W271 Assignment 3

Erin Werner

November 29th, 2020

Contents

```r
library(ggplot2)
library(dplyr)
library(car)
library(readr)
library(tseries)
library(forecast)
library(fable)
library(fpp2)
library(fpp3)
library(gridExtra)
library(grid)
library(tsibble)
library(tibble)
library(mvtnorm)
library(vars)
library(lubridate)
library(tidyverse)
```

Question 1 - Time Series Linear Model

The data set `Q1.csv` concerns the monthly sales figures of a shop which opened in January 1987 and sells gifts, souvenirs, and novelties. The shop is situated on the wharf at a beach resort town in Queensland, Australia. The sales volume varies with the seasonal population of tourists. There is a large influx of visitors to the town at Christmas and for the local surfing festival, held every March since 1988. Over time, the shop has expanded its premises, range of products, and staff.

```r
Q1 <- read_csv(paste0(here::here(),"/assignments/assignment_3/Q1.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   sales = col_double()
## )
```

To help with my analysis, I will create additional variables to help represent the time series in different ways. This includes individual month and year variables as well as a formal date variable. These additional features will provide different forms of time representation in my model.

```r
Q1$Month <- factor(rep(1:12, times = 7), labels = month.abb)
Q1$Year <- factor(rep(1987:1993, each = 12))
Q1$date <- as.Date(paste(Q1$Year, Q1$Month, '01',sep = '-'),format = '%Y-%b-%d')
```

The time series spans from the beginning of 1987 to the end of 1993.

```r
t1 <- head(Q1)[,c(2,3,4)]
t2 <- tail(Q1)[,c(2,3,4)]
grid.arrange(tableGrob(t1), tableGrob(t2), ncol = 2)
```

	sales	Month	Year
1	1664.81	Jan	1987
2	2397.53	Feb	1987
3	2840.71	Mar	1987
4	3547.29	Apr	1987
5	3752.96	May	1987
6	3714.74	Jun	1987

	sales	Month	Year
1	26155.15	Jul	1993
2	28586.52	Aug	1993
3	30505.41	Sep	1993
4	30821.33	Oct	1993
5	46634.38	Nov	1993
6	104660.67	Dec	1993

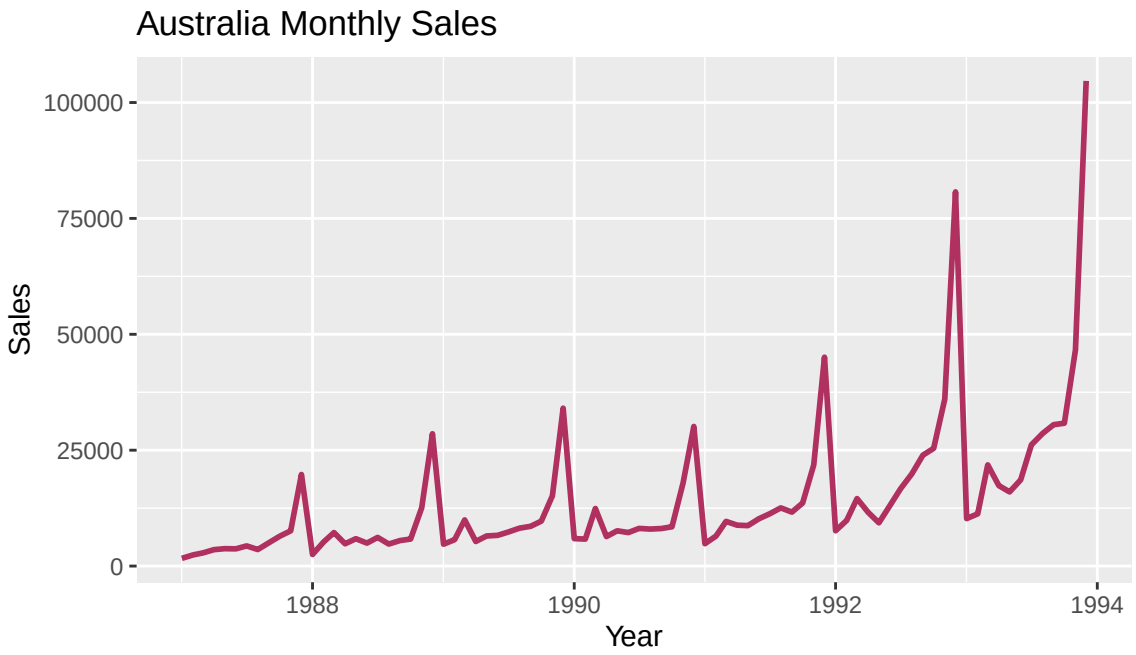
a) Produce a time plot of the data and describe the patterns in the graph. Identify any unusual or unexpected fluctuations in the time series.

First, I need to convert the data into a suitable time series object.

```
sales_ts <- as_tsibble(Q1, index = date, regular = TRUE)
```

Then, I am able to produce a time plot for the series.

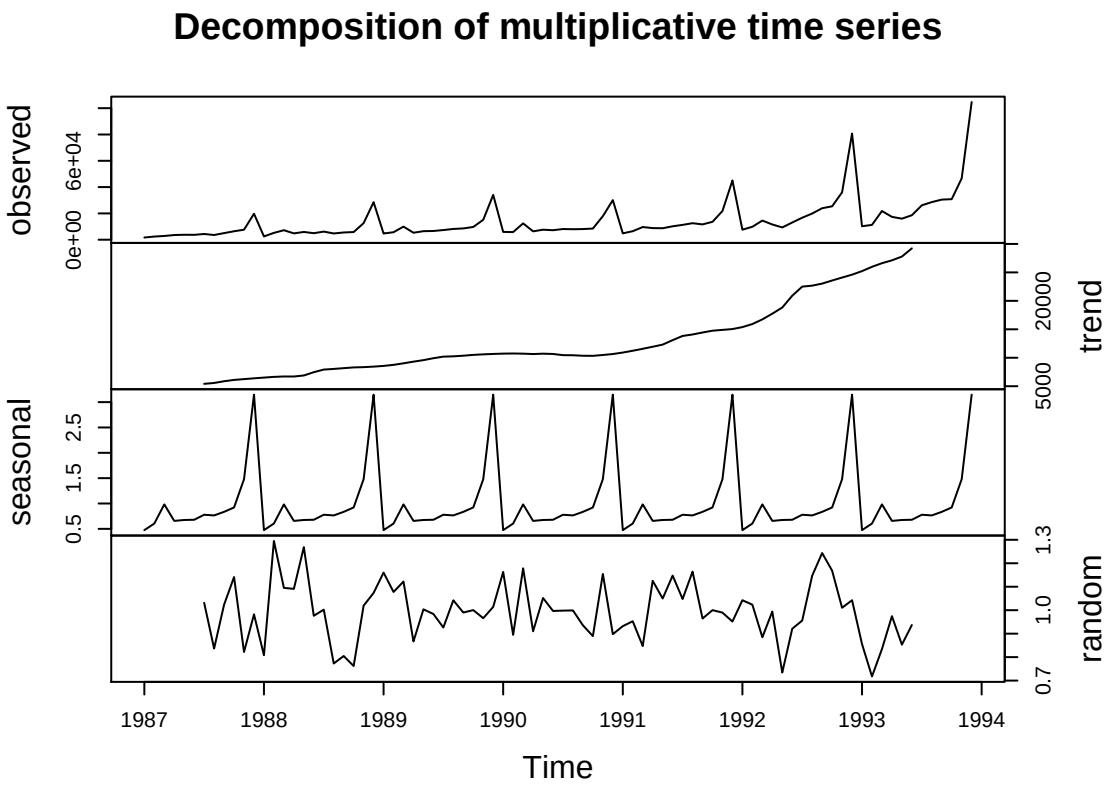
```
sales_ts %>% autoplot(sales,colour="maroon",size=1) +
  labs(title = "Australia Monthly Sales", x = "Year", y = "Sales")
```



From the time series plot, I can see a generally positive trend with reoccurring seasonal peaks. Overall, it appears that a similar pattern repeats each year as the sales start off low and increase gradually then experience a sharp peak followed by a smaller peak. Although the seasonal peaks are not unexpected due to the varying tourism, they are unusual in regards to their extreme increase in value. Additionally, the size of the seasonal fluctuations and random fluctuations seem to increase with the level of the time series, growing almost exponentially. As a result, the additive model is not appropriate for describing this time series. A multiplicative seasonal change would be better as the magnitude of the seasonal change increases over time as the data values increase. However, the extra variability can make multiplicative seasonal changes harder to forecast accurately. Thus, it may be necessary to transform the time series in order to produce a series that is viable for my forecasting model.

It is often helpful to split a time series into several components, each representing an underlying pattern category. More generally, in order address the the trend, seasonal and irregular elements that are in the series, I can multiplicatively decompose the original time series into trend, seasonal and error components.

```
sts <- ts(sales_ts$sales, start = c(1987,1), end = c(1993,12), frequency = 12)
sales.deco <- decompose(sts, type = "multiplicative")
plot(sales.deco, yax.flip = TRUE)
```

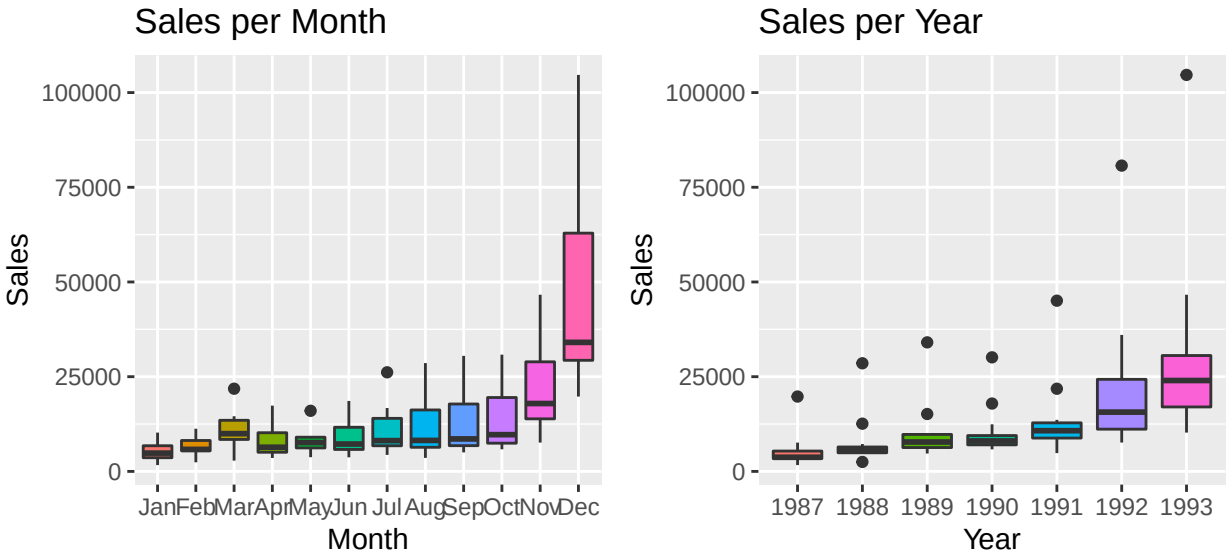


From the plots, I can see that:

- the overall sales have an upward trend with seasonality.
- the sales trend is strong and upward. It appears to grow almost exponentially.
- the sales seasonality fluctuates consistently, with repeating cycles of major peaks followed by smaller peaks.
- the variation in the remainder does not appear like white noise. There is inconsistent oscillation over time.

I can also further examine these fluctuations by taking a closer look at the monthly and annual box plots.

```
p1 <- ggplot(Q1,aes(Month,sales)) + theme(legend.position = "none") +
  geom_boxplot(varwidth = T, aes(fill = Month)) +
  labs(x = "Month", y = "Sales", title = "Sales per Month")
p2 <- ggplot(Q1,aes(Year,sales)) + theme(legend.position = "none") +
  geom_boxplot(varwidth = T, aes(fill = Year)) +
  labs(x = "Year", y = "Sales", title = "Sales per Year")
egg::ggarrange(p1, p2, nrow = 1)
```



From the box plots, I can see that:

- with respect to the change by the month, there is apparent seasonality in the series. There is a much higher distribution of sales in December compared to the rest of the year, with a slightly higher distribution in March as well. Yet, this is consistent with the seasonal tourist travel behavior.
- with respect to the change by the year, there is a strong and consistent upward trend in sales. There is also a noticeably larger distribution of sales in more recent years, which is consistent with the shop's expansion of products and staff. This is indicative of exponential growth.

These patterns in the time series indicate that it is necessary to perform a data transformation in order to build a successful model. This will lead to more reliable forecasts.

b) Explain why it is necessary to take logarithms of these data before fitting a model.

It is necessary to take logarithms of the sales data before fitting the model. This is because the size of the peaks in the data is not constant. As mentioned earlier, the size of the seasonal fluctuations and random fluctuations seem to increase with the level of the time series, growing almost exponentially. The extra variability can make multiplicative seasonal changes harder to forecast accurately. As a result, a transformation is required and log transforms are effective at removing exponential variance. So, this transformation should be applied to the time series before building my model.

Furthermore, logarithm transformations are useful because they are more easily interpretable in analysis. This is because the interpretation reflects that changes in a log value are relative to percentage changes on the original scale.

c) Use R to fit a regression model to the logarithms of these sales data with a linear trend, seasonal dummies and a "surfing festival" dummy variable.

The logarithmic model would take the form as follows:

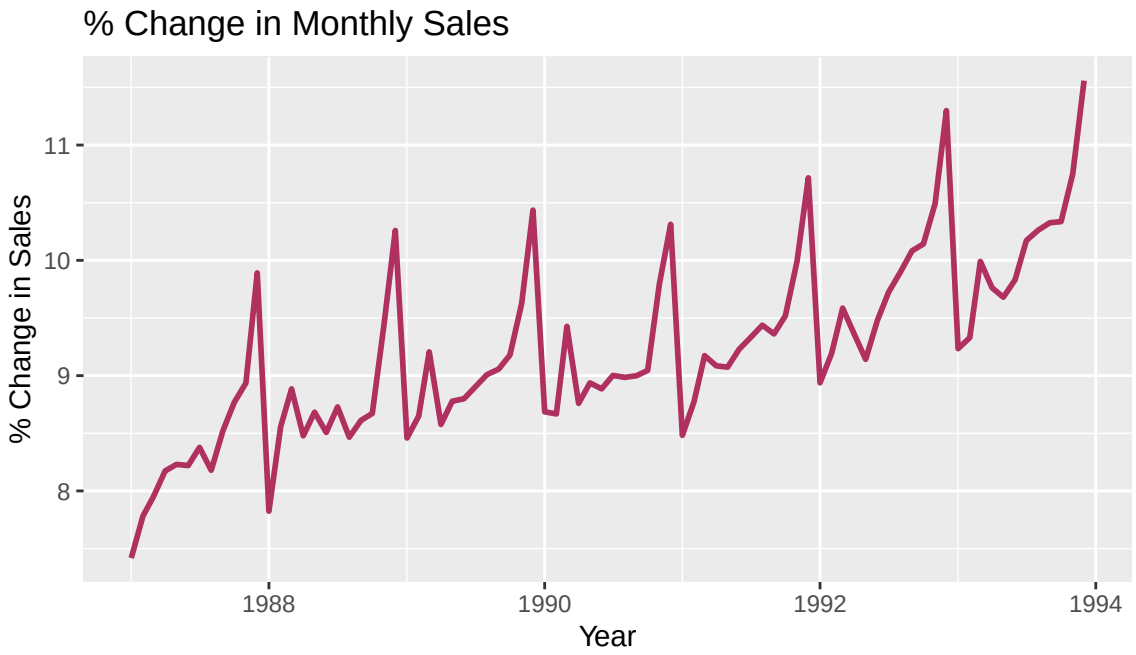
$$log(y_t) = \beta_0 + \beta_1 t + \beta_2 t + \beta_3 t + \epsilon_t y_t = e^{\beta_0 + \beta_1 t + \beta_2 t + \beta_3 t + \epsilon_t}$$

First, I need to perform the log transformation on the data and create a suitable time series object.

```
Q1$sales_log <- log(Q1$sales)
sales_ts_log <- as_tsibble(Q1, index = date, regular = TRUE)
```

Then, I am able to produce a time plot for the log-transformed series.

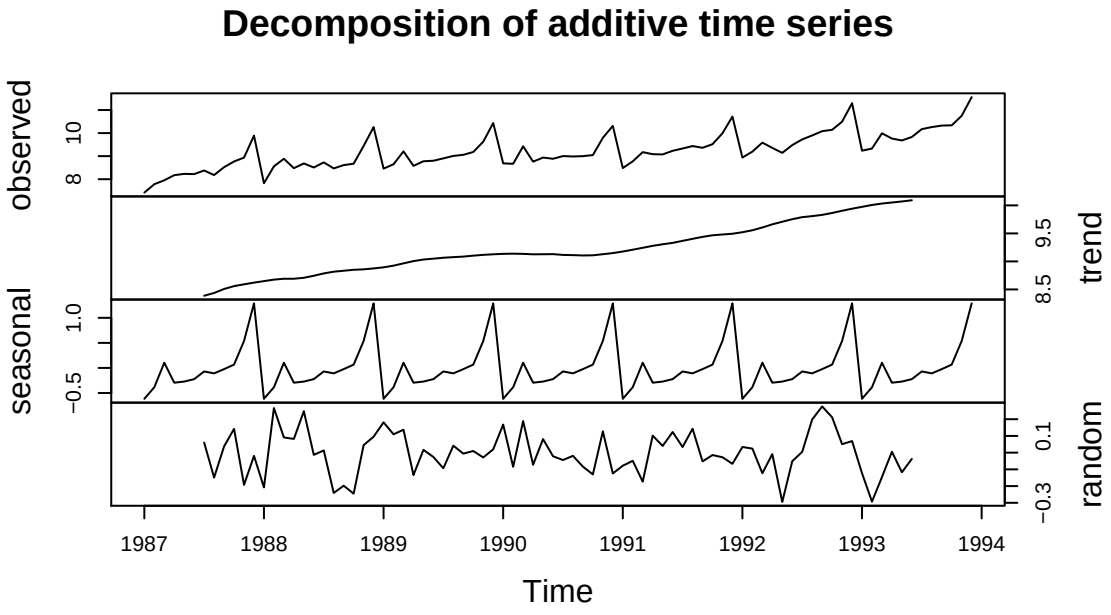
```
sales_ts_log %>% autoplot(sales_log, colour = "maroon", size = 1) +
  labs(title = "% Change in Monthly Sales", x = "Year", y = "% Change in Sales")
```



Now, I can see that the size of the seasonal and random fluctuations in the log-transformed time series seem to be roughly constant over time. They also do not depend on the level of the time series. The trend is still present, but appears to be more linear rather than exponential. Thus, the log-transformed time series can probably be described using an additive model. However, there is still a prevalent positive trend and oscillating seasonal variation in the log-series.

I can also decompose the log-series and observe each of the trend, seasonal, and random components.

```
sts_log <- ts(sales_ts_log$sales_log,start=c(1987,1),end=c(1993,12),frequency=12)
sales.deco_log <- decompose(sts_log, type = "additive")
plot(sales.deco_log, yax.flip = TRUE)
```



From the plots, I can see that each of the components are very similar to the original series decomposition, although the trend is now linear. This will be important to consider in my model analysis, as the trend and seasonality are still prevalent and might require more manipulations or transformations.

Before I build my model, the local surfing festival, which started in 1988, is held every March. So, I need to create a dummy variable to represent this effect.

```
sales_ts_log$surf <- ifelse(sales_ts_log$Month=="Mar"&sales_ts_log$Year!=1987,1,0)
```

Next, I am able to build my regression model to the logarithms of the sales data with a linear trend, seasonal dummies and a "surfing festival" dummy variable.

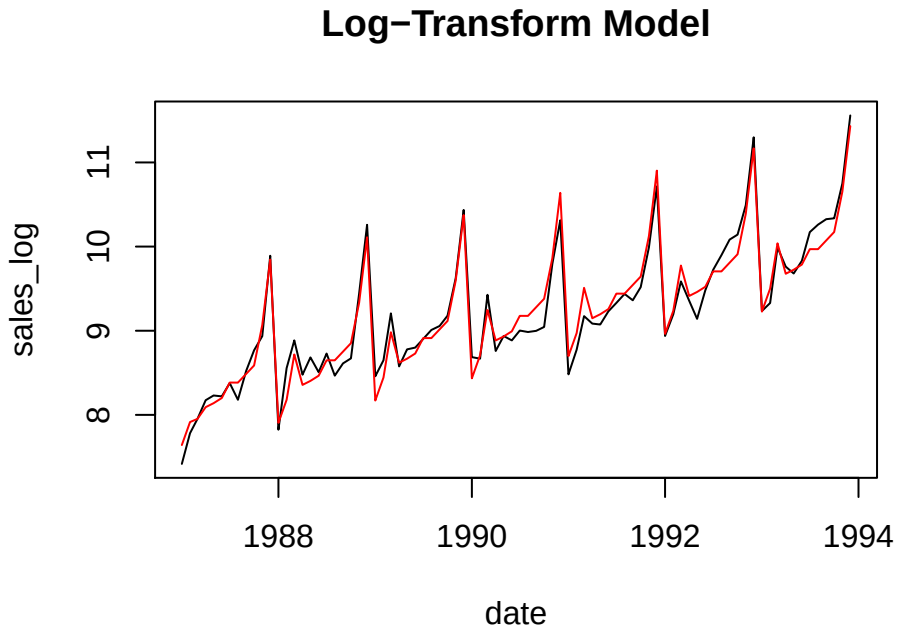
```
exp.trend.fit <- lm(sales_log ~ date + Month + surf, data = sales_ts_log)
summary(exp.trend.fit)
```

```
##
## Call:
## lm(formula = sales_log ~ date + Month + surf, data = sales_ts_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33639 -0.12729  0.00262  0.10887  0.37695
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.150e+00  2.095e-01  15.034  < 2e-16 ***
## date        7.234e-04  2.715e-05  26.646  < 2e-16 ***
## MonthFeb    2.510e-01  9.564e-02   2.625 0.010644 *
## MonthMar    2.675e-01  1.933e-01   1.384 0.170895
## MonthApr    3.848e-01  9.567e-02   4.022 0.000144 ***
## MonthMay    4.106e-01  9.569e-02   4.290 5.62e-05 ***
## MonthJun    4.495e-01  9.572e-02   4.696 1.28e-05 ***
## MonthJul    6.114e-01  9.576e-02   6.385 1.62e-08 ***
## MonthAug    5.885e-01  9.581e-02   6.143 4.38e-08 ***
## MonthSep    6.695e-01  9.586e-02   6.984 1.34e-09 ***
## MonthOct    7.479e-01  9.592e-02   7.797 4.33e-11 ***
## MonthNov    1.207e+00  9.599e-02  12.572  < 2e-16 ***
## MonthDec    1.963e+00  9.606e-02  20.430  < 2e-16 ***
## surf        5.012e-01  1.963e-01   2.553 0.012870 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1789 on 70 degrees of freedom
## Multiple R-squared:  0.9568, Adjusted R-squared:  0.9487
## F-statistic: 119.1 on 13 and 70 DF,  p-value: < 2.2e-16
```

As a result, almost all the terms are statistically significant for $\alpha = 0.05$ and the resulting adjusted r-squared value is quite high with a value of 0.95.

Then, I can plot the actual values with the model to visually asses the fit.

```
plot(sales_log ~ date, data = sales_ts_log, type="l",main="Log-Transform Model")
lines(fitted(exp.trend.fit) ~ date, data = sales_ts_log, col = "red")
```



From the plot, I can see that the model fits the trend and seasonal fluctuations very well. It follows the trend and major fluctuations, including the extreme peak values, in the series.

I can also take a look at the residuals to further assess the model fit.

d) Plot the residuals against time and against the fitted values. Do these plots reveal any problems with the model?

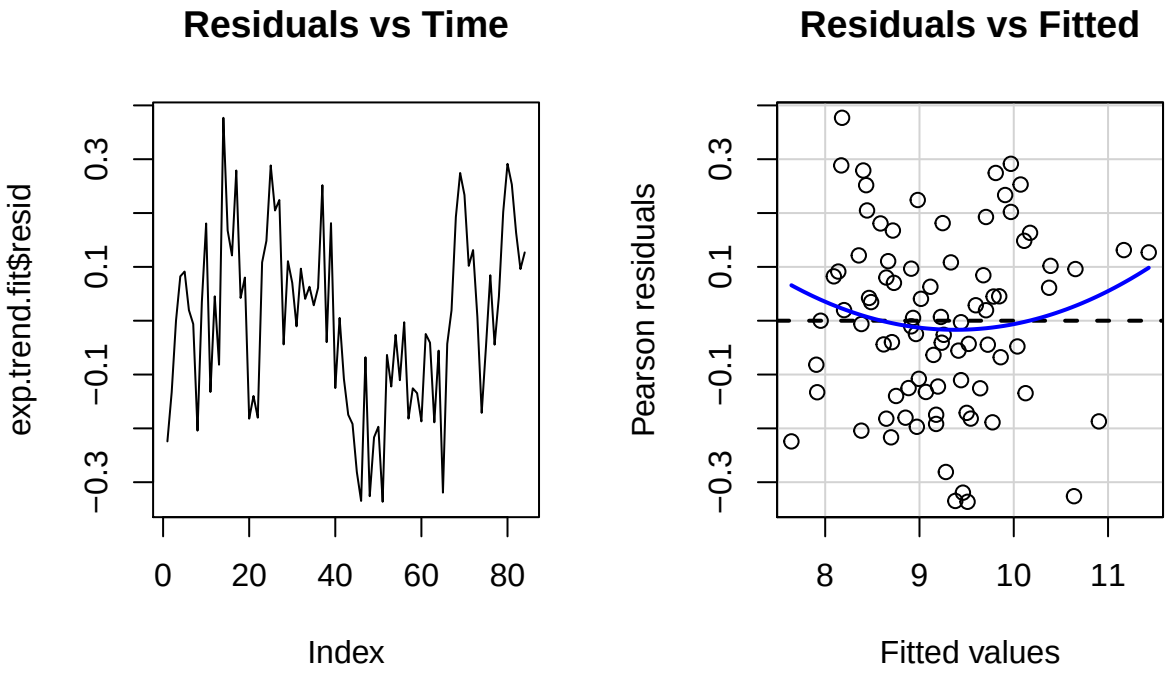
The residuals, in a time series model, are what is left over after fitting a model. The residuals are equal to the difference between the observations and the corresponding fitted values. Residuals are useful in checking whether a model has adequately captured the information in the data.

In order to produce good forecasts, the residuals must:

- be uncorrelated. If there are correlations, than there is still information in them. So, they should not be used for forecasting.
- have zero mean. If they do not, then the forecasts will be biased.
- have constant variance.
- be normally distributed.

To start, I can plot the residuals against time and against the fitted values.

```
par(mfrow=c(1,2))
plot(exp.trend.fit$resid, type="l", main="Residuals vs Time")
residualPlot(exp.trend.fit, main="Residuals vs Fitted")
```



From the residual vs time plot (above - left), there is no major apparent trend although the seasonality is still largely unaccounted for. Additionally, the variation of the residuals is not consistent across the data, as there are fluctuations of varying heights throughout the series. Therefore, the assumption of constant variance is violated for the model.

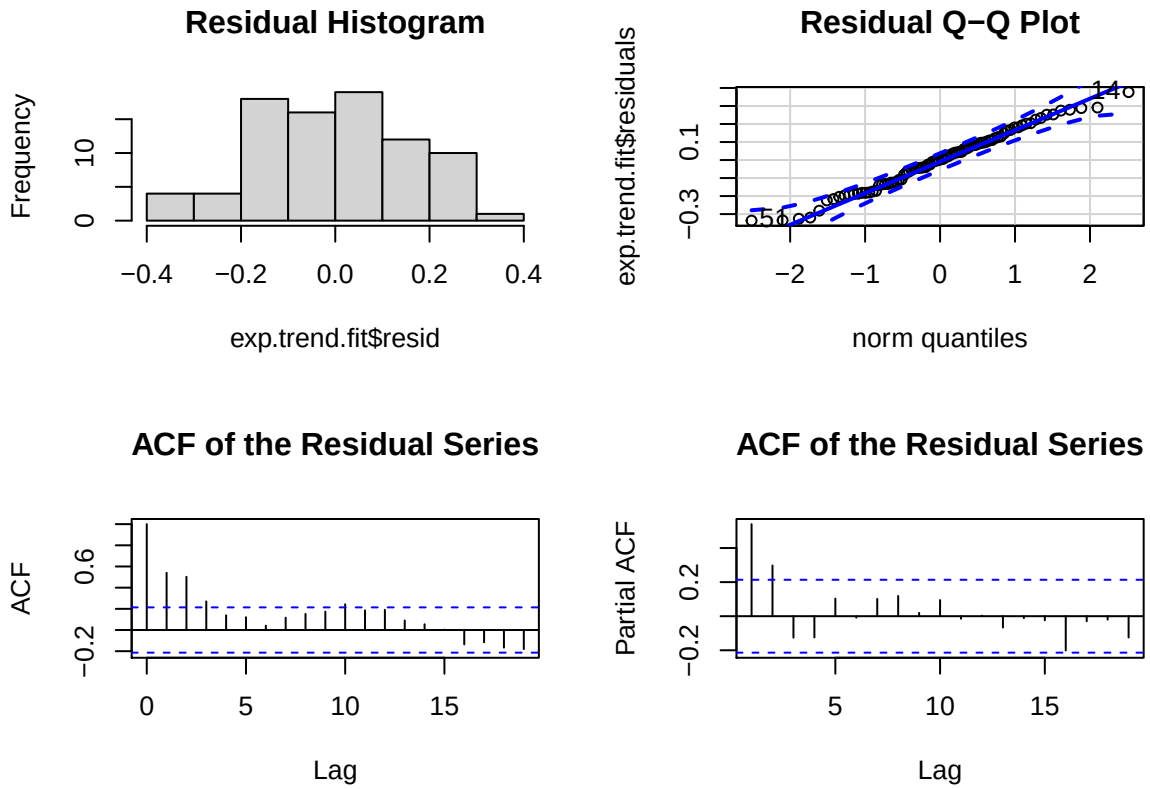
From the residual vs fitted plot (above - right), I can see that the residuals follow a slight curve. Yet, the data is generally centered and dispersed around the zero-line and the mean does not change too drastically from left to right. So, the values are not necessarily different for different values of x. There is a somewhat flat band of points for the plot and the error is mostly zero in expectation. Thus, I can confirm the assumption of having zero mean as there is no apparent pattern among the data points.

Yet, I can also take a closer look at the the histogram, the Q-Q Plot, the ACF, and the PACF of the residuals in order to further analyze the model fit.

```
par(mfrow=c(2,2))
hist(exp.trend.fit$resid, main = "Residual Histogram")
qqPlot(exp.trend.fit$residuals, main = "Residual Q-Q Plot")
```

```
## [1] 14 51
```

```
acf(exp.trend.fit$resid, main="ACF of the Residual Series")
pacf(exp.trend.fit$resid, main="ACF of the Residual Series")
```



From the additional residual plots, I can see that:

- The histogram and Q-Q plot show a roughly normal distribution of residuals, with a slight departure around the extreme values. This is an indication of the normality of the residuals.
- the ACF appears to gradually decline and oscillate around zero. This means I can expect a negative AR parameter for the random elements. Yet, there is a small spike around lag 10 indicating that there may be some correlation in the residuals.
- the PACF appears to gradually decline and oscillate around zero. The PACF drastically cuts off after the second lag, indicating the potential need for a second order model. In general though, there does not seem to be major violations of non-correlation.

Collectively, the residuals violate some of the underlying assumptions. Therefore, the plots reveal problems with constant variance in the model.

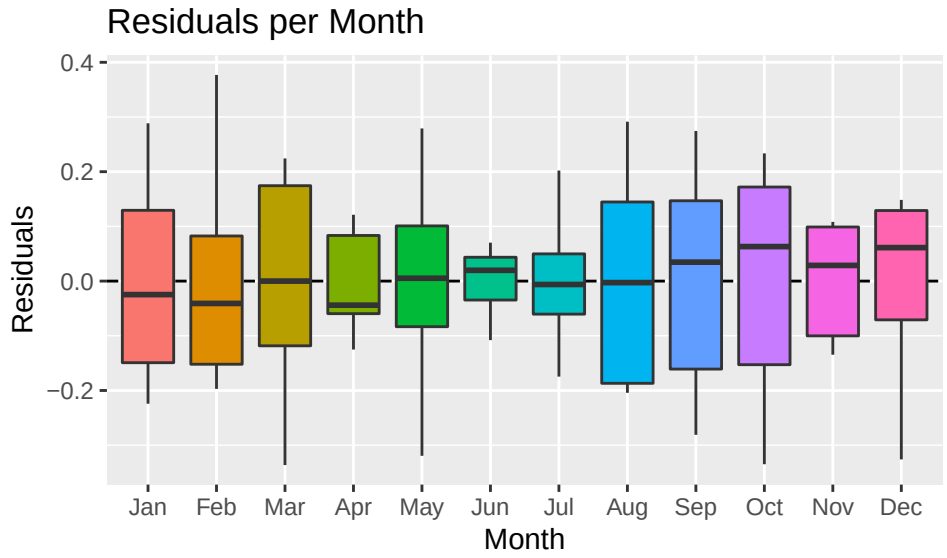
e) Do box plots of the residuals for each month. Does this reveal any problems with the model?

First, I need to create a monthly representation for my model residuals.

```
resid_month <- factor(rep(1:12, times = 7), labels = month.abb)
resid_values <- exp.trend.fit$resid
resid_df <- data.frame("Month" = resid_month, "Value" = resid_values)
```

Then, I am able to produce a box plot for the monthly residuals.

```
ggplot(resid_df,aes(Month,Value)) + theme(legend.position = "none") +
  geom_hline(yintercept=0,lty="dashed")+geom_boxplot(varwidth=T,aes(fill=Month))+
  labs(x = "Month", y = "Residuals", title = "Residuals per Month")
```



From the box plot, I can see that the residuals roughly maintain zero conditional mean per month. Although each of the months have varying residual distributions, they are all approximately centered around the zero-line. This assumption was satisfied for the overall data in the residuals vs fitted plot and grouping the data by month reaffirms this. Furthermore, as there are varying residual distributions, the box plot indicates that there is not constant variance in the model per month. This assumption was violated in the residual vs time plot and is also violated in the monthly box plot. This proves that our model does not satisfy all of the residual model assumptions.

Therefore, the residuals do not satisfy each of the underlying assumptions, as the plot reveals no problems with the assumption of zero conditional mean but reveals problems with the assumption of constant variance with regards to monthly residuals.

f) What do the values of the coefficients tell you about each variable?

To start, almost all of the coefficients for my model were statistically significant.

```
summary(exp.trend.fit)
```

```
##
## Call:
## lm(formula = sales_log ~ date + Month + surf, data = sales_ts_log)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33639 -0.12729  0.00262  0.10887  0.37695
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.150e+00  2.095e-01  15.034  < 2e-16 ***
## date        7.234e-04  2.715e-05  26.646  < 2e-16 ***
## MonthFeb    2.510e-01  9.564e-02   2.625 0.010644 *
## MonthMar    2.675e-01  1.933e-01   1.384 0.170895
## MonthApr    3.848e-01  9.567e-02   4.022 0.000144 ***
## MonthMay    4.106e-01  9.569e-02   4.290 5.62e-05 ***
## MonthJun    4.495e-01  9.572e-02   4.696 1.28e-05 ***
## MonthJul    6.114e-01  9.576e-02   6.385 1.62e-08 ***
## MonthAug    5.885e-01  9.581e-02   6.143 4.38e-08 ***
## MonthSep    6.695e-01  9.586e-02   6.984 1.34e-09 ***
```

```
## MonthOct    7.479e-01  9.592e-02   7.797 4.33e-11 ***
## MonthNov    1.207e+00  9.599e-02  12.572  < 2e-16 ***
## MonthDec    1.963e+00  9.606e-02  20.430  < 2e-16 ***
## surf        5.012e-01  1.963e-01   2.553 0.012870 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1789 on 70 degrees of freedom
## Multiple R-squared:  0.9568, Adjusted R-squared:  0.9487
## F-statistic: 119.1 on 13 and 70 DF,  p-value: < 2.2e-16
```

According to the model,

- a unit increase in `date` (one day) shows an increase of 0.0007 percent in sales. This would roughly be a 0.02 percent increase in sales per month.
- a unit increase in `MonthFeb` (change to February) shows a 0.25 percent increase in sales.
- a unit increase in `MonthMar` (change to March) shows an increase of 0.27 percent in sales.
- a unit increase in `MonthApr` (change to April) shows a 0.38 percent increase in sales.
- a unit increase in `MonthMay` (change to May) shows a 0.41 percent increase in sales.
- a unit increase in `MonthJun` (change to June) shows an increase of 0.45 percent in sales.
- a unit increase in `MonthJul` (change to July) shows a 0.61 percent increase in sales.
- a unit increase in `MonthAug` (change to August) shows an increase of 0.59 percent in sales.
- a unit increase in `MonthSep` (change to September) shows a 0.67 percent increase in sales.
- a unit increase in `MonthOct` (change to October) shows a 0.74 percent increase in sales.
- a unit increase in `MonthNov` (change to November) shows an increase of 1.207 percent in sales.
- a unit increase in `MonthDec` (change to December) shows a 1.963 percent increase in sales.
- a unit increase in `surf` (change to the surfing festival in March) shows an increase of 0.50 percent in sales.

These values indicate that there is a generally positive trend in sales, as each variable leads to an increase in percent of sales over time.

g) What does the Breusch-Godfrey test tell you about your model?

The Breusch-Godfrey test checks for higher-order serial correlation. This is conducted by using the `bgtest()` function. In the context of an BG test, H_0 is that there is no serial correlation of any order up to p, with rejection (H_A) indicating that the series does have serial correlation up to order p. I can apply my Breusch-Godfrey test for order 1 on my series, as my series is of order 0.

```
bgtest(exp.trend.fit, order = 1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  exp.trend.fit
## LM test = 24.986, df = 1, p-value = 5.775e-07
```

The resulting p-value of the BG test is 5.775e-07, which is less than $\alpha = 0.05$. So, I will reject the null hypothesis that the series has no serial correlation of up to order 1. This is an indication that my model has serial correlation in it.

I can also test for the significance of the correlation in the model residuals using the Ljung–Box test.

```
Box.test(exp.trend.fit$residuals)
```

```
##
##  Box-Pierce test
##
## data:  exp.trend.fit$residuals
## X-squared = 24.519, df = 1, p-value = 7.359e-07
```

The resulting p-value of the LB test is 7.359e-07, which is less than $\alpha = 0.05$. As a result, I can reject the null hypothesis that the series is uncorrelated. This is further evidence that correlation does exist in the model.

Therefore, the Breusch-Godfrey test tells me that there is correlation in my model.

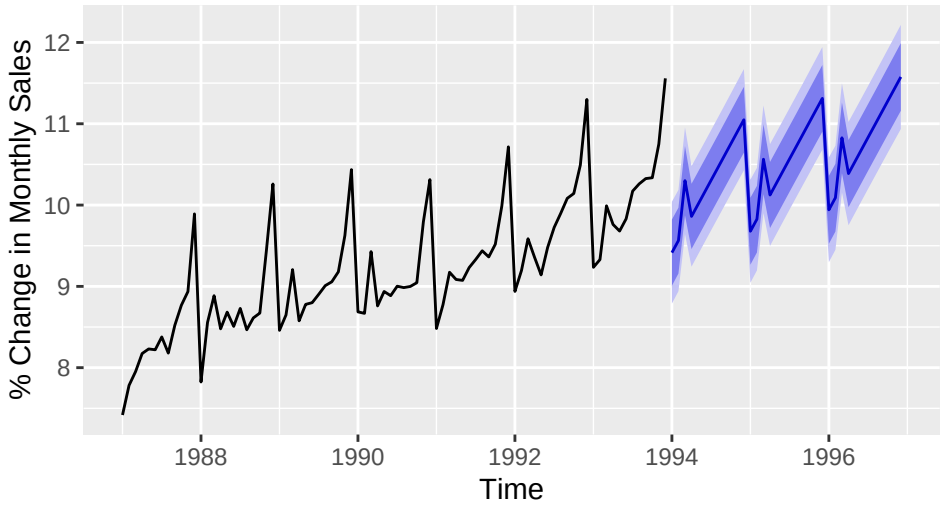
h) Regardless of your answers to the above questions, use your regression model to predict the monthly sales for 1994, 1995, and 1996. Produce prediction intervals for each of your forecasts.

First, I can generate forecasts from my model for 1994, 1995, and 1996.

```
sts_log <- ts(sales_ts_log,start=c(1987,1),end=c(1993,12),frequency=12)
fit.forcast <- tslm(sales_log ~ trend + Month + surf, data = sts_log)
h <- 3
new_data_s <- data.frame(surf = c(0,0,1,0,0,0,0,0,0,0,0,0),
                         Month=c(1,2,3,4,5,6,7,8,9,10,11,12))
fcast.up <- forecast(fit.forcast, newdata = data.frame(
                 surf = rep(new_data_s$surf,h),Month = rep(new_data_s$Month,h)))
```

Then, I can plot the results. This will include the actual values as well as the predicted values and their confidence intervals, up to the end of 1996.

```
autoplot(sts_log[,"sales_log"]) + ylab("% Change in Monthly Sales") +
  autolayer(fcast.up, PI = TRUE)
```



From the plot, I can see that the predicted values follow the positive upwards trend and oscillate in a similar, yet slightly less extreme pattern of repeated peaks. The confidence intervals follow this behavior as well.

I can also calculate the confidence intervals for each the annual forecasts and print out the prediction intervals for each of my forecasts in 1994, 1995, and 1996.

```
fcast.up
```

```
##          Point Forecast     Lo 80     Hi 80     Lo 95    Hi 95
```

```
## Jan 1994       9.414599  9.007952  9.821246  8.788350 10.04085
## Feb 1994       9.563182  9.158711  9.967653  8.940284 10.18608
## Mar 1994      10.300272  9.874849 10.725694  9.645108 10.95544
## Apr 1994       9.860349  9.459043 10.261656  9.242325 10.47837
## May 1994      10.008932  9.608606 10.409259  9.392417 10.62545
## Jun 1994      10.157516  9.757763 10.557269  9.541884 10.77315
## Jul 1994      10.306099  9.906512 10.705687  9.690722 10.92148
## Aug 1994      10.454683 10.054853 10.854513  9.838932 11.07043
## Sep 1994      10.603266 10.202786 11.003746  9.986514 11.22002
## Oct 1994      10.751849 10.350313 11.153386 10.133471 11.37023
## Nov 1994      10.900433 10.497438 11.303428 10.279809 11.52106
## Dec 1994      11.049016 10.644165 11.453867 10.425533 11.67250
## Jan 1995       9.678060  9.266344 10.089776  9.044005 10.31211
## Feb 1995       9.826643  9.417090 10.236197  9.195919 10.45737
## Mar 1995      10.563733 10.134100 10.993365  9.902085 11.22538
## Apr 1995      10.123810  9.717409 10.530211  9.497940 10.74968
## May 1995      10.272394  9.866974 10.677814  9.648034 10.89675
## Jun 1995      10.420977 10.016137 10.825817  9.797511 11.04444
## Jul 1995      10.569561 10.164898 10.974223  9.946367 11.19275
## Aug 1995      10.718144 10.313255 11.123033 10.094603 11.34169
## Sep 1995      10.866727 10.461210 11.272245 10.242218 11.49124
## Oct 1995      11.015311 10.608764 11.421857 10.389217 11.64140
## Nov 1995      11.163894 10.755920 11.571868 10.535602 11.79219
## Dec 1995      11.312478 10.902683 11.722272 10.681382 11.94357
## Jan 1996       9.941521  9.523720 10.359323  9.298095 10.58495
## Feb 1996      10.090105  9.674447 10.505762  9.449980 10.73023
## Mar 1996      10.827194 10.392356 11.262032 10.157530 11.49686
## Apr 1996      10.387272  9.974746 10.799797  9.751970 11.02257
## May 1996      10.535855 10.124310 10.947400  9.902063 11.16965
## Jun 1996      10.684438 10.273478 11.095399 10.051547 11.31733
## Jul 1996      10.833022 10.422249 11.243794 10.200420 11.46562
## Aug 1996      10.981605 10.570624 11.392587 10.348681 11.61453
## Sep 1996      11.130189 10.718601 11.541776 10.496332 11.76405
## Oct 1996      11.278772 10.866184 11.691360 10.643374 11.91417
## Nov 1996      11.427355 11.013374 11.841336 10.789812 12.06490
## Dec 1996      11.575939 11.160177 11.991701 10.935653 12.21622
```

From the forecast data frame, I can see that the predicted values (`Point Forecast`) oscillate, but generally trend upwards. This is also the case for each of the confidence intervals.

i) Transform your predictions and intervals to obtain predictions and intervals for the raw data.

In order to obtain the predictions and intervals for the raw data, I can transform my results by exponentiating them, as this will "undo" the log transformation.

```
fs <- data.frame(fcast.up)
exp_fs <- exp(fs)
exp_fs
```

```
##          Point.Forecast     Lo.80     Hi.80     Lo.95     Hi.95
## Jan 1994       12266.15  8167.776  18420.98  6557.406  22944.82
## Feb 1994       14231.06  9496.808  21325.39  7633.367  26531.29
## Mar 1994       29740.69 19435.352  45510.31 15446.039  57264.44
## Apr 1994       19155.57 12823.601  28614.12 10325.015  35538.55
## May 1994       22224.10 14892.391  33165.29 11997.062  41169.29
```

```
## Jun 1994       25784.17 17287.910  38455.96 13931.167  47722.01
## Jul 1994       29914.52 20060.576  44608.81 16166.920  55352.44
## Aug 1994       34706.51 23268.425  51767.24 18749.684  64243.33
## Sep 1994       40266.14 26978.234  60098.88 21731.414  74609.12
## Oct 1994       46716.35 31266.833  69799.76 25171.593  86701.60
## Nov 1994       54199.82 36222.588  81099.14 29138.303 100816.46
## Dec 1994       62882.07 41947.121  94265.21 33709.439 117301.10
## Jan 1995       15963.50 10576.019  24095.39  8467.626  30095.01
## Feb 1995       18520.69 12296.746  27894.84  9856.818  34799.85
## Mar 1995       38705.34 25187.424  59478.22 19971.978  75010.25
## Apr 1995       24929.58 16604.169  37429.39 13332.236  46615.13
## May 1995       28923.04 19282.891  43382.61 15491.308  54000.74
## Jun 1995       33556.21 22384.789  50302.86 17988.915  62595.16
## Jul 1995       38931.56 25975.203  58350.51 20876.250  72602.43
## Aug 1995       45167.99 30129.350  67712.95 24211.980  84261.89
## Sep 1995       52403.43 34933.792  78609.26 28063.308  97854.45
## Oct 1995       60797.91 40488.126  91295.56 32507.188 113709.81
## Nov 1995       70537.10 46906.920 106071.40 37631.712 132215.16
## Dec 1995       81836.41 54321.937 123287.17 43537.686 153825.30
## Jan 1996       20775.33 13680.408  31549.80 10917.199  39535.25
## Feb 1996       24103.32 15905.930  36525.36 12707.906  45717.20
## Mar 1996       50372.16 32609.406  77810.52 25784.541  98406.05
## Apr 1996       32444.02 21477.185  49010.83 17188.064  61241.03
## May 1996       37641.22 24942.033  56806.17 19971.526  70944.07
## Jun 1996       43670.95 28954.413  65867.39 23191.633  82234.47
## Jul 1996       50666.58 33598.927  76404.29 26914.485  95379.94
## Aug 1996       58782.83 38972.970  88662.00 31215.847 110694.47
## Sep 1996       68199.23 45188.648 102927.07 36182.525 128546.45
## Oct 1996       79124.03 52374.957 119534.47 41913.941 149368.27
## Nov 1996       91798.88 60680.304 138875.94 48523.924 173667.61
## Dec 1996      106504.10 70275.390 161409.61 56142.759 202040.72
```

From the forecast data frame, I can once again see that the predicted values for the raw data (`Point.Forecast`) oscillate, but generally trend upwards. This is also the case for the confidence intervals.

j) How could you improve these predictions by modifying the model?

A primary assumption for time series forecasting is that the data are stationary. Stationarity means that the series are normally distributed and the mean and variance are constant over a long time period. Therefore, the approaches in which I would modify my model would help to ensure this assumption.

One approach to improving my predictions would be to take the first difference of the series. First-differencing a time series will remove a linear trend. The log transformation helped reduce the exponential growth observed in the model, but there was still an obvious linear trend in the log-series. So, differencing the series by an order of 1 would help to address the trend that is still prevalent in the series. This would then lead to a stationary series that would produce more accurate forecasts.

I can evaluate the stationarity of the series by using a KPSS test.

```
print("Original series: ")
```

```
## [1] "Original series: "
```

```
kpss.test(sales_ts_log$sales_log)
```

```
## Warning in kpss.test(sales_ts_log$sales_log): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  sales_ts_log$sales_log
## KPSS Level = 1.7909, Truncation lag parameter = 3, p-value = 0.01
```

```
print("First Difference: ")
```

```
## [1] "First Difference: "
```

```
kpss.test(diff(sales_ts_log$sales_log, 1))
```

```
## Warning in kpss.test(diff(sales_ts_log$sales_log, 1)): p-value greater than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(sales_ts_log$sales_log, 1)
## KPSS Level = 0.062948, Truncation lag parameter = 3, p-value = 0.1
```

The resulting p-value for the original log-series of the KPSS test is 0.01, which is less than $\alpha = 0.05$. So, I can reject the null hypothesis that the series is stationary. Then, testing the first difference of the log-series, I see that the series becomes stationary and no higher degree of differencing is warranted. Thus, the first-difference modification to my model would help to improve my model stationarity and my forecasting predictions.

Another approach would be to decompose the model and remove the trend and seasonal variation. As seen in `Part 1A` and `Part 1C`, the decomposition model reduces the time series into three components: trend, seasonal effects, and random errors. In order to forecast accurately, I want to model the random errors as some form of stationary process. From the decomposition plots, I know that there is still prevalent trend and seasonality in both the original and log transformed series. So, to achieve stationarity, I could decompose the time series and remove these effects. This would be accomplished by dividing the series by the trend and seasonal variation in the original multiplicative series or subtracting the components from the additive log-series. Thus, the model would then be built on a stationary series, satisfying model assumptions, that would lead to more reliable forecasts.

Overall, these data modifications to my model would help to improve my predictions.

Question 2 - Cross-Validation

The `gafa_stock` data set from the `tsibbledata` package contains historical stock price data for Google, Amazon, Facebook and Apple.

a) Define the accuracy measures returned by the `accuracy` function. Explain how the given code calculates these measures using cross-validation.

To start, I can take a closer look at the measures returned by the `accuracy()` function.

```
google_stock <- gafa_stock %>% filter(Symbol == "GOOG") %>%
  mutate(day = row_number()) %>% update_tsibble(index = day, regular = TRUE)
google_2015 <- google_stock %>% filter(year(Date) == 2015)
google_fit <- google_2015 %>%
  model(Mean = MEAN(Close), `Naïve` = NAIVE(Close), Drift = RW(Close ~ drift()))
google_jan_2016 <- google_stock %>% filter(yearmonth(Date)==yearmonth("2016 Jan"))
google_fc <- google_fit %>% forecast(google_jan_2016)

google_fc %>% accuracy(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
```

From the accuracy results, I can see that:

- `ME` represents the Mean Error. The mean error refers to the average of all the errors in a set. An "error" in this context is an uncertainty in a measurement, or the difference between the measured value and true/correct value (residual).

$$ME = \frac{1}{n}\sum_{t=1}^{n} e_t$$

- `RMSE` represents the Root Mean Squared Error. Root Mean Square Error (RMSE) is the standard deviation of the residuals, as it refers to the square root of the average of the squared error terms. The RMSE is a measure of how spread out these errors/residuals are.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

- `MAE` represents the Mean Absolute Error. The mean absolute error refers to the average of the absolute value of all the residuals in a set.

$$MAE = \frac{1}{n}\sum_{t=1}^{n} |e_t|$$

- `MPE` represents the Mean Percentage Error. The MPE is the computed average of percentage errors by which forecasts of a model differ from actual values of the quantity being forecast. This metric performs better in the case when the forecast and actual measurements have a trend.

$$MPE = \frac{100\%}{n}\sum_{t=1}^{n} \frac{e_t}{y_t}$$

- `MAPE` represents the Mean Absolute Percentage Error. The mean absolute percentage error is the computed absolute average of percentage errors by which forecasts of a model differ from actual values of the quantity being forecast. This metric looks at the relative error in the forecast and 'actual' measurements.

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{e_t}{y_t} \right|$$

- `MASE` represents the Mean Absolute Scaled Error. It is the mean absolute error of the forecast values, divided by the mean absolute error of the in-sample one-step naive forecast. The MASE is a scale-free error metric that gives each error as a ratio compared to a baseline's average error.

$$MASE = \frac{MAE}{MAE_{in-sample,naive}}$$

- `ACF1` represents the Autocorrelation of errors at lag 1. A lag 1 autocorrelation is the correlation between values that are one time period apart. It is a measure of how much is the current value influenced by the previous values in a time series.

$$r_k = \frac{\sum_{i=1}^{N-k}(Y_i - Y)(Y_{i+k} - Y)}{\sum_{i=1}^{N}(Y_i - Y)^2}$$

In general, cross-validation, or out-of-sample testing, is a validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is useful for prediction as it estimates how accurately a model will perform in practice. This out-of-sample cross-validation is accomplished by withholding some of the known data to act as a test set. The remaining data is used to train the model and then the withheld data is used to measure the accuracy of the models predictions. This is achieved as the various measures take the difference between the observed known outcome values and the values predicted by the model.

More specifically, the given code calculates all of the above measures using cross-validation. This is accomplished as the `accuracy()` function measures test set forecast accuracy based on ["actual data" - "forecast data"]. In this context, `google_stock` is the observed known outcomes ("actual data") and `google_fc` is a forecast object that contains the predicted values ("forecast data").

Therefore, cross-validation can be used in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

b) Obtain Facebook stock data from the `gafa_stock` dataset. Use cross-validation to compare the RMSE forecasting accuracy of naive and drift models for the `Volume` series, as the forecast horizon is allowed to vary.

To start, I will retrieve the Facebook stock data from the `gafa_stock` dataset and focus on the `Volume` series.

```
facebook_stock <- gafa_stock %>% filter(Symbol == "FB") %>%
  mutate(day = row_number()) %>% update_tsibble(index = day, regular = TRUE)
```

```
t1 <- head(facebook_stock)[,c(2,8)]
t2 <- tail(facebook_stock)[,c(2,8)]
grid.arrange(tableGrob(t1), tableGrob(t2), ncol = 2)
```

	Date	Volume
1	2014-01-02	43195500
2	2014-01-03	38246200
3	2014-01-06	68852600
4	2014-01-07	77207400
5	2014-01-08	56682400
6	2014-01-09	92253300

	Date	Volume
1	2018-12-21	56901500
2	2018-12-24	22066000
3	2018-12-26	39723400
4	2018-12-27	31202500
5	2018-12-28	22627600
6	2018-12-31	24625300

From the tables, I can see that the data spans from the start of 2014 to the end of 2018. So, my year of interest will be 2018, which will be filtered out as a 'test set' in order to perform my cross-validation.

I will create a single test set (`fb_2018`), that includes all of 2018, as well as a series of 'test sets' (`fb_2018_tr`), each consisting of one observation and corresponding to a 'training set' consisting of the prior observations.

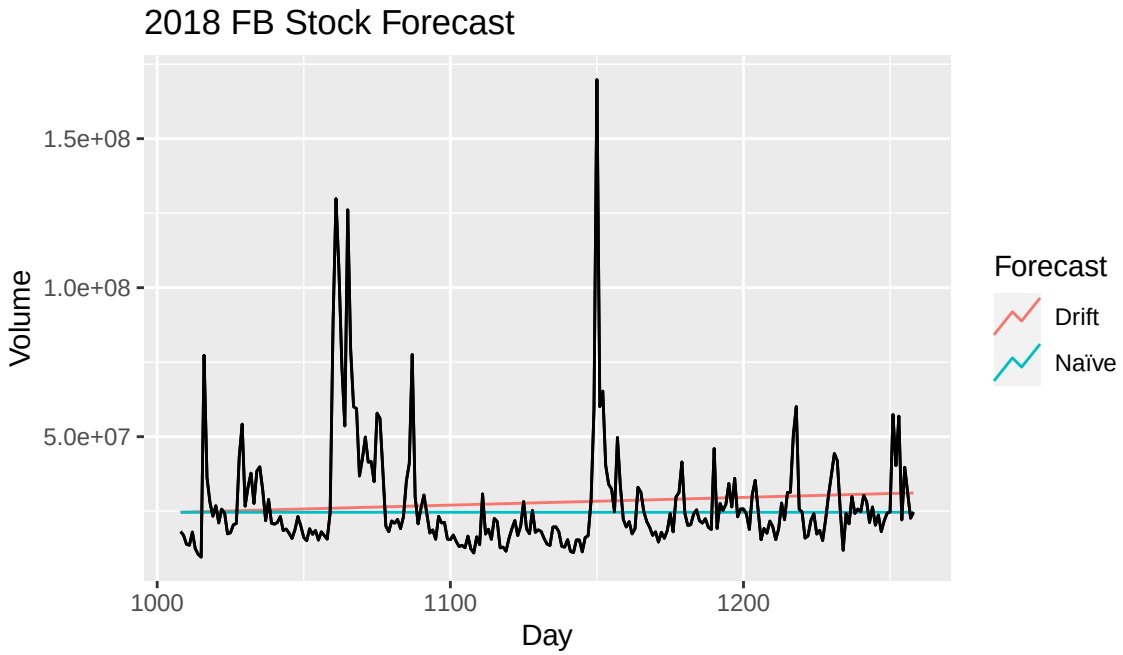
```
fb_2018 <- facebook_stock %>% filter(year(Date) == 2018)
fb_2018_tr <- fb_2018 %>% slice(1:(n()-1)) %>% stretch_tsibble(.init=3, .step=1)
```

Then, I can first fit the naive and drift models to the Facebook data for 2018 and plot the results with the actual values for the single test set (`fb_2018`).

```
fb_fit <- fb_2018 %>% model(`Naïve` = NAIVE(Volume),Drift = RW(Volume ~ drift()))
```

```
fb_jan_2018 <- facebook_stock %>% filter(year(Date) == "2018")
fb_fc_p <- fb_fit %>% forecast(fb_jan_2018)

fb_fc_p %>% autoplot(fb_2018,level=NULL) + autolayer(fb_jan_2018,Volume,color='black') +
  xlab("Day") + ggtitle("2018 FB Stock Forecast") + ylab("Volume") +
  guides(colour=guide_legend(title="Forecast"))
```



From the plot, I can see that both models are fairly conservative as they are strictly linear. Neither model fits the fluctuations of the the data, but they do appear to roughly span the mean value of the series. Yet,

the Drift model has a positive slope and increases slightly overtime, where the Naïve model maintains a zero-slope. So, I can also compare the RMSE scores of each model in order to determine which one actually better fits the series.

```
fb_fc_p %>% accuracy(facebook_stock)
```

```
## # A tibble: 2 x 10
##   .model Symbol .type       ME      RMSE       MAE   MPE  MAPE  MASE  ACF1
##   <chr>  <chr>  <chr>    <dbl>     <dbl>     <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift  FB     Test  -237864. 19337527. 11932302. -27.0  45.5  1.44 0.634
## 2 Naïve  FB     Test  3024729. 19362901. 10665670. -12.4  36.2  1.28 0.626
```

In general, lower values of RMSE indicate better fit. However, because the RMSE has the same unit as the dependent variable, there is no set scale that would suggest how small is good or how large is bad - there is no absolute standard. Yet, as the dependent variable for both models is `Volume`, the two scores will have the same units and can be compared to one another.

From the single-fold cross-validation accuracy results, I can see that the RMSE score is:

- 19337527 for the Drift model.
- 19362901 for the Naïve Model.

As a result, the Drift model has a smaller score, which means that it would be a better fit for the series. However, this was performed on a single test set. So, I can also use my series of 'test sets' (`fb_2018_tr`) in order to increase the number of folds in my cross-validation. This will help reduce some bias and variation in my forecast results.

```
fb_fc <- fb_2018_tr %>% model(`Naïve` = NAIVE(Volume), Drift = RW(Volume ~ drift())) %>%
  forecast(h=1)
```

Then, I can use the k-fold cross-validation to compare the RMSE forecasting accuracy of the Naïve and Drift models.

```
fb_fc %>% accuracy(facebook_stock)
```

```
## # A tibble: 2 x 10
##   .model Symbol .type       ME      RMSE      MAE   MPE  MAPE  MASE   ACF1
##   <chr>  <chr>  <chr>    <dbl>     <dbl>    <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 Drift  FB     Test  -103330. 16828057. 8802102. -6.39  27.1  1.05 -0.261
## 2 Naïve  FB     Test    43324. 16632932. 8693054. -6.13  26.8  1.04 -0.262
```

From the k-fold cross-validation accuracy results, I can see that the RMSE score is:

- 16828057 for the Drift model.
- 16632932 for the Naïve Model.

Once again, lower values of RMSE indicate better fit and the dependent variable for both models is `Volume`, so the two scores can be compared to one another. Therefore, I can see that the Naïve Model now has a lower RMSE score, which indicates that it is actually a better fit for the series.

Overall, I was able to use cross-validation to compare the RMSE forecasting accuracy of Naïve and Drift models for the 2018 Facebook `Volume` series.

Question 3 - ARIMA Model

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

```
library(fma)
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.

First, I need to convert the data into a suitable time series object.

```
sheep_ts <- as_tsibble(sheep, regular = TRUE)
```

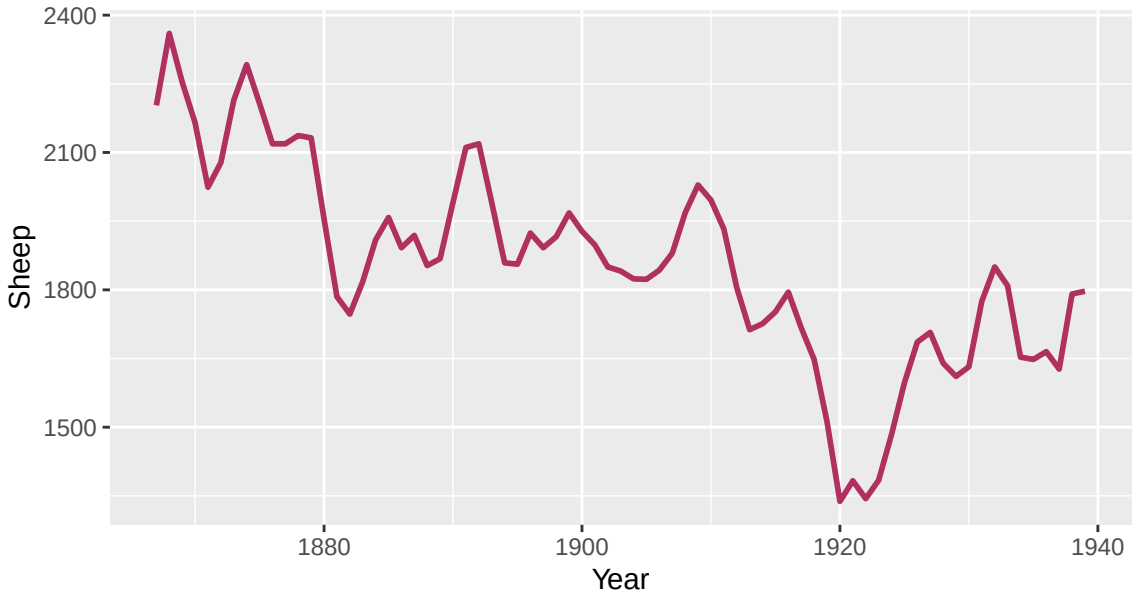
```
t1 <- head(sheep_ts)
t2 <- tail(sheep_ts)
grid.arrange(tableGrob(t1), tableGrob(t2), ncol = 2)
```

	index	value
1	1867	2203
2	1868	2360
3	1869	2254
4	1870	2165
5	1871	2024
6	1872	2078

	index	value
1	1934	1653
2	1935	1648
3	1936	1665
4	1937	1627
5	1938	1791
6	1939	1797

Then, I am able to produce a time plot for the series.

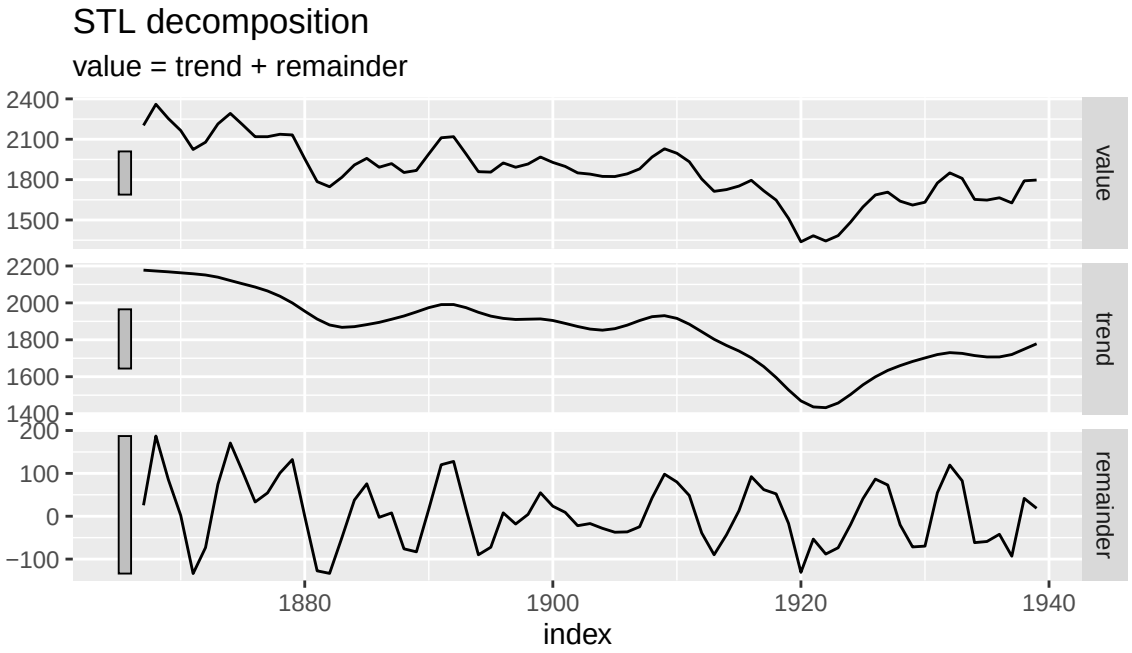
```
sheep_ts %>% autoplot(value,colour="maroon",size=1) + labs(x="Year",y="Sheep")
```



From the plot, I can see that the annual sheep population generally has a negative trend. The value fluctuates over a downward trend, with peaks and valleys of varying heights. The maximum sheep population is about 2360 (in 1868) and the minimum is 1350 (in 1920). Yet, this minimum value occurred in a particularly low period and the population rose slightly afterwards. Overall, the series appears to oscillate downwards.

I can further examine the series by decomposing the original time series into its trend and random components.

```
sheep_ts %>% model(STL(value)) %>% components() %>% autoplot()
```



From the plots, I can see that:

- the overall sheep population has a fluctuating downward trend with seasonality.
- the population trend is generally downward. However, the downward trend is not extremely persistent as there are also some upward fluctuations.
- the random element fluctuates somewhat consistently, with oscillating peaks.

These trend and random patterns in the time series will be important to consider in building my model in order to generate reliable forecasts.

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)? Express this ARIMA model using backshift operator notation.

An Auto Regressive model takes the form of the following:

$$Y_t = \sum_{j=1}^{p} \phi_j Y_{t-j} + \omega_t$$

where $\omega_t \sim \mathrm{N}(0, \sigma^2)$.

Furthermore, the formula for a differenced series is:

$$Y_t^{'} = Y_t - Y_{t-1}$$

Therefore, based on the given formula, the model I am trying to fit is an auto regressive model with an order of 3 (an AR(3)). It is also of the first difference of the series. This is due to the fact that there are three AR terms, which are represented by the difference between two subsequent time periods. There are also no moving average components present in the model. Therefore, in this model, **p = 3, d = 1, and q = 0.**

```
mod <- sheep_ts %>% model(ARIMA(value ~ 0 + pdq(3,1,0)))
mod %>% report()
```

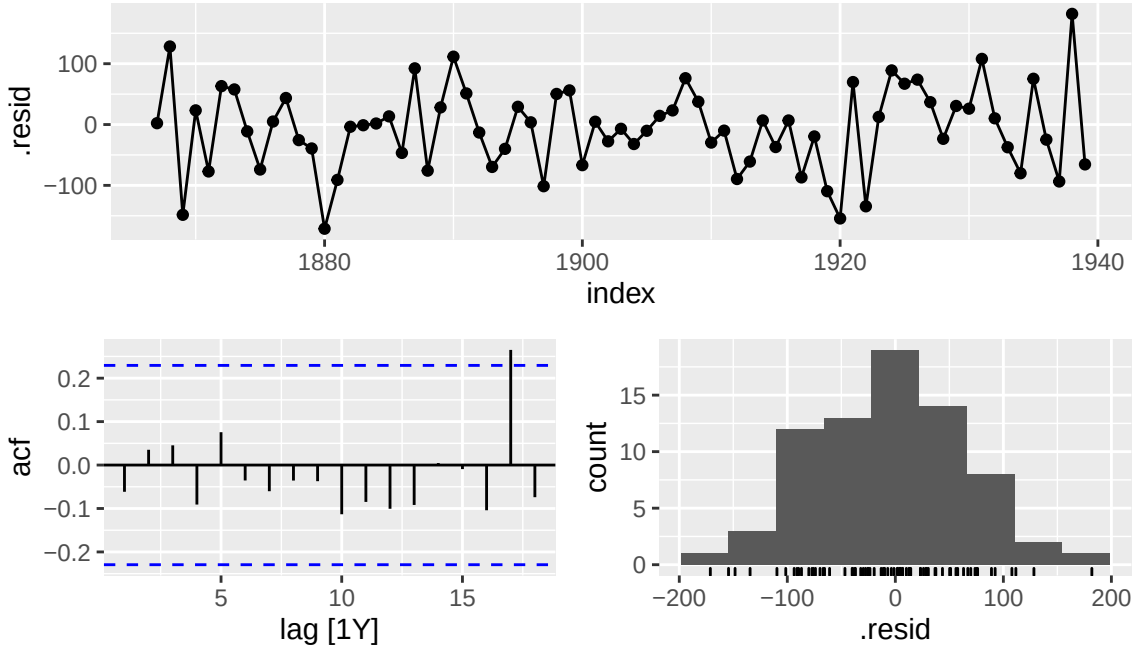
```
## Series: value
## Model: ARIMA(3,1,0)
##
## Coefficients:
##          ar1      ar2      ar3
##       0.4210  -0.2018  -0.3044
## s.e.  0.1193   0.1363   0.1243
##
## sigma^2 estimated as 4991:  log likelihood=-407.56
## AIC=823.12   AICc=823.71   BIC=832.22
```

The model in back-shift notation is below.

$$(1 - 0.4210B + 0.2018B^2 + 0.3044B^3)(1 - B)y_t = c + \epsilon_t$$

I can then assess the fit of this model by examining the residuals.

```
mod %>% gg_tsresiduals()
```



The residuals are normally distributed and do not have an apparent trend, indicating leftover white noise. The ACF does not have an apparent trend and oscillates around zero. As a result, my model satisfies all of the necessary assumptions.

Therefore, the AR(3) of the differenced series is an appropriate model for the sheep population time series.

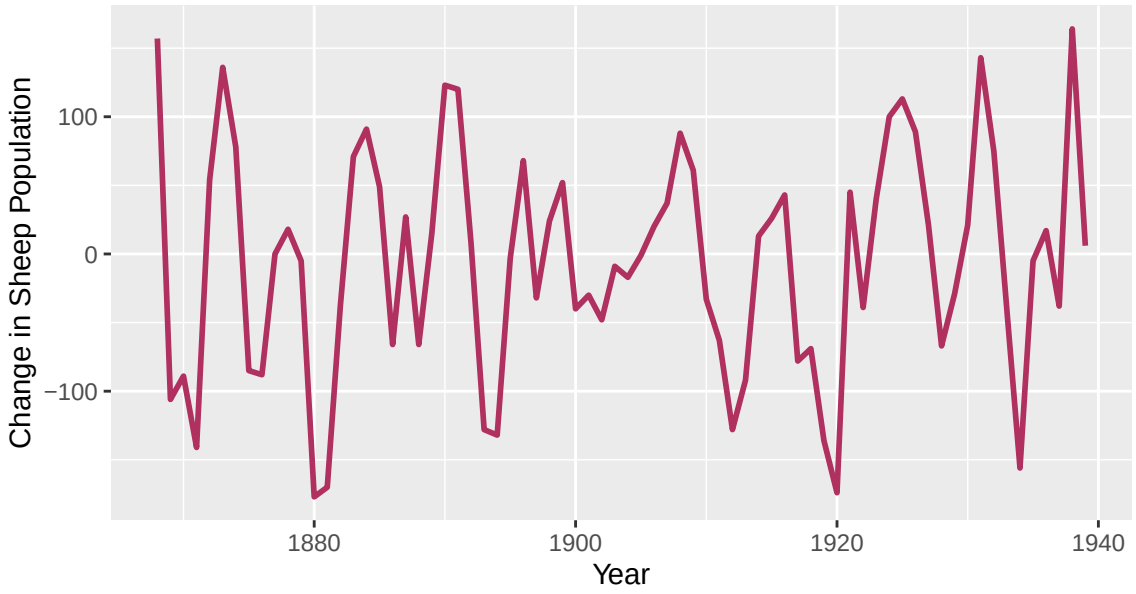
c) By examining the ACF and PACF of the differenced data, explain why this model is appropriate.

First, I need to perform the first order difference and convert the data into a suitable time series object.

```
diff_sheep_ts <- as_tsibble(diff(sheep), regular = TRUE)
```

Then, I am able to produce a time plot for the differenced series.

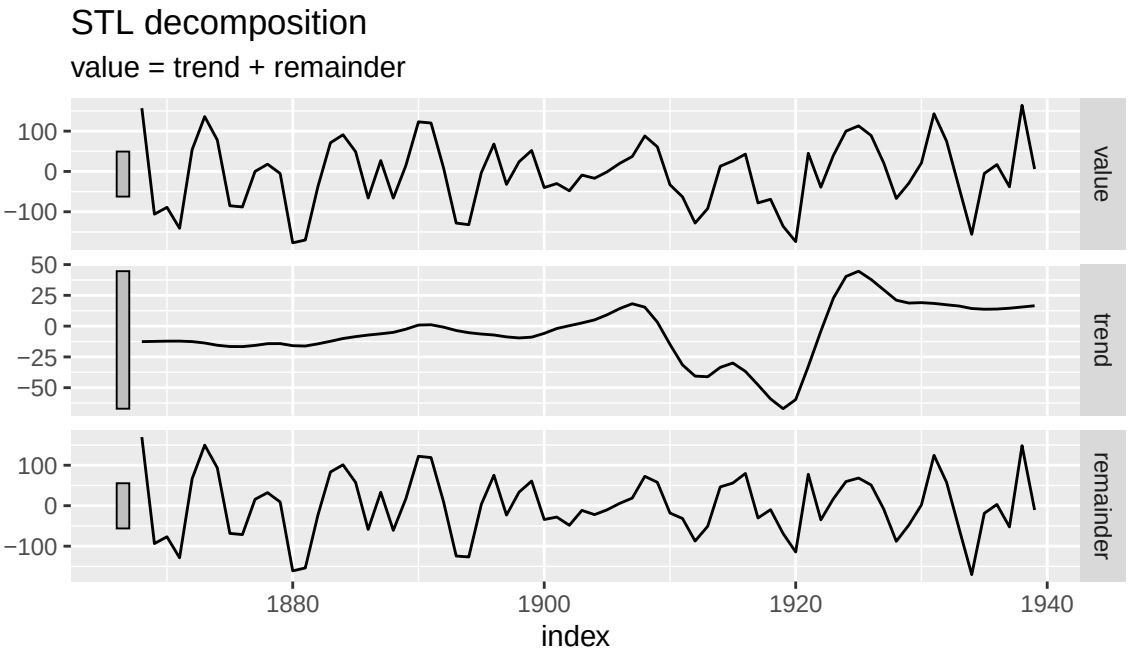
```
diff_sheep_ts %>% autoplot(value, colour = "maroon", size = 1) +
  labs(x = "Year", y = "Change in Sheep Population")
```



From the plot, I can see that there is no major trend remaining in the series, as the series now oscillates around the zero-line. There is still prevalent seasonality, which fluctuates somewhat consistently throughout the time series.

I can further examine the series by decomposing the differenced data into its trend and random components.

```
diff_sheep_ts %>% model(STL(value)) %>% components() %>% autoplot()
```

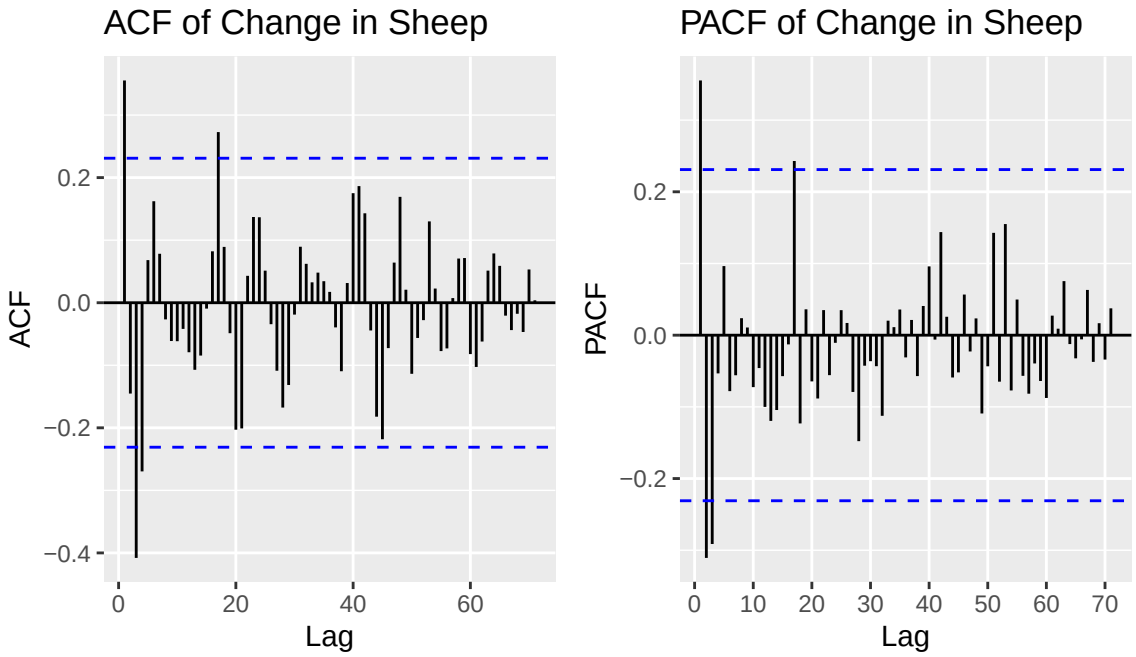


From the plots, I can see that:

- the overall differenced sheep population appears to be stationary and has fluctuating seasonality with no major trend.
- the differenced population trend is generally flat. However, there are some deviating fluctuations in the more recent years, both upwards and downwards.
- the differenced random element fluctuates somewhat persistently, with nearly constant variance.

Then, I can examine the ACF and PACF of the differenced data

```
p1 <- ggAcf(diff_sheep_ts$value, lag.max = 104, main = "ACF of Change in Sheep")
p2 <- ggPacf(diff_sheep_ts$value, lag.max = 104, main = "PACF of Change in Sheep")
egg::ggarrange(p1, p2, nrow = 1)
```



From the plot, I can see that:

- the ACF gradually declines and oscillates around the zero-line. This means I can expect a negative AR parameter for the random elements. There is a slight spike around lag 17, which indicates that there could be some correlation, not just a statistical fluke, in the model. But, this is only a slight deviation and there does not appear to be any other indications of correlation in the differenced series.
- the PACF drastically cuts off after the third lag, reinforcing the notion that I should a apply an AR of order 3 in my model. It then gradually declines and oscillates around the zero-line. Although there is a slight spike at lag 17, there doesn't appear to any other major indications of correlation present in the model.

As a result, the differenced series does not violate the assumption of stationarity or of non-correlation for building a model. Therefore, the differenced model is appropriate.

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 $\phi_1 = 0.42$, $\phi_2 = -0.20$, & $\phi_3 = -0.30$. Without using the forecast function, calculate forecasts for the next three years (1940–1942).

Given the model:

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

I am able to calculate forecasts for the next three years (1940-1942) by hand. This is done by using the values for the past four years in the model, given the ϕ_i variables.

First, I will create a data frame for the last five years and set the ϕ_i variables.

```
phi_1 <- 0.42
phi_2 <- -0.20
phi_3 <- -0.30
last_five <- data.frame(Year = c(1935,1936,1937,1938,1939),
                        Sheep = c(1648,1665,1627,1791,1797))
```

Then, I can determine my forecasts. In order to predict values for 1940-1942, I will first need to calculate the prediction for 1940, based on the previous four years, and subsequently use that value in my following calculations. This is accomplished in the loop below.

```
set.seed(8580)
my_year <- c(1940,1941,1942)
my_yt <- c()

for(i in my_year){
  y_t1 <- last_five[which(last_five$Year==(i-1)),]$Sheep
  y_t2 <- last_five[which(last_five$Year==(i-2)),]$Sheep
  y_t3 <- last_five[which(last_five$Year==(i-3)),]$Sheep
  y_t4 <- last_five[which(last_five$Year==(i-4)),]$Sheep

  y_t <- y_t1 + phi_1*(y_t1 - y_t2) + phi_2*(y_t2 - y_t3) +
    phi_3*(y_t3 - y_t4)  + rnorm(1)
  my_yt <- c(my_yt, y_t)
  last_five <- rbind(last_five,c(i,y_t))
}

results <- data.frame(Year = my_year, Predictions = my_yt)
results
```

```
##   Year Predictions
## 1 1940    1776.534
## 2 1941    1716.878
## 3 1942    1695.411
```

Now that I have manually computed the forecast values, I can compare my results with the actual `forecast()` function.

```
mod %>% forecast(h = 3)
```

```
## # A fable: 3 x 4 [1Y]
## # Key:     .model [1]
##   .model                          index          value .mean
##   <chr>                           <dbl>         <dist> <dbl>
## 1 ARIMA(value ~ 0 + pdq(3, 1, 0))  1940  N(1778, 4991) 1778.
## 2 ARIMA(value ~ 0 + pdq(3, 1, 0))  1941 N(1719, 15070) 1719.
## 3 ARIMA(value ~ 0 + pdq(3, 1, 0))  1942 N(1696, 24804) 1696.
```

My by-hand results are very similar to the `forecast()` function results (`.mean`), with the variation most likely due to the random element in the calculations. As a result, I predict that the sheep population will decline from 1940 to 1942.

e) Find the roots of your model's characteristic equation and explain their significance.

The roots of a model's characteristic equation are significant as they are an indication of stationarity in the model. This is a crucial assumption for time series models in order to produce reliable forecasts.

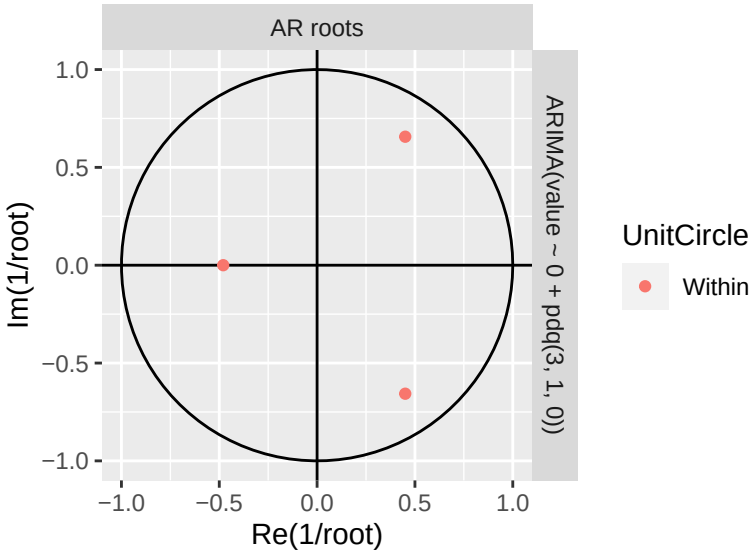
First, I can produce the value of the roots for my model's characteristic equation.

```
glance(mod)$ar_roots
```

```
## [[1]]
## [1]  0.710303+1.035517i -2.083645+0.000000i  0.710303-1.035517i
```

I can also plot the root values on the unit circle. There is an indication of stationarity in the model if the roots are within the bounds of the unit circle.

```
gg_arma(mod)
```



From the plot, I can see that each of the roots are within the unit circle, which means that the AR(3) model of the first differenced series satisfies the assumption of stationarity.

I can also statistically test for unit roots in the series. In the context of an ADF test, H_0 is the presence of a unit root, with rejection (H_A) indicating that the series does not have a unit root. The presence of a unit root would mean that the series is not stationary.

```
adf.test(diff_sheep_ts$value)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_sheep_ts$value
## Dickey-Fuller = -3.8322, Lag order = 4, p-value = 0.02223
## alternative hypothesis: stationary
```

The resulting p-value of the ADF test is 0.02, which is less than $\alpha = 0.05$. So, I can reject the null hypothesis that the series does have a unit root. This is an indication that the series is actually stationary.

Therefore, the roots of the model's characteristic equation are significant as they affirm the important assumption of model stationarity.

Question 4 - Model Averaging

Apply a Holt-Winters model to the ECOMPCTNSA time series data recorded in `Q4.csv`. Compare this model's forecasting performance to that of a seasonal ARIMA model using cross-validation. Then compare both of these models to the performance of a simple average of the ARIMA and Holt-Winters models.

A Holt-Winters Filtering of a time series is a classical form of exponential smoothing model. Exponential smoothing models are categorized by error, trend and seasonal components, which if present may be additive or multiplicative.

To start, I can load the data set and examine the `ECOMPCTNSA` variable.

```
Q4 <- read_csv(paste0(here::here(),"/assignments/assignment_3/Q4.csv"))
```

```
## Parsed with column specification:
## cols(
##   DATE = col_character(),
##   ECOMPCTNSA = col_double()
## )
```

```
t1 <- head(Q4,10)
t2 <- tail(Q4,10)
grid.arrange(tableGrob(t1), tableGrob(t2), ncol = 2)
```

	DATE	ECOMPCTNSA
1	1/10/1999	0.7
2	1/1/2000	0.8
3	1/4/2000	0.8
4	1/7/2000	0.9
5	1/10/2000	1.2
6	1/1/2001	1.1
7	1/4/2001	1.0
8	1/7/2001	1.0
9	1/10/2001	1.3
10	1/1/2002	1.3

	DATE	ECOMPCTNSA
1	1/7/2017	8.6
2	1/10/2017	10.6
3	1/1/2018	9.4
4	1/4/2018	9.2
5	1/7/2018	9.3
6	1/10/2018	11.4
7	1/1/2019	10.3
8	1/4/2019	10.1
9	1/7/2019	10.6
10	1/10/2019	12.7

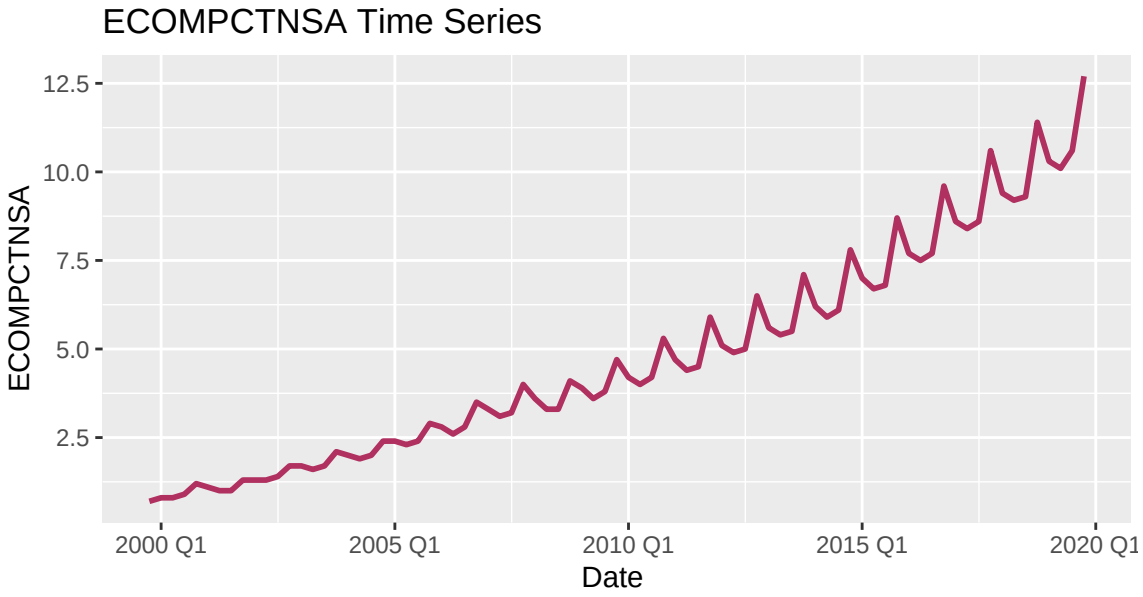
From the tables, I can see that the time series records values quarterly. There is a consistent time pattern with regular gaps between the observations.

In order to conduct my analysis, I first need to convert my data into a suitable time series object.

```
ecom <- Q4 %>% mutate(DATE = dmy(DATE)) %>% as_tsibble(index = 'DATE') %>%
  mutate(DATE = yearquarter(DATE)) %>% rename(date = DATE, value = ECOMPCTNSA)
ecomp_ts <- as_tsibble(ecom, index = date, regular = TRUE)
```

Then, I can plot the quarterly ECOMPCTNSA time series.

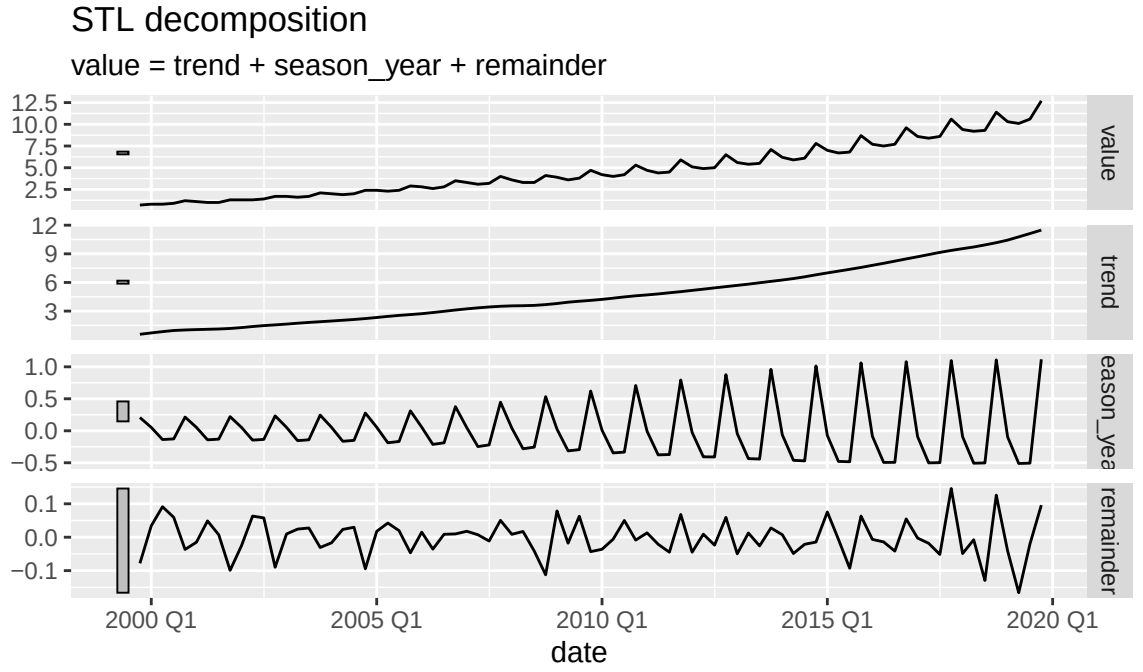
```
ecomp_ts %>% autoplot(value, colour = "maroon", size = 1) +
  labs(title = "ECOMPCTNSA Time Series", x = "Date ", y = "ECOMPCTNSA")
```



From the plot, I can see that there is a strong upward trend with prevalent seasonality. The size of the trend, seasonal, and random fluctuations seem to increase with the level of the time series, growing almost exponentially. As a result, a multiplicative seasonal change might be most applicable as the magnitude of the trend and seasonal change appear to increase over time as the data values increase. Yet, I can still apply both additive and multiplicative models and compare their forecasting performance.

I can also examine the series by decomposing the time series into its trend and seasonal components.

```
ecomp_ts %>% model(STL(value)) %>% components() %>% autoplot()
```



From the plots, I can see that:

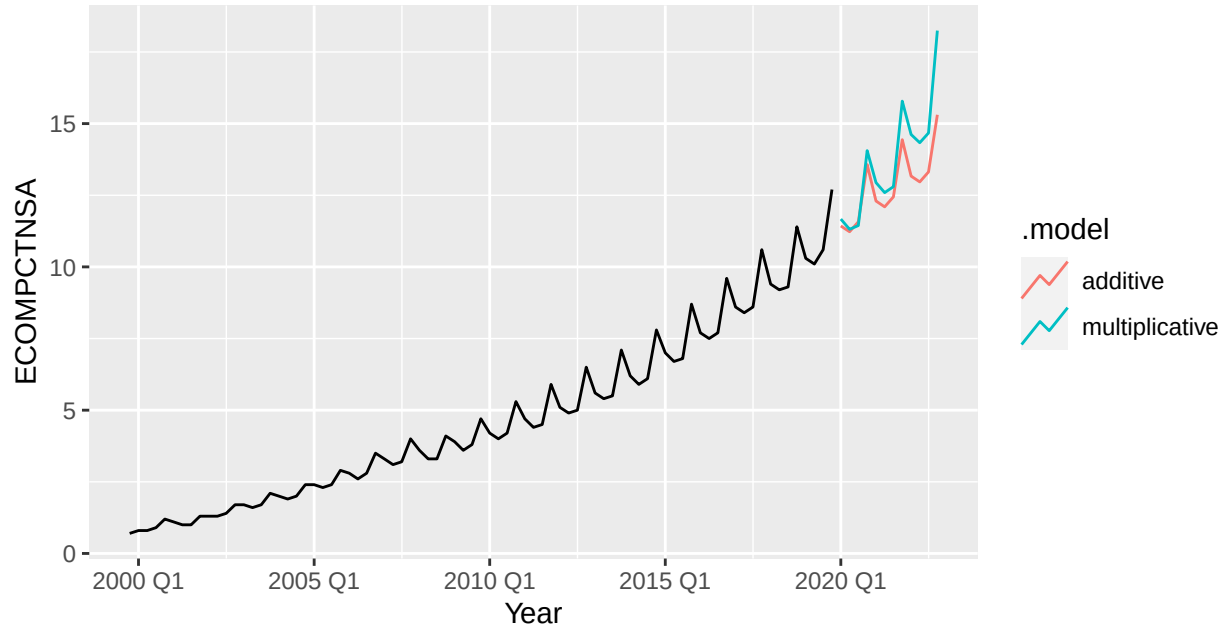
- the overall series appears to have a strong upward trend with fluctuating seasonality. Both the trend and seasonality seem to grow larger over time.
- the trend is strong and consistent in its upward rise. It appears to grow linearly.

- the seasonality fluctuates persistently, with variance that increases over time.
- the random element appears somewhat like white noise. Its variation oscillates almost randomly throughout the series.

As the trend is more linear and the seasonality is more exponential, it is possible that the additive model might actually be more applicable in this context. Overall, these trend and seasonal patterns in the time series will be important to consider in building my model in order to generate reliable forecasts.

Then, I can apply a Holt-Winters model to the ECOMPCTNSA time series data, in both additive and multiplicative variants.

```
fit <- ecomp_ts %>%
  model(additive = ETS(value ~ trend("A") + season("A") + error("A")),
    multiplicative = ETS(value ~ trend("M") + season("M") + error("M")))
fc <- fit %>% forecast(h = "3 years")
fc %>% autoplot(ecomp_ts, level = NULL) + xlab("Year") + ylab("ECOMPCTNSA")
```

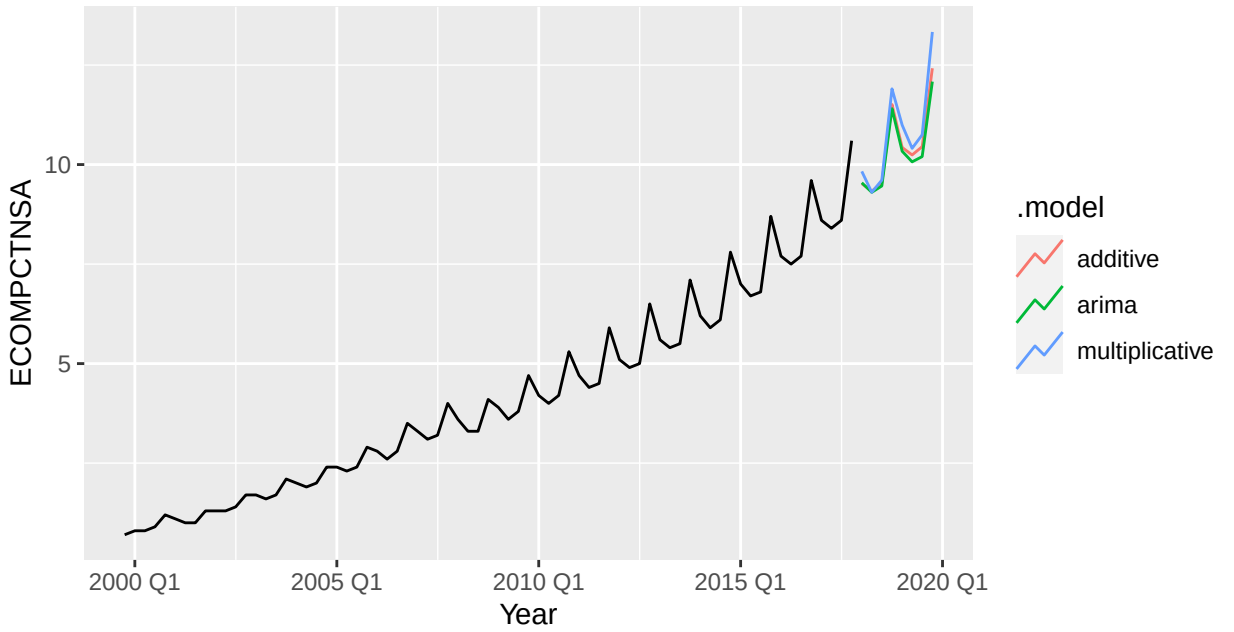


From the plot, I can see that both model forecasts follow the increasing trend and fluctuating seasonality of the series. Yet, the multiplicative forecast increases almost exponentially where the additive model increases more linearly.

Next, I can compare each model's forecasting performance to that of a seasonal ARIMA model using cross-validation. In order to use cross-validation, I need to split my current data into a training and test set. The training set will be used to build the different models. The test set, which will consist of the values from the last two years of the series, will be withheld and used to measure cross-validation accuracy between each of the models.

```
ecomp_train <- ecomp_ts %>% filter_index('1999 Q4'~'2017 Q4')
ecomp_test <- ecomp_ts %>% filter_index('2017 Q4'~'2019 Q4')
```

```
fit <- ecomp_train %>%
  model(additive = ETS(value ~ trend("A") + season("A") + error("A")),
    multiplicative = ETS(value ~ trend("M") + season("M") + error("M")),
    arima = ARIMA(value ~ trend() + season()))
fc <- fit %>% forecast(h = "2 years")
fc %>% autoplot(ecomp_train, level = NULL) + xlab("Year") + ylab("ECOMPCTNSA")
```



In order to compare each model's forecasting, I will consider the RMSE and MAPE score. As mentioned in `Part 2`, the RMSE is the standard deviation of the residuals, as it refers to the square root of the average of the squared error terms, and the MAPE is the computed absolute average of percentage errors by which forecasts of a model differ from actual values of the quantity being forecast. In general, lower values of RMSE or MAPE indicate a better fit.

```
fc %>% accuracy(ecomp_test)
```

```
## # A tibble: 3 x 9
##   .model         .type      ME  RMSE   MAE    MPE  MAPE  MASE    ACF1
##   <chr>          <chr>   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 additive       Test  -0.0554 0.173 0.163 -0.663  1.57   NaN  0.423
## 2 arima          Test   0.0747 0.274 0.187  0.519  1.71   NaN  0.496
## 3 multiplicative Test  -0.391  0.438 0.391 -3.70   3.70   NaN -0.0833
```

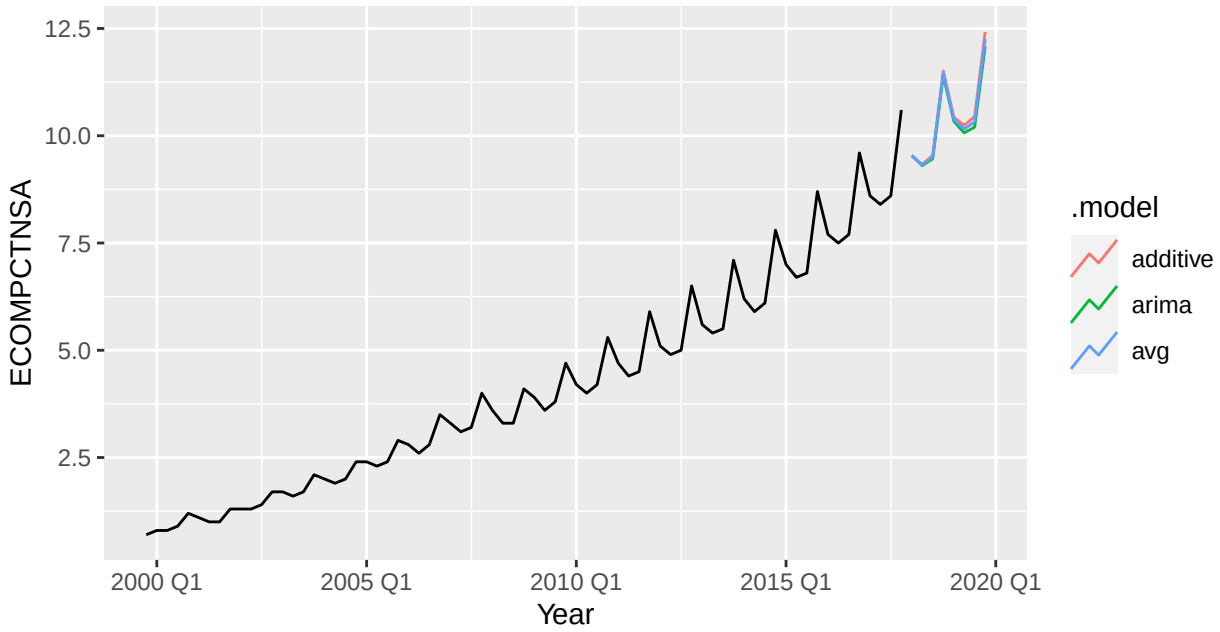
From the results, I can see that:

- the RMSE is 0.17 and the MAPE is 1.57 for the additive model
- the RMSE is 0.44 and the MAPE is 3.64 for the multiplicative model
- the RMSE is 0.27 and the MAPE is 1.71 for the ARIMA model

Thus, the additive model has the lowest RMSE and MAPE, which means that it fits the ECOMPCTNSA series the best out of these models.

Last, I can then compare these models to the performance of a simple average of the ARIMA and Holt-Winters model. As the additive model performed better than the multiplicative model, I will average the additive model with the ARIMA model for this comparison.

```
fit <- ecomp_train %>%
  model(additive = ETS(value ~ trend("A") + season("A") + error("A")),
        arima = ARIMA(value ~ trend() + season())) %>%
  mutate(avg = (additive + arima)/2)
fc <- fit %>% forecast(h = "2 years")
fc %>% autoplot(ecomp_train, level = NULL) + xlab("Year") + ylab("ECOMPCTNSA")
```



Once again, I will consider the RMSE and MAPE scores in order to asses each model's performance.

```
fc %>% accuracy(ecomp_test)
```

```
## # A tibble: 3 x 9
##   .model   .type       ME  RMSE   MAE     MPE  MAPE  MASE  ACF1
##   <chr>    <chr>    <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 additive Test  -0.0554   0.173 0.163 -0.663   1.57   NaN 0.423
## 2 arima    Test   0.0747   0.274 0.187  0.519   1.71   NaN 0.496
## 3 avg      Test   0.00963  0.213 0.171 -0.0720  1.60   NaN 0.472
```

From the results, I can see that:

- the RMSE is 0.17 and the MAPE is 1.57 for the additive model
- the RMSE is 0.27 and the MAPE is 1.71 for the ARIMA model
- the RMSE is 0.21 and the MAPE is 1.60 for the average of the two models

Thus, the additive model still has the lowest RMSE and MAPE, which means that it has the best performance and fits the ECOMPCTNSA series the best.

Question 5 - Vector Autoregression

Conduct an EDA and develop a VAR model for the period 1946-2003. Forecast the last three years, 2004-2006, conducting residual diagnostics. Examine the relative advantages of logarithmic transformations and the use of differences.

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 Q5.csv. All of the observations are measured in billions of dollars, after adjustment by the Consumer Price Index (CPI).

```
Q5 <- read_csv(paste0(here::here(),"/assignments/assignment_3/Q5.csv"))
```

```
## Parsed with column specification:
## cols(
##   Year = col_double(),
##   RMC = col_double(),
##   RCC = col_double(),
##   RDPI = col_double()
## )
```

```
t1 <- head(Q5)
t2 <- tail(Q5)
grid.arrange(tableGrob(t1), tableGrob(t2), ncol = 2)
```

	Year	RMC	RCC	RDPI
1	1946	117.9	49.7	826.2
2	1947	126	59.2	767.7
3	1948	137.3	67.6	790.9
4	1949	157.1	81.5	800
5	1950	186.7	99.2	871.8
6	1951	198.8	97.7	888.5

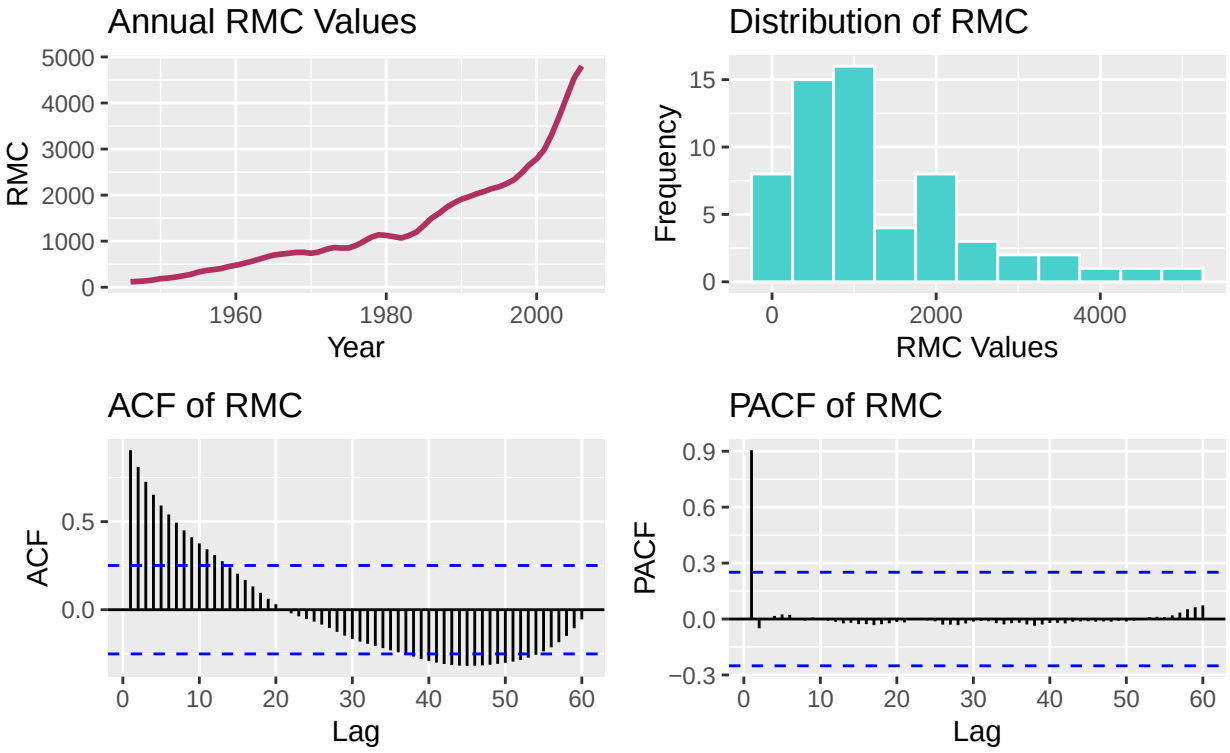
	Year	RMC	RCC	RDPI
1	2001	2985	1072.6	4227.4
2	2002	3317.3	1118.5	4352.5
3	2003	3709	1150.1	4436.1
4	2004	4133.6	1181.7	4595.9
5	2005	4548.5	1191.2	4626.8
6	2006	4799.5	1209.2	4723.8

First, I need to convert the data into a suitable time series object.

```
credit_ts <- as_tsibble(Q5, index = Year, regular = TRUE)
```

Then, I can begin my EDA. I will start this exploration by examining each series individually, starting with RMC.

```
p1 <- credit_ts %>% ggplot(aes(x=Year,y=RMC)) + geom_line(colour="maroon",size=1) +
  ggtitle('Annual RMC Values') + theme(legend.position = "none")
p2 <- ggplot(credit_ts, aes(RMC)) +
  geom_histogram(binwidth = 500, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Distribution of RMC", x = "RMC Values",y="Frequency")
p3 <- ggAcf(credit_ts$RMC, lag.max = 104, main = "ACF of RMC")
p4 <- ggPacf(credit_ts$RMC, lag.max = 104, main = "PACF of RMC")
egg::ggarrange(p1, p2, p3, p4, nrow = 2)
```

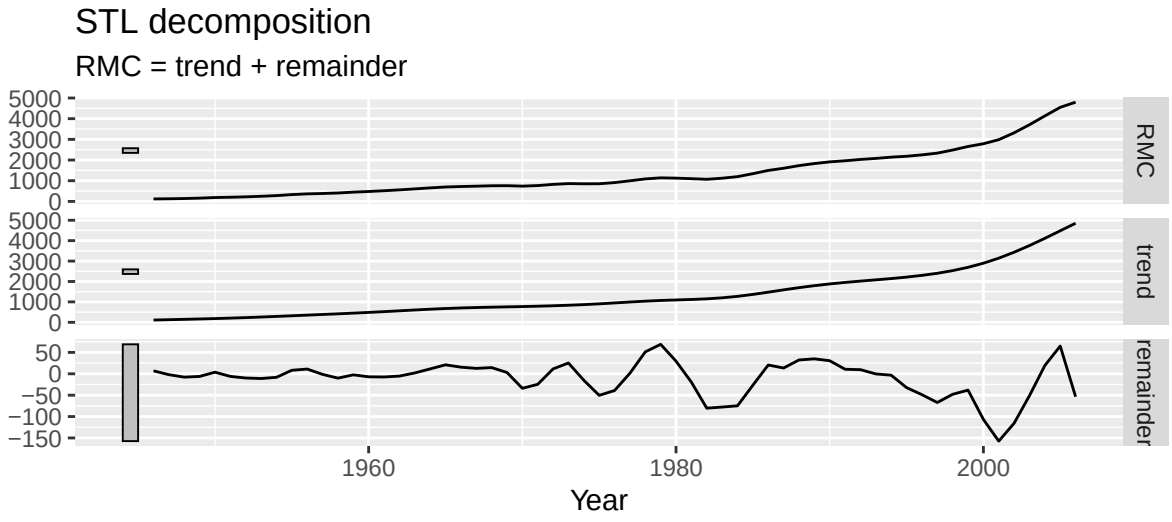


From the plots, I can see that:

- the time series has a strong upward trend, with no apparent seasonality. The series appears to grow almost exponentially as it increases more drastically in the more recent years.
- the histogram shows that the values are unimodal but not necessarily normal. There is a positive right skew as there is a high frequency of lower values with a slight tail towards higher RMC scores.
- the ACF declines very gradually and oscillates around the zero line.
- the PACF has a sharp cut off after the first lag, then oscillates around the zero-line.

Then, I can further examine the `RMC` series by decomposing it into its trend and random elements.

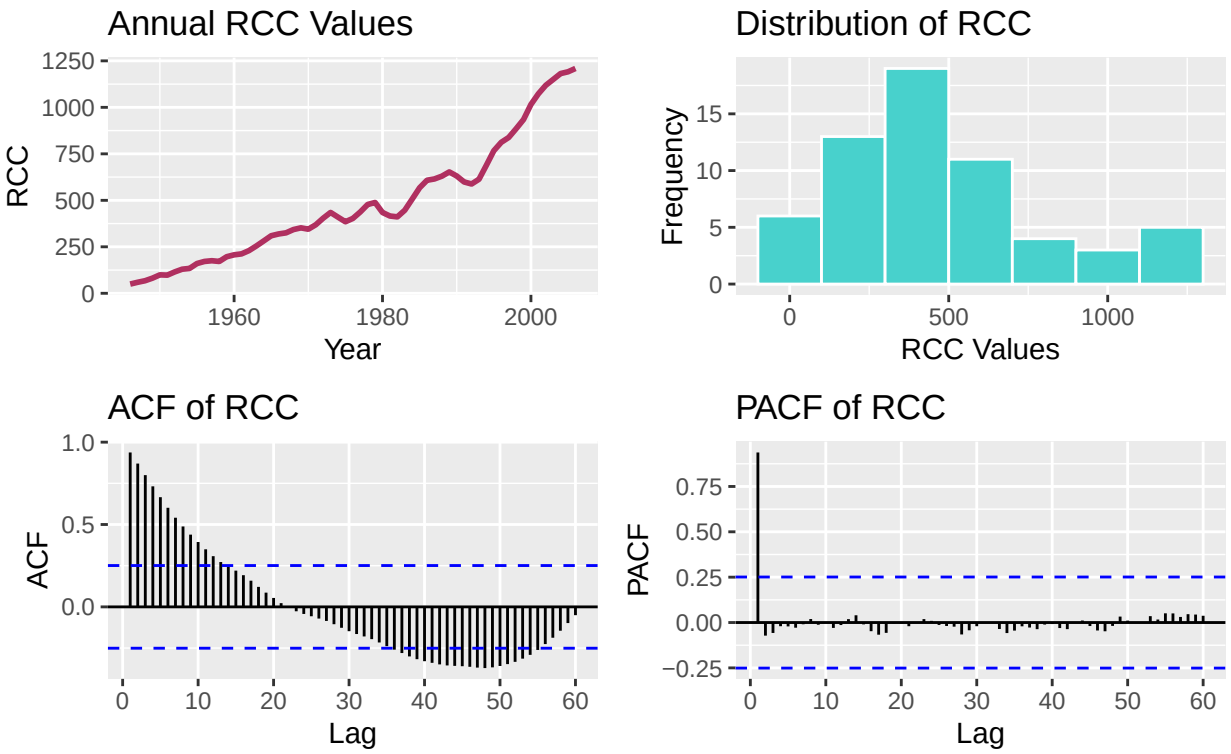
```
credit_ts %>% model(STL(RMC)) %>% components() %>% autoplot()
```



The decomposition shows that there is a strong, almost exponential, upward trend. There is also some inconsistent random variation in the series.

Then I can examine the RCC.

```
p1 <- credit_ts %>% ggplot(aes(x=Year,y=RCC)) + geom_line(colour="maroon",size=1) +
  ggtitle('Annual RCC Values') + theme(legend.position = "none")
p2 <- ggplot(credit_ts, aes(RCC)) +
  geom_histogram(binwidth = 200, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Distribution of RCC", x = "RCC Values",y="Frequency")
p3 <- ggAcf(credit_ts$RCC, lag.max = 104, main = "ACF of RCC")
p4 <- ggPacf(credit_ts$RCC, lag.max = 104, main = "PACF of RCC")
egg::ggarrange(p1, p2, p3, p4, nrow = 2)
```

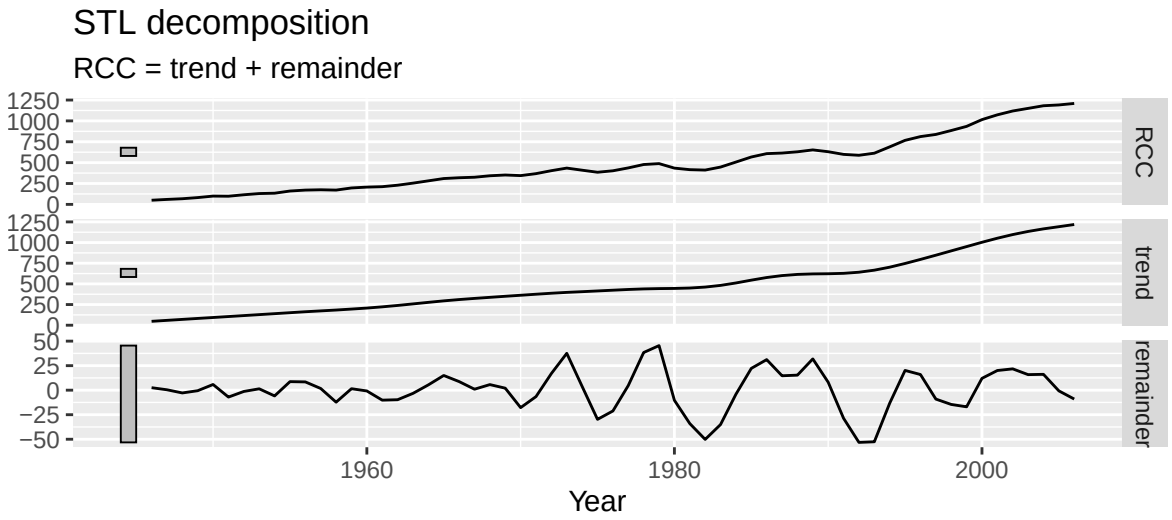


From the plots, I can see that:

- the time series has a strong upward trend, with no apparent seasonality. The series appears to grow almost linearly, with some variations throughout the series.
- the histogram shows that the values are mostly unimodal but not necessarily normal. There is a slight positive right skew as there is a higher frequency of lower values with a slight tail towards higher RCC scores.
- the ACF declines very gradually and oscillates around the zero line.
- the PACF has a sharp cut off after the first lag, then oscillates around the zero-line.

Then, I can once again examine the RCC series by decomposing it into its trend and random elements.

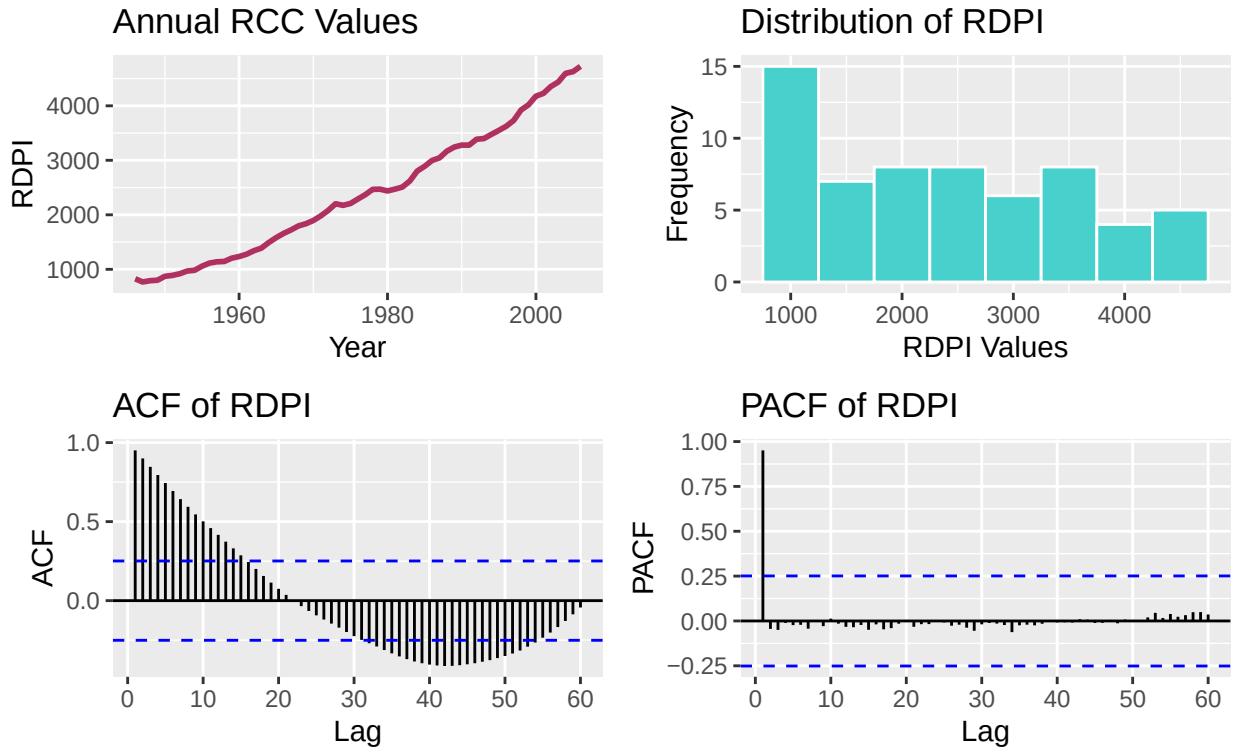
```
credit_ts %>% model(STL(RCC)) %>% components() %>% autoplot()
```



The decomposition shows that there is a strong, mostly linear, upward trend. There is also some inconsistent random variation in the series.

Last, I can examine the `RDPI`.

```
p1 <- credit_ts %>% ggplot(aes(x=Year,y=RDPI)) + geom_line(colour="maroon",size=1) +
  ggtitle('Annual RCC Values') + theme(legend.position = "none")
p2 <- ggplot(credit_ts, aes(RDPI)) +
  geom_histogram(binwidth = 500, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Distribution of RDPI", x = "RDPI Values",y="Frequency")
p3 <- ggAcf(credit_ts$RDPI, lag.max = 104, main = "ACF of RDPI")
p4 <- ggPacf(credit_ts$RDPI, lag.max = 104, main = "PACF of RDPI")
egg::ggarrange(p1, p2, p3, p4, nrow = 2)
```

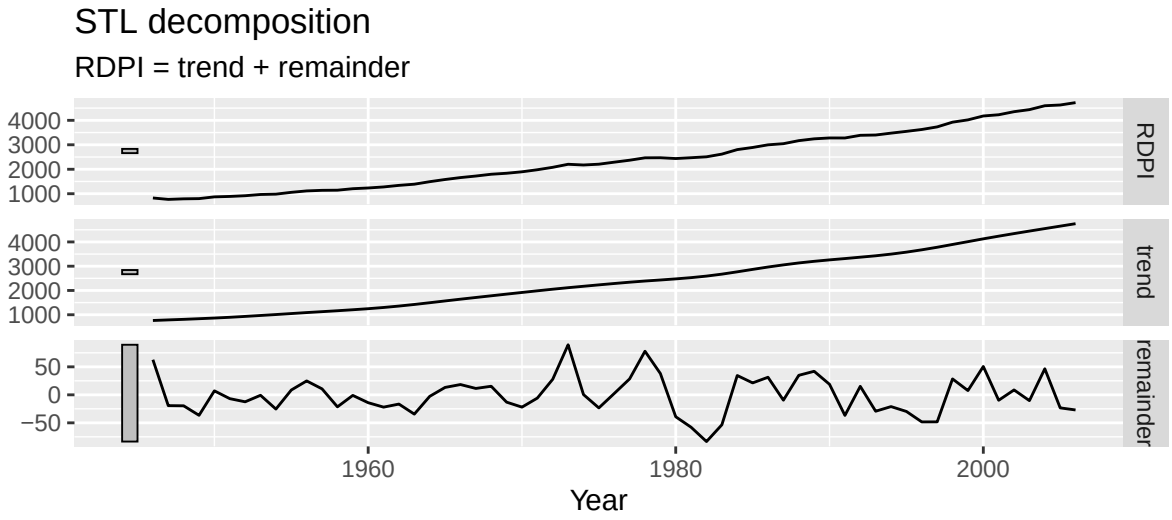


From the plots, I can see that:

- the time series has a strong upward trend, with no apparent seasonality. The series appears to grow almost linearly, with some slight variations throughout the series.
- the histogram shows that the values are somewhat unimodal but definitely not normal. There is a slight positive right skew as there is a higher frequency of lower values with a prominent tail towards higher RDPI scores.
- the ACF declines very gradually and oscillates around the zero line.
- the PACF has a sharp cut off after the first lag, then oscillates around the zero-line.

Then, I can further examine the `RDPI` series by decomposing it into its trend and random elements.

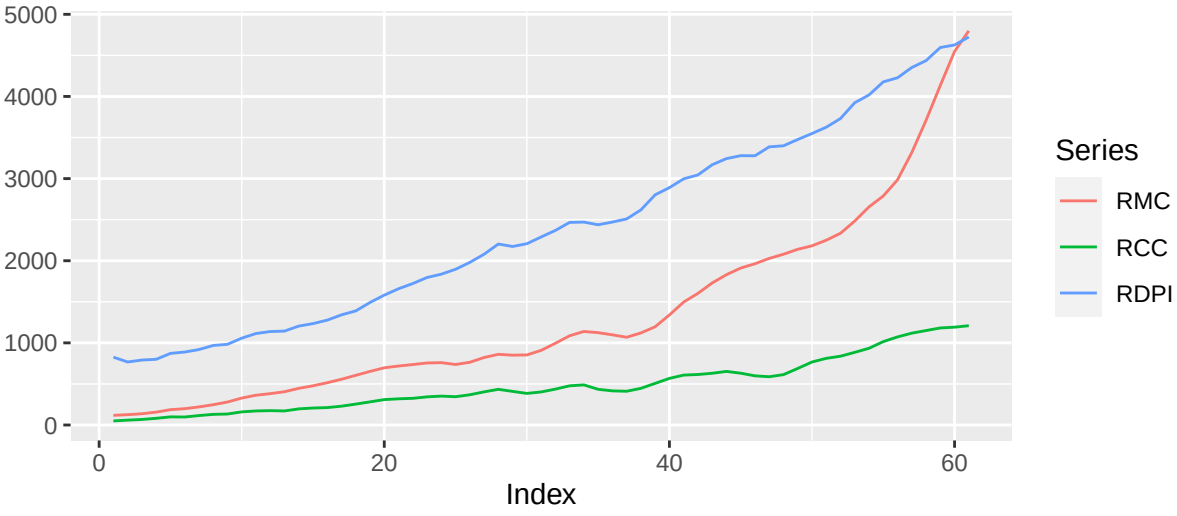
```
credit_ts %>% model(STL(RDPI)) %>% components() %>% autoplot()
```



The decomposition shows that there is a strong, mostly linear, upward trend. There is also some inconsistent random variation in the series.

I can then observe each of the series in relation with each other. First, I can get an overall idea of how each series compares by examining their collective time plot.

```
credit_zoo <- credit_ts %>% zoo
autoplot(credit_zoo[,c(2,3,4)], main = "", facet=NULL)
```

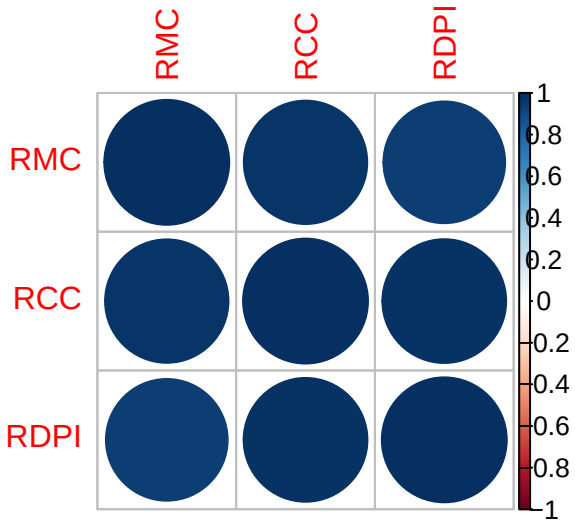


From the plot, I can see that `RDPI` has a consistently high linear growth. The `RMC` grows almost exponentially

as it start low and rises to the level of the RDPI in the more recent years. The RCC has a consistently low linear growth, in relation to the other series.

Next, I want to examine the correlation between the three series.

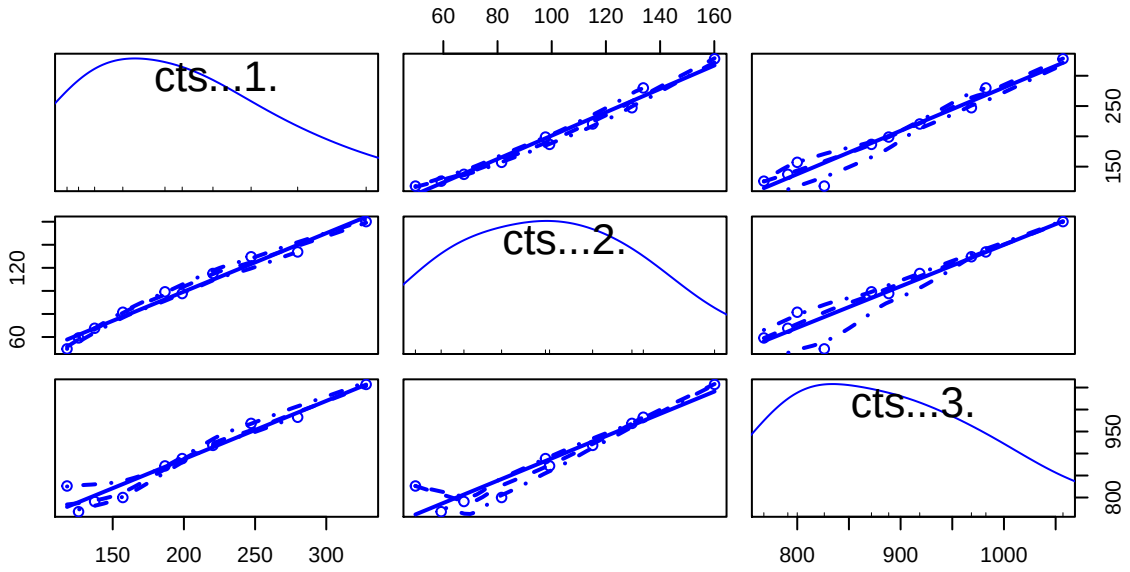
```
corrplot::corrplot(cor(credit_ts[ ,c(2,3,4)]))
```



I can see from the correlation plot that each series is very correlated with one another.

Then, I can also take a look at their scatter plot matrix, which is a representation of the extent of the contemporaneous correlation between the three series.

```
cts <- ts(credit_ts[,c(2,3,4)], start=c(1994), end=c(2003), frequency=1)
scatterplotMatrix(~cts[,1]+cts[,2]+cts[,3])
```

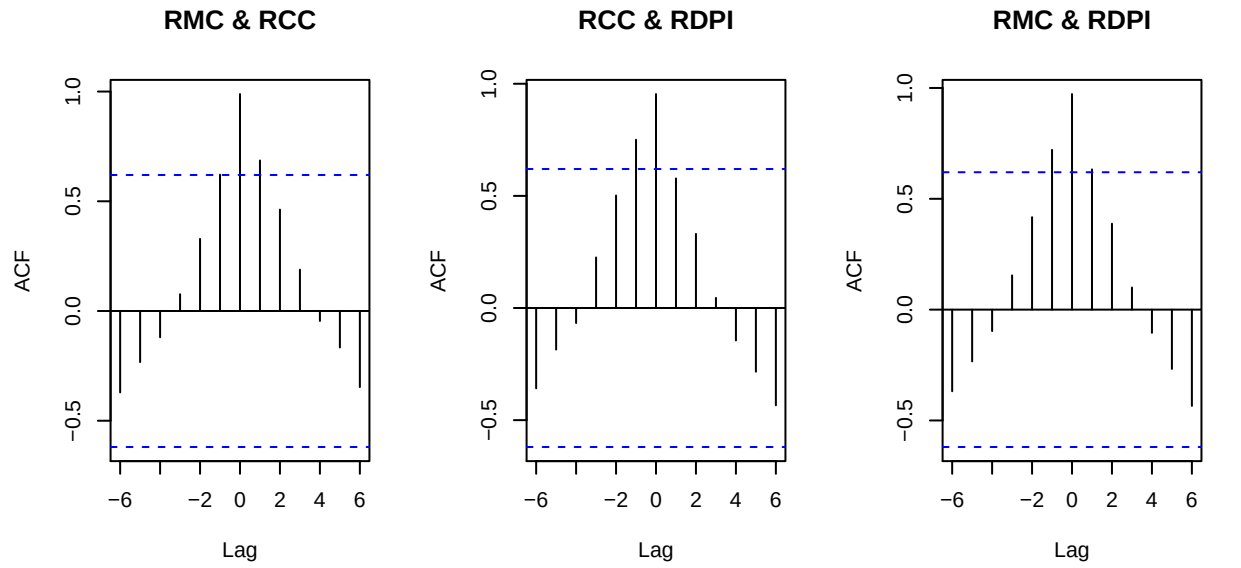


From the plots, I can see that each combination of the series has a positive correlation, with a strong upward trend. More specifically, RMC (represented by cts[,1]) is very positively correlated with RCC (represented by cts[,2]). There are slightly more deviations in correlation for both RMC and RCC with RDPI (represented by cts[,3]), but there is still a strong positive correlation present.

I can also consider each series' cross-correlation, which is achieved using the ccf() function. Cross correlation presents a technique for comparing two time series and finding objectively how they match up with each

other, and in particular where the best match occurs. It can also reveal any periodicities in the data.

```
par(mfrow=c(1,3))
p1 <- ccf(cts[,1],cts[,2], main = "RMC & RCC")
p2 <- ccf(cts[,2],cts[,3], main = "RCC & RDPI")
p3 <- ccf(cts[,1],cts[,3], main = "RMC & RDPI")
```



Each CCF plot reveals that each combination of the series has significant cross-correlation. Although each plot is slightly different, they are very similar in terms of their cross-correlation representation. Overall, the cross-correlation plots show that there is meaningful cross-correlation after several lags.

I can then use the Augmented Dickey-Fuller Test to test if each series has a unit root.

```
adf.test(credit_ts$RMC)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  credit_ts$RMC
## Dickey-Fuller = 1.3586, Lag order = 3, p-value = 0.99
## alternative hypothesis: stationary
```

```
adf.test(credit_ts$RCC)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  credit_ts$RCC
## Dickey-Fuller = -0.016205, Lag order = 3, p-value = 0.99
## alternative hypothesis: stationary
```

```
adf.test(credit_ts$RDPI)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  credit_ts$RDPI
## Dickey-Fuller = -0.93887, Lag order = 3, p-value = 0.9402
## alternative hypothesis: stationary
```

The resulting p-value of each ADF test is approximately 0.9, which is greater than $\alpha = 0.05$. So, I can not reject the null hypothesis that each series does have a unit root. So, it is likely that each series has a unit root present. This is an indication that each series is actually not stationary and a transformation might be necessary for my model.

Then, I can use the Phillips-Ouliaris Co-integration Test to test the null hypothesis that the series are not co-integrated. The correlation is used to check for the linear relationship (or linear interdependence) between two series while co-integration is used to check for the existence of a long-run relationship between two or more series.

```
po.test(credit_ts)
```

```
## Warning in po.test(credit_ts): p-value greater than printed p-value
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  credit_ts
## Phillips-Ouliaris demeaned = -13.886, Truncation lag parameter = 0,
## p-value = 0.15
```

The resulting p-value of the PO test is 0.15, which is greater than $\alpha = 0.05$. So, I can not reject the null hypothesis that the series are not co-integrated.

Now that I have completed my EDA and have a good understanding of the time series data that I am working with, I can build my VAR model for the period 1946-2003.

Although there is prevalent correlation and non-stationarity in the data, I will start by building a basic VAR model with no transformations. Then, I will explore the relative advantages of logarithmic transformations and the use of differences. The purpose of these manipulations would be to help maintain model stationarity and improve the resulting forecasts.

In order to build this basic VAR model, I will first split my data into a training (1946-2003) and test (2004-2006) set.

```
credit_train <- Q5[which(Q5$Year <= 2003),c(2,3,4)]
credit_test <- Q5[which(Q5$Year > 2003),c(2,3,4)]
```

I can use `VARselect` to to compare VAR models of different lags.

```
VARselect(credit_train, lag.max = 10, type = "both")$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     10     10      2      9
```

All the metrics produce different lags. The `AIC` and the `HQ` indicate that the lag $p = 10$ where the `FPE` produces $p = 9$. However, the `SC` indicates that the lag $p = 2$. In order to best generalize my model, I want to pick a lower lag value. As a result, I know that I need to build a VAR model of lag $p = 2$.

```
var.fit <- VAR(credit_train, p = 2, type = "both")
summary(var.fit)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: RMC, RCC, RDPI
## Deterministic variables: both
## Sample size: 56
## Log Likelihood: -768.931
## Roots of the characteristic polynomial:
```

```
## 1.157 0.8639 0.8639 0.8374 0.6331 0.0473
## Call:
## VAR(y = credit_train, 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   1.799678   0.146932  12.248  < 2e-16 ***
## RCC.l1   0.204995   0.258493   0.793    0.432
## RDPI.l1 -0.025617   0.158550  -0.162    0.872
## RMC.l2  -0.786066   0.146643  -5.360 2.34e-06 ***
## RCC.l2   0.001376   0.253267   0.005    0.996
## RDPI.l2  0.001162   0.142219   0.008    0.994
## const    8.805267  32.526993   0.271    0.788
## trend   -1.312656   3.413383  -0.385    0.702
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 34.44 on 48 degrees of freedom
## Multiple R-Squared: 0.9987,  Adjusted R-squared: 0.9985
## F-statistic:  5114 on 7 and 48 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   0.002156   0.087117   0.025    0.980
## RCC.l1   1.472530   0.153262   9.608 9.23e-13 ***
## RDPI.l1  0.054159   0.094005   0.576    0.567
## RMC.l2   0.058906   0.086945   0.678    0.501
## RCC.l2  -0.659156   0.150163  -4.390 6.21e-05 ***
## RDPI.l2 -0.075895   0.084322  -0.900    0.373
## const   16.277333  19.285373   0.844    0.403
## trend    1.657949   2.023807   0.819    0.417
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 20.42 on 48 degrees of freedom
## Multiple R-Squared: 0.9954,  Adjusted R-squared: 0.9947
## F-statistic:  1469 on 7 and 48 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   0.090310   0.182094   0.496 0.622189
## RCC.l1   0.423064   0.320351   1.321 0.192889
## RDPI.l1  0.824607   0.196491   4.197 0.000116 ***
## RMC.l2  -0.070556   0.181735  -0.388 0.699559
## RCC.l2  -0.314406   0.313874  -1.002 0.321513
## RDPI.l2  0.005734   0.176253   0.033 0.974184
## const   93.061946  40.310784   2.309 0.025316 *
## trend    9.071795   4.230214   2.145 0.037082 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 42.68 on 48 degrees of freedom
## Multiple R-Squared: 0.9986,  Adjusted R-squared: 0.9984
## F-statistic:  4983 on 7 and 48 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##         RMC   RCC   RDPI
## RMC  1186.1 406.4  735.6
## RCC   406.4 416.9  642.2
## RDPI  735.6 642.2 1821.6
##
## Correlation matrix of residuals:
##         RMC    RCC   RDPI
## RMC  1.0000 0.5780 0.5004
## RCC  0.5780 1.0000 0.7369
## RDPI 0.5004 0.7369 1.0000
```

Most of the auto-regressive coefficients are not significant in this model. There are a few significant terms, each at different lags for each series. The constant term and the trend term are generally not distinguishable from zero.

To check if the VAR(2) with a constant and a trend is a stable process, I will need to check if the moduli of the eigenvalues of the companion matrix are all less than one.

```
roots(var.fit)
```

```
## [1] 1.15729313 0.86389207 0.86389207 0.83741311 0.63307489 0.04730256
```

There is an eigenvalue greater than 1. This is further indication that this is not necessarily a stationary model and a data transformation (such a a log or difference) will be necessary.

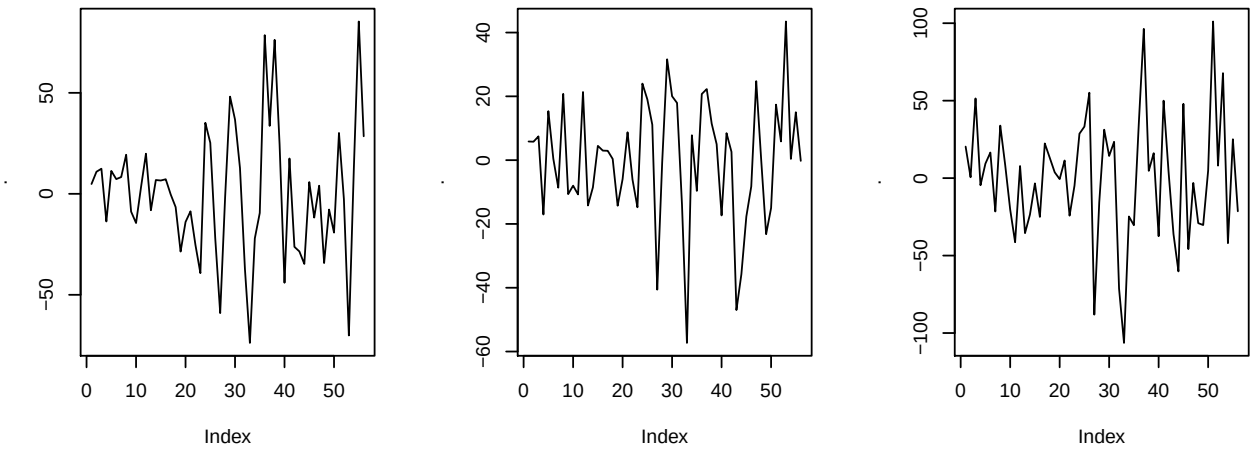
I can then assess the fit of the model by taking a closer look at the model residuals. As mentioned in `Part 1`, in order to produce good forecasts, the residuals must:

- be uncorrelated. If there are correlations, than there is still information in them. So, they should not be used for forecasting.
- have zero mean. If they do not, then the forecasts will be biased.
- have constant variance.
- be normally distributed.

To start my residual diagnostics, I can first observe the residual time series.

```
par(mfrow=c(1,3))
var.fit %>% resid %>% .[, "RMC"] %>% plot(type = "l")
```

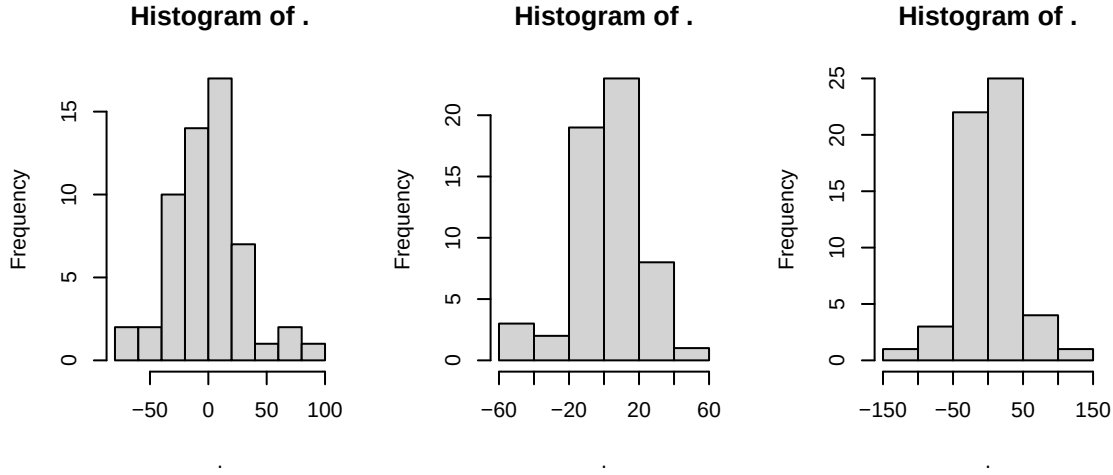
```
var.fit %>% resid %>% .[, "RCC"] %>% plot(type = "l")
var.fit %>% resid %>% .[, "RDPI"] %>% plot(type = "l")
```



From the plots I can see that each residual time series does not necessarily have constant variance. There are fluctuations of varying heights that are not consistent throughout the entire series. Yet, these fluctuations all appear to be centered around the zero-line. Therefore, the assumption of constant variance is violated and the assumption of zero conditional mean is maintained.

I can also examine the normality of the residuals by looking at their histograms and Q-Q plots.

```
par(mfrow=c(1,3))
var.fit %>% resid %>% .[, "RMC"] %>% hist
var.fit %>% resid %>% .[, "RCC"] %>% hist
var.fit %>% resid %>% .[, "RDPI"] %>% hist
```



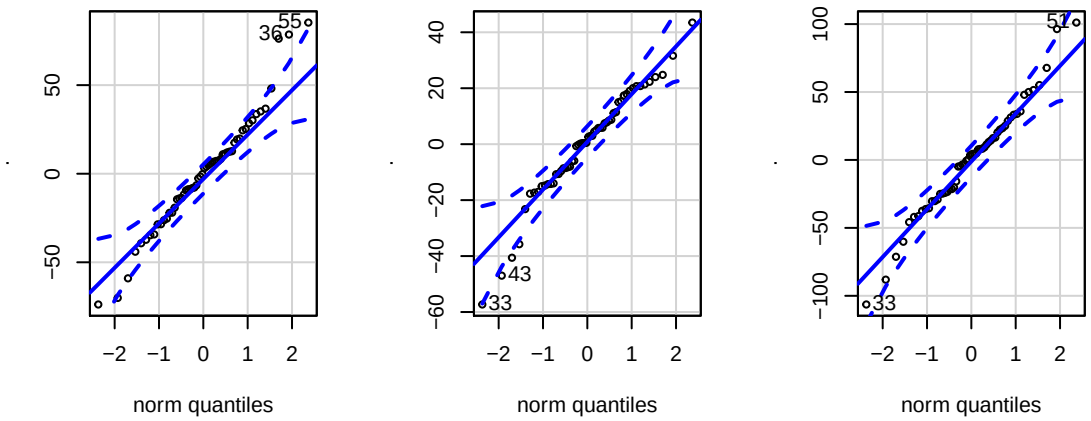

```
par(mfrow=c(1,3))
var.fit %>% resid %>% .[, "RMC"] %>% qqPlot
```

```
## [1] 55 36
```

```
var.fit %>% resid %>% .[, "RCC"] %>% qqPlot
```

```
## [1] 33 43
```

```
var.fit %>% resid %>% .[, "RDPI"] %>% qqPlot
```




```
## [1] 33 51
```

From the plots, I can see that, although there are some outlying points, the residuals are normally distributed for each series. Thus, the assumption of residual normality appears to be maintained.

I can also statistically test for the normality of the model using the `normality.test()` function. This function computes a multivariate Jarque-Bera test and multivariate skewness and kurtosis tests for the residuals of a VAR model. The null hypothesis is that the residuals are normally distributed as well as the skewness being zero and the excess kurtosis being zero.

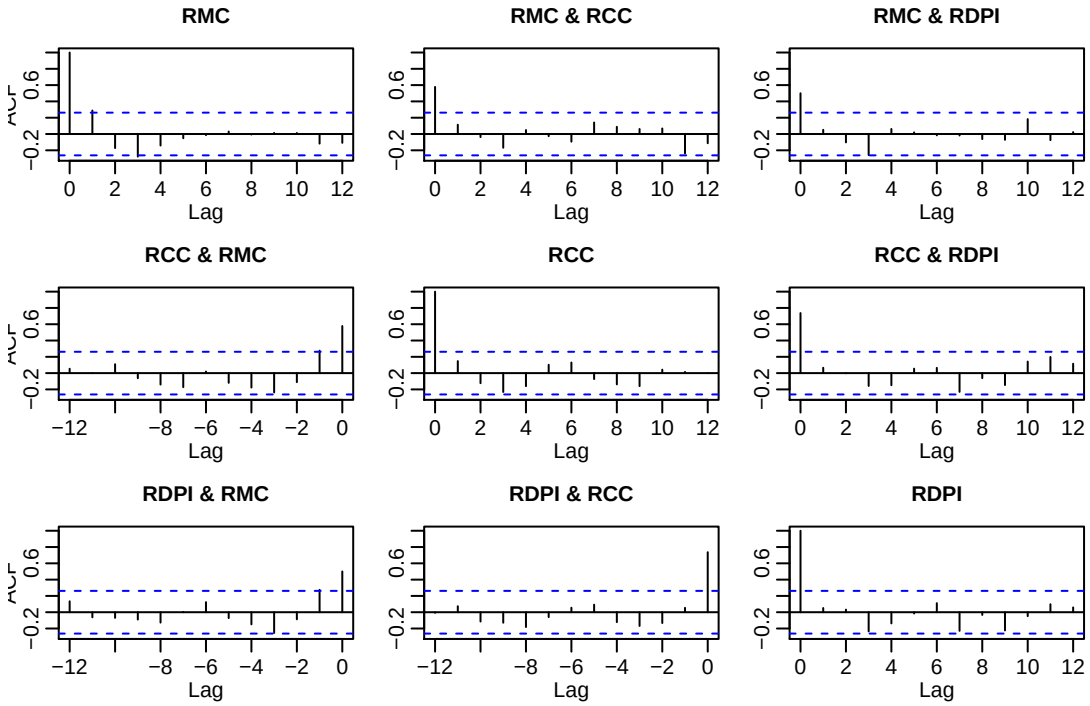
```
var.fit.norm <- normality.test(var.fit, multivariate.only = TRUE)
var.fit.norm
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.fit
## Chi-squared = 78.114, df = 6, p-value = 8.771e-15
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.fit
## Chi-squared = 16.908, df = 3, p-value = 0.0007382
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.fit
## Chi-squared = 61.206, df = 3, p-value = 3.247e-13
```

From each of the results, the p-values are all less than $\alpha = 0.05$. As a result, even though the histograms and Q-Q plots indicate normality, the residuals for the VAR model are not necessarily normally distributed.

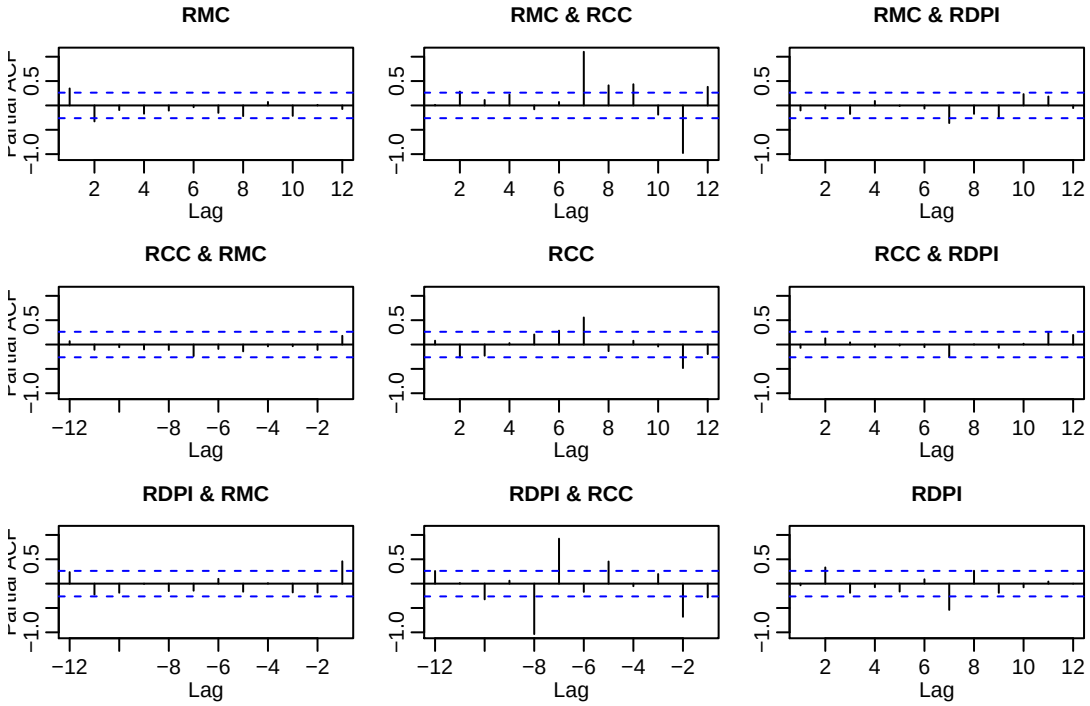
In order to assess the assumption of non-correlation, I can also consider the residual ACF and PACF.

```
var.fit %>% resid %>% acf
```



From the residual ACF plots, I can see that each series gradually declines and oscillates around the zero line. There does not appear to be any major spikes, indicating that there is no serious correlation in the residuals.

```
var.fit %>% resid %>% pacf
```



From the residual PACF plots, I can see that each series gradually declines and oscillates around the zero line. The `RCC` plots appear to have some spikes in subsequent lags. Otherwise, there does not appear to be many major spikes, indicating that there is generally no serious correlation in the residuals. So, overall, the assumption of non-correlation appears to be maintained.

I can also statistically test for no serial correlation in the VAR model. This `serial.test()` function computes the multivariate Portmanteau test for serially correlated errors. The Portmanteau statistic tests the absence of serially correlated disturbances in a stable VAR. In the context of a serial test, H_0 is that there is no serial correlation of any order up to `lags.pt`, with rejection (H_A) indicating that the series does have serial correlation up to lag p.

```
var.fit.ptasy <- serial.test(var.fit, lags.pt = 10, type = "PT.asymptotic")
var.fit.ptasy
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.fit
## Chi-squared = 92.824, df = 72, p-value = 0.04988
```

The resulting p-value of the serial test is 0.049, which is slightly less than $\alpha = 0.05$. So, I can reject the null hypothesis that the series has no serial correlation of up to lag 10. This is an indication that my model actually has serial correlation in it.

Then, I can test for the absence of ARCH effect. A time series exhibiting conditional heteroscedasticity—or autocorrelation in the squared series—is said to have autoregressive conditional heteroscedastic (ARCH) effects. The function `arch.test()` assesses the null hypothesis that a series of residuals exhibits no conditional heteroscedasticity (ARCH effects), against the alternative that an ARCH(L) model describes the series.

```
var.fit.arch <- arch.test(var.fit)
var.fit.arch
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.fit
## Chi-squared = 229.03, df = 180, p-value = 0.00786
```

The resulting p-value of the ARCH test is 0.007, which is less than $\alpha = 0.05$. So, I can reject the null hypothesis that the series exhibits no conditional heteroscedasticity. This is an indication that my model has heteroscedasticity in it, which violates model assumptions.

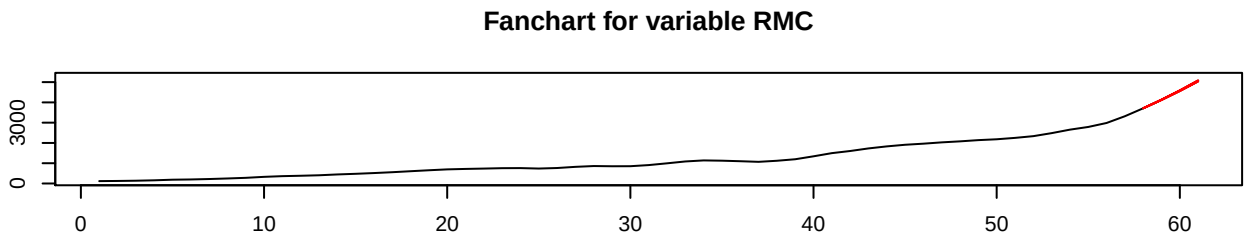
Then, I can forecast the last three years, which are 2004-2006, and plot the results.

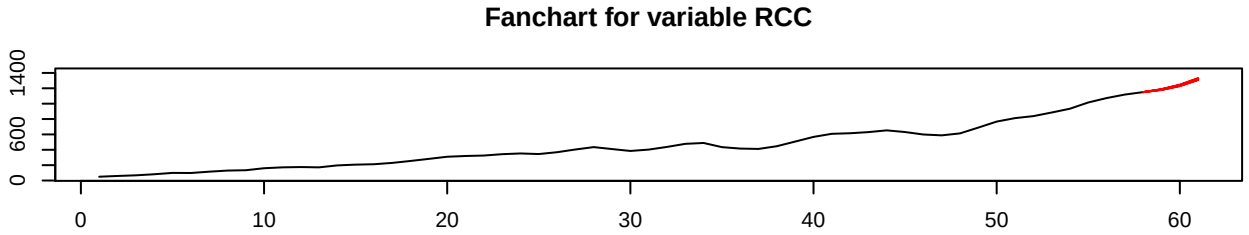
```
predictions <- var.fit %>% predict(n.ahead = 3, ci = 0.95)
predictions
```

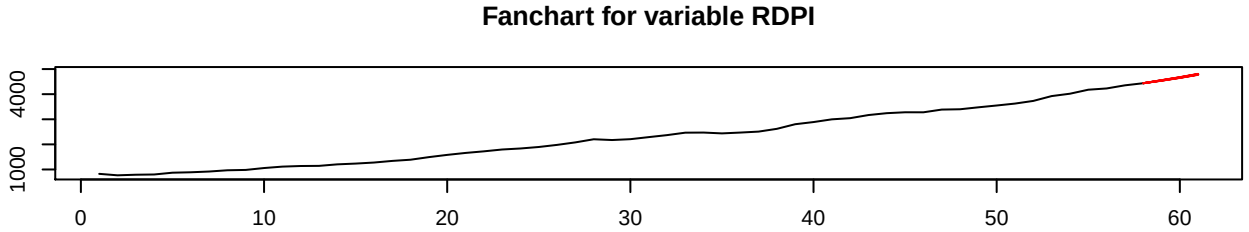
```
## $RMC
##          fcst    lower    upper        CI
## [1,] 4127.466 4059.966 4194.965  67.49954
## [2,] 4575.543 4433.239 4717.846 142.30380
## [3,] 5059.963 4830.919 5289.007 229.04401
##
## $RCC
##          fcst    lower    upper        CI
## [1,] 1183.718 1143.697 1223.739  40.02073
## [2,] 1237.693 1163.536 1311.850  74.15690
## [3,] 1320.121 1219.994 1420.247 100.12636
##
## $RDPI
##          fcst    lower    upper        CI
## [1,] 4547.101 4463.449 4630.754  83.65235
## [2,] 4662.627 4542.871 4782.383 119.75559
```

```
## [3,] 4790.805 4642.249 4939.360 148.55579
```

```
predictions %>% fanchart(colors = c("black","red"))
```







From the forecast charts, I can see that each of the series are projected to continue the general trend of the data. They all continue to increase at a similar rate as the prior data, with somewhat tight confidence intervals.

I can also conduct residual diagnostics on my forecasts by comparing the actual values with the predicted values.

```
years <- c(2004,2005,2006)
actual_rmc <- credit_test$RMC
actual_rcc <- credit_test$RCC
actual_rdpi <- credit_test$RDPI
p_rmc <- predictions$fcst$RMC[,1]
p_rcc <- predictions$fcst$RCC[,1]
p_rdpi <- predictions$fcst$RDPI[,1]
```

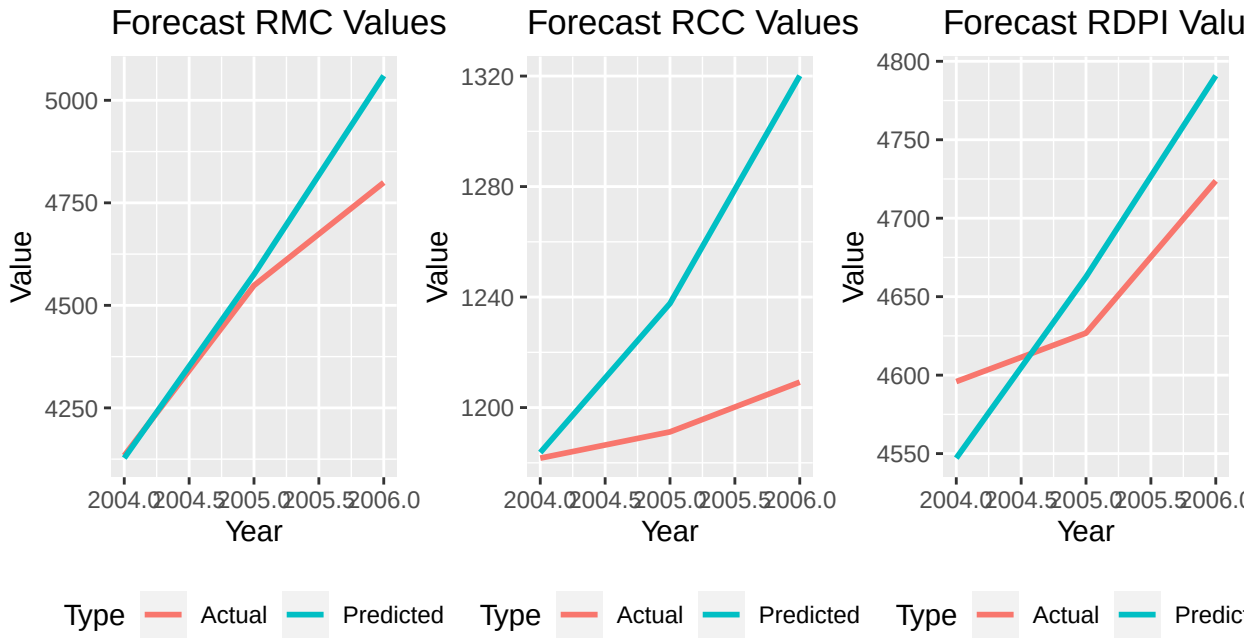
```
d1 <- data.frame(Year = years, Value = actual_rmc, Type = rep("Actual",3))
d2 <- data.frame(Year = years, Value = p_rmc, Type = rep("Predicted",3))
df_rmc <- rbind(d1,d2)
p1 <- df_rmc %>% ggplot(aes(x=Year,y=Value)) + theme(legend.position="bottom") +
  geom_line(size=1, aes(colour=Type)) + ggtitle('Forecast RMC Values')

d1 <- data.frame(Year = years, Value = actual_rcc, Type = rep("Actual",3))
```

```
d2 <- data.frame(Year = years, Value = p_rcc, Type = rep("Predicted",3))
df_rcc <- rbind(d1,d2)
p2 <- df_rcc %>% ggplot(aes(x=Year,y=Value)) + theme(legend.position="bottom") +
  geom_line(size=1, aes(colour=Type)) + ggtitle('Forecast RCC Values')

d1 <- data.frame(Year = years, Value = actual_rdpi, Type = rep("Actual",3))
d2 <- data.frame(Year = years, Value = p_rdpi, Type = rep("Predicted",3))
df_rdpi <- rbind(d1,d2)
p3 <- df_rdpi %>% ggplot(aes(x=Year,y=Value)) + theme(legend.position="bottom") +
  geom_line(size=1, aes(colour=Type)) + ggtitle('Forecast RDPI Values')

egg::ggarrange(p1, p2, p3, nrow = 1)
```



From the plots, I can see that both the actual and predicted values increase and follow the general trend of the series. The predicted `RMC` and the `RDPI` are fairly close to the actual values, whereas the `RCC` is slightly less accurate.

Then, I can calculate the RMSE score of the predicted values. As mentioned previously, the RMSE represents the standard deviation of the residuals, as it refers to the square root of the average of the squared error terms.

```
print(paste0("RMC RMSE: ",round(Metrics::rmse(actual_rmc, p_rmc),4)))
```

```
## [1] "RMC RMSE: 151.2282"
```

```
print(paste0("RCC RMSE: ",round(Metrics::rmse(actual_rcc, p_rcc),4)))
```

```
## [1] "RCC RMSE: 69.4481"
```

```
print(paste0("RDPI RMSE: ",round(Metrics::rmse(actual_rdpi, p_rdpi),4)))
```

```
## [1] "RDPI RMSE: 52.1361"
```

In general, lower values of RMSE indicate better fit. So, I can see that the `RDPI` series has the lowest score and performs the best out of each of the series.

Yet, it is possible to improve this model. As previously discovered in my EDA and residual diagnostics, I

know that each series has a unit root and is not necessarily stationary. This is a good reason to explore the different advantages of logarithmic transformations and the applications of differences. These manipulations could help to improve the stationarity, which would improve the VAR model and its forecasts.

Logarithmic transformations would be helpful to ensure stationarity in this context because there is a strong trend that is present in each series. Log transformation is useful when the size of the trend, seasonal, or random fluctuations seem to increase with the level of the time series. This type of transformation is especially useful where there is evidence of exponential growth, which was apparent in the `RMC` series. The extra variability can make trend and seasonal changes harder to forecast accurately. As a result, a transformation would required and log transforms are effective at removing variance. However, it is possible for null values to be produced, so they would need to be replaced with interpolated values. Nevertheless, a log transformation could be applied to the time series before building my model.

Furthermore, logarithm transformations are useful because they are more easily interpretable in analysis. This is because the interpretation reflects that changes in a log value are relative to percentage changes on the original scale.

The use of differences would be beneficial in ensuring stationarity because first-differencing a time series will help to remove a linear trend. The first difference of a time series is the series of changes from one period to the next. Differencing can be used to remove the series dependence on time, so-called temporal dependence. This includes structures like trends and seasonality. This type of transformation is especially useful where there is evidence of linear growth, which was apparent in the `RCC` and the `RDPI` series. The non-stationarity can make trend and seasonal changes harder to forecast accurately. Thus, a difference transformation could be applied to the time series in order to improve the VAR model forecasts.

Additionally, the first difference transformation would change the model interpretation to the change between subsequent time periods on the original scale.

As there was apparent non-stationarity in the series, I can now build a VAR model for the first difference of the log transformed data in order to improve my model stationarity. The purpose of performing both transformations would be to remove the apparent trend and seasonality that is present in each time series.

In order to build this first differenced and log transformed VAR model, I will first transform and split my data into a training (1946-2003) and test (2004-2006) set.

```
credit_train_log <- data.frame(Year = credit_ts$Year,RMC = log(credit_ts$RMC),
                               RCC = log(credit_ts$RCC),RDPI = log(credit_ts$RDPI))
credit_train_log$RMC <- ifelse(is.nan(credit_train_log$RMC),NA,credit_train_log$RMC)
credit_train_log$RMC <- na.approx(credit_train_log$RMC)
credit_train_log$RCC <- ifelse(is.nan(credit_train_log$RCC),NA,credit_train_log$RCC)
credit_train_log$RCC <- na.approx(credit_train_log$RCC)
credit_train_log$RDPI<-ifelse(is.nan(credit_train_log$RDPI),NA,credit_train_log$RDPI)
credit_train_log$RDPI[1] <- min(credit_train_log$RDPI, na.rm = T)
credit_train_log$RDPI <- na.approx(credit_train_log$RDPI)
```

```
Q5_diff <- data.frame(Year=credit_train_log$Year[-1],RMC=diff(credit_train_log$RMC),
                      RCC=diff(credit_train_log$RCC),RDPI=diff(credit_train_log$RDPI))
credit_train_t <- Q5_diff[which(Q5_diff$Year <= 2003),c(2,3,4)]
credit_test_t <- Q5_diff[which(Q5_diff$Year > 2003),c(2,3,4)]
```

I can then use `VARselect` to to compare VAR models of different lags.

```
VARselect(credit_train_t, lag.max = 10, type = "both")$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     10      2      1      2
```

All the metrics produce different lags. The `AIC` indicates that the lag $p = 10$ where the `HQ` and `FPE` produces $p = 2$. However, the `SC` indicates that the lag $p = 1$. In order to best generalize my model, I want to pick a

lower, more general, lag value. As a result, I know that I need to build a VAR model of lag $p = 2$.

As a result, I can build a transformed VAR model with lag 2.

```
var.fit_t <- VAR(credit_train_t, p = 2, type = "both")
summary(var.fit_t)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: RMC, RCC, RDPI
## Deterministic variables: both
## Sample size: 55
## Log Likelihood: 383.994
## Roots of the characteristic polynomial:
## 0.6249 0.6249 0.6149 0.4861 0.3545 0.3545
## Call:
## VAR(y = credit_train_t, 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   1.1679080  0.2059950   5.670 8.48e-07 ***
## RCC.l1  -0.0837442  0.1217906  -0.688  0.49508
## RDPI.l1 -0.5953045  0.2661332  -2.237  0.03008 *
## RMC.l2  -0.6065872  0.2013709  -3.012  0.00417 **
## RCC.l2   0.1291733  0.1022547   1.263  0.21273
## RDPI.l2  0.1181111  0.2499500   0.473  0.63873
## const    0.0527865  0.0158224   3.336  0.00167 **
## trend   -0.0004762  0.0002867  -1.661  0.10341
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.02962 on 47 degrees of freedom
## Multiple R-Squared: 0.6089,  Adjusted R-squared: 0.5506
## F-statistic: 10.45 on 7 and 47 DF,  p-value: 7.827e-08
##
##
## 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   0.7124724  0.3977339   1.791  0.07968 .
## RCC.l1   0.2687209  0.2351526   1.143  0.25893
## RDPI.l1 -0.8458290  0.5138484  -1.646  0.10642
## RMC.l2  -0.9113694  0.3888057  -2.344  0.02336 *
## RCC.l2   0.0591594  0.1974327   0.300  0.76577
## RDPI.l2  0.3288772  0.4826020   0.681  0.49892
## const    0.0985867  0.0305497   3.227  0.00228 **
## trend   -0.0012353  0.0005536  -2.231  0.03047 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.05719 on 47 degrees of freedom
## Multiple R-Squared: 0.3189,  Adjusted R-squared: 0.2175
## F-statistic: 3.144 on 7 and 47 DF,  p-value: 0.008295
##
##
## 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   0.3686959  0.1209289   3.049 0.003765 **
## RCC.l1  -0.0180309  0.0714969  -0.252 0.801993
## RDPI.l1 -0.1907459  0.1562329  -1.221 0.228211
## RMC.l2  -0.4249260  0.1182143  -3.595 0.000776 ***
## RCC.l2  -0.0232756  0.0600284  -0.388 0.699957
## RDPI.l2  0.3896655  0.1467326   2.656 0.010775 *
## const    0.0459858  0.0092885   4.951 9.93e-06 ***
## trend   -0.0005133  0.0001683  -3.049 0.003763 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.01739 on 47 degrees of freedom
## Multiple R-Squared: 0.3905,  Adjusted R-squared: 0.2997
## F-statistic: 4.302 on 7 and 47 DF,  p-value: 0.0009534
##
##
##
## Covariance matrix of residuals:
##             RMC       RCC      RDPI
## RMC  0.0008774 0.0012233 0.0001992
## RCC  0.0012233 0.0032709 0.0005715
## RDPI 0.0001992 0.0005715 0.0003024
##
## Correlation matrix of residuals:
##         RMC    RCC   RDPI
## RMC  1.0000 0.7221 0.3868
## RCC  0.7221 1.0000 0.5747
## RDPI 0.3868 0.5747 1.0000
```

More of the auto-regressive coefficients are significant in this model. There are a some significant terms, each at different lags for each series. The constant term and the trend term are generally distinguishable from zero.

To check if the VAR(2) with a constant and a trend is a stable process, I will need to check if the moduli of the eigenvalues of the companion matrix are all less than one.

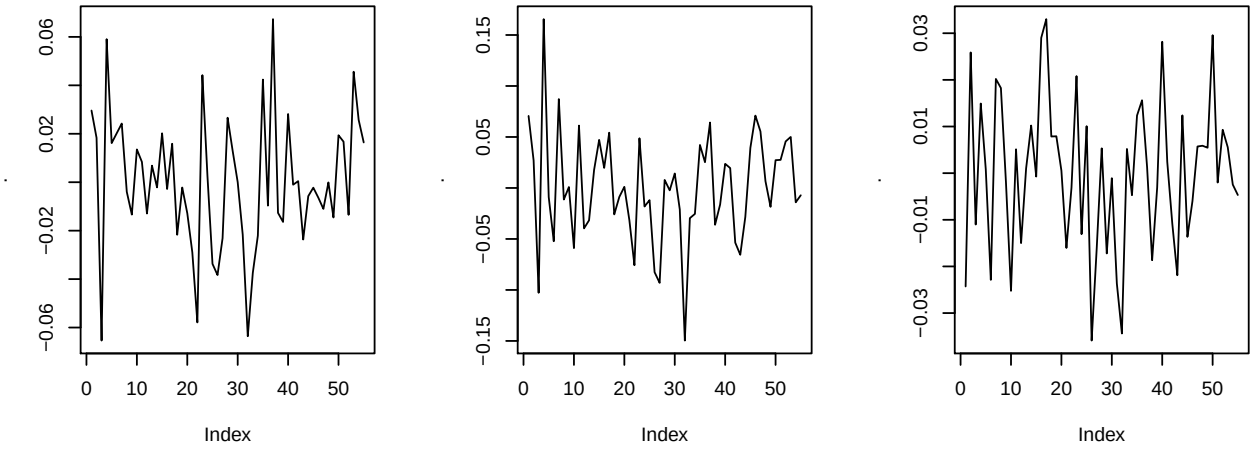
```
roots(var.fit_t)
```

```
## [1] 0.6249280 0.6249280 0.6148877 0.4860748 0.3545087 0.3545087
```

There are no eigenvalues greater than 1. As a result, the first differenced and log transformed data is stationary and maintains important model assumptions.

I can then assess the fit of the model by taking a closer look at the model residuals.

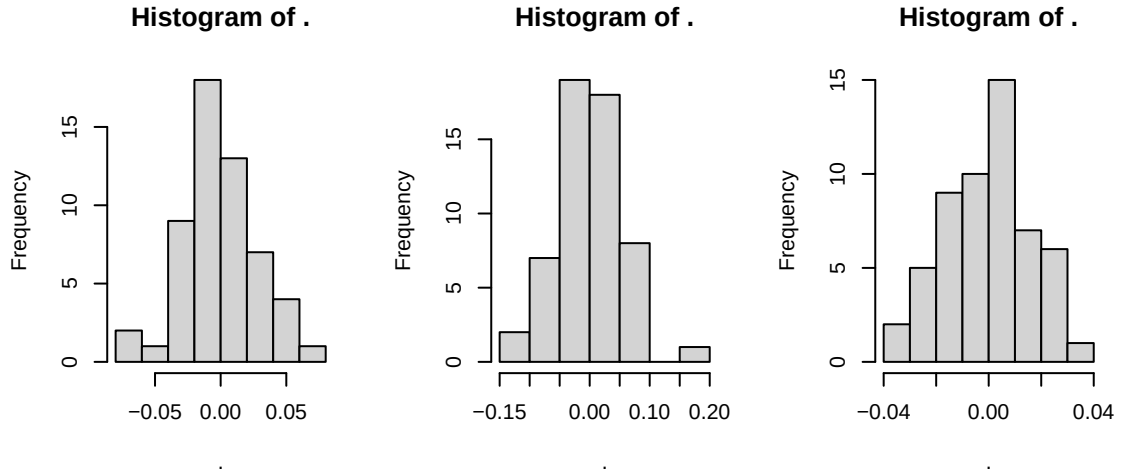
```
par(mfrow=c(1,3))
var.fit_t %>% resid %>% .[, "RMC"] %>% plot(type = "l")
var.fit_t %>% resid %>% .[, "RCC"] %>% plot(type = "l")
var.fit_t %>% resid %>% .[, "RDPI"] %>% plot(type = "l")
```



The residuals almost appear more like white noise. From the plots, I can see that each residual time series has mostly constant variance that is centered around the zero-line. Therefore, the assumptions of constant variance and zero conditional mean are maintained.

I can also examine the normality of the residuals by looking at their histograms and Q-Q plots.

```
par(mfrow=c(1,3))
var.fit_t %>% resid %>% .[, "RMC"] %>% hist
var.fit_t %>% resid %>% .[, "RCC"] %>% hist
var.fit_t %>% resid %>% .[, "RDPI"] %>% hist
```



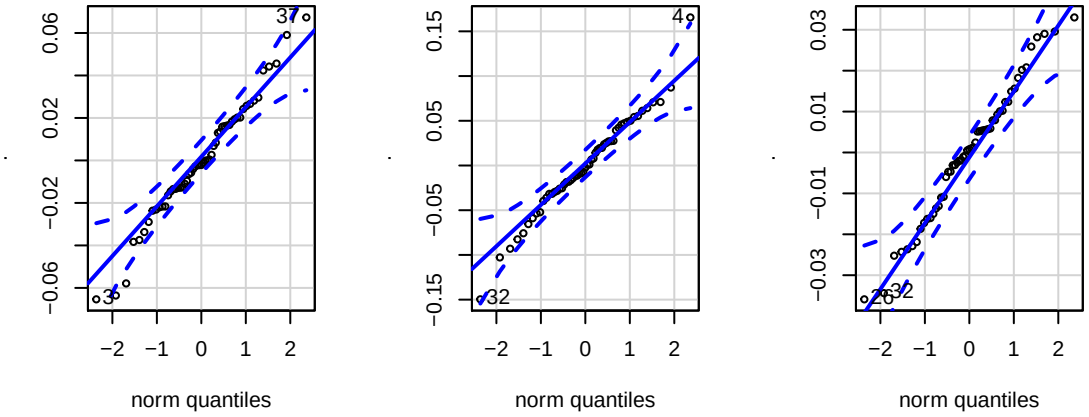

```
par(mfrow=c(1,3))
var.fit_t %>% resid %>% .[, "RMC"] %>% qqPlot
```

```
## [1] 37  3
```

```
var.fit_t %>% resid %>% .[, "RCC"] %>% qqPlot
```

```
## [1]  4 32
```

```
var.fit_t %>% resid %>% .[, "RDPI"] %>% qqPlot
```




```
## [1] 26 32
```

From the plots, I can see that, although there are some outlying points, the residuals are normally distributed for each series. Thus, the assumption of residual normality appears to be maintained.

I can also statistically test for the normality of the model. Once again, the null hypothesis is that the residuals are normally distributed as well as the skewness being zero and the excess kurtosis being zero.

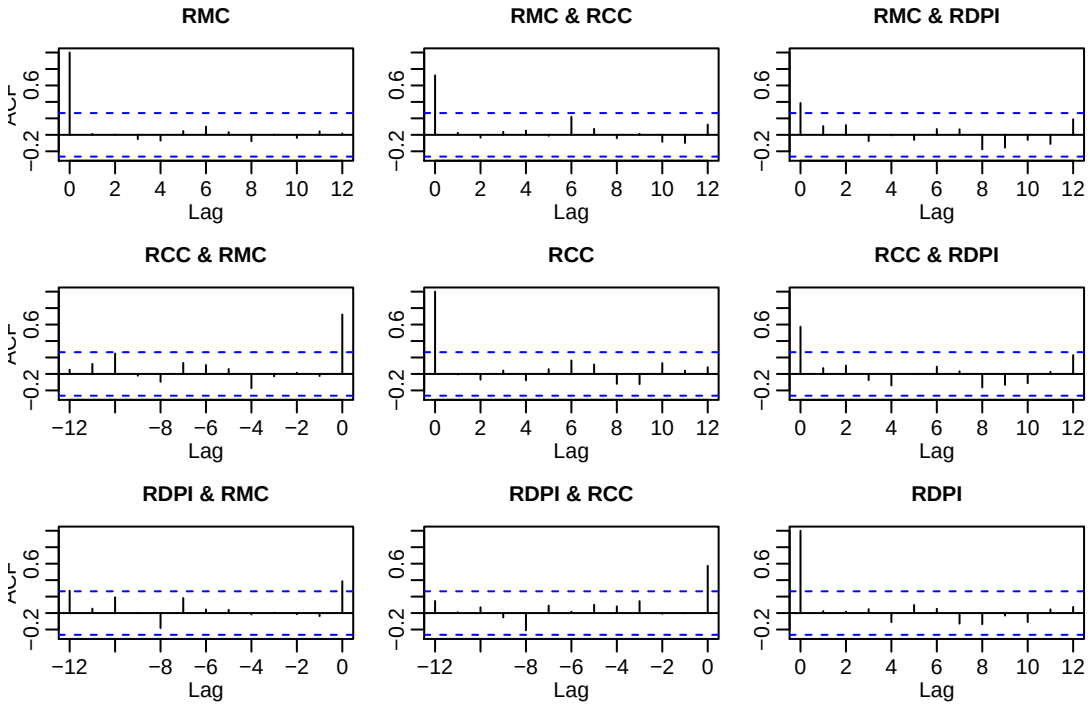
```
var.fit.norm_t <- normality.test(var.fit_t, multivariate.only = TRUE)
var.fit.norm_t
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object var.fit_t
## Chi-squared = 0.37535, df = 6, p-value = 0.999
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object var.fit_t
## Chi-squared = 0.17416, df = 3, p-value = 0.9816
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object var.fit_t
## Chi-squared = 0.20119, df = 3, p-value = 0.9774
```

From each of the results, the p-values are all greater than $\alpha = 0.05$. As a result, the residuals for the transformed VAR model are normally distributed.

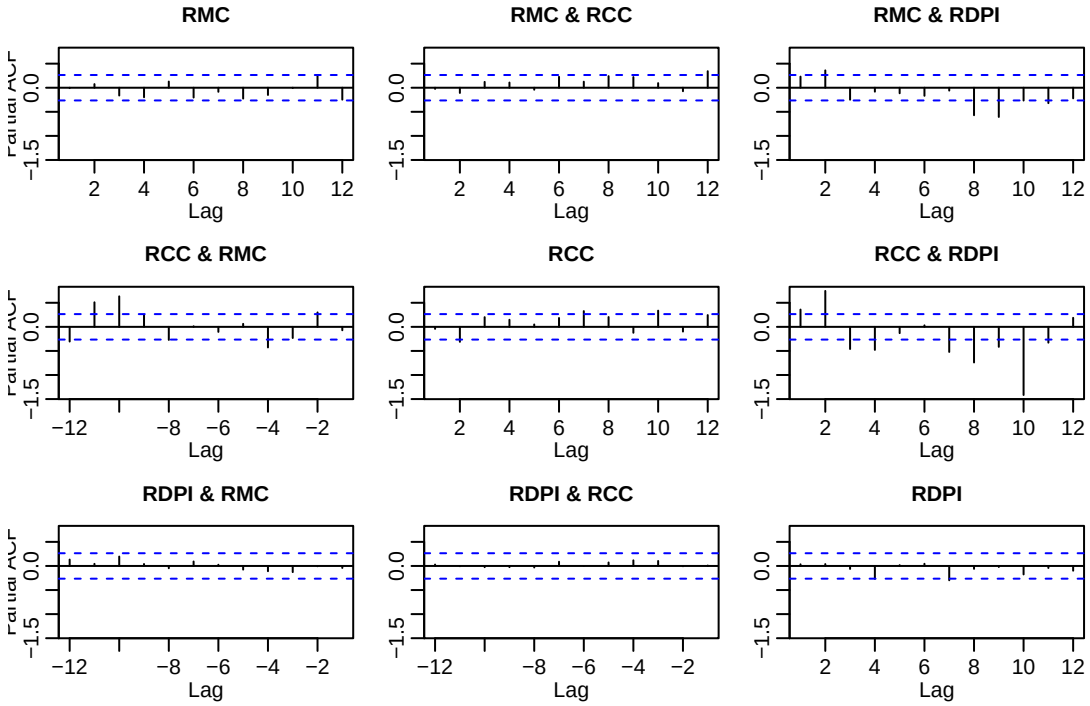
In order to assess the assumption of non-correlation, I can also consider the residual ACF and PACF.

```
var.fit_t %>% resid %>% acf
```



From the residual ACF plots, I can see that each series gradually declines and oscillates around the zero line. There does not appear to be any major spikes, indicating that there is no serious correlation in the residuals.

```
var.fit_t %>% resid %>% pacf
```



From the residual PACF plots, I can see that each series gradually declines and oscillates around the zero line. The `RCC & RDPI` plot appears to have some spikes in subsequent lags. Otherwise, there does not appear to be many major spikes, indicating that there is generally no serious correlation in the residuals. So, overall, the assumption of non-correlation visually appears to be maintained.

Then, I can statistically test for serial correlation in the model. In the context of an serial test, H_0 is that there is no serial correlation of any order up to `lags.pt`, with rejection (H_A) indicating that the series does have serial correlation up to lag p.

```
var.fit.ptasy_t <- serial.test(var.fit_t, lags.pt = 10, type = "PT.asymptotic")
var.fit.ptasy_t
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var.fit_t
## Chi-squared = 55.049, df = 72, p-value = 0.9312
```

The resulting p-value of the serial test is 0.93, which is greater than $\alpha = 0.05$. So, I can not reject the null hypothesis that the series has no serial correlation of up to lag 10. This is an indication that my model now has no serial correlation in it.

Then, I can test for the absence of ARCH effect. The null hypothesis is that a series of residuals exhibits no conditional heteroscedasticity (ARCH effects), against the alternative that an ARCH(L) model describes the series.

```
var.fit.arch_t <- arch.test(var.fit_t)
var.fit.arch_t
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var.fit_t
## Chi-squared = 179.86, df = 180, p-value = 0.4889
```

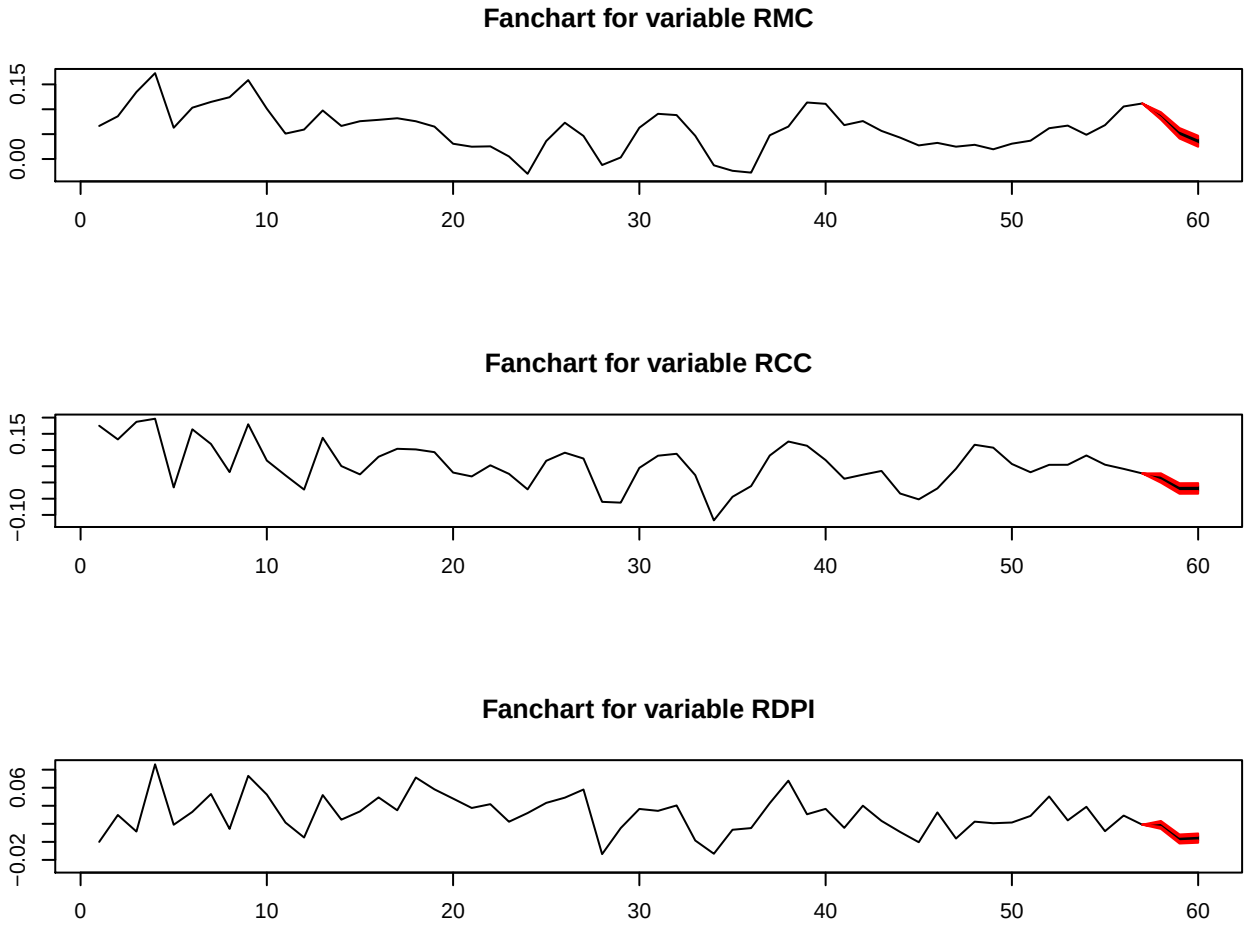
The resulting p-value of the ARCH test is 0.49, which is greater than $\alpha = 0.05$. So, I can not reject the null hypothesis that the series exhibits no conditional heteroscedasticity. This is an indication that my model has no heteroscedasticity in it, which satisfies model assumptions.

Then, I can forecast and look at the transformed predictions for 2004-2006. The interpretation will now be in relation to the percent change from each subsequent time period.

```
predictions_t <- var.fit_t %>% predict(n.ahead = 3, ci = 0.95)
predictions_t
```

```
## $RMC
##            fcst       lower     upper         CI
## [1,] 0.08668938  0.02863379 0.1447450 0.05805559
## [2,] 0.05175341 -0.03011882 0.1336256 0.08187223
## [3,] 0.03572399 -0.05044340 0.1218914 0.08616738
##
## $RCC
##             fcst       lower     upper        CI
## [1,]  0.01372593 -0.09836746 0.1258193 0.1120934
## [2,] -0.01853820 -0.14460628 0.1075299 0.1260681
## [3,] -0.01834785 -0.14474902 0.1080533 0.1264012
##
## $RDPI
##             fcst       lower      upper         CI
## [1,] 0.018773465 -0.01530795 0.05285488 0.03408141
## [2,] 0.003176063 -0.03569600 0.04204813 0.03887207
## [3,] 0.004159725 -0.03568066 0.04400011 0.03984038
```

```
predictions_t %>% fanchart(colors = c("black","red"))
```



From the plots, I can see that the forecasts estimate that the percent change value from one year to the next will decrease for each series.

As a result, there were advantages to performing logarithmic transformations and applying the use of differences. These data manipulations improved model assumptions of correlation and constant variance, which ultimately helped ensure model stationarity. Model stationarity thus satisfied the main forecasting assumptions, producing reliable predictions. Overall, I was able to conduct a thorough EDA, develop a basic and a transformed VAR model for the period 1946-2003, forecast the last three years (2004-2006), conduct residual diagnostics, and examine the relative advantages of logarithmic transformations and the use of differences for the multivariate data series.