up until 2020 the point estimate remains relatively close to the actual realisations. However, by the end of 2020, the actual realisations well surpass the prediction intervals of the forecast.
- I consider there are several reasons for this:
    - As noted in part (b), even though our ARMA(12,3) model is our preferred model for the cyclical component of the time series, it is not a very good model given that its errors are not white noise. This means its unlikely the estimates produced by this model truly reflect the underlying dependence structure in the data, which likely means we have omitted some important variable or relationship from this model. Regardless of the root cause, its likely this factor is contributing to poor performance from 2020 onwards.
    - Another obvious reason that poor performance occurs from 2020 onwards is the COVID-19 pandemic. 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 we are using univariate model for forecasting. 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.

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

#CREATE DATAFRAME FOR FORECASTING

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  %>%
  mutate(time = seq(1,length(data_fc$price_level))) %>%
  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 ################################

#FORECAST TREND COMPONENT

trend_forecast <- predict(price_model_linear, newdata = data_fc)

#FORECAST CYCLICAL COMPONENT

cyclical_forecast <- forecast(best_model, h = forecast_intervals, xreg = trend_forecast) %>%
  data.frame() %>%
  select(point_estimate = Point.Forecast, low_prediction_interval = Lo.95, high_prediction_inter
val = Hi.95)




############################## EXTRACTING MODEL OUTPUTS ##########################

# COMBINING BACK INTO A SINGLE DATAFRAME

data_fc <- data_fc %>%
  cbind(trend_forecast, cyclical_forecast)


#CREATING FINAL FORECAST PROFILE

data_fc <- data_fc %>%
  mutate(

    point_estimate_fc = trend_forecast + point_estimate,
    high_prediction_interval_fc = trend_forecast + high_prediction_interval,
    low_prediction_interval_fc = trend_forecast + low_prediction_interval

    ) %>%
  select(date, point_estimate_fc, high_prediction_interval_fc, low_prediction_interval_fc)


# CREATING FINAL DATAFRAME
```

```
data_final <- read_csv("csindex.csv") %>%
  select(date = DATE, price_level = CSUSHPISA) %>%
    mutate(date = dmy(date)) %>%
  na.omit() %>%
  filter(date < as.Date("2022-02-01")) %>%
  left_join(data_fc, by = "date")
```

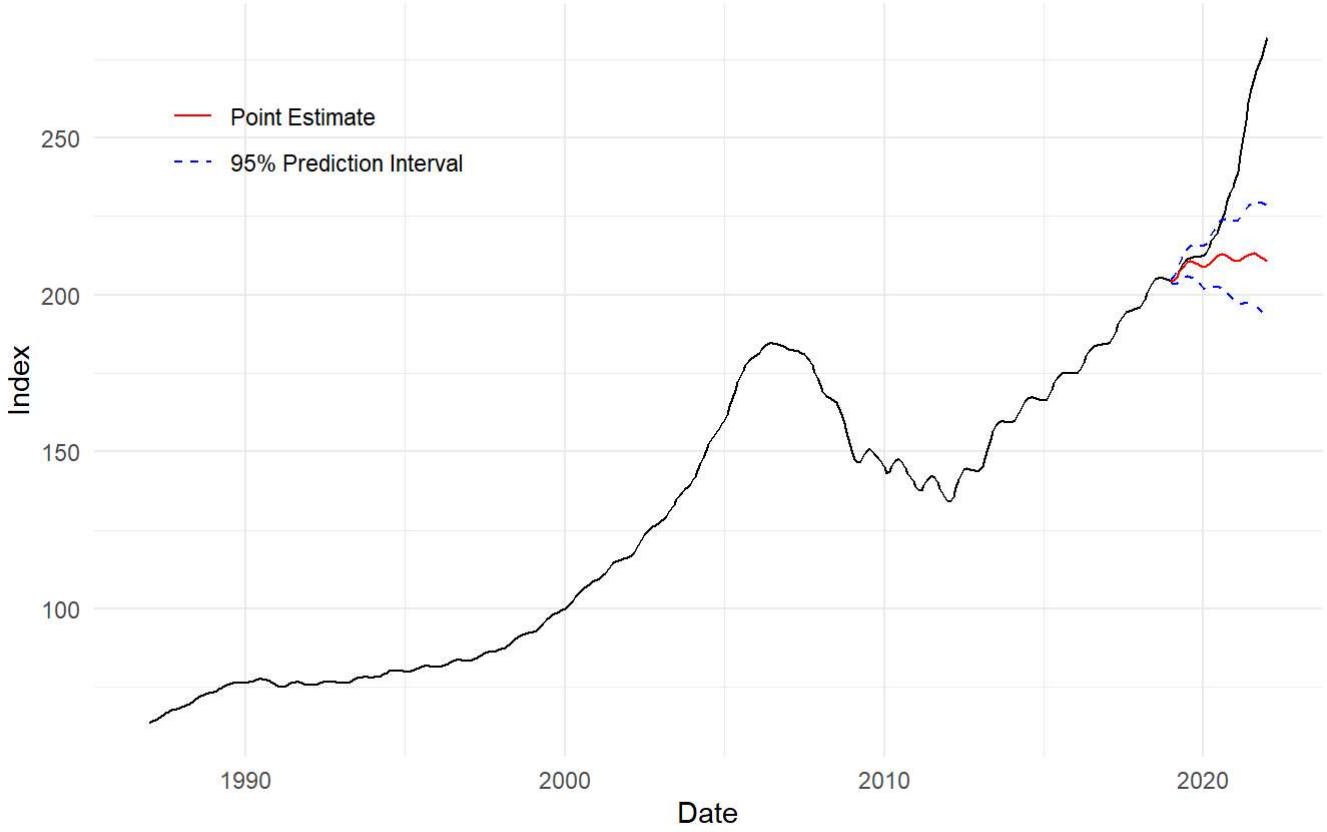
```
## Rows: 433 Columns: 2
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): DATE
## dbl (1): CSUSHPISA
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE`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

```
################################## CHARTING FINAL RESULTS ########################


ggplot(data_final, aes(x = date)) +
  geom_line(aes(y = price_level)) +
  geom_line(aes(y = point_estimate_fc, color = "Point Estimate")) +
  geom_line(aes(y = high_prediction_interval_fc, color = "95% Prediction Interval"), linetype =
"dashed") +
  geom_line(aes(y = low_prediction_interval_fc, color = "95% Prediction Interval"), linetype =
"dashed") +
  scale_color_manual(values = c("Point Estimate" = "red", "95% Prediction Interval" = "blue"),
                     breaks = c("Point Estimate", "95% Prediction Interval"),
                     guide = guide_legend(title = NULL)) +
  ggtitle("Chart 10: Price level of existing single family-homes in the US from January 1987 to
January 2022 \n with forecast (deterministic trend and ARMA(13,3) cyclical component)") +
  labs(x = "Date", y = "Index", caption = "Source: S&P CoreLogic, Case-Shiller Home Price Index
& author's calculations.") +
  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()`).
```

Chart 10: Price level of existing single family-homes in the US from January 1987 to January 2022 with forecast (deterministic trend and ARMA(13,3) cyclical component)



Source: S&P CoreLogic, Case-Shiller Home Price Index & author's calculations.