W271 Assignment 3

Due 11:59pm Pacific Time Sunday April 5 2020

Instructions (Please Read Carefully):

- No page limit, but be reasonable
- Do not modify fontsize, margin or line_spacing settings
- This assignment needs to be completed individually; this is not a group project. Each student needs to submit their homework to the course github repo by the deadline; submission and revisions made after the deadline will not be graded
- Answers should clearly explain your reasoning; do not simply 'output dump' the results of code without explanation
- Submit two files:
  1. A pdf file that details your answers. Include all R code used to produce the answers. Do not suppress the codes in your pdf file
  2. The R markdown (Rmd) file used to produce the pdf file

  The assignment will not be graded unless **both** files are submitted
- Use the following file-naming convensation:
  - StudentFirstNameLastName_HWNumber.fileExtension
  - For example, if the student's name is Kyle Cartman for assignment 1, name your files follows:
    - KyleCartman_assignment3.Rmd
    - KyleCartman_assignment3.pdf
- Although it sounds obvious, please write your name on page 1 of your pdf and Rmd files
- For statistical methods that we cover in this course, use the R libraries and functions that are covered in this course. If you use libraries and functions for statistical modeling that we have not covered, you must provide an explanation of why such libraries and functions are used and reference the library documentation. For data wrangling and data visualization, you are free to use other libraries, such as dplyr, ggplot2, etc.
- For mathematical formulae, type them in your R markdown file. Do not e.g. write them on a piece of paper, snap a photo, and use the image file.
- Incorrectly following submission instructions results in deduction of grades
- Students are expected to act with regard to UC Berkeley Academic Integrity

```r
library(dplyr)
library(lubridate)
library(tsibble)
library(feasts)
library(forecast)
library(fpp3)
library(fma)
```

Question 1 (1 point)

Time Series Linear Model

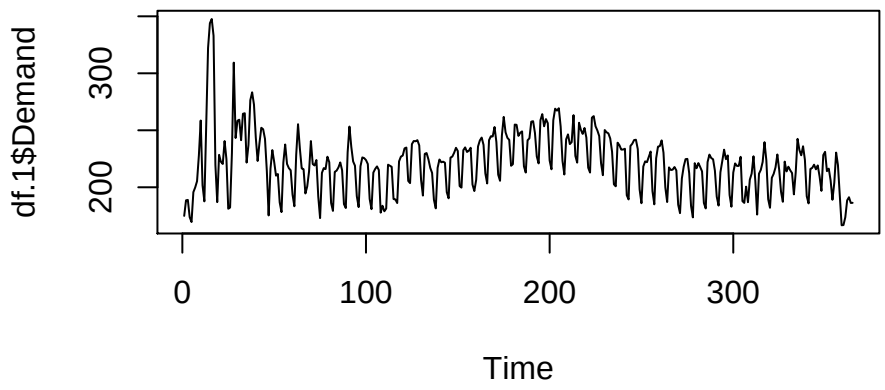
Daily electricity demand and temperature (in degrees Celsius) is recorded in `Q1.csv`.

a) Plot electricy demand and temperature as time series. Find the regression model for demand with temperature as an explanatory variable. Why do you think there a positive relationship?

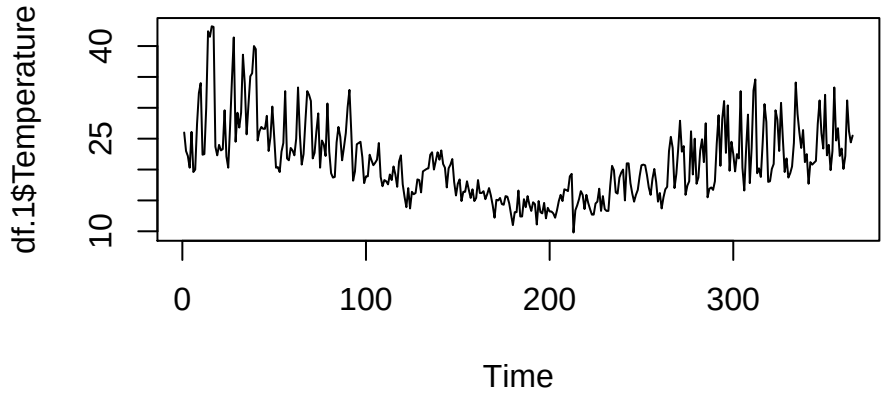
```r
df.1 <- read.csv("Q1.csv", header = T)
head(df.1)
```

```
##   X   Demand WorkDay Temperature
## 1 1 174.8963       0        26.0
## 2 2 188.5909       1        23.0
## 3 3 188.9169       1        22.2
## 4 4 173.8142       0        20.3
## 5 5 169.5152       0        26.1
## 6 6 195.7288       1        19.6
```

```r
plot.ts(df.1$Demand)
```



```r
plot.ts(df.1$Temperature)
```



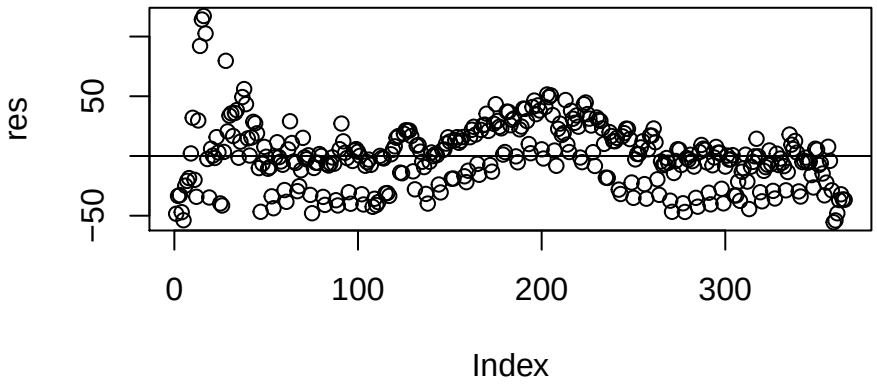

```r
summary(lm(Demand ~  Temperature, df.1))
```

```
##
## Call:
## lm(formula = Demand ~ Temperature, data = df.1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.474 -15.719  -0.336  15.767 117.184
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 212.3856     5.0080  42.409   <2e-16 ***
## Temperature   0.4182     0.2263   1.848   0.0654 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.55 on 363 degrees of freedom
## Multiple R-squared:  0.009322,   Adjusted R-squared:  0.006592
## F-statistic: 3.416 on 1 and 363 DF,  p-value: 0.0654
```

There is a positive relationship, but it is not significant. It is likely the case as temperature increases, AC is needed, thus increasing the demand for electricity.

b) Produce a residual plot. Is the model adequate? Describe any outliers or influential observations, and discuss how the model could be improved.

```r
res = resid(lm(Demand ~ Temperature, data = df.1))
plot(res)
abline(0, 0)
```



There are some outliers in the 0 index. Moreover, there seems to be a visible pattern in the residuals.

c) Use a model to forecast the electricity demand (with prediction intervals) that you would expect for the next day if the maximum temperature was 15°. Compare this with the forecast if the with maximum temperature was 35°. Do you believe these forecasts?

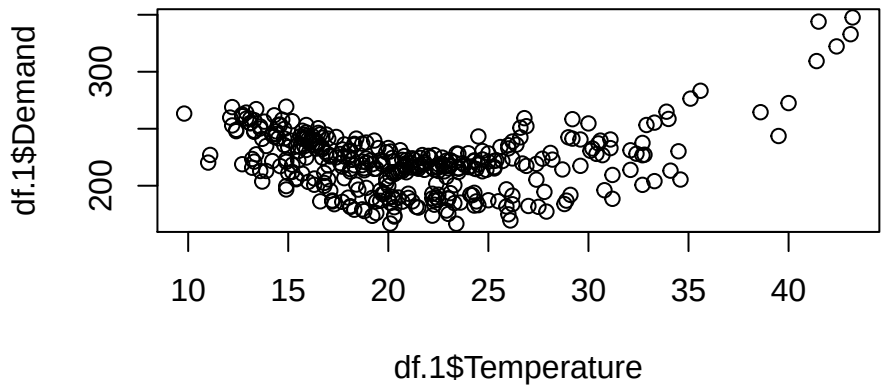
```
forecast(lm(Demand ~ Temperature, data = df.1), newdata=data.frame(Temperature=c(15,35)))
```

```
##        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 218.66       218.6592 184.4779 252.8406 166.3039 271.0146
## 227.02       227.0242 192.6585 261.3898 174.3866 279.6617
```

It's hard to believe these point forecasts because the confidence intervals are massive, and they even overlap. Hard to make much of them. This model is terrible.

d) Plot Demand vs Temperature for all of the available data in `Q1.csv`. What does this say about your model?

```
plot(df.1$Temperature, df.1$Demand)
```



There is a positive relationship between the two variables, but it is not consisitent over time. The best way to fit this data would be with a parabolic line.

Question 2 (1 point)

Elasticity

A log-log functional form is specified as $\log y = \beta_0 + \beta_1 \log x + \epsilon$

In this model, the slope β_1 can be interpreted as an *elasticity* coefficient, representing the expected percentage change in y (e.g. quantity demanded) resulting from a percentage increase in x (e.g. price).

Mathematically, the elasticity is defined as $\frac{dy}{y} \div \frac{dx}{x}$

Show that this expression is equivalent to β_1 when considering the conditional expectation of $\log y$ given x.

Log functional form

$$\log y = \beta_0 + \beta_1 \log x + \epsilon$$

Want to show

$$\frac{dy}{y} * \frac{x}{dx} = \beta_1$$

Thus, you differentiate the intial equation:

$$\frac{dy}{y} = \beta_1 * \frac{dx}{x}$$

Then, rearrange

$$\frac{dy}{y} / \frac{dx}{x} = \beta_1$$

Which comes out to what we were trying to solve for:

$$\frac{dy}{y} * \frac{x}{dx} = \beta_1$$

Question 3 (1 point)

Cross validation

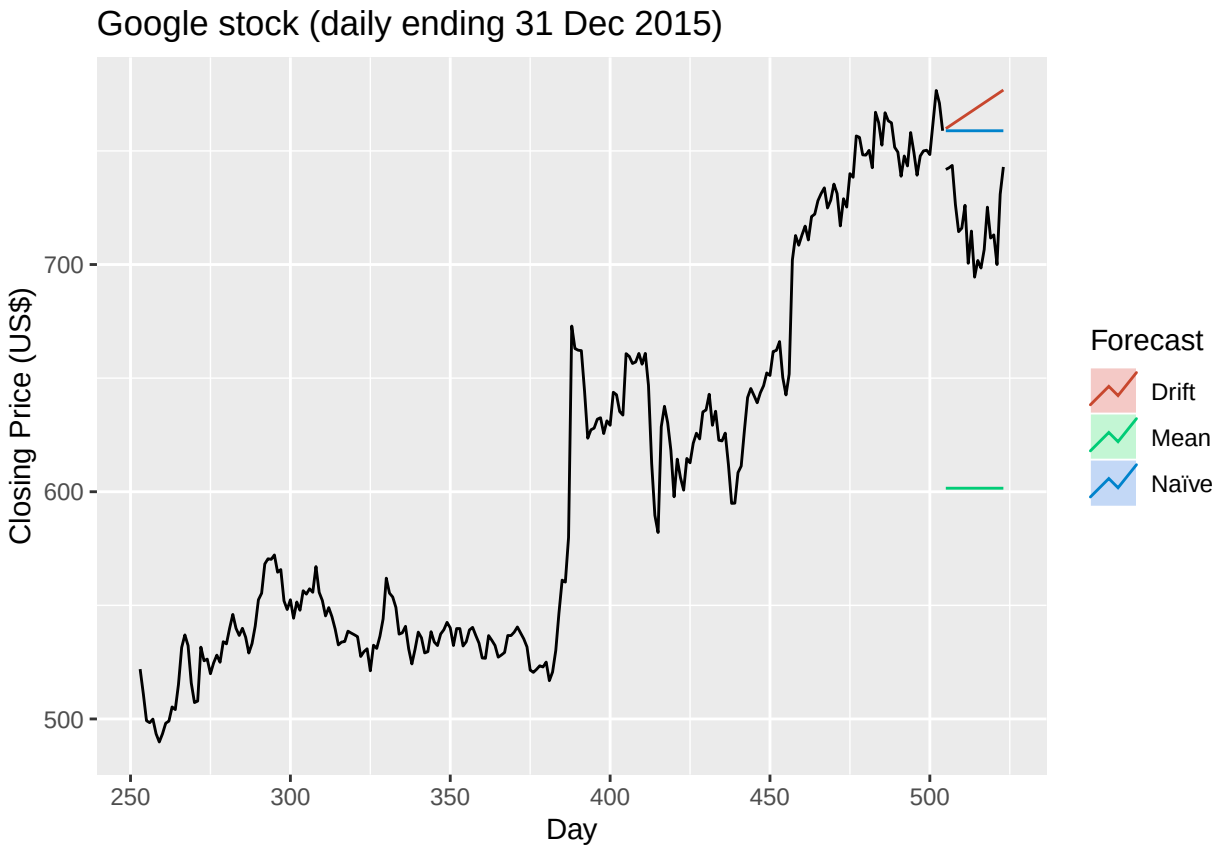
The `gafa_stock` data set from the `tsibbledata` package contains historical stock price data for Google, Amazon, Facebook and Apple. The following code fits the following models to a 2015 training set of Google stock prices:

- `MEAN()`: the *average method*, forecasting all future values to be equal to the mean of the historical data
- `NAIVE()`: the *naive method*, forecasting all future values to be equal to the value of the latest observation
- `RW()`: the *drift method*, forecasting all future values to continue following the average rate of change between the last and first observations. This is equivalent to forecasting using a model of a random walk with drift.

```
# Re-index based on trading days
google_stock <- gafa_stock %>%
  filter(Symbol == "GOOG") %>%
  mutate(day = row_number()) %>%
  update_tsibble(index = day, regular = TRUE)

# Filter the year of interest
google_2015 <- google_stock %>% filter(year(Date) == 2015)

# Fit models
google_fit <- google_2015 %>%
  model(
    Mean = MEAN(Close),
    `Naïve` = NAIVE(Close),
    Drift = RW(Close ~ drift())
  )
```

The following creates a test set of January 2016 stock prices, and plots this against the forecasts from the average, naive and drift models:

```
google_jan_2016 <- google_stock %>%
  filter(yearmonth(Date) == yearmonth("2016 Jan"))
google_fc <- google_fit %>% forecast(google_jan_2016)

# Plot the forecasts
google_fc %>%
  autoplot(google_2015, level = NULL) +
    autolayer(google_jan_2016, Close, color='black') +
    ggtitle("Google stock (daily ending 31 Dec 2015)") +
    xlab("Day") + ylab("Closing Price (US$)") +
    guides(colour=guide_legend(title="Forecast"))
```



Forecasting performance can be measured with the `accuracy()` function:

```
accuracy(google_fc, google_stock)
```

```
## # A tibble: 3 x 10
##   .model Symbol .type    ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>  <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift  GOOG   Test  -49.8  53.1  49.8 -6.99  6.99  7.84 0.604
## 2 Mean   GOOG   Test  117.  118.  117.  16.2  16.2  18.4  0.496
## 3 Naïve  GOOG   Test  -40.4  43.4  40.4 -5.67  5.67  6.36 0.496
```

These measures compare model performance over the entire test set. An alternative version of pseudo-out-of-sample forecasting is **time series cross-validation**.

In this procedure, there may be a series of 'test sets', each consisting of a limited number of observations one or multiple steps ahead. The corresponding training set consists only of observations that occurred prior to the observations that forms the test set. The forecast accuracy is computed by averaging over the test sets. Since it is not possible to obtain a reliable forecast based on a small training set, the earliest observations are not considered as test sets.

```
# Time series cross-validation accuracy
google_2015_tr <- google_2015 %>%
  slice(1:(n()-1)) %>%
  stretch_tsibble(.init = 3, .step = 1)

fc <- google_2015_tr %>%
```

```
  model(RW(Close ~ drift())) %>%
  forecast(h=60)

fc %>% accuracy(google_2015)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as
## 59 observations are missing between 505 and 563
```

```
## # A tibble: 1 x 10
##   .model             Symbol .type    ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>              <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 RW(Close ~ drift(~ GOOG   Test   24.6  74.4  48.7  3.67  7.89  6.84 0.972
```

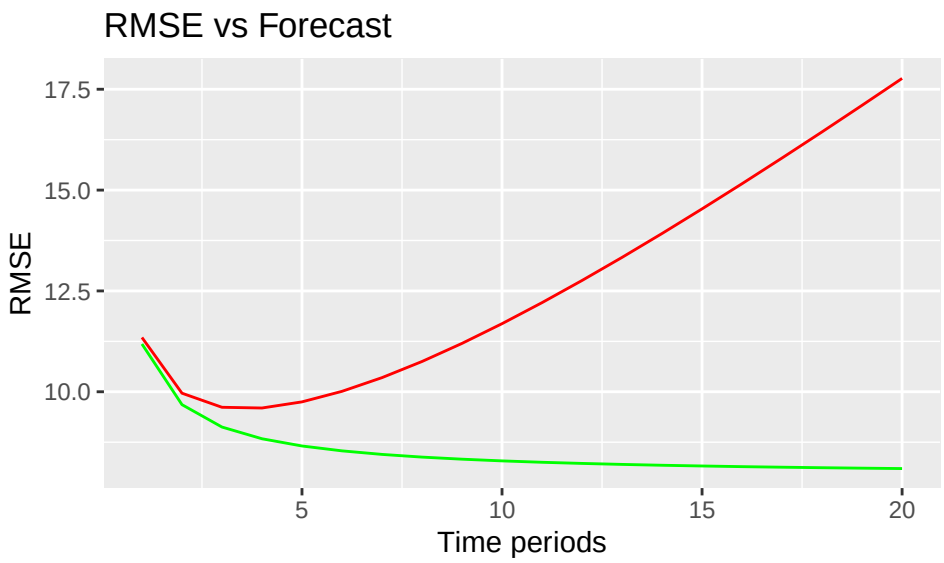
Use cross-validation to compare the forecasting accuracy of the naive and drift models, as the number of steps ahead is allowed to vary. Define the accuracy measure(s) you are using to make the comparison.

```
google_2015_tr <- google_2015 %>%
  slice(1:(n() - 1)) %>%
  stretch_tsibble(.init = 3, .step = 1)

final_df = data.frame()

# Loop over to get various values for the naive/drift model given number of steps ahead
for (p in seq(1, 20)) {
  hvalue <- p

  fc <- google_2015_tr %>%
    model(RW(Close ~ drift())) %>%
    forecast(h = hvalue)

  e <- tsCV(fc$Close, rwf, drift = TRUE, h = hvalue)

  # -------------------------------
  fc2 <- google_2015_tr %>%
    model(NAIVE(Close)) %>%
    forecast(h = hvalue)

  e2 <- tsCV(fc2$Close, rwf, h = hvalue)

  result = data.frame(h=hvalue, Drift=sqrt(mean(e*e, na.rm = TRUE)), Naive = sqrt(mean(e2*e2, r
  total_df <- rbind(final_df, result)
  final_df <- total_df
}

final_df
```

```
##    h     Drift     Naive
## 1  1 11.346552 11.184185
```

```
## 2   2  9.964540  9.680910
## 3   3  9.615630  9.125718
## 4   4  9.597825  8.835198
## 5   5  9.749816  8.656255
## 6   6 10.010789  8.534896
## 7   7 10.350498  8.447154
## 8   8 10.750890  8.380750
## 9   9 11.199811  8.328740
## 10 10 11.688427  8.286899
## 11 11 12.209994  8.252509
## 12 12 12.759213  8.223742
## 13 13 13.331831  8.199322
## 14 14 13.924390  8.178334
## 15 15 14.534050  8.160100
## 16 16 15.158460  8.144113
## 17 17 15.795658  8.129980
## 18 18 16.444000  8.117398
## 19 19 17.102099  8.106123
## 20 20 17.768778  8.095963
```

```
ggplot(final_df, aes(h)) +
  geom_line(aes(y=Drift), colour="red") +
  geom_line(aes(y=Naive), colour="green") + ggtitle("RMSE vs Forecast") +
  xlab("Time periods") + ylab("RMSE")
```



Question 4 (1 point)

Harmonic regression

A Fourier series is a periodic function composed of series of sine and cosine terms of varying frequencies. Fourier terms can be incorporated into a time series regression to model seasonal patterns. This type of model element is sometimes known as 'trigonometric seasonality'.

If m is the seasonal period, then the first few terms of a Fourier series are given by $x_{1,t} = \sin(2\pi tm)$, $x_{2,t} = \cos(2\pi tm)$, $x_{3,t} = \sin(4\pi tm)$, $x_{4,t} = \cos(4\pi tm)$, $x_{5,t} = \sin(6\pi tm)$, $x_{5,t} = \cos(6\pi tm)$

and so on.

If we have monthly seasonality, and we use the first 11 of these predictor variables, then we will get exactly the same forecasts as using 11 dummy variables.

With Fourier terms, we often need fewer predictors than with dummy variables, especially when m is large. This makes them useful for weekly data, for example, where $m \approx 52$ (for short seasonal periods (e.g., quarterly data), there is little advantage in using Fourier terms over seasonal dummy variables).

These Fourier terms are produced using the `fourier(x, K)` function in R, where `x` is a seasonal time series and `K` is the order of Fourier terms, up to a maximum of $K = m/2$. A regression model containing Fourier terms can be called a harmonic regression.

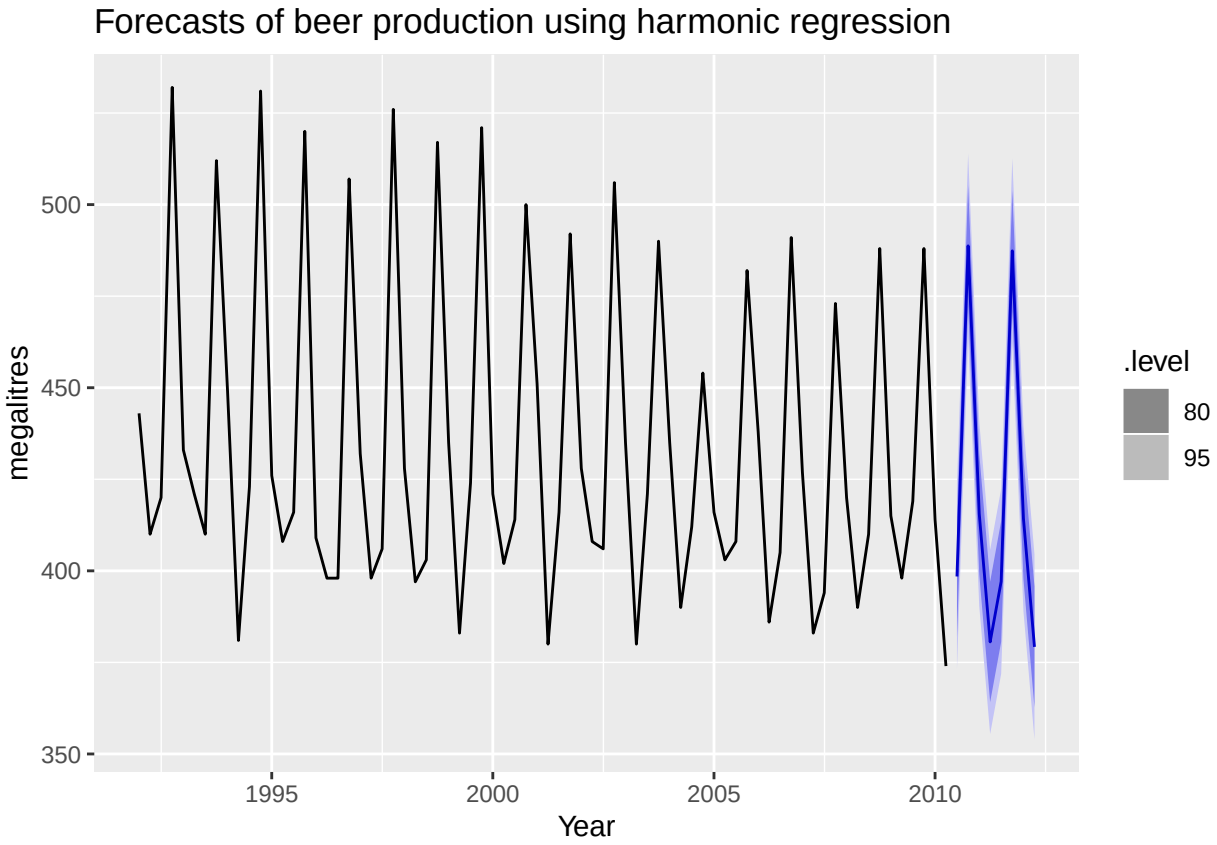
For example, Australian beer production data can be modelled like this:

```
recent_production <- aus_production %>%
  filter(year(Quarter) >= 1992)
fourier_beer <- recent_production %>%
  model(TSLM(Beer ~ trend() + fourier(K=2)))
report(fourier_beer)
```

```
## Series: Beer
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -42.9029  -7.5995  -0.4594   7.9908  21.7895
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         446.87920    2.87321 155.533  < 2e-16 ***
## trend()              -0.34027    0.06657  -5.111 2.73e-06 ***
## fourier(K = 2)C1_4    8.91082    2.01125   4.430 3.45e-05 ***
## fourier(K = 2)S1_4 -53.72807    2.01125 -26.714  < 2e-16 ***
## fourier(K = 2)C2_4 -13.98958    1.42256  -9.834 9.26e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.23 on 69 degrees of freedom
```

```
## Multiple R-squared: 0.9243,  Adjusted R-squared: 0.9199
## F-statistic: 210.7 on 4 and 69 DF, p-value: < 2.22e-16
```

```
fc_beer <- forecast(fourier_beer)
fc_beer %>%
  autoplot(recent_production) +
  ggtitle("Forecasts of beer production using harmonic regression") +
  xlab("Year") + ylab("megalitres")
```

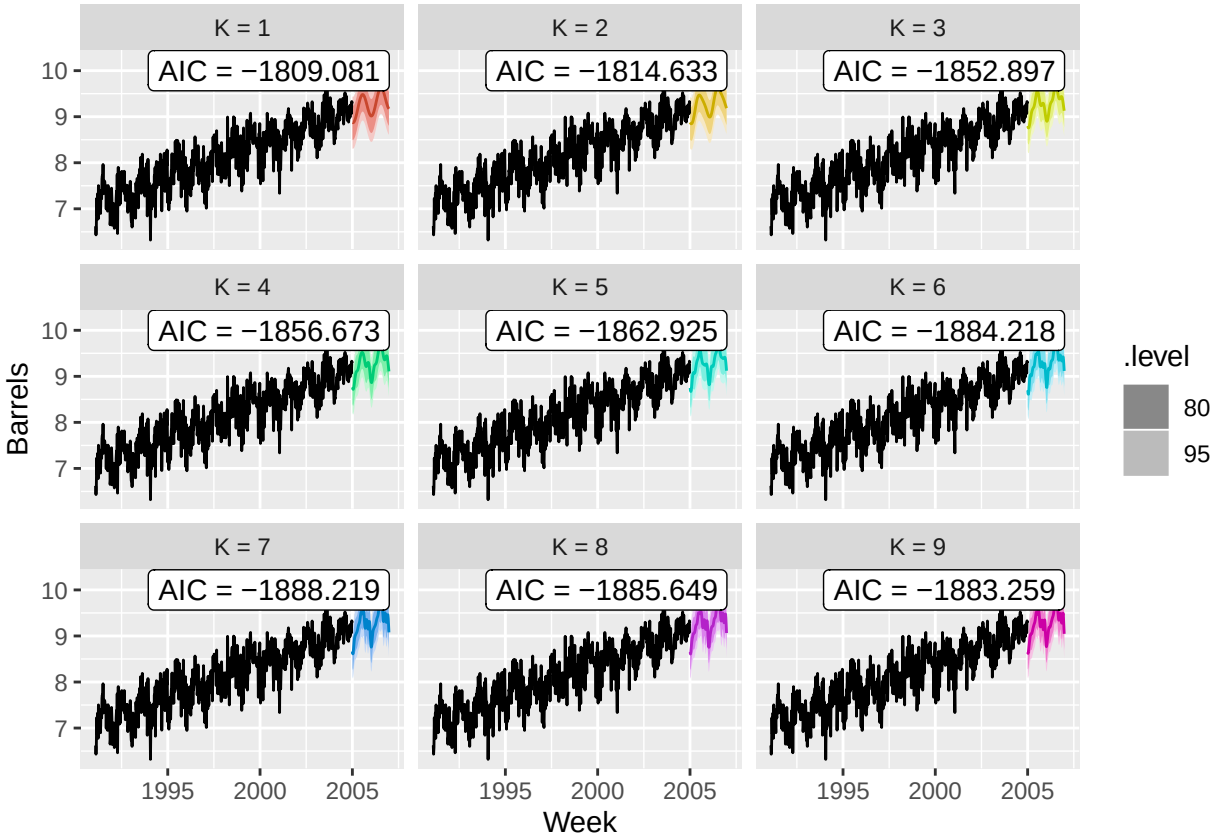


The `us_gasoline` series from the `fpp3` package consists of weekly data for supplies of US finished motor gasoline product, from 2 February 1991 to 20 January 2017. The units are in "million barrels per day". Consider only the data to the end of 2004.

a) Fit a harmonic regression with trend to the data. Select the appropriate number of Fourier terms to include by minimising the AIC value. Plot and describe the observed and fitted gasoline values.

```
# Only until end of 2004
old_production <- us_gasoline %>%
  filter(year(Week) < 2005)

# Fit various values for the fourier()
fit <- old_production %>%
  model(
    `K = 1` = TSLM(Barrels ~ fourier(K = 1) + trend()),
    `K = 2` = TSLM(Barrels ~ fourier(K = 2) + trend()),
    `K = 3` = TSLM(Barrels ~ fourier(K = 3) + trend()),
```

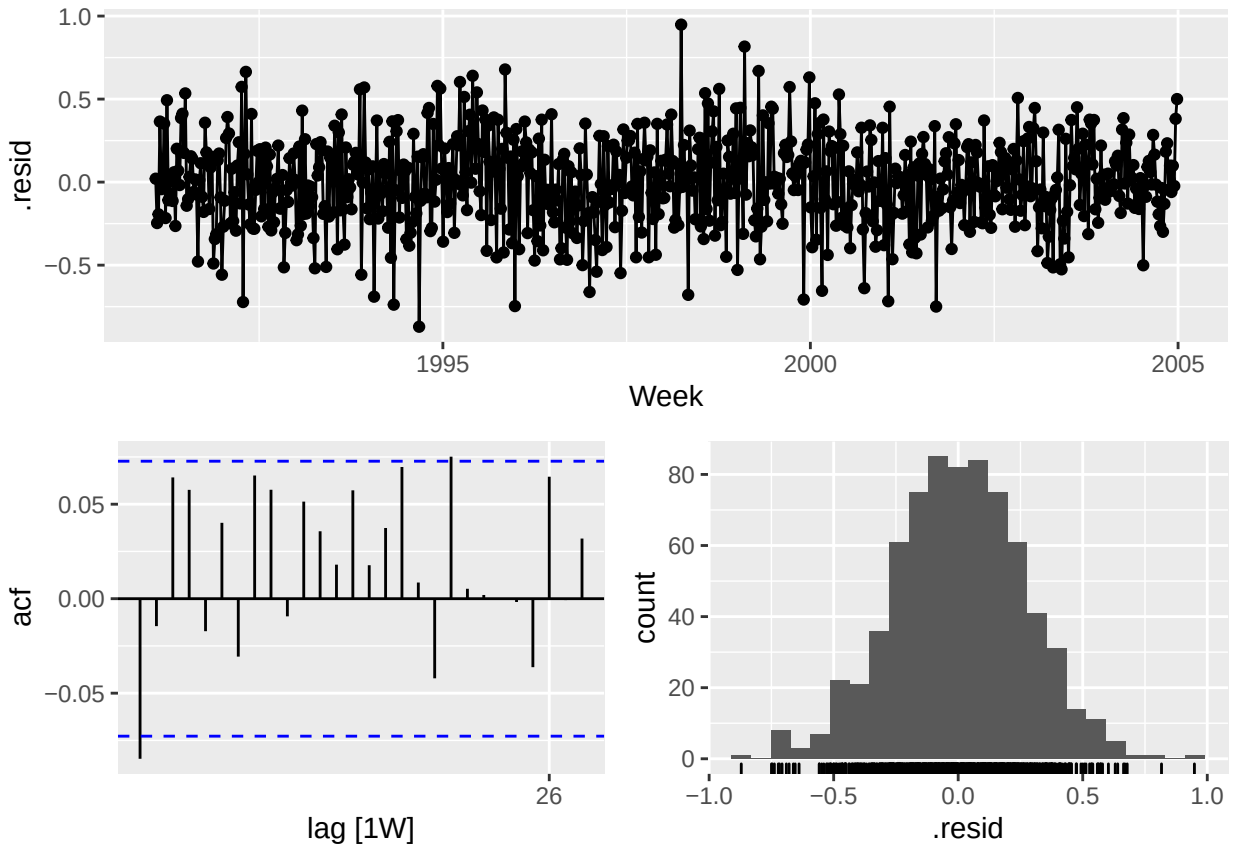
```
    `K = 4` = TSLM(Barrels ~ fourier(K = 4) + trend()),
    `K = 5` = TSLM(Barrels ~ fourier(K = 5) + trend()),
    `K = 6` = TSLM(Barrels ~ fourier(K = 6) + trend()),
    `K = 7` = TSLM(Barrels ~ fourier(K = 7) + trend()),
    `K = 8` = TSLM(Barrels ~ fourier(K = 8) + trend()),
    `K = 9` = TSLM(Barrels ~ fourier(K = 9) + trend())
  )

# Plot the various fits over 2 years to get the lowest AIC value
fit %>%
  forecast(h = "2 years") %>%
  autoplot(old_production) +
  facet_wrap(vars(.model), ncol = 3) +
  guides(colour = FALSE) +
  geom_label(
    aes(x = yearmonth("2007 Jan"), hjust=1, y = 10, label = paste0("AIC = ", format(AIC))),
    data = glance(fit)
  )
```



The lowest AIC is with K = 7, meaning I will pick that model to model the seasonal variation.

b) Check the residuals of the final model using `gg_tsresiduals()`. Use a Ljung-Box test to check for correlation in the residuals. Even though the residuals fail the correlation tests, the results are probably not severe enough to make much difference to the forecasts and prediction intervals. (Note that the correlations are reltively small, even though they are significant.)

```
# Fit model with lowest AIC
model.7 <- old_production %>%
  model(
    `K = 7` = TSLM(Barrels ~ fourier(K = 7) + trend()),
  )

# See residuals
model.7  %>% gg_tsresiduals()
```

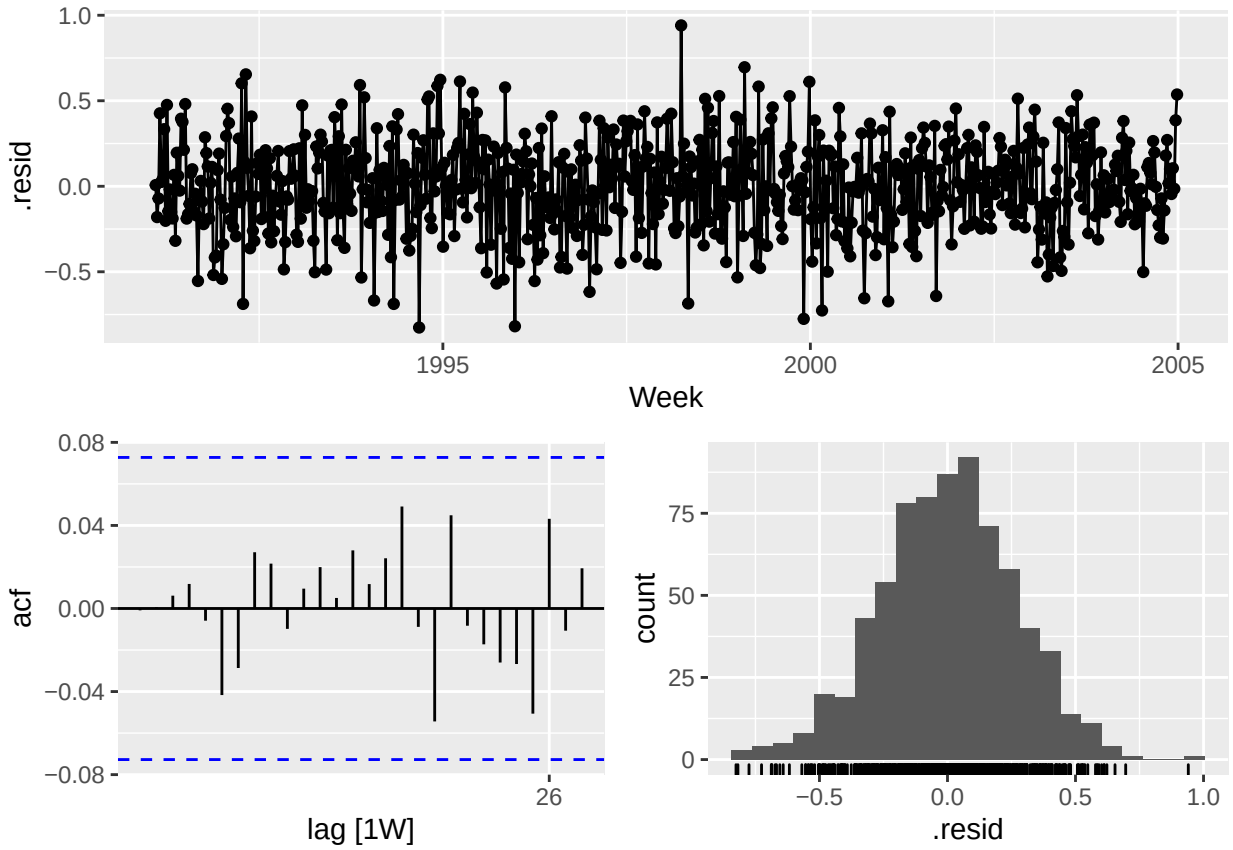



```
# Ljung Box Test
augment(model.7) %>%
  features(.resid, ljung_box, lag = 1)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 K = 7     5.23    0.0222
```

```
# -------
# Repeat the process, but use ARIMA instead
model.7.2 <- old_production %>%
  model(
    `K = 7` = ARIMA(Barrels ~ fourier(K = 7) + PDQ(0,0,0)),
  )
```

```
model.7.2  %>% gg_tsresiduals()
```




```
augment(model.7.2) %>%
  features(.resid, ljung_box, lag = 1)
```

```
## # A tibble: 1 x 3
##   .model  lb_stat lb_pvalue
##   <chr>     <dbl>     <dbl>
## 1 K = 7  0.000558     0.981
```

The Ljung-Box test shows the residuals to be white noise for the ARIMA model, but not the TSLM with the same value of the Fourier term. We can prefer the ARIMA model.

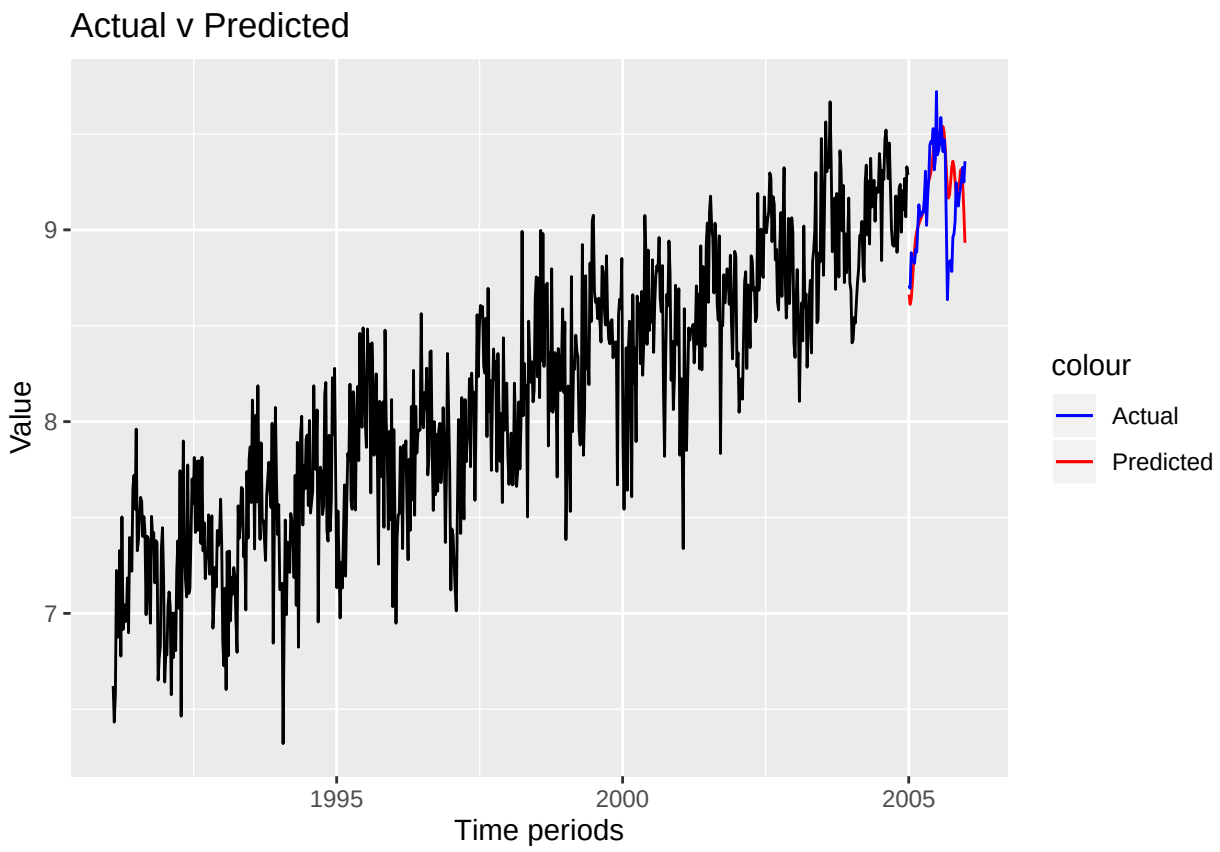
c) Plot forecasts along with the actual data for 2005. What do you find?

```
# Get predicted v actual values
prediction <- as_data_frame(model.7 %>% forecast(h=52))[2:3]
```

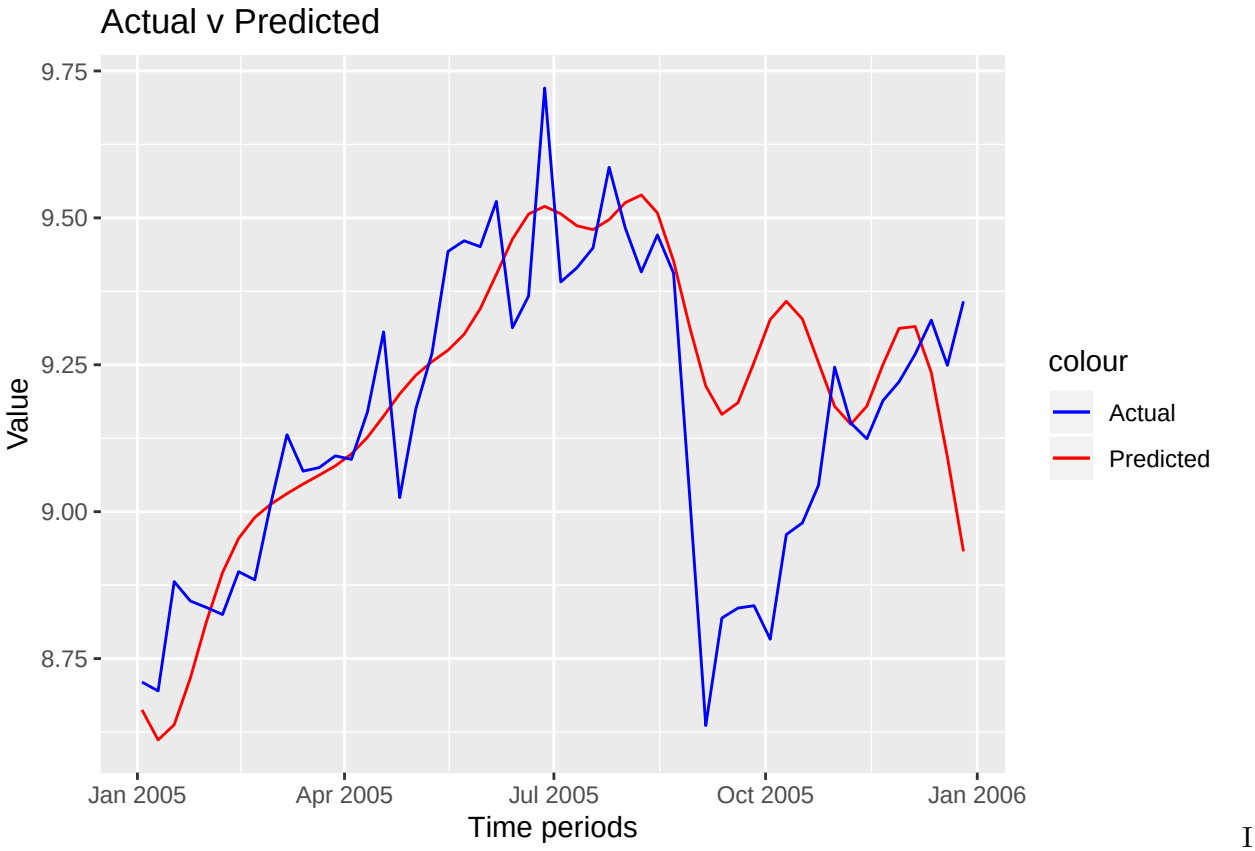
```
## Warning: `as_data_frame()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```

```
actual <- as_data_frame(us_gasoline %>% filter(year(Week) < 2006)  %>%  filter(year(Week) >200

# Combine them to graph
combined <- merge(prediction, actual, by="Week")
```

```
# Graph
ggplot(combined, aes(Week)) +
  geom_line(aes(y=Barrels.x, color = "Predicted")) +
  geom_line(aes(y=Barrels.y, color="Actual")) +
  autolayer(old_production, Barrels, color='black') +
  ggtitle("Actual v Predicted") +
  xlab("Time periods") + ylab("Value")  +
  scale_color_manual(values = c("blue","red"))
```




```
# Focus in on the year itsself
ggplot(combined, aes(Week)) +
  geom_line(aes(y=Barrels.x, color = "Predicted")) +
  geom_line(aes(y=Barrels.y, color="Actual")) +
  ggtitle("Actual v Predicted") +
  xlab("Time periods") + ylab("Value")  +
  scale_color_manual(values = c("blue","red"))
```



I see the predicted is quite smoothed, and has mainly a similar shape to the actual. The first half the series is predicted well, the 2nd half not so much so.

Question 5 (1 point):

ARIMA model

Consider `fma::sheep`, the sheep population of England and Wales from 1867–1939.

```
head(fma::sheep)
```

```
## Time Series:
## Start = 1867
## End = 1872
## Frequency = 1
## [1] 2203 2360 2254 2165 2024 2078
```

a) Produce a time plot of the time series.

```
# Show time series
sheep_ts <- as_tsibble(sheep)
sheep_ts %>% autoplot(value) + ylab("Sheep Population") + xlab("Year")
```

b) Assume you decide to fit the following model:

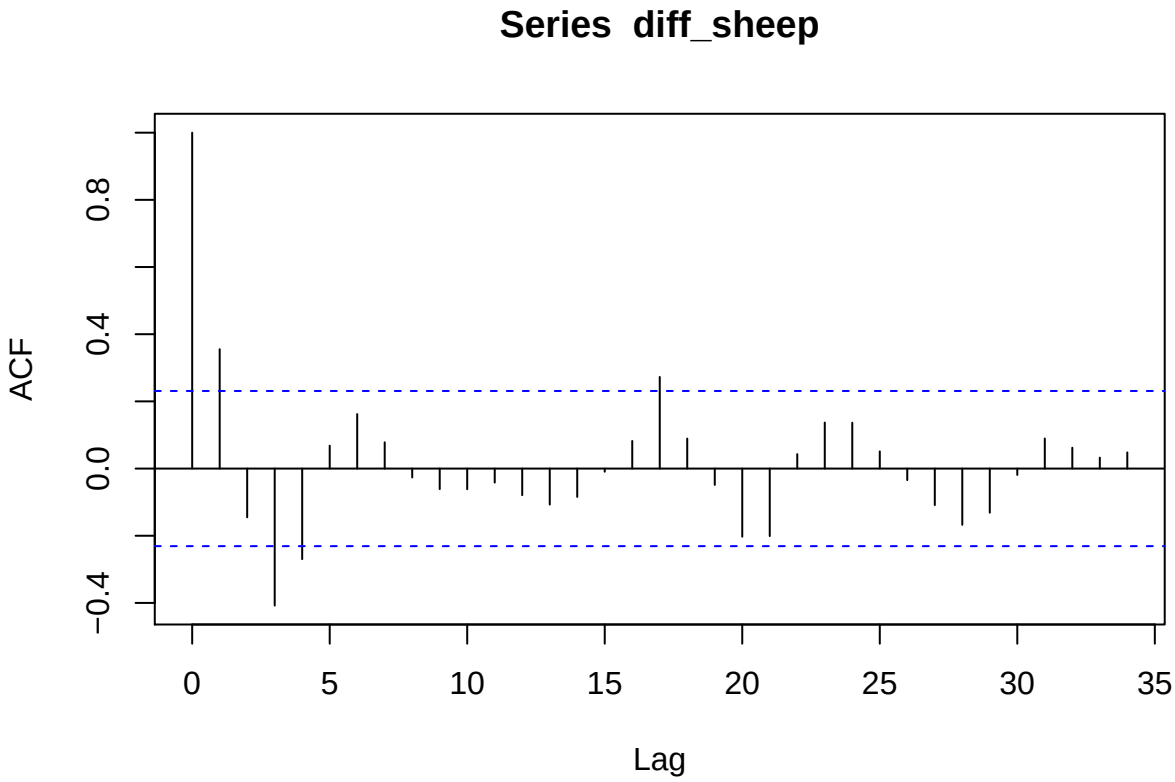
$$y_t = y_{t-1} + \phi_1(y_{t-1} - y_{t-2}) + \phi_2(y_{t-2} - y_{t-3}) + \phi_3(y_{t-3} - y_{t-4}) + \epsilon_t$$

where ϵ_t is a white noise series. What sort of ARIMA model is this (i.e., what are p, d, and q)?

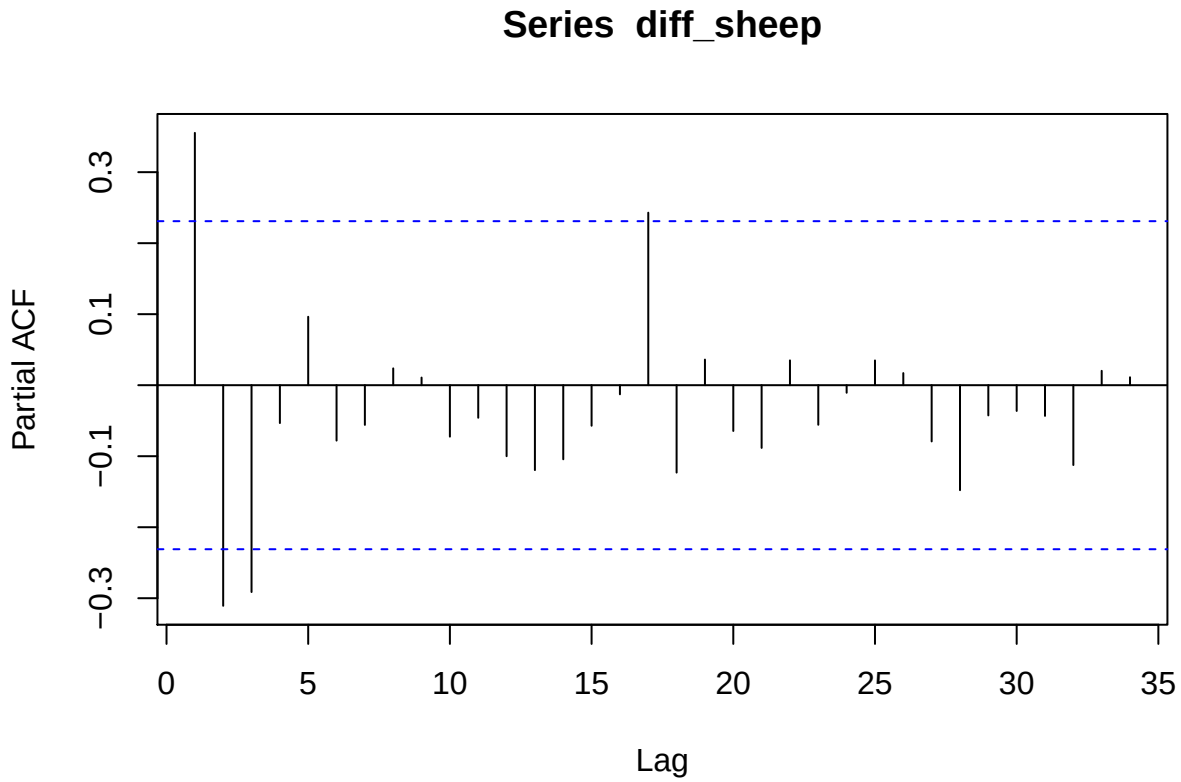
$$ARIMA(3, 1, 0)$$

c) By examining the ACF and PACF of the differenced data, explain why this model is appropriate.

```
# Differenced series
diff_sheep <- diff(sheep_ts$value, differences=1)
acf(diff_sheep, lag.max=34)
```




```
pacf(diff_sheep, lag.max=34)
```



When look at the ACF, there is a line at lag 3 which far overshoots the CI. Thus, we would use an AR(3) process in order to model it.

d) The last five values of the series are given below:

Year	1935	1936	1937	1938	1939
Millions of sheep	1648	1665	1627	1791	1797

The estimated parameters are $\phi_1 = 0.42$, $\phi_2 = -0.20$, and $\phi_3 = -0.30$.

Without using the forecast function, calculate forecasts for the next three years (1940–1942).

```
# Use formulas based of phi
value1939 = 1797
value1938 = 1791
value1937 = 1627
value1936 = 1665
phi1 = 0.42
phi2 = -0.20
phi3 = -0.30

predicted1940 <-  phi1*(value1939 - value1938) + phi2*(value1938 - value1937) + phi3*(value193

predicted1941 <- phi1*(predicted1940 - value1939) + phi2*(value1939 - value1938) + phi3*(value

predicted1942 <- phi1*(predicted1941 - predicted1940) + phi2*(predicted1940 - value1939) + phi


predicted1940
```

```
## [1] 1778.12
```

```
predicted1941
```

```
## [1] 1719.79
```

```
predicted1942
```

```
## [1] 1697.268
```

Question 6 (1 point):

Part 1

Backshift Operator Expression

Write down the following two models in terms of (1) backshift operators and (2) the fully-expressed form as y_t as a function of lags of y_t and the shock ω_t.

For example, for the $ARIMA(1,0,1)(0,0,0)_4$ model, you would write down:

1. $(1-\phi_1 B)y_t = (1+\theta_1 B)\omega_t$
2. $y_t = \phi_1 y_{t-1} + \omega_t + \theta_1 \omega_{t-1}$

a) $ARIMA(2,0,2)(1,0,1)_4$

$$(1-\Phi_1 B^4)(1-\phi_B-\phi_2 B^2)y_t = (1-\theta_1 B+\theta_2 B^2)(1+\Theta_1 B^4)\omega_t$$

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \Phi y_{t-4} + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \Theta \omega_{t-4} + \omega_t$$

b) $ARIMA(2,1,2)(1,1,1)_4$

$$(1-\Phi_1 B^4)(1-\phi_B-\phi_2 B^2)(1-B^4)(1-B)y_t = (1+\theta_2 B+\theta_2 B^2)(1+\Theta_1 B^4)\omega_t$$

$$y_t = y_{t-1} + y_{t-4} + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \Phi y_{t-4} + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \Theta \omega_{t-4} + \omega_t$$

Part 2

Parameter Redundancy, Stationarity, and Invertibility

In each of the following cases, (1) check for parameter redundancy and ensure that the $ARMA(p,q)$ notation is expressed in the simplest form, and (2) determine whether they are stationary and/or invertible.

a)

$$y_t = y_{t-1} - \frac{1}{4}y_{t-2} + \omega_t + \frac{1}{2}\omega_{t-1}$$

This is stationary, because the root of the AR process is |2|. Moreover, the root of the MA process is also 2, meaning it is invertible. There is no parameter redundancy.

b)

$$y_t = \frac{7}{10}y_{t-1} - \frac{1}{10}y_{t-2} + \omega_t + \frac{3}{2}\omega_{t-1}$$

This is stationary, because the root of the AR process is |2|. Moreover, the root of the MA process is 2/3, meaning it is not invertible. There is no parameter redundancy.

Question 7 (1 point):

Seasonal ARIMA model

Download the series of E-Commerce Retail Sales as a Percent of Total Sales from:

https://fred.stlouisfed.org/series/ECOMPCTNSA

(Feel free to explore the `fredr` package and API if interested.)

Build a Seasonal ARIMA model for this series, following all appropriate steps for a univariate time series model: checking the raw data, conducting a thorough EDA, justifying all modeling decisions (including transformation), testing model assumptions, and clearly articulating why you chose your given model. Measure and discuss your model's in-sample and pseudo-out-of-sample model performance, including with cross-validation. Use your model to generate a twelve-month forecast, and discuss its plausibility.

Data Processing

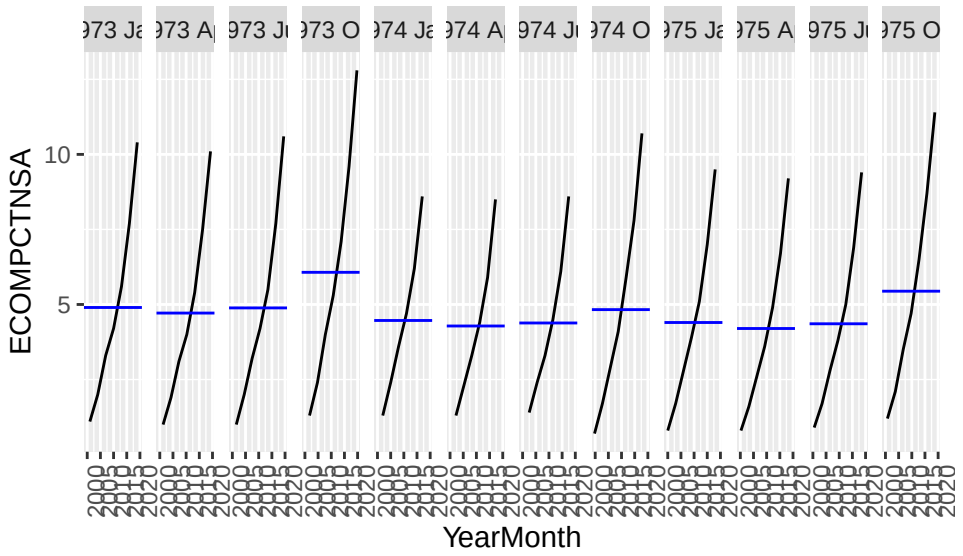
```
df.7 <- read.csv('ECOMPCTNSA.csv', header = TRUE, stringsAsFactors = FALSE)
df.7$DATE <- as.Date(fast_strptime(df.7$DATE, '%Y-%m-%d'))
df.7.tsibble <- as_tsibble(df.7, index = DATE)
# Make sure there are no gaps in the data
df.7.tsibble <- df.7.tsibble %>%
  mutate(YearMonth = yearmonth(as.character(DATE))) %>%
  as_tsibble(index = YearMonth)  %>% select(ECOMPCTNSA, YearMonth)
head(df.7.tsibble)
```

```
## # A tsibble: 6 x 2 [3M]
##   ECOMPCTNSA YearMonth
##        <dbl>     <mth>
## 1        0.7  1999 Oct
## 2        0.8  2000 Jan
## 3        0.8  2000 Apr
## 4        0.9  2000 Jul
## 5        1.2  2000 Oct
## 6        1.1  2001 Jan
```

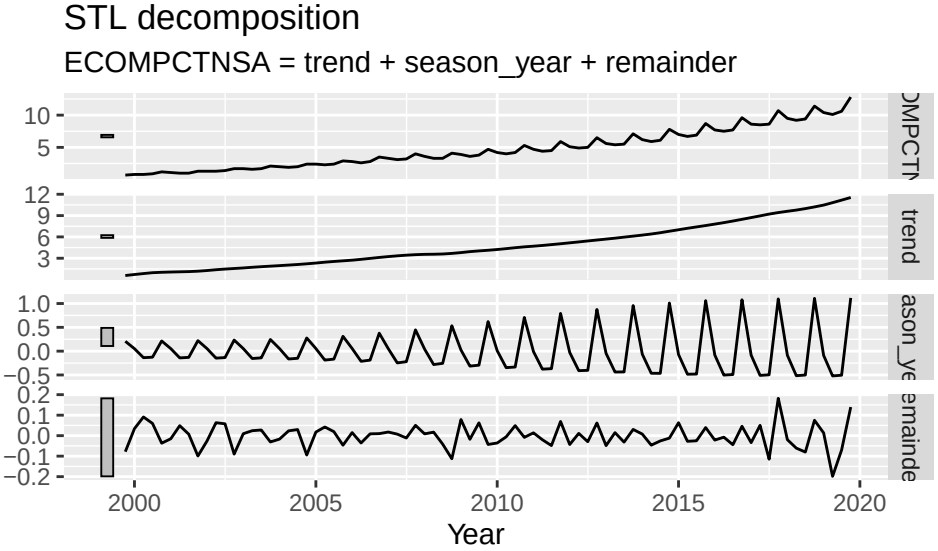
We can see the case for seasonality

```
df.7.tsibble %>% gg_subseries(period = 12)
```

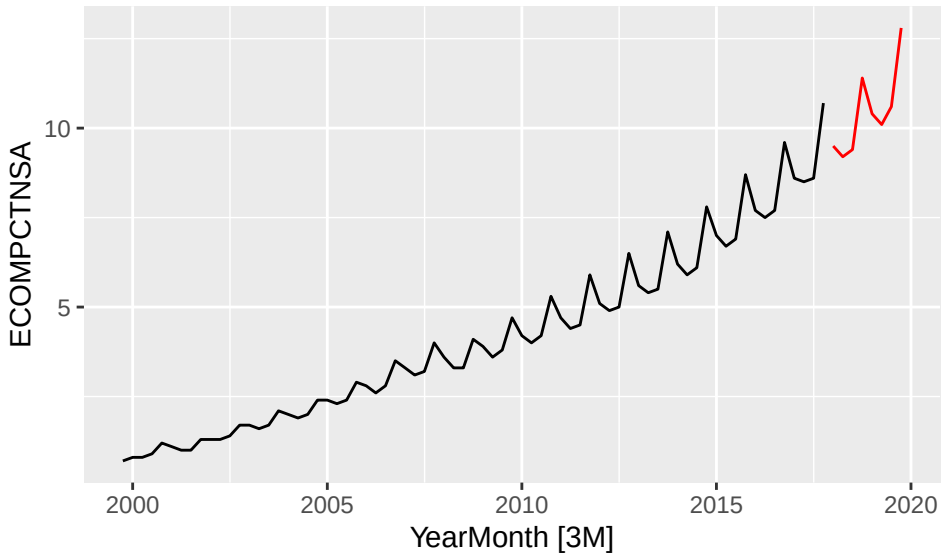
```
## Plot variable not specified, automatically selected `y = ECOMPCTNSA`
```




```
dcmp <- df.7.tsibble %>%
  model(STL(ECOMPCTNSA))

components(dcmp) %>% autoplot() + xlab("Year")
```




```
# 73/8 split on test and train data
df.7.tsibble.train <-  df.7.tsibble %>% filter(YearMonth < yearmonth('2018 Jan'))
df.7.tsibble.test<- df.7.tsibble %>% filter(YearMonth >= yearmonth('2018 Jan'))

# The graph with the ACTUAL train data
autoplot(df.7.tsibble.train, ECOMPCTNSA) +
  autolayer(df.7.tsibble.test, ECOMPCTNSA, colour = 'red')
```

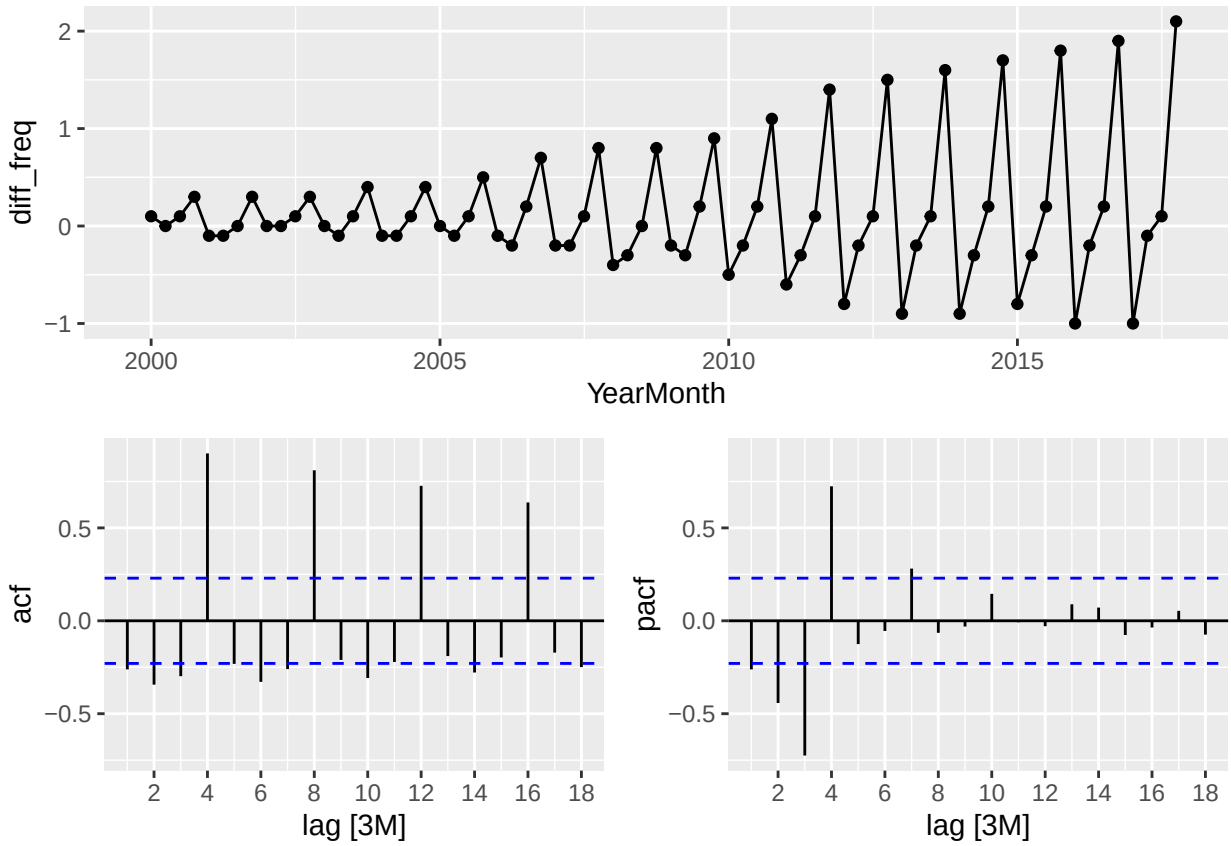


Here, we examine the graph as we difference the values. A single difference is not that strong, as there is still a heavy seasonal pattern. It seems as if differencing along with non seasonal differencing gives the best outcome.

```
df.7.tsibble.train <-  mutate(df.7.tsibble.train,
                              diff_freq = difference(ECOMPCTNSA),
                              seasdiff_freq = difference(ECOMPCTNSA, lag = 4),
                              diffseasdiff_freq = difference(difference(ECOMPCTNSA, lag = 4


df.7.tsibble.train %>% gg_tsdisplay(diff_freq, plot_type = 'partial')
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```

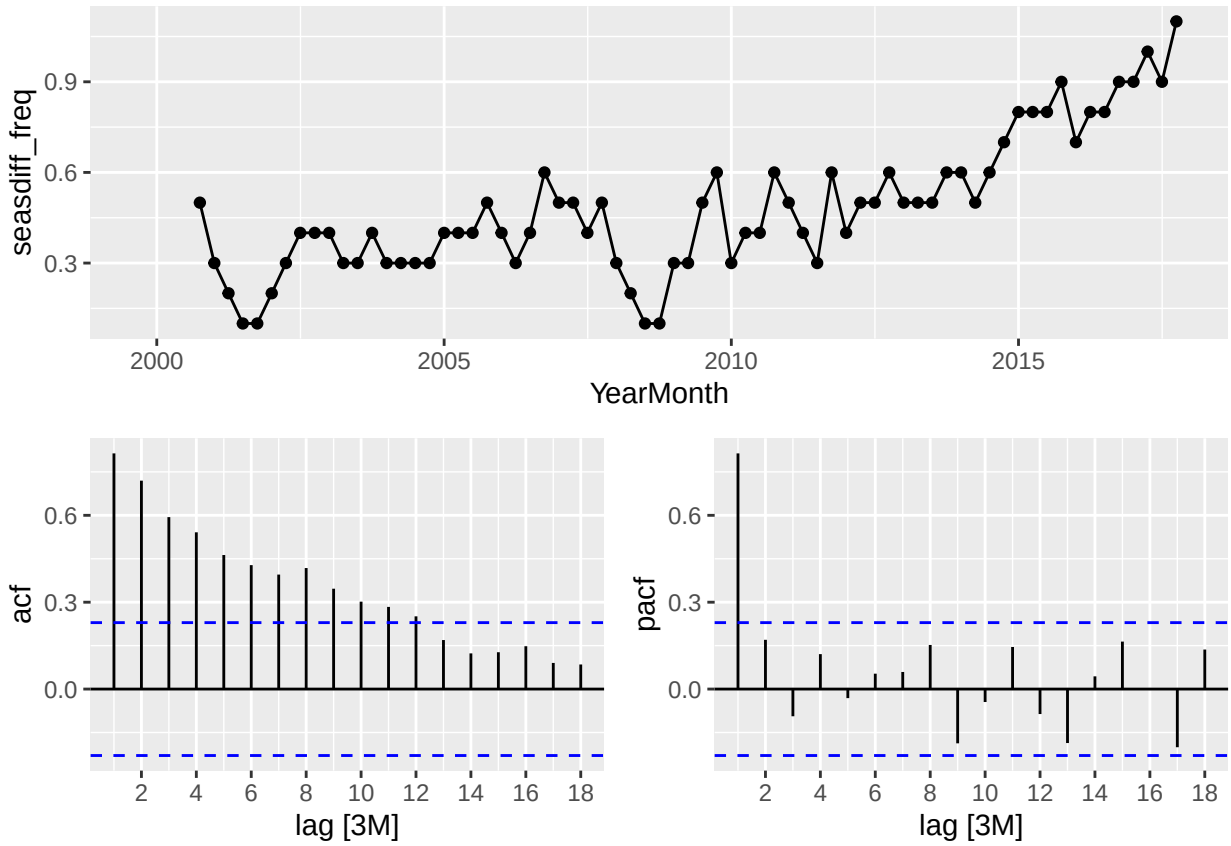
```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
df.7.tsibble.train %>% gg_tsdisplay(seasdiff_freq, plot_type = 'partial')
```

```
## Warning: Removed 4 rows containing missing values (geom_path).
```

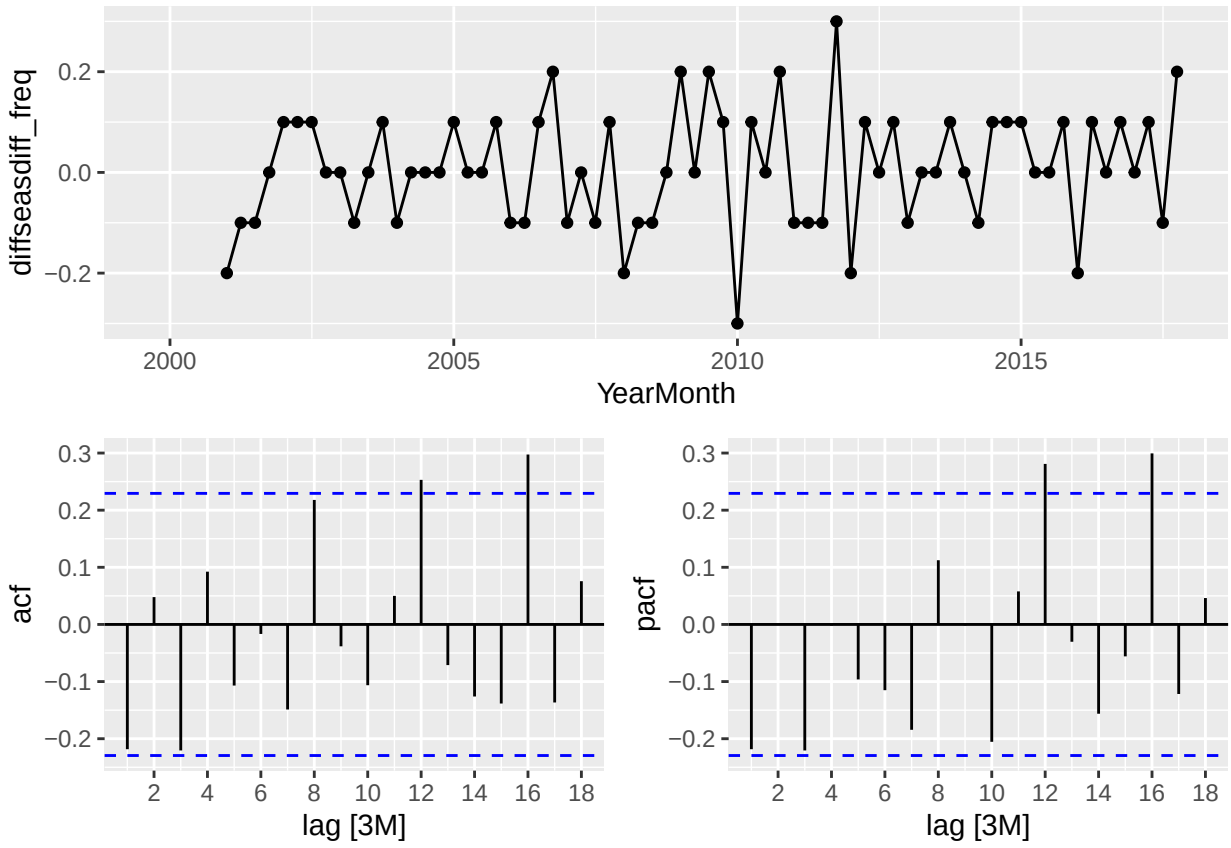
```
## Warning: Removed 4 rows containing missing values (geom_point).
```




```
df.7.tsibble.train %>% gg_tsdisplay(diffseasdiff_freq, plot_type = 'partial')
```

```
## Warning: Removed 5 rows containing missing values (geom_path).
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```



We start by modeling the nonseasonal-ARIMA component.

```
for (p in 0:2) {
  for (q in 0:2) {
    tryCatch({
      fit <- df.7.tsibble.train %>% model(arima = ARIMA(ECOMPCTNSA ~ pdq(p, 1, q) + PDQ(0, 0, 0
      print(paste(p, q, fit$arima[[1]]$fit[[3]]$AIC))
    },
    error = function(e) {
    })

  }
}
```

```
## [1] "0 0 147.726488765469"
## [1] "0 1 119.529296561541"
## [1] "0 2 98.1040428186016"
## [1] "1 0 143.95814692223"
## [1] "1 1 121.367280576153"
## [1] "1 2 123.309766968571"
## [1] "2 0 126.612817074058"
## [1] "2 1 101.839411662438"
## [1] "2 2 52.6697779029157"
```

The lowest AIC is with an ARIMA (2, 1, 2) model. Now move on to the seasonal part.

```
for (P in 0:1) {
  for (Q in 0:1) {
    tryCatch({
      fit <- df.7.tsibble.train %>% model(arima = ARIMA(ECOMPCTNSA ~ pdq(2, 1, 2) + PDQ(P, 1, 
      print(paste(P, Q, fit$arima[[1]]$fit[[3]]$AIC))
    },
    error = function(e) {
    })

  }
}
```

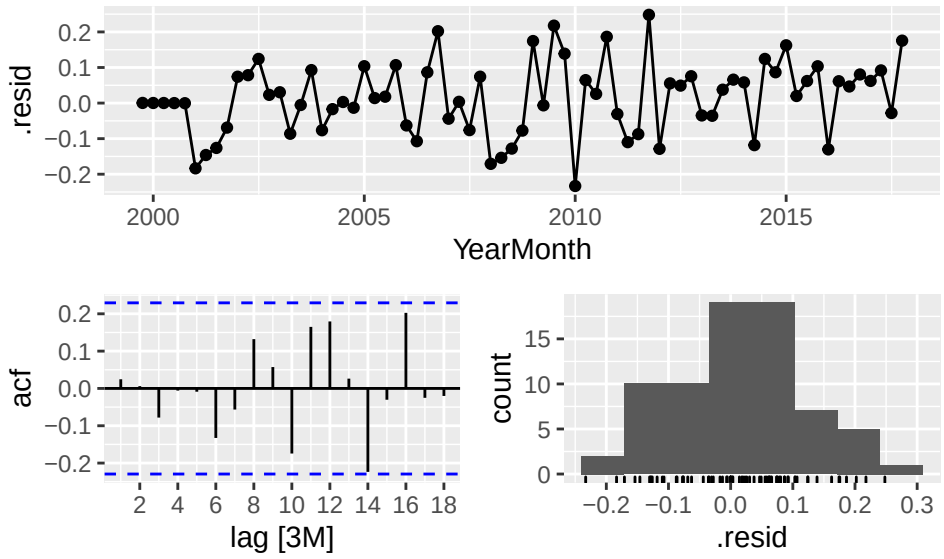
```
## [1] "0 0 -101.039853228265"
## [1] "0 1 -99.0403859783124"
## [1] "1 0 -99.038731158674"
## [1] "1 1 -97.4173283773743"
```

Lowest AIC is in (0,0)

```
fit <- df.7.tsibble.train %>% model(arima = ARIMA(ECOMPCTNSA ~ pdq(2, 1, 2) + PDQ(0, 1, 0)))
fit %>% report()
```

```
## Series: ECOMPCTNSA
## Model: ARIMA(2,1,2)(0,1,0)[4]
##
## Coefficients:
##           ar1     ar2     ma1      ma2
##       -0.4196  0.5744  0.1873  -0.7586
## s.e.   0.2039  0.2003  0.1579   0.1415
##
## sigma^2 estimated as 0.01196:  log likelihood=55.52
## AIC=-101.04   AICc=-100.07   BIC=-89.94
```

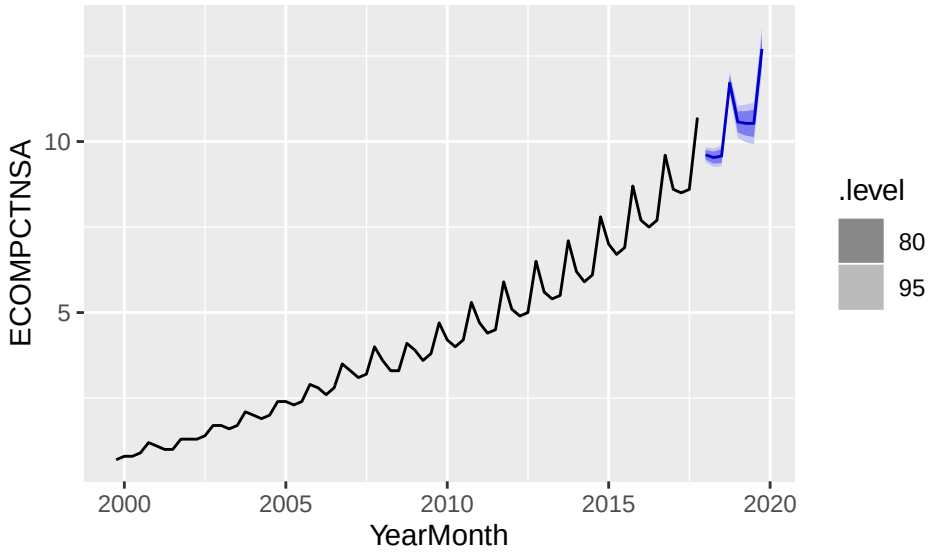
```
fit %>% gg_tsresiduals()
```



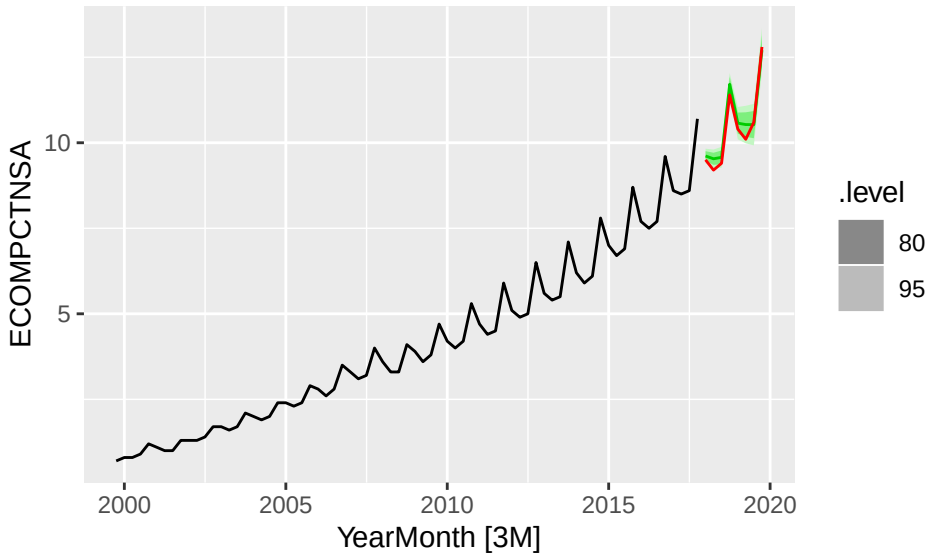

```
augment(fit) %>% features(.resid, ljung_box)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 arima   0.0458     0.831
```

```
fit %>% forecast(h=8) %>% autoplot(df.7.tsibble.train)
```

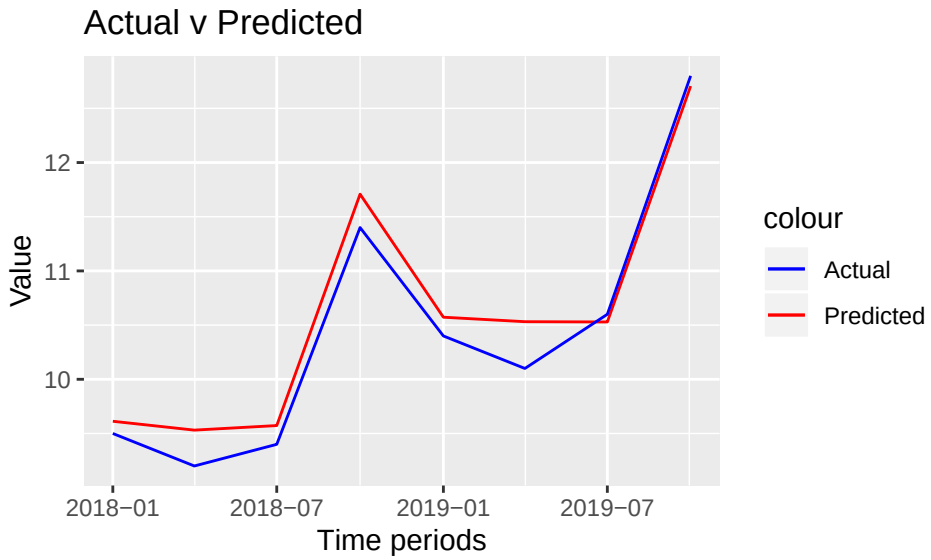



```
predicted <- fit %>% forecast(h=8)

autoplot(df.7.tsibble.train, ECOMPCTNSA) +
  autolayer(predicted, ECOMPCTNSA, colour = 'green') +
  autolayer(df.7.tsibble.test, ECOMPCTNSA, colour = 'red')
```



Conduct out of sample tests

```
predicted <- as_data_frame(fit %>% forecast(h=8))[2:3]
combined <- merge(predicted, df.7.tsibble.test, by="YearMonth")

ggplot(combined, aes(YearMonth)) +
  geom_line(aes(y=ECOMPCTNSA.x, color = "Predicted")) +
  geom_line(aes(y=ECOMPCTNSA.y, color="Actual")) +
  ggtitle("Actual v Predicted") +
  xlab("Time periods") + ylab("Value")  +
  scale_color_manual(values = c("blue","red"))
```




```
# Find the end RMSE
accuracy(combined$ECOMPCTNSA.y, combined$ECOMPCTNSA.x)
```

```
##                 ME      RMSE       MAE      MPE     MAPE
## Test set 0.1705183 0.2442204 0.2118001 1.67531 2.029043
```

12 month forecast looks extremely accurate vs the actual values–it is plausible.

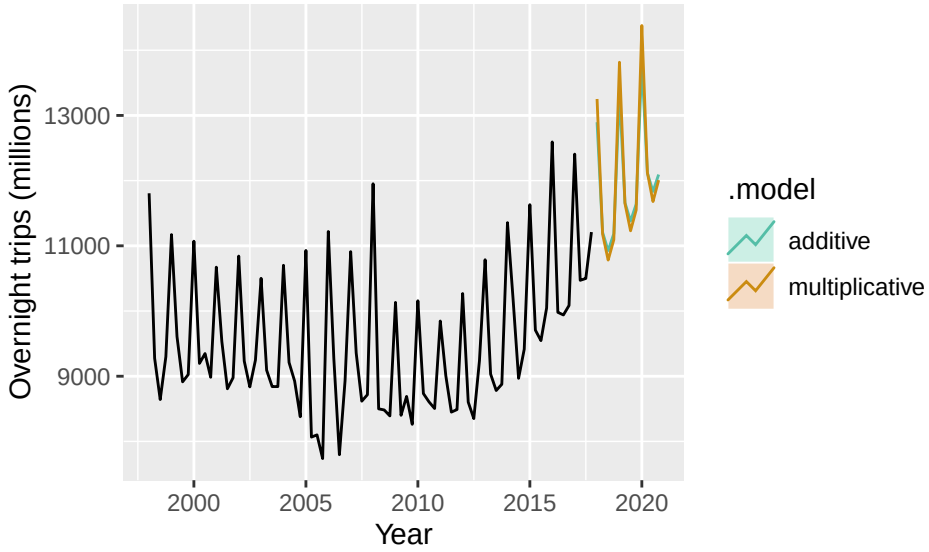
Question 8 (1 point):

Model averaging

The `HoltWinters()` function from the base R `stats` package computes a Holt-Winters Filtering of a time series. This is a classical form of exponential smoothing model, an approach to time series modeling that predates Box and Jenkins' ARIMA methodology. Exponential smoothing models are categorized by error, trend and seasonal components, which if present may be additive or multiplicative. Detail is given in the (optional) readings from Cowpertwait and Metcalfe (Chapter 3.4) and Hyndman and Athanasopoulos (Chapter 8.3).

The Holt-Winters method (in additive and multiplicative variants) can also be applied using the `ETS()` function from the `fable` package, as per the following example:

```
aus_holidays <- tourism %>%
  filter(Purpose == "Holiday") %>%
  summarise(Trips = sum(Trips))

# using ETS() function from fable
fit <- aus_holidays %>%
  model(
    additive = ETS(Trips ~ error("A") + trend("A") + season("A")),
    multiplicative = ETS(Trips ~ error("M") + trend("A") + season("M"))
  )
fc <- fit %>% forecast(h = "3 years")

fc %>%
  autoplot(aus_holidays, level = NULL) + xlab("Year") +
  ylab("Overnight trips (millions)") +
  scale_color_brewer(type = "qual", palette = "Dark2")
```



Apply a Holt-Winters model to the ECOMPCTNSA time series from Question 7, and compare its forecasting performance to that of the ARIMA model you developed. Then compare both to the

performance of a simple average of the ARIMA and Holt-Winters models.

```
# using ETS() function from fable
fit <- df.7.tsibble.train %>%
  model(
    multiplicative = ETS(ECOMPCTNSA ~ error("M") + trend("A") + season("M"))
  )
fc <- fit %>% forecast(h = 8)
fc
```

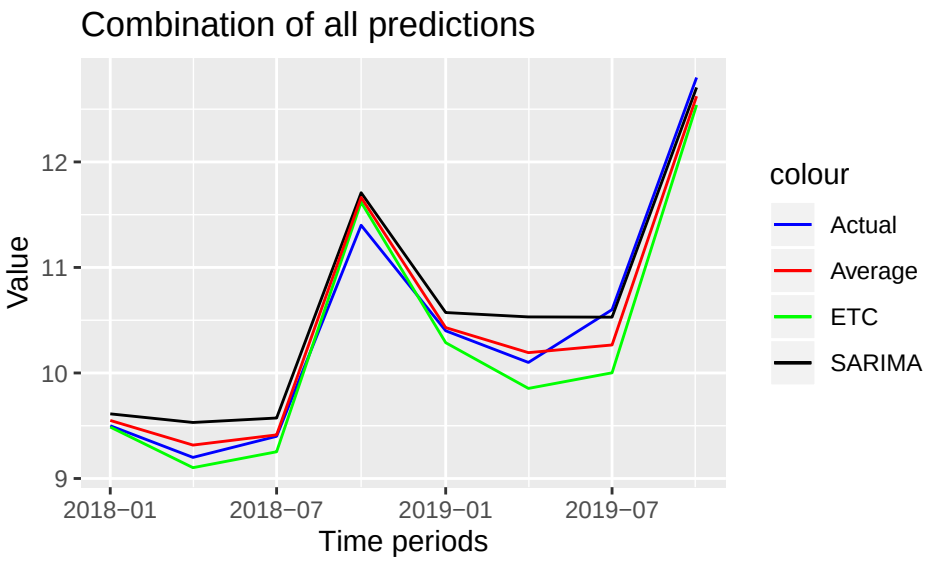
```
## # A fable: 8 x 4 [3M]
## # Key:     .model [1]
##   .model         YearMonth ECOMPCTNSA .distribution
##   <chr>              <mth>      <dbl> <dist>
## 1 multiplicative  2018 Jan       9.49 N( 9.5, 0.11)
## 2 multiplicative  2018 Apr       9.10 N( 9.1, 0.20)
## 3 multiplicative  2018 Jul       9.25 N( 9.3, 0.32)
## 4 multiplicative  2018 Oct      11.6  N(11.6, 0.71)
## 5 multiplicative  2019 Jan      10.3  N(10.3, 0.81)
## 6 multiplicative  2019 Apr       9.85 N( 9.9, 0.95)
## 7 multiplicative  2019 Jul      10.0  N(10.0, 1.21)
## 8 multiplicative  2019 Oct      12.5  N(12.5, 2.33)
```

```
# Combined dataframes, predictions, and average
fc <- as_data_frame(fc <- fit %>% forecast(h = 8))[2:3]
full_combined <- merge(predicted, fc, by="YearMonth")
full_combined <- merge(df.7.tsibble.test, full_combined, by="YearMonth")
colnames(full_combined)[2] <- "Actual"
colnames(full_combined)[3] <- "SARIMA"
colnames(full_combined)[4] <- "ETC"
full_combined$Average <- (full_combined$SARIMA + full_combined$ETC)/2
full_combined
```

```
##   YearMonth Actual    SARIMA       ETC   Average
## 1  2018 Jan    9.5  9.612639  9.487714  9.550176
## 2  2018 Apr    9.2  9.531109  9.102619  9.316864
## 3  2018 Jul    9.4  9.573178  9.253319  9.413248
## 4  2018 Oct   11.4 11.708096 11.618533 11.663315
## 5  2019 Jan   10.4 10.572807 10.287130 10.429968
## 6  2019 Apr   10.1 10.531445  9.853864 10.192655
## 7  2019 Jul   10.6 10.529129 10.001659 10.265394
## 8  2019 Oct   12.8 12.705744 12.539654 12.622699
```

```
ggplot(full_combined, aes(YearMonth)) +
  geom_line(aes(y=Actual, color = "Actual")) +
  geom_line(aes(y=SARIMA, color="SARIMA")) +
  geom_line(aes(y=ETC, color="ETC")) +
  geom_line(aes(y=Average, color="Average")) +
  ggtitle("Combination of all predictions") +
  xlab("Time periods") + ylab("Value")  +
```

```
  scale_color_manual(values = c("blue","red", "green", "black"))
```




```
accuracy(full_combined$Actual, full_combined$ETC)
```

```
##                     ME      RMSE       MAE       MPE     MAPE
## Test set -0.1569384 0.2687983 0.2115718 -1.569654 2.039879
```

```
accuracy(full_combined$Actual, full_combined$SARIMA)
```

```
##                   ME      RMSE       MAE     MPE     MAPE
## Test set 0.1705183 0.2442204 0.2118001 1.67531 2.029043
```

```
accuracy(full_combined$Actual, full_combined$Average)
```

```
##                      ME      RMSE       MAE         MPE     MAPE
## Test set 0.006789957 0.1726864 0.1347667 0.08878631 1.25483
```

The Average of the two models seems to give the best result.

Question 10 (1 point):

Vector autoregression

Annual values for real mortgage credit (RMC), real consumer credit (RCC) and real disposable personal income (RDPI) for the period 1946-2006 are recorded in `Q10.csv`. All of the observations are measured in billions of dollars, after adjustment by the Consumer Price Index (CPI). Develop a VAR model for these data for the period 1946-2003, and then forecast the last three years, 2004-2006. Examine the relative advantages of a logarithmic transform and the use of differences.

```
# Redo with log and differences
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(vars)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following objects are masked from 'package:fma':
##
##     cement, housing, petrol
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following object is masked from 'package:tsibble':
##
##     index
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

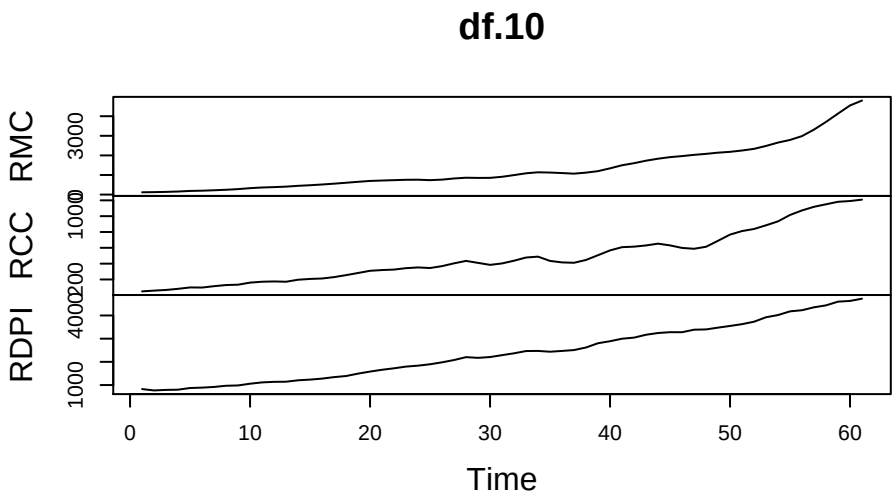
```
## Loading required package: urca
```

```
## Loading required package: lmtest
```

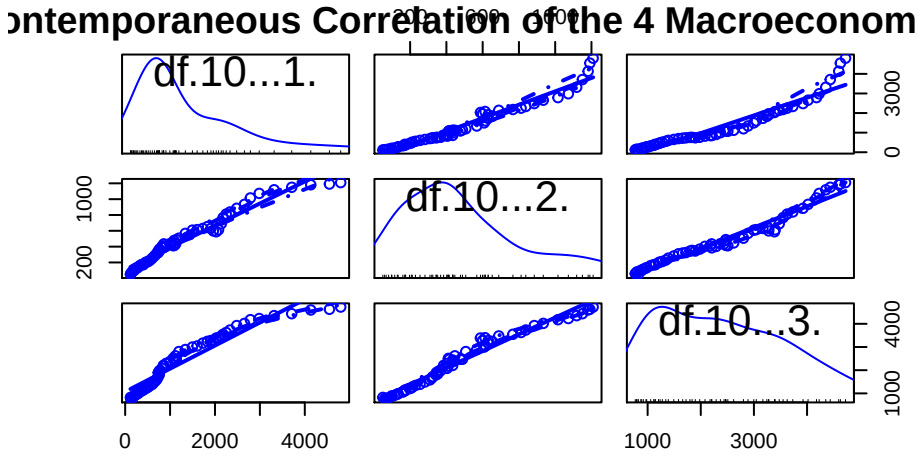
```
##
## Attaching package: 'vars'
```

```
## The following object is masked from 'package:fable':
##
##     VAR
```

```
df.10 <- read.csv("Q10.csv", header = T, row.names = 1)

# df.10$RMC <- log(df.10$RMC)
# df.10$RCC <- log(df.10$RCC)
# df.10$RDPI<- log(df.10$RDPI)
# df.10
plot.ts(df.10)
```

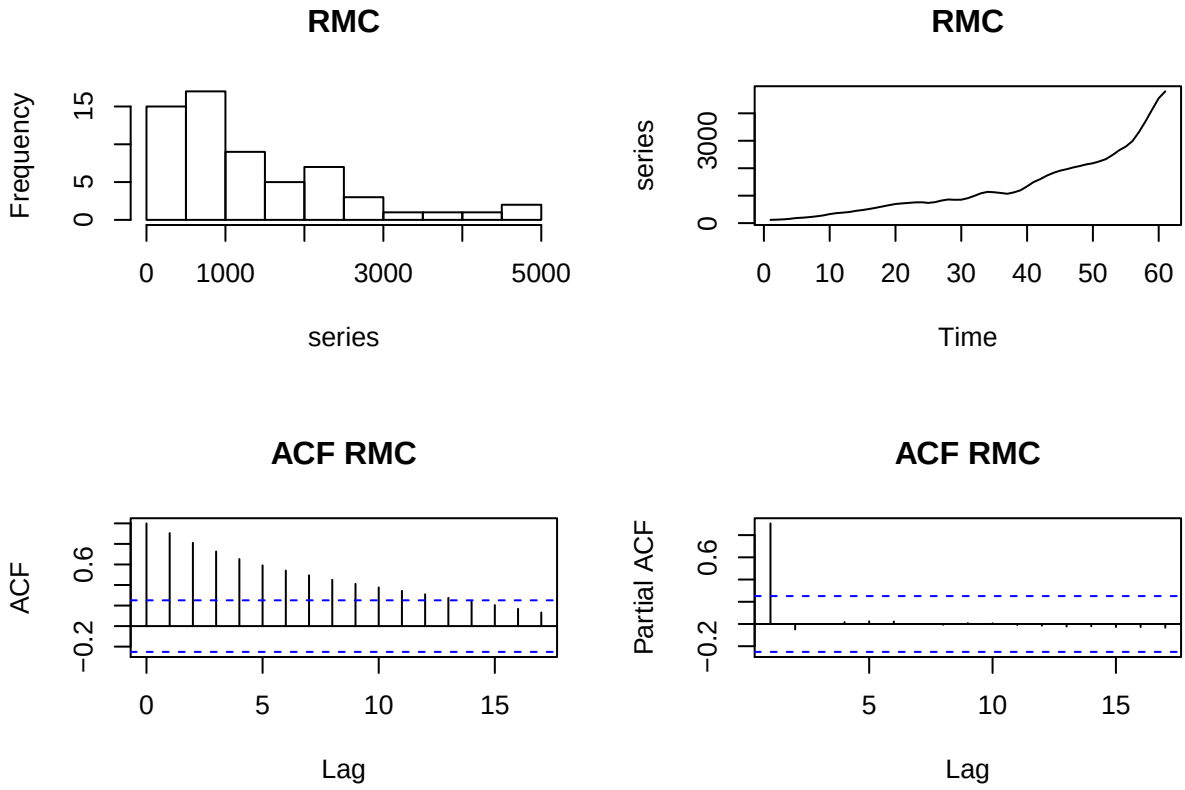



```
scatterplotMatrix(~df.10[,1]+df.10[,2]+df.10[,3]);
  title("Contemporaneous Correlation of the 4 Macroeconomic Series ")
```

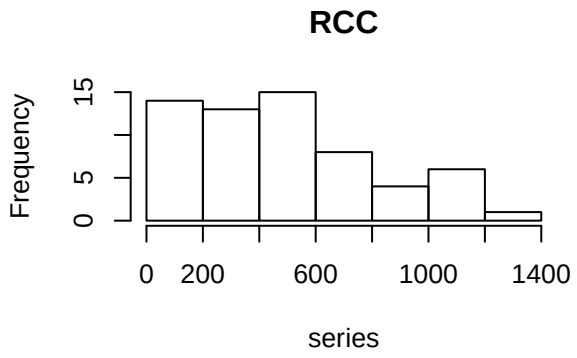


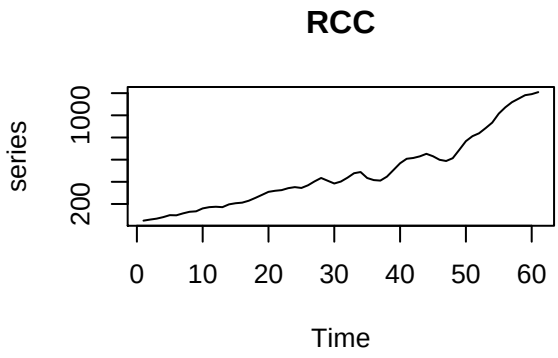

```
# From live session 10
tsplot <- function(series, title) {
  par(mfrow=c(2,2))
```

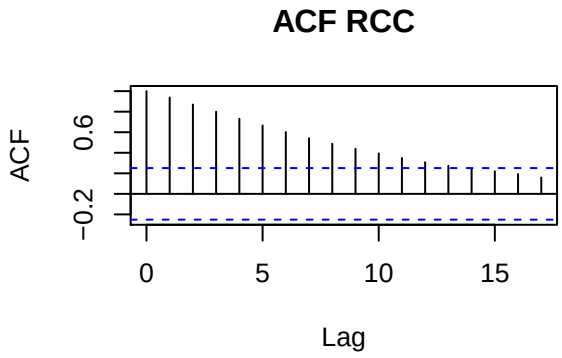
```
  hist(series, main=""); title(title)
  plot.ts(series, main=""); title(title)
  acf(series, main=""); title(paste("ACF",title))
  pacf(series, main=""); title(paste("ACF",title))
}

tsplot(df.10[,1], "RMC")
```

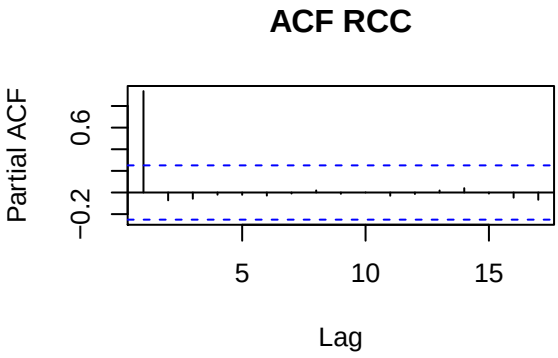



```
tsplot(df.10[,2], "RCC")
```

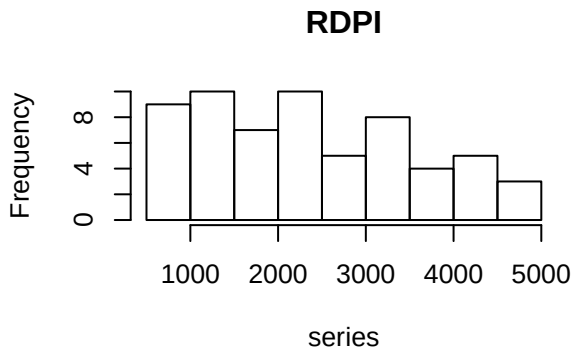


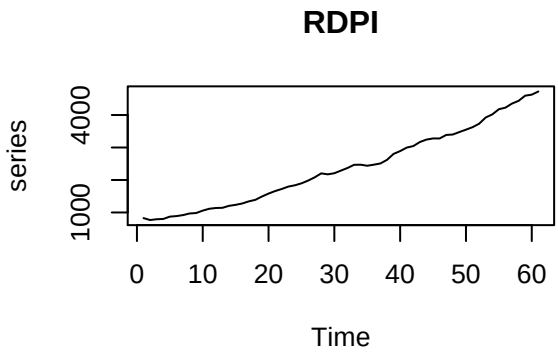


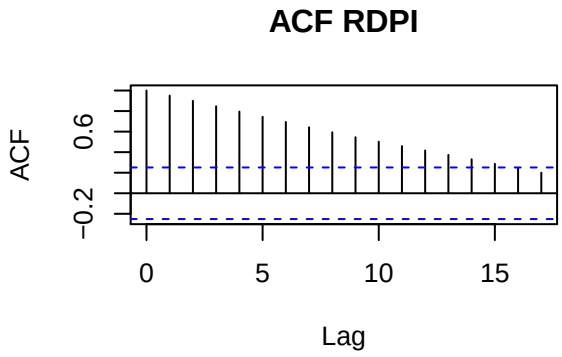


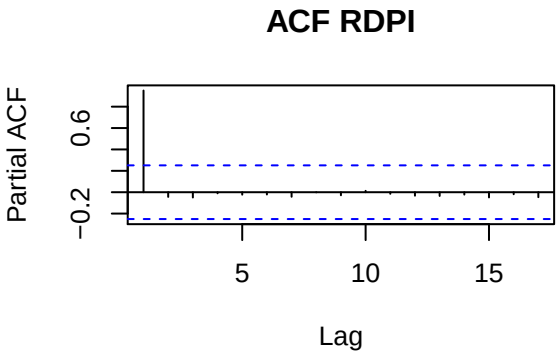



```
tsplot(df.10[,3], "RDPI")
```









```r
VARselect(df.10, lag.max = 8, type = "both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      8      8      2      8
##
## $criteria
##                   1            2            3            4            5
## AIC(n) 2.146434e+01 2.030492e+01 2.002378e+01 2.002587e+01 2.019057e+01
## HQ(n)  2.167878e+01 2.064802e+01 2.049554e+01 2.062629e+01 2.091966e+01
## SC(n)  2.202197e+01 2.119713e+01 2.125056e+01 2.158723e+01 2.208651e+01
## FPE(n) 2.101742e+09 6.627392e+08 5.060443e+08 5.174134e+08 6.298360e+08
##                   6            7            8
## AIC(n) 2.011386e+01 1.976309e+01 1.924910e+01
## HQ(n)  2.097161e+01 2.074950e+01 2.036418e+01
## SC(n)  2.234438e+01 2.232819e+01 2.214878e+01
## FPE(n) 6.115867e+08 4.605469e+08 3.021814e+08
```

```r
var.fit1 <- VAR(df.10, p = 2, type = "both")

var.fit1
```

```
##
## VAR Estimation Results:
## ========================
##
## Estimated coefficients for equation RMC:
## ========================================
## Call:
## RMC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##       RMC.l1       RCC.l1      RDPI.l1       RMC.l2       RCC.l2
##   1.68341589   0.33188470   0.06852067  -0.74293750  -0.01136669
##      RDPI.l2        const        trend
##  -0.02713700 -30.28937637  -3.81748419
##
##
## Estimated coefficients for equation RCC:
## ========================================
## Call:
## RCC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##      RMC.l1      RCC.l1     RDPI.l1      RMC.l2      RCC.l2     RDPI.l2
## -0.04279738  1.55176384  0.03879275  0.08378372 -0.70568059 -0.04597915
##       const       trend
##  7.54642659  1.21727487
##
##
```

```
## Estimated coefficients for equation RDPI:
## =========================================
## Call:
## RDPI = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##      RMC.l1      RCC.l1     RDPI.l1      RMC.l2      RCC.l2     RDPI.l2
##  0.08685910  0.53622836  0.73799523 -0.08889031 -0.37914327  0.09435291
##       const       trend
## 89.58746087  9.26398830
```

```
summary(var.fit1)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: RMC, RCC, RDPI
## Deterministic variables: both
## Sample size: 59
## Log Likelihood: -819.208
## Roots of the characteristic polynomial:
## 1.052 0.9147 0.9147 0.8403 0.6232 0.141
## Call:
## VAR(y = df.10, p = 2, type = "both")
##
##
## Estimation results for equation RMC:
## ====================================
## RMC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##          Estimate Std. Error t value Pr(>|t|)
## RMC.l1    1.68342    0.13105  12.846  < 2e-16 ***
## RCC.l1    0.33188    0.26853   1.236    0.222
## RDPI.l1   0.06852    0.16879   0.406    0.686
## RMC.l2   -0.74294    0.14502  -5.123 4.66e-06 ***
## RCC.l2   -0.01137    0.27301  -0.042    0.967
## RDPI.l2  -0.02714    0.15236  -0.178    0.859
## const   -30.28938   32.58617  -0.930    0.357
## trend    -3.81748    3.65732  -1.044    0.302
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 38.47 on 51 degrees of freedom
## Multiple R-Squared: 0.999,   Adjusted R-squared: 0.9988
## F-statistic:  7096 on 7 and 51 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation RCC:
```

```
## ====================================
## RCC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##         Estimate Std. Error t value Pr(>|t|)
## RMC.l1  -0.04280    0.06938  -0.617    0.540
## RCC.l1   1.55176    0.14216  10.915 5.98e-15 ***
## RDPI.l1  0.03879    0.08936   0.434    0.666
## RMC.l2   0.08378    0.07677   1.091    0.280
## RCC.l2  -0.70568    0.14453  -4.883 1.07e-05 ***
## RDPI.l2 -0.04598    0.08066  -0.570    0.571
## const    7.54643   17.25118   0.437    0.664
## trend    1.21727    1.93619   0.629    0.532
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 20.37 on 51 degrees of freedom
## Multiple R-Squared: 0.9964,  Adjusted R-squared: 0.9959
## F-statistic:  2009 on 7 and 51 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation RDPI:
## ====================================
## RDPI = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##         Estimate Std. Error t value Pr(>|t|)
## RMC.l1   0.08686    0.14649   0.593 0.555845
## RCC.l1   0.53623    0.30017   1.786 0.079982 .
## RDPI.l1  0.73800    0.18868   3.911 0.000272 ***
## RMC.l2  -0.08889    0.16211  -0.548 0.585849
## RCC.l2  -0.37914    0.30518  -1.242 0.219783
## RDPI.l2  0.09435    0.17032   0.554 0.582009
## const   89.58746   36.42576   2.459 0.017346 *
## trend    9.26399    4.08826   2.266 0.027726 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 43.01 on 51 degrees of freedom
## Multiple R-Squared: 0.9988,  Adjusted R-squared: 0.9986
## F-statistic:  6092 on 7 and 51 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##         RMC   RCC RDPI
## RMC  1480.3 425.7  729
## RCC   425.7 414.9  651
```

```
## RDPI  729.0 651.0 1850
##
## Correlation matrix of residuals:
##         RMC    RCC   RDPI
## RMC  1.0000 0.5432 0.4405
## RCC  0.5432 1.0000 0.7431
## RDPI 0.4405 0.7431 1.0000
```

```
names(var.fit1)
```

```
##  [1] "varresult"    "datamat"      "y"            "type"
##  [5] "p"            "K"            "obs"          "totobs"
##  [9] "restrictions" "call"
```

```
roots(var.fit1)
```

```
## [1] 1.0523974 0.9146892 0.9146892 0.8403369 0.6232106 0.1410174
```

```
# Test of normality:
var.fit1.norm <- normality.test(var.fit1, multivariate.only = TRUE)
# names(var.fit1.norm)
var.fit1.norm
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.fit1
## Chi-squared = 40.106, df = 6, p-value = 4.342e-07
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.fit1
## Chi-squared = 8.4757, df = 3, p-value = 0.03714
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.fit1
## Chi-squared = 31.63, df = 3, p-value = 6.262e-07
```

```
# Test of no serial correlation:
var.fit1.ptasy <- serial.test(var.fit1, lags.pt = 12, type = "PT.asymptotic")
var.fit1.ptasy
```

```
##
```

```
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.fit1
## Chi-squared = 127.05, df = 90, p-value = 0.006181
```

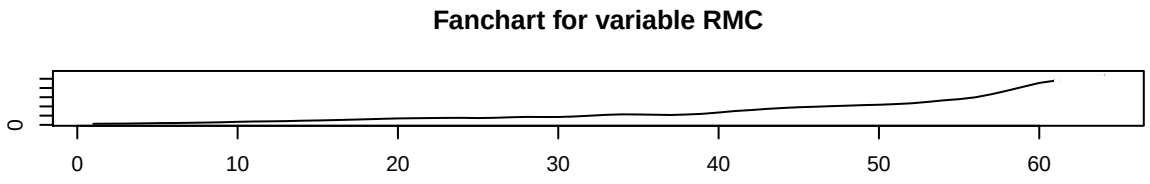
```
# Test of the absence of ARCH effect:
var.fit1.arch <- arch.test(var.fit1)
names(var.fit1.arch)
```

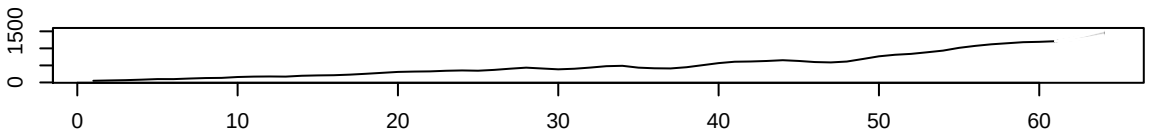
```
## [1] "resid"    "arch.mul"
```

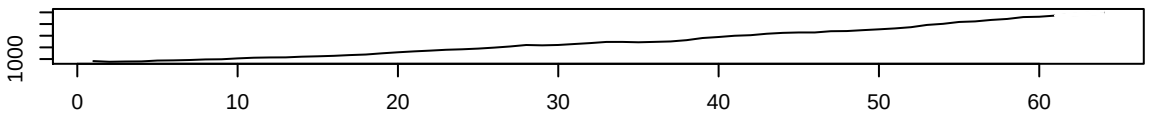
```
var.fit1.arch
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.fit1
## Chi-squared = 210.85, df = 180, p-value = 0.0575
```

```
# Predictions forward
var.fit1 %>%  predict(n.ahead = 3, ci = 0.95) %>% fanchart()
```








```
var.fit1 %>%  predict(n.ahead = 3, ci = 0.95)
```

```
## $RMC
##          fcst    lower    upper        CI
## [1,] 5019.225 4943.817 5094.633  75.40799
## [2,] 5219.448 5062.615 5376.282 156.83339
## [3,] 5421.361 5173.492 5669.231 247.86963
##
```

```
## $RCC
##          fcst    lower    upper        CI
## [1,] 1265.001 1225.080 1304.922  39.92113
## [2,] 1350.071 1275.728 1424.415  74.34347
## [3,] 1453.994 1352.751 1555.238 101.24351
##
## $RDPI
##          fcst    lower    upper        CI
## [1,] 4795.983 4711.690 4880.276  84.29322
## [2,] 4887.541 4769.360 5005.723 118.18136
## [3,] 4993.506 4845.283 5141.729 148.22264
```