W271 Assignment 3

Due 11:59pm Pacific Time Sunday November 29 2020

Shu Ying (Amber) Chen

Instructions (Please Read Carefully):

- No page limit, but be reasonable
- Do not modify fontsize, margin or line_spacing settings
- This assignment needs to be completed individually; this is not a group project. Each student needs to submit their homework to the course github repo by the deadline; submission and revisions made after the deadline will not be graded
- Answers should clearly explain your reasoning; do not simply 'output dump' the results of code without explanation
- Submit two files:

  1. A pdf file that details your answers. Include all R code used to produce the answers. Do not suppress the codes in your pdf file
  2. The R markdown (Rmd) file used to produce the pdf file

  The assignment will not be graded unless **both** files are submitted

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

- Incorrectly following submission instructions results in deduction of grades
- Students are expected to act with regard to UC Berkeley Academic Integrity

Question 1 (2 points)

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.

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.

```r
# Load data
df <- read.csv("Q1.csv")
sales.ts <- ts(as.numeric(df$sales), start = c(1987, 1), frequency = 12) %>%
    as_tsibble() %>% rename(sales = value)
ggplot(sales.ts, aes(x = index, y = sales)) + geom_line() + ggtitle("Monthly Store Sales")
```

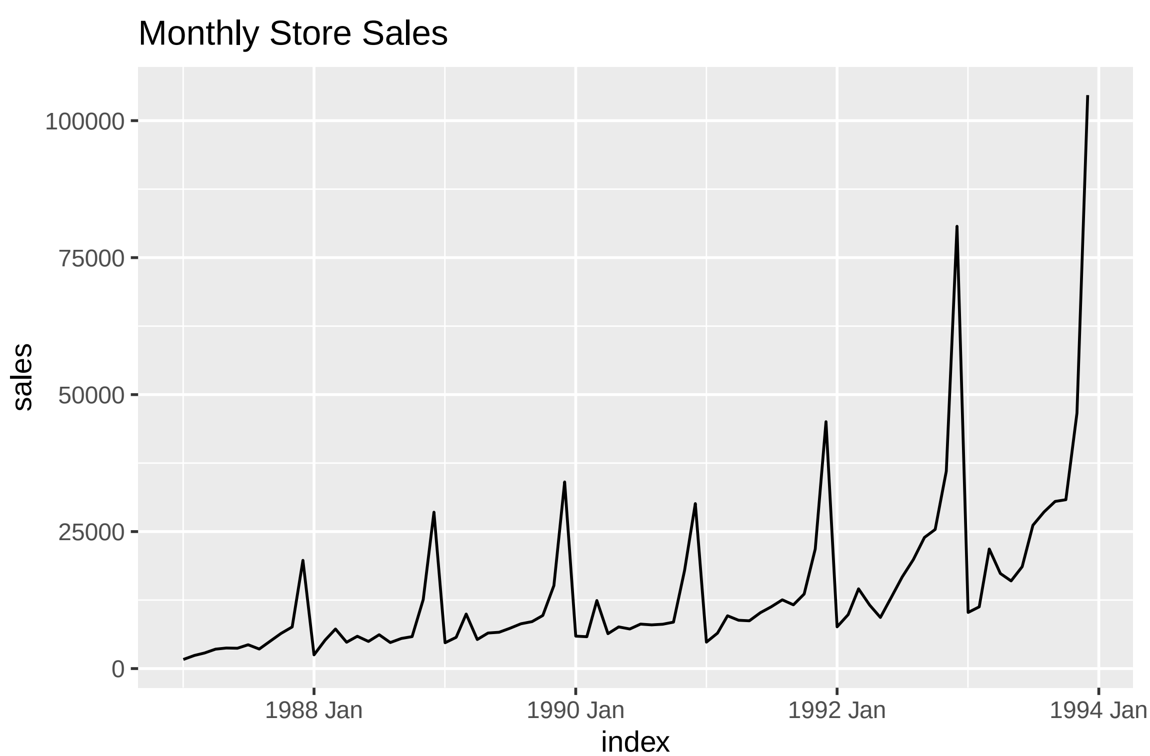


Figure 1: Monthly store sales from 1987 to 1994

The data contains monthly sales figures from January 1987 to December 1993. Within each year the sales figure follows an increasing trend from January to December. The yearly pattern has a stable increase trend from January to November, and has a significant spike in December. Looking at the trend from January to November, we can see the increasing trend steepens year over year. The year-over-year (YoY) figures appear to be an increasing trend for each month, especially December figures seem to be increasing in an exponential trend rather than in a linear trend.

Below two seasonal plots provide a clear picture of YoY increasing trend of each month. The left plot shows that all months have an exponential pattern. The right plot shows that the mean of each month over time does not appear to be different significantly from January to October, but the mean for November and December shows a significant difference from other months.

```
p1 <- sales.ts %>% gg_season(sales) + ylab("Sales") + xlab("Month") +
    ggtitle("Seasonal plot: Store Sales")

p2 <- sales.ts %>% gg_subseries(sales) + ylab("Sales") + xlab("Month") +
    ggtitle("Seasonal subseries plot: Store Sales")

grid.arrange(p1, p2, ncol = 2)
```

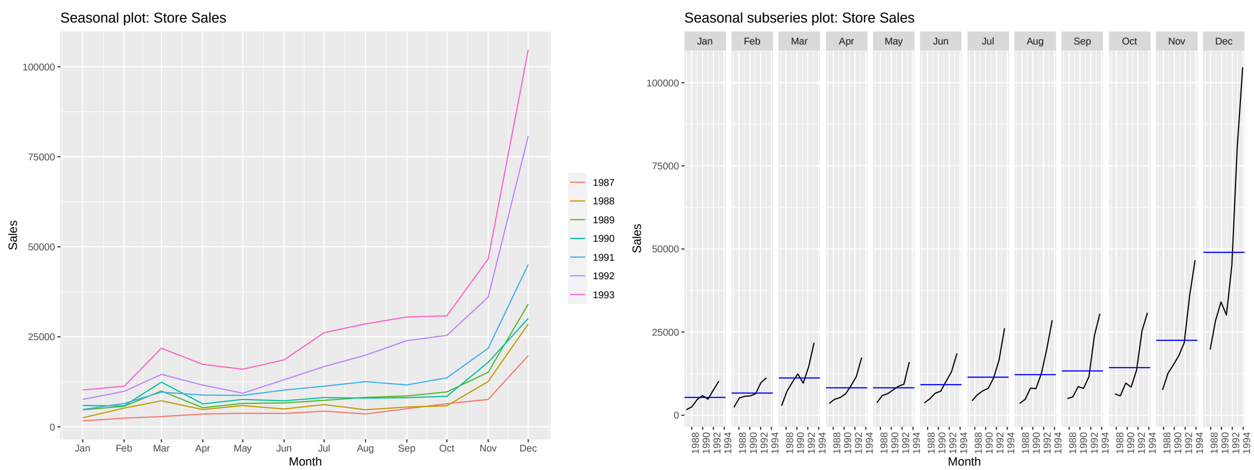


Figure 2: Seasonal plots for monthly sales figures from 1987 to 1994

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

From Fig 1 and 2, we observe an exponential increasing trend for both year over year and month over month within each year. This means that the variance of sales is not constant over time. We can see the seasonal component from the additive decompositiion (Fig. 3) increases over time and the irregular component possesses a similar pattern each year. On the other side, the multiplicative decomposition has a more stable variance and a more random irregular component. Hence, we determine that logarithmic transformation is needed to stablize the variance.

```
# Compare additive and multiplicative decompositions
p1 <- sales.ts %>% model(x11 = feasts:::X11(sales, type = "additive")) %>%
    components() %>% autoplot()
p2 <- sales.ts %>% model(x11 = feasts:::X11(sales, type = "multiplicative")) %>%
    components() %>% autoplot()
grid.arrange(p1, p2, ncol = 2)
```

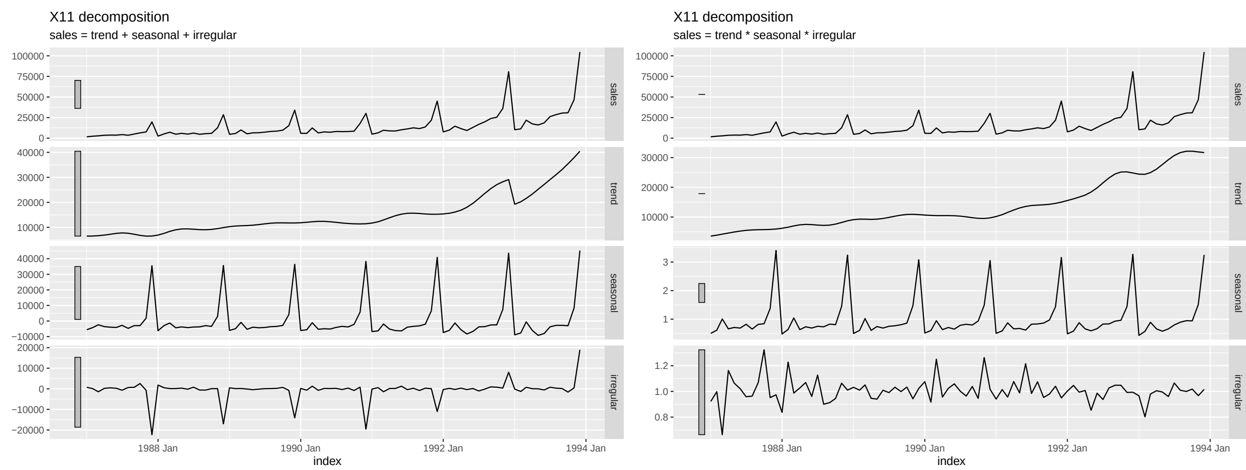


Figure 3: Additive and multiplicative decomposition of the sales data

```r
p1 <- ggplot(sales.ts, aes(x = index, y = log(sales))) + geom_line() +
    ggtitle("Log Monthly Store Sales")
p2 <- sales.ts %>% model(x11 = feasts:::X11(log(sales), type = "additive")) %>%
    components() %>% autoplot()
grid.arrange(p1, p2, ncol = 2)
```

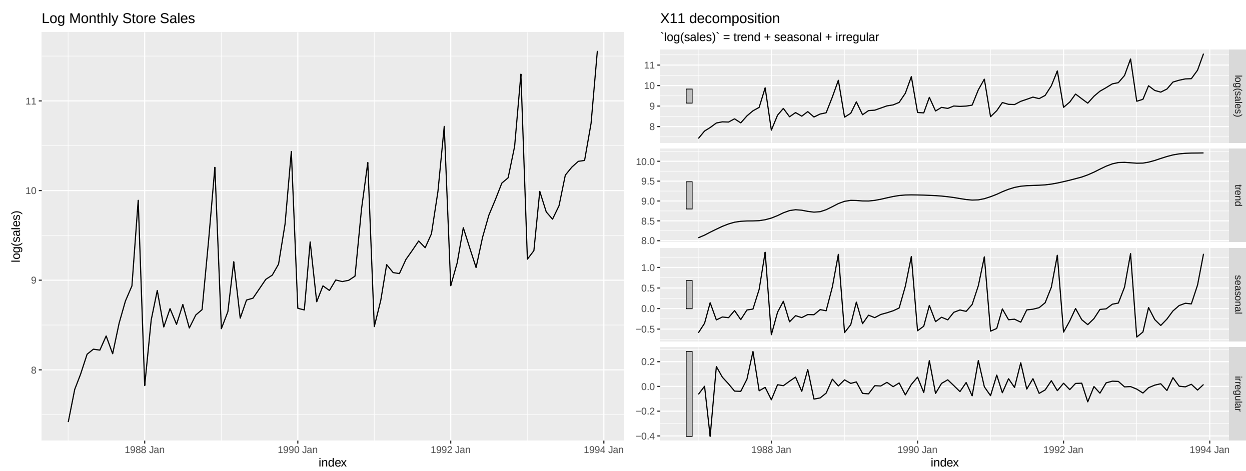


Figure 4: Logarithmic transformation of the sales data and its additive decomposition

After taking log of the time series, the variance becomes more stable year over year and the trend in the irregular component is removed.

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.

We define the regression model as follows:

$$log(sales) = \beta_0 + \beta_1 trend + \beta_2 season_2 + ... + \beta_{12} season_{12} + \beta_{13} SurfingFestival$$

,where $SurfingFestival = 1$ if $month = March$ and $year > 1987$, or 0 if otherwise

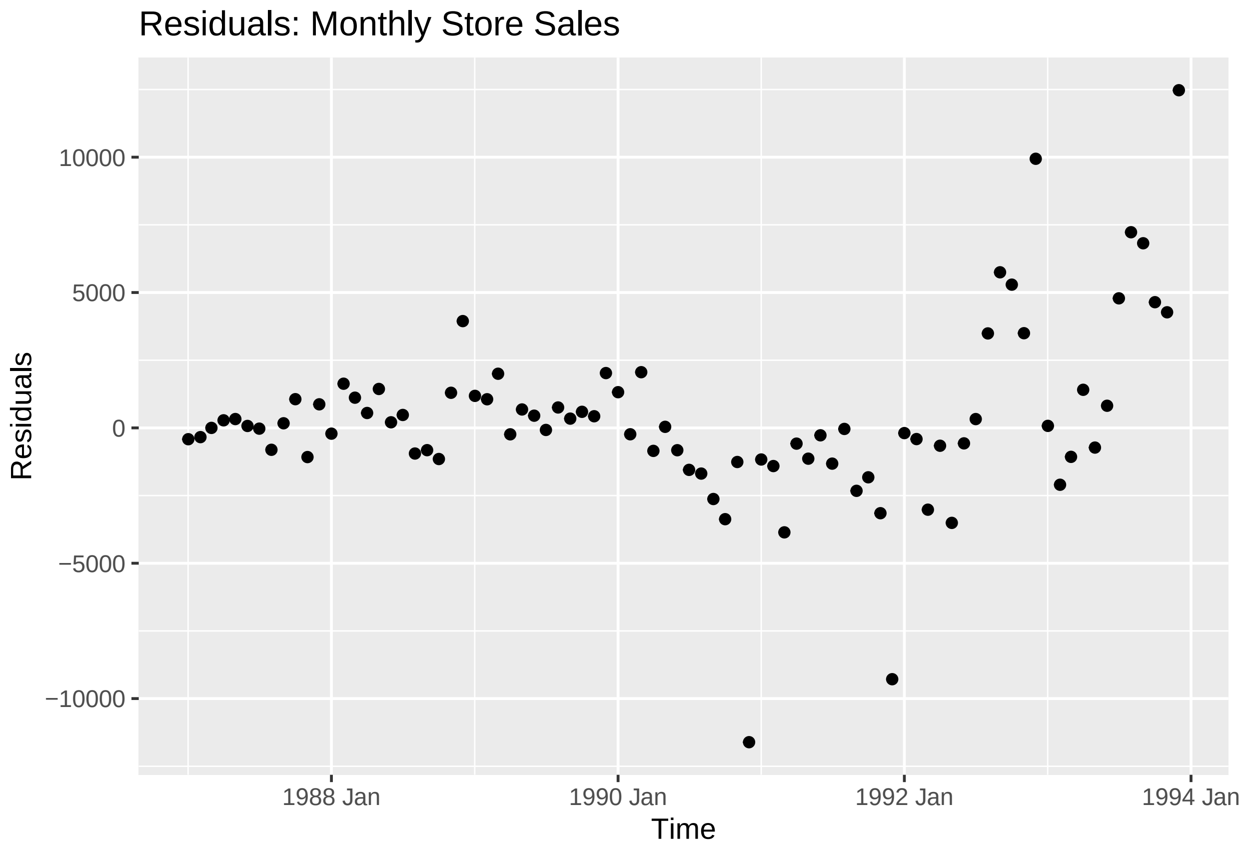
```
sales.ts <- sales.ts %>% mutate(surfing_festival = ifelse(month(index) ==
    3, 1, 0))
sales.ts$surfing_festival[3] = 0

sales.seasonal.lm <- sales.ts %>% model(TSLM(log(sales) ~ trend() +
    season() + surfing_festival))
report(sales.seasonal.lm)
```

```
## Series: sales
## Model: TSLM
## Transformation: log(.x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.336727 -0.127571  0.002568  0.109106  0.376714
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.6196670  0.0742471 102.626  < 2e-16 ***
## trend()          0.0220198  0.0008268  26.634  < 2e-16 ***
## season()year2    0.2514168  0.0956790   2.628 0.010555 *
## season()year3    0.2660828  0.1934044   1.376 0.173275
## season()year4    0.3840535  0.0957075   4.013 0.000148 ***
## season()year5    0.4094870  0.0957325   4.277 5.88e-05 ***
## season()year6    0.4488283  0.0957647   4.687 1.33e-05 ***
## season()year7    0.6104545  0.0958039   6.372 1.71e-08 ***
## season()year8    0.5879644  0.0958503   6.134 4.53e-08 ***
## season()year9    0.6693299  0.0959037   6.979 1.36e-09 ***
## season()year10   0.7473919  0.0959643   7.788 4.48e-11 ***
## season()year11   1.2067479  0.0960319  12.566  < 2e-16 ***
## season()year12   1.9622412  0.0961066  20.417  < 2e-16 ***
## surfing_festival 0.5015151  0.1964273   2.553 0.012856 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.179 on 70 degrees of freedom
## Multiple R-squared: 0.9567,   Adjusted R-squared: 0.9487
## F-statistic:   119 on 13 and 70 DF, p-value: < 2.22e-16
```

The model has a high adjusted R^2 of 0.9487. Though most of the coefficients are statistically significant, the one for March seasonal variable is not, probably due to the inclusion of surfing_festival, which captures the small surge in sales every March except the first year.

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

```r
# Plot residuals against time
augment(sales.seasonal.lm) %>% ggplot(aes(x = index, y = .resid)) +
    geom_point() + ylab("Residuals") + xlab("Time") + ggtitle("Residuals: Monthly Store Sales")
```



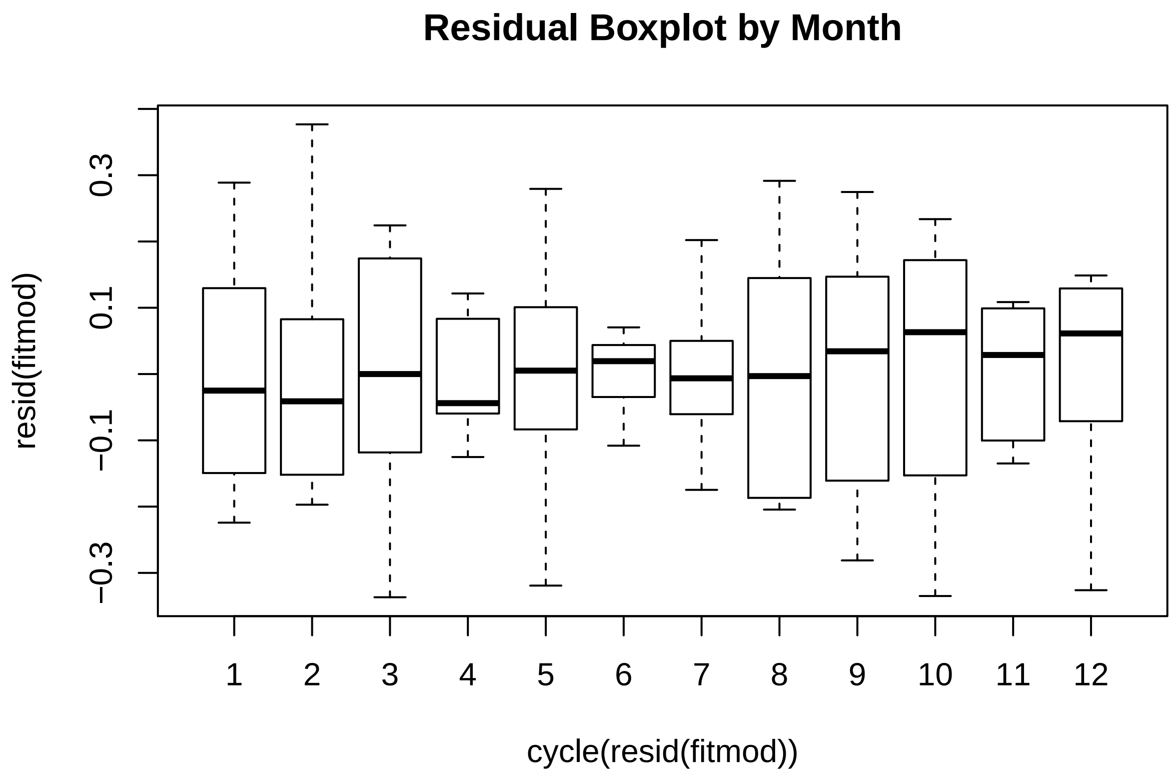

```r
# Plot residuals against the fitted values
augment(sales.seasonal.lm) %>% ggplot(aes(x = .fitted, y = .resid,
    colour = factor(quarter(index)))) + geom_point() + ylab("Residuals") +
    xlab("Fitted values") + ggtitle("Monthly Store Sales") +
    scale_colour_brewer(palette = "Dark2", name = "Quarter")
```

The two plots reveal two problems:

1. The variance of the residuals increase as time progresses, so the residual term is not homoscedastic;
2. The residual term incorporates some seasonal sales trend. The residuals of the 4th quarter data are the most significant amongst all quarters, whereas the residuals of the 1st and 2nd quarters are close to 0.

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

```
logsales <- ts(log(sales.ts$sales), frequency = 12, start = c(1987,
    1))
surfing_festival <- ts(as.numeric(sales.ts$surfing_festival),
    frequency = 12, start = c(1987, 1))
sales.df <- data.frame(logsales, surfing_festival)
fitmod <- tslm(logsales ~ trend + season + surfing_festival,
    data = sales.df)
boxplot(resid(fitmod) ~ cycle(resid(fitmod)), main = "Residual Boxplot by Month")
```



We can see the residuals drift up and down throughout the year, especially from August to December. The residuals stay closer to 0 in April, June and July while spreading out more during August to October.

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

The positive coefficient of trend() tells us an increasing sales trend from 1987 to 1994. The coefficient of surfing festival is statistically significant and captures the positive surge in March sales for all years except the first year. January sales figures are modeled using the intercept, The coefficients of all seasonal variables are positive, indicating that sales of all months are greater than January sales figures and especially November and December have significant increases in sales.

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

```r
bgtest(fitmod)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  fitmod
## LM test = 25.031, df = 1, p-value = 5.642e-07
```

The Breusch-Godfrey test result suggest to reject null hypothesis of that there is no serial correlation of order 1 in residuals. Hence the model is not a good fit as some seasonal factor is remained in the residual term.

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.

```r
# create a surfing festival variable for the forecast period.
# It is 1 for every March and 0 if otherwise.
fore.surf_fest <- rep(0, 36)
fore.surf_fest[seq_along(fore.surf_fest)%%12 == 3] = 1

sales_pred <- forecast(fitmod, newdata = data.frame(surfing_festival = fore.surf_fest))
sales_pred
```

```
##          Point Forecast     Lo 80     Hi 80     Lo 95    Hi 95
## Jan 1994       9.491352  9.238522  9.744183  9.101594  9.88111
## Feb 1994       9.764789  9.511959 10.017620  9.375031 10.15455
## Mar 1994      10.302990 10.048860 10.557120  9.911228 10.69475
## Apr 1994       9.941465  9.688635 10.194296  9.551707 10.33122
## May 1994       9.988919  9.736088 10.241749  9.599161 10.37868
## Jun 1994      10.050280  9.797449 10.303110  9.660522 10.44004
## Jul 1994      10.233926  9.981095 10.486756  9.844168 10.62368
## Aug 1994      10.233456  9.980625 10.486286  9.843698 10.62321
## Sep 1994      10.336841 10.084010 10.589671  9.947083 10.72660
## Oct 1994      10.436923 10.184092 10.689753 10.047165 10.82668
## Nov 1994      10.918299 10.665468 11.171129 10.528541 11.30806
## Dec 1994      11.695812 11.442981 11.948642 11.306054 12.08557
## Jan 1995       9.755590  9.499844 10.011336  9.361338 10.14984
## Feb 1995      10.029027  9.773281 10.284773  9.634775 10.42328
## Mar 1995      10.567228 10.310518 10.823938 10.171489 10.96297
## Apr 1995      10.205703  9.949957 10.461449  9.811451 10.59996
## May 1995      10.253157  9.997411 10.508903  9.858904 10.64741
## Jun 1995      10.314518 10.058772 10.570264  9.920265 10.70877
## Jul 1995      10.498164 10.242418 10.753910 10.103911 10.89242
## Aug 1995      10.497694 10.241948 10.753440 10.103441 10.89195
## Sep 1995      10.601079 10.345333 10.856825 10.206826 10.99533
## Oct 1995      10.701161 10.445415 10.956907 10.306908 11.09541
## Nov 1995      11.182537 10.926791 11.438282 10.788284 11.57679
## Dec 1995      11.960050 11.704304 12.215796 11.565797 12.35430
## Jan 1996      10.019828  9.760564 10.279093  9.620151 10.41951
## Feb 1996      10.293265 10.034000 10.552530  9.893588 10.69294
## Mar 1996      10.831466 10.571566 11.091365 10.430810 11.23212
## Apr 1996      10.469941 10.210677 10.729206 10.070264 10.86962
## May 1996      10.517395 10.258130 10.776659 10.117718 10.91707
## Jun 1996      10.578756 10.319491 10.838021 10.179079 10.97843
## Jul 1996      10.762402 10.503137 11.021667 10.362725 11.16208
## Aug 1996      10.761932 10.502667 11.021196 10.362254 11.16161
## Sep 1996      10.865317 10.606052 11.124582 10.465640 11.26499
## Oct 1996      10.965399 10.706134 11.224664 10.565722 11.36508
## Nov 1996      11.446774 11.187510 11.706039 11.047097 11.84645
## Dec 1996      12.224288 11.965023 12.483552 11.824611 12.62396
```

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

```
exp(as.data.frame(sales_pred))
```

```
##          Point Forecast     Lo 80     Hi 80      Lo 95     Hi 95
## Jan 1994       13244.70  10285.82  17054.73   8969.583  19557.43
## Feb 1994       17409.81  13520.45  22418.00  11790.284  25707.73
## Mar 1994       29821.65  23129.40  38450.24  20155.412  44123.68
## Apr 1994       20774.16  16133.21  26750.16  14068.696  30675.62
## May 1994       21783.73  16917.24  28050.15  14752.395  32166.37
## Jun 1994       23162.27  17987.81  29825.24  15685.969  34201.95
## Jul 1994       27831.56  21613.98  35837.72  18848.111  41096.73
## Aug 1994       27818.48  21603.82  35820.87  18839.249  41077.41
## Sep 1994       30848.42  23956.87  39722.43  20891.193  45551.50
## Oct 1994       34095.57  26478.61  43903.67  23090.230  50346.32
## Nov 1994       55176.84  42850.31  71049.28  37366.903  81475.41
## Dec 1994      120067.79  93244.59 154607.08  81312.400 177294.90
## Jan 1995       17250.40  13357.65  22277.59  11629.938  25587.08
## Feb 1995       22675.20  17558.28  29283.31  15287.252  33633.55
## Mar 1995       38840.85  30046.98  50208.44  26146.972  57697.39
## Apr 1995       27057.06  20951.33  34942.16  18241.435  40133.06
## May 1995       28371.96  21969.51  36640.25  19127.918  42083.42
## Jun 1995       30167.42  23359.80  38958.95  20338.387  44746.58
## Jul 1995       36248.88  28068.91  46812.70  24438.412  53767.06
## Aug 1995       36231.84  28055.72  46790.69  24426.922  53741.78
## Sep 1995       40178.16  31111.50  51887.06  27087.467  59595.26
## Oct 1995       44407.37  34386.35  57348.77  29938.733  65868.34
## Nov 1995       71864.42  55647.40  92807.48  48449.831 106594.69
## Dec 1995      156380.86 121091.75 201954.08 105429.448 231955.81
## Jan 1996       22467.57  17336.40  29117.46  15065.329  33506.86
## Feb 1996       29533.04  22788.25  38274.14  19802.984  44043.89
## Mar 1996       50587.81  39009.73  65602.25  33887.802  75517.62
## Apr 1996       35240.15  27191.96  45670.42  23629.808  52555.15
## May 1996       36952.72  28513.41  47889.88  24778.151  55109.18
## Jun 1996       39291.20  30317.82  50920.48  26346.183  58596.65
## Jul 1996       47211.93  36429.60  61185.57  31657.322  70409.18
## Aug 1996       47189.73  36412.48  61156.80  31642.439  70376.07
## Sep 1996       52329.57  40378.47  67817.91  35088.887  78041.33
## Oct 1996       57837.85  44628.77  74956.52  38782.394  86256.08
## Nov 1996       93598.96  72222.70 121302.09  62761.521 139588.15
## Dec 1996      203676.38 157160.50 263959.89 136572.460 303751.35
```

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

To improve the predictions, we can apply an ARIMA model to better capture seasonality within the time series.

Question 2 (2 points)

Cross-validation

This question is based on section 5.9 of *Forecasting: Principles and Practice Third Edition* (Hyndman and Athanasopoulos).

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

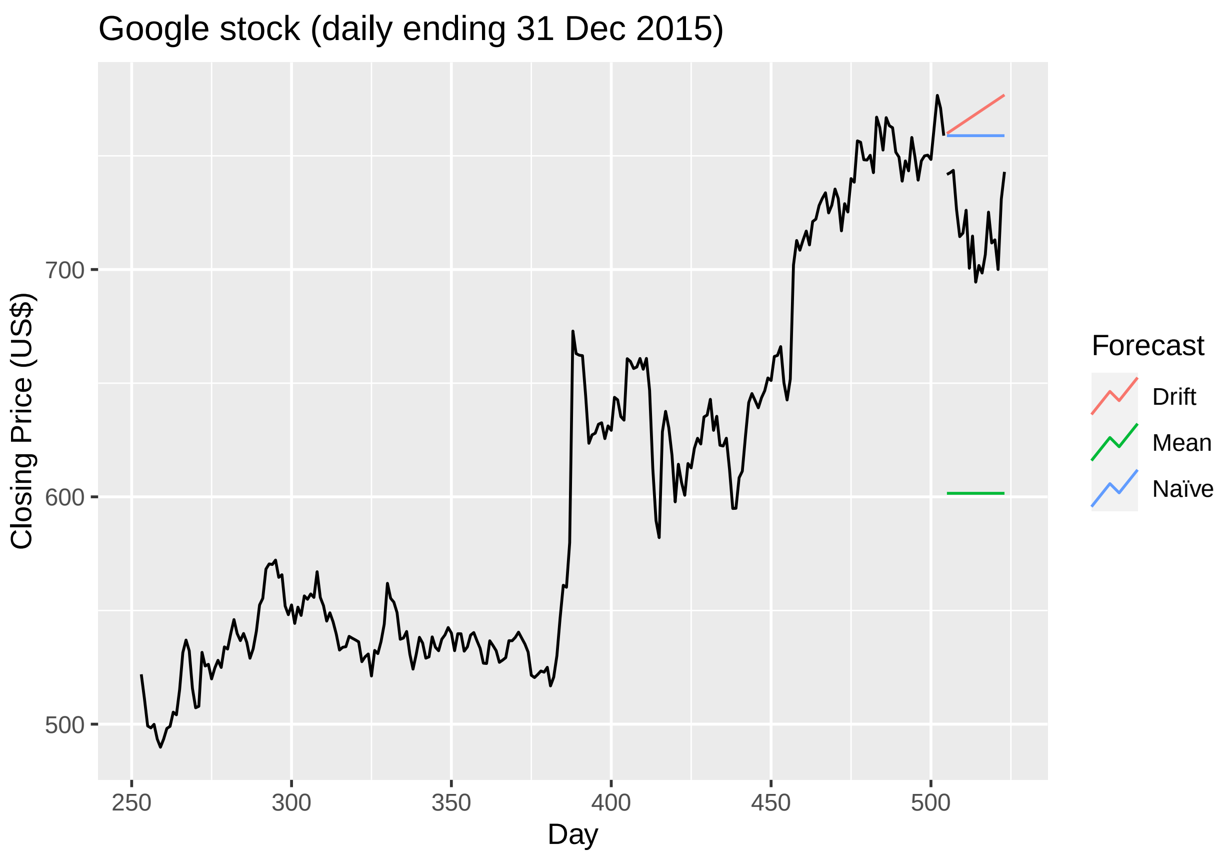
The following code fits the following models to a 2015 training set of Google stock prices:

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

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

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

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

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

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

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



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

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

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

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

In this procedure, there may be a series of 'test sets', each consisting of one observation and corresponding to a 'training set' consisting of the prior observations.

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

fc <- google_2015_tr %>% model(RW(Close ~ drift())) %>% forecast(h = 1)

fc %>% accuracy(google_2015)
```

```
## # A tibble: 1 x 10
##   .model              Symbol .type    ME  RMSE   MAE   MPE  MAPE  MASE   ACF1
##   <chr>               <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 RW(Close ~ drift()) GOOG   Test  0.726  11.3  7.26 0.112  1.19  1.02 0.0985
```

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

RMSE is calculated as the square root of mean of squared errors. The forecast method that minimizes the RMSE will lead to forecasts of the mean.

MAE is calculated as the mean of absolute of error term. The forecast method that minimizes the MAE will lead to forecasts of the median.

MPE is defined as the mean of percentage error. Since the measure is in percentage, it is unit free and can be used to compare forecast performances between data sets.

MAPE is defined as the mean of absolute percentage error.

MASE is decided as the mean of absolute scaled error. It is scaled using MAE of the training set naive forecasts for non-seasonal time series - the Google stock price.

ACF1 is the autocorrelation of errors at lag 1.

The cross-validation procedure produces 4-step-ahead forecast and evaluates forecasts on a rolling forecast basis. The entire year 2015 dataset is splitted into N - 3 training sets and all observations except the first three for the test set. All of the accuracy measures are smaller using cross-validation because of the following two reasons:

- this method trains and validates the model on the same dataset whereas the first method trains year 2015 data and forecast out to January 2016.
- The accuracy measures from cross-validation method are calculated as the average of accuracy in each training set. Since the size of each slice is small relative to the number of training set slices and the data points within each slice are consecutive, the variance of the error term is small and can be improved over the runs.

b) Obtain Facebook stock data from the `gafa_stock` dataset.

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

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.

```
# Filter the year of interest
facebook_2015 <- facebook_stock %>% filter(year(Date) == 2015)

# Time series cross-validation accuracy
facebook_2015_tr <- facebook_2015 %>% slice(1:(n() - 1)) %>%
    stretch_tsibble(.init = 3, .step = 1)
```

```r
fb_fc <- facebook_2015_tr %>% model(Naïve = NAIVE(Volume), Drift = RW(Volume ~
    drift())) %>% forecast(h = 1)

fb_fc %>% accuracy(facebook_2015)
```

```
## # 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  -194181. 10800377. 7500810. -6.74  27.8  1.01 -0.138
## 2 Naïve  FB     Test   -36549. 10713951. 7430153. -6.18  27.4  1.00 -0.139
```

By comparing the RMSE forecasing accuracy of both models, we found that the naive model is a better model than the drift model to predict the volume of Facebook stocks.

Question 3 (2 points):

ARIMA model

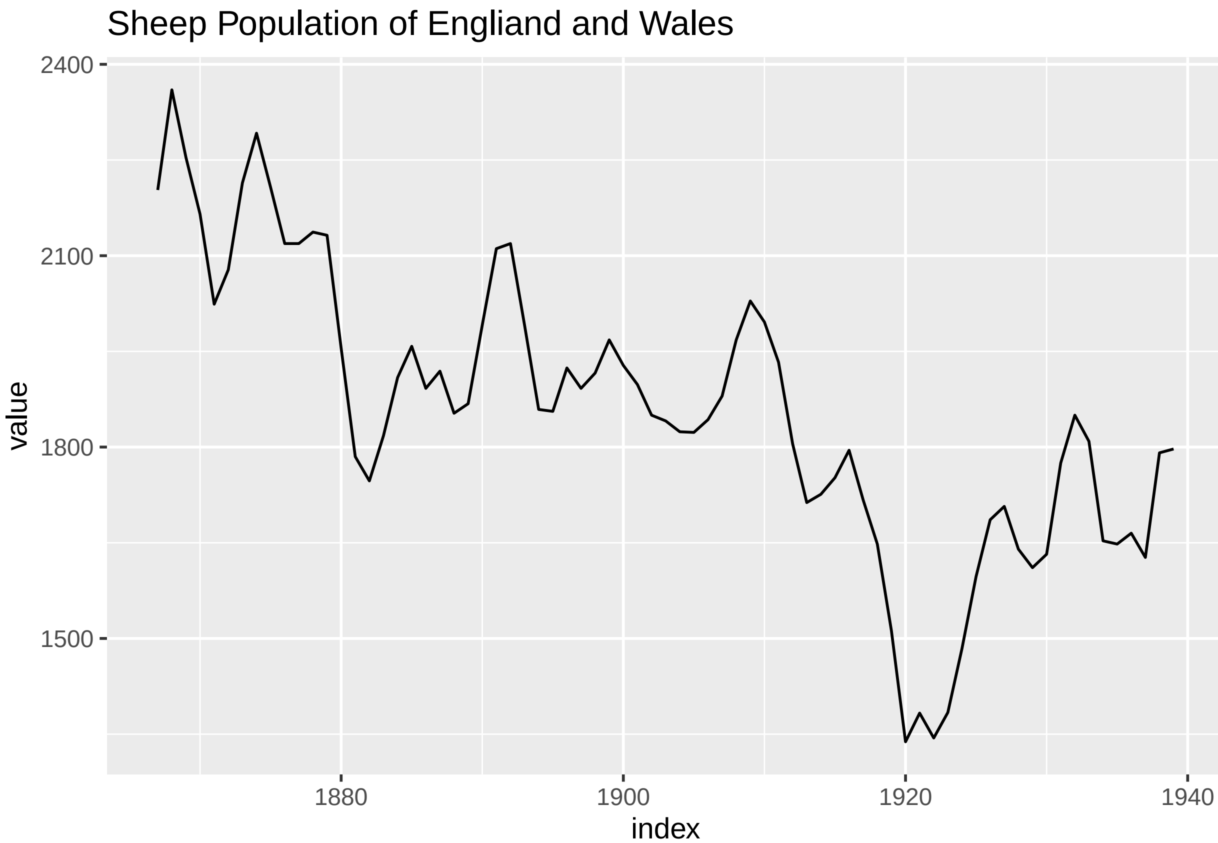
Consider `fma::sheep`, the sheep population of England and Wales from 1867–1939.

```
# install.packages('fma')
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.

```
# the original ts object ended in 1872. It seems that the
# frequency is not set up properly convert to tsibble and fix
# the frequency
sheep_tsibble <- ts(as.numeric(fma::sheep), start = c(1867, 1),
    frequency = 1) %>% as_tsibble()
ggplot(sheep_tsibble, aes(x = index, y = value)) + geom_line() +
    ggtitle("Sheep Population of Engliand and Wales")
```



From the time plot, we can see the sheep population has a general decreasing trend from 1867 to 1940 with some seasonality. The population dipped to the lowest in year 1920 and then bounced back.

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.

The ARIMA model has one degree of first-differencing and autoregressive at lag 3. The model is pdq(3,1,0) with no seasonal component.

The model can be expressed as follows using backshift operator:

$$(1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3)(1 - B)y_t = \epsilon_t$$

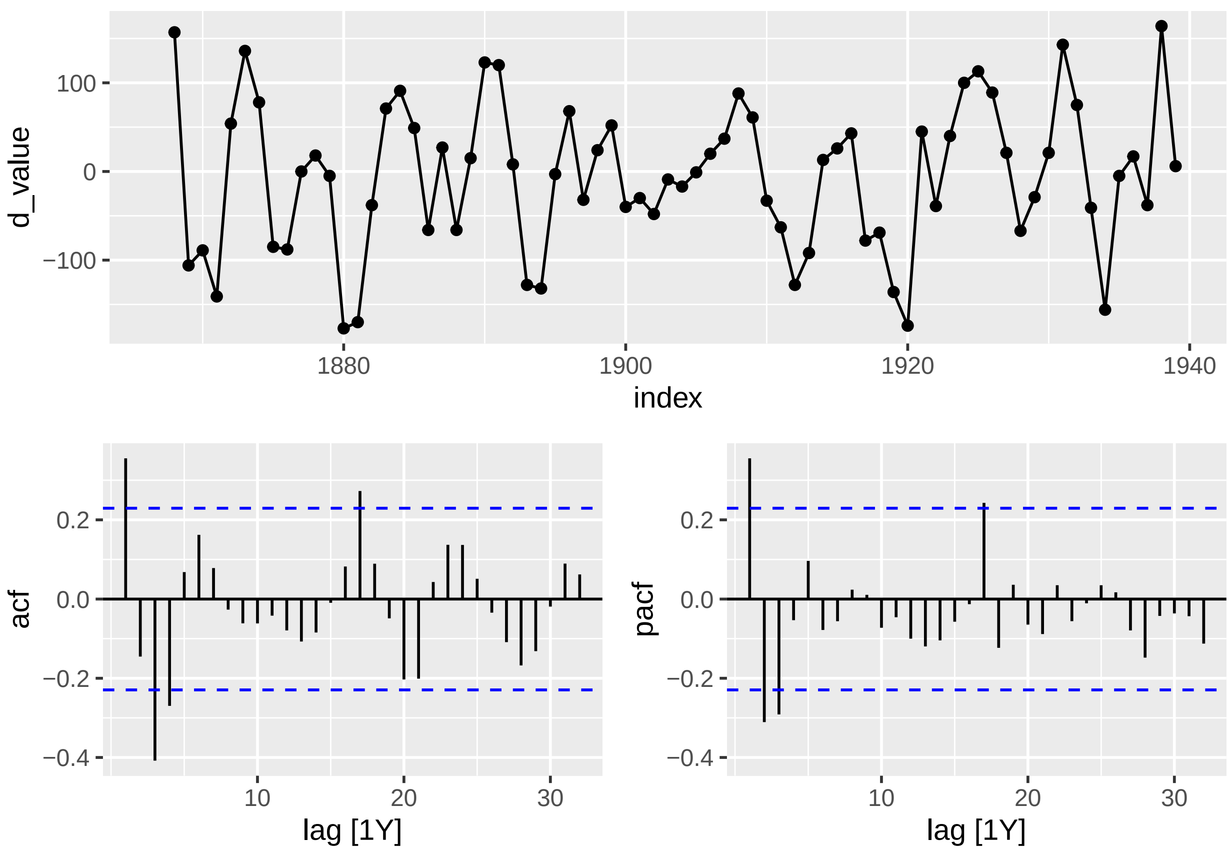
c) By examining the ACF and PACF of the differenced data, explain why this model is appropriate.

```
# Apply first order differencing
sheep_tsibble <- sheep_tsibble %>% mutate(d_value = difference(value,
    1))
```

```r
sheep_tsibble %>% gg_tsdisplay(y = d_value, plot = "partial",
    lag_max = 32)
```

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

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



After the first-differencing is applied, the time series has a stable mean around 0. Its acf tails off after lag 3 and pacf cuts off sharply after lag 3, indicating AR of lag 3 while not showing a MA characteristics. Hence the model pdq(3,1,0) 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 $\phi_1 = 0.42$, $\phi_2 = -0.20$, and $\phi_3 = -0.30$.

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

```r
# extract the last five values from the time series
last5_fc = sheep_tsibble %>% filter(index >= 1935)

# a vector of phi's
phi = cbind(0.42, -0.2, -0.3)

n = last5_fc$index
y = last5_fc$value

# calculates the next 3-year's values using the formula
# provided in question (b)
for (i in 1:3) {
    len = length(n)
    n = c(n, n[len] + 1)
    y = c(y, round(y[len] + phi[1] * (y[len] - y[len - 1]) +
        phi[2] * (y[len - 1] - y[len - 2]) + phi[3] * (y[len -
        2] - y[len - 3]) + rnorm(1)))
}

sheep_3yr_fc <- tibble(index = year(as.Date(as.character(tail(n,
    3)), format = "%Y")), value = tail(y, 3)) %>% as_tsibble(index = index)
sheep_3yr_fc
```

```
## # A tsibble: 3 x 2 [1Y]
##   index value
##   <dbl> <dbl>
## 1  1940  1776
## 2  1941  1717
## 3  1942  1694
```

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

Rearranging the characteristic equation to

$$(1-(\phi_1+1)B+(\phi_1-\phi_2)B^2+(\phi_2-\phi_3)B^3+\phi_3B^4)y_t=\epsilon_t$$

```r
Mod(polyroot(c(1, -(phi[1] + 1), (phi[1] - phi[2]), (phi[2] -
    phi[3]), phi[3])))
```

```
## [1] 1.000000 1.261473 1.261473 2.094704
```

The model is non-stationary due to the presence of one unit root, though all other roots exceed unity in absolute value.

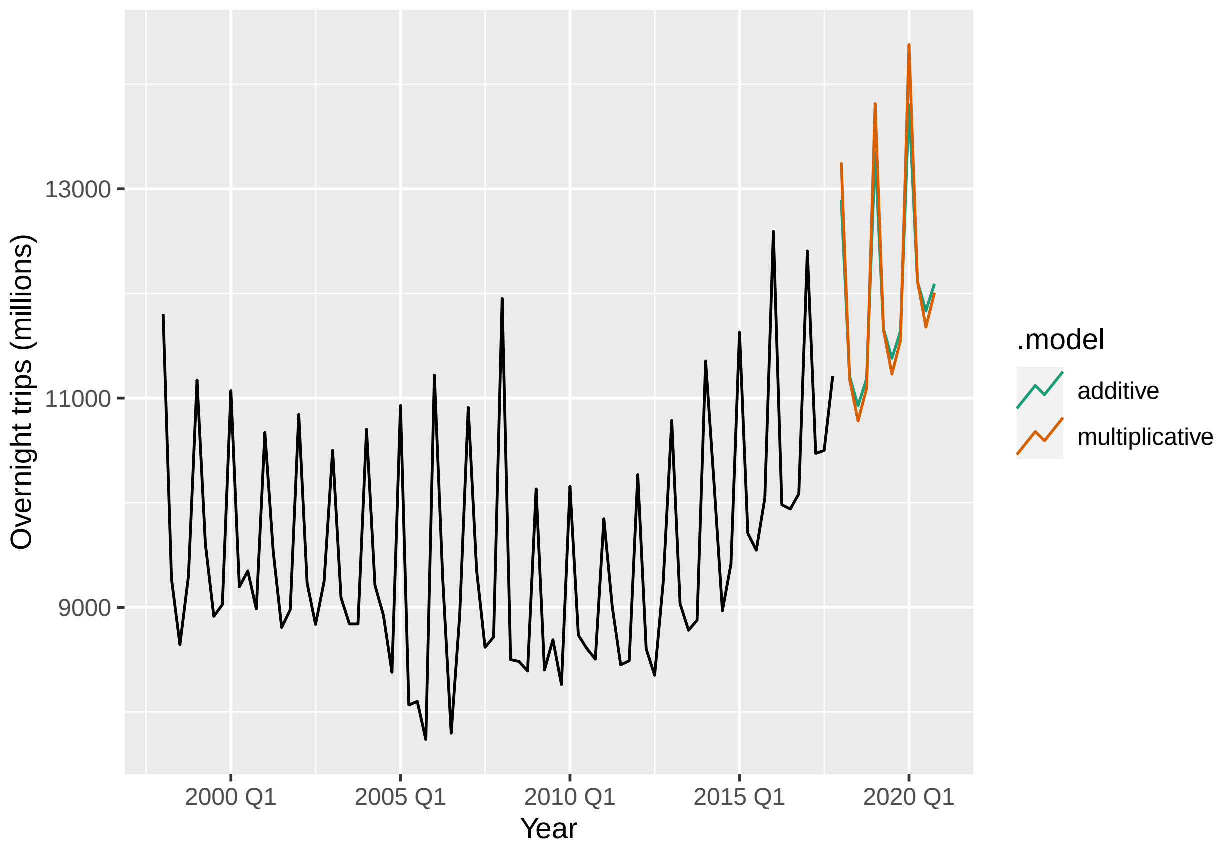
Question 4 (2 points):

Model averaging

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

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

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

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

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



Apply a Holt-Winters model to the ECOMPCTNSA time series data recorded in `Q4.csv` (these are the same Federal Reserve Economic Database data from the Week 9 Live Session). 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.

```
# Load data
ecom <- read.csv("Q4.csv") %>% mutate(DATE = dmy(DATE)) %>% as_tsibble(index = "DATE") %>%
    mutate(DATE = yearquarter(DATE)) %>% rename(date = DATE,
    value = ECOMPCTNSA)

# Inspect data
str(ecom)
```

```
## tsibble [81 x 2] (S3: tbl_ts/tbl_df/tbl/data.frame)
##  $ date : qtr [1:81] 1999 Q4, 2000 Q1, 2000 Q2, 2000 Q3, 2000 Q4, 2001 Q1, 2001 ...
##    ..@ fiscal_start: num 1
##  $ value: num [1:81] 0.7 0.8 0.8 0.9 1.2 1.1 1 1 1.3 1.3 ...
##  - attr(*, "key")= tibble [1 x 1] (S3: tbl_df/tbl/data.frame)
##   ..$ .rows: list<int> [1:1]
##   .. ..$ : int [1:81] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..@ ptype: int(0)
##  - attr(*, "index")= chr "date"
```

```
##   ..- attr(*, "ordered")= logi TRUE
##  - attr(*, "index2")= chr "date"
##  - attr(*, "interval")= interval [1:1] 1Q
##   ..@ .regular: logi TRUE
```

```r
head(ecom)
```

```
## # A tsibble: 6 x 2 [1Q]
##      date value
##     <qtr> <dbl>
## 1 1999 Q4   0.7
## 2 2000 Q1   0.8
## 3 2000 Q2   0.8
## 4 2000 Q3   0.9
## 5 2000 Q4   1.2
## 6 2001 Q1   1.1
```

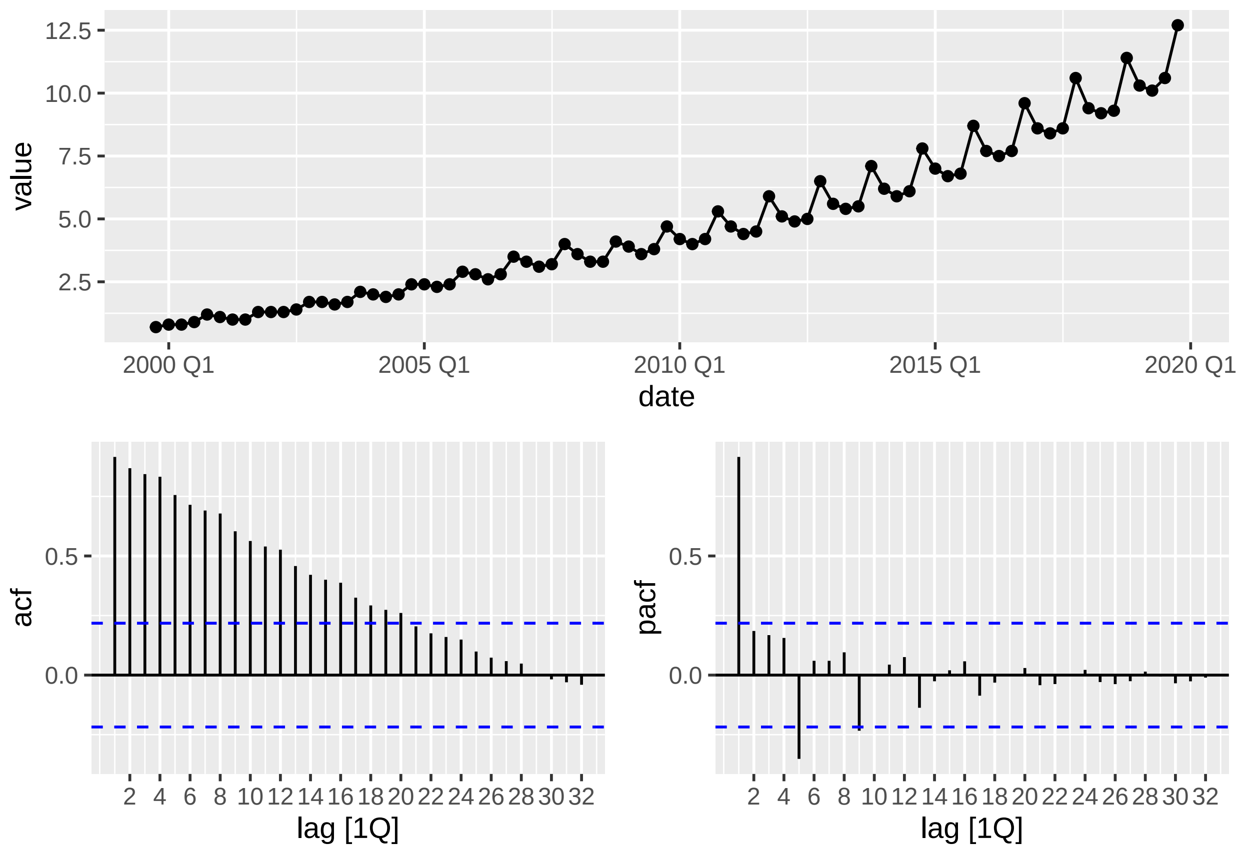
```r
tail(ecom)
```

```
## # A tsibble: 6 x 2 [1Q]
##      date value
##     <qtr> <dbl>
## 1 2018 Q3   9.3
## 2 2018 Q4  11.4
## 3 2019 Q1  10.3
## 4 2019 Q2  10.1
## 5 2019 Q3  10.6
## 6 2019 Q4  12.7
```

```r
# Aggregate statistics
summary(ecom$value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.700   2.300   4.100   4.749   6.800  12.700
```

```r
# Time series plot, autocorrelation and partial
# autocorrelation function

ecom %>% gg_tsdisplay(y = value, plot = "partial", lag_max = 32)
```



The time series has an upward trend with quarterly seasonal pattern, and this seasonality cycle does not seem to be consistent, but increasing through the period. Autocorrelation decays slowly while partial autocorrelation cuts off after lag 1 and have some spikes at seasonal lags.

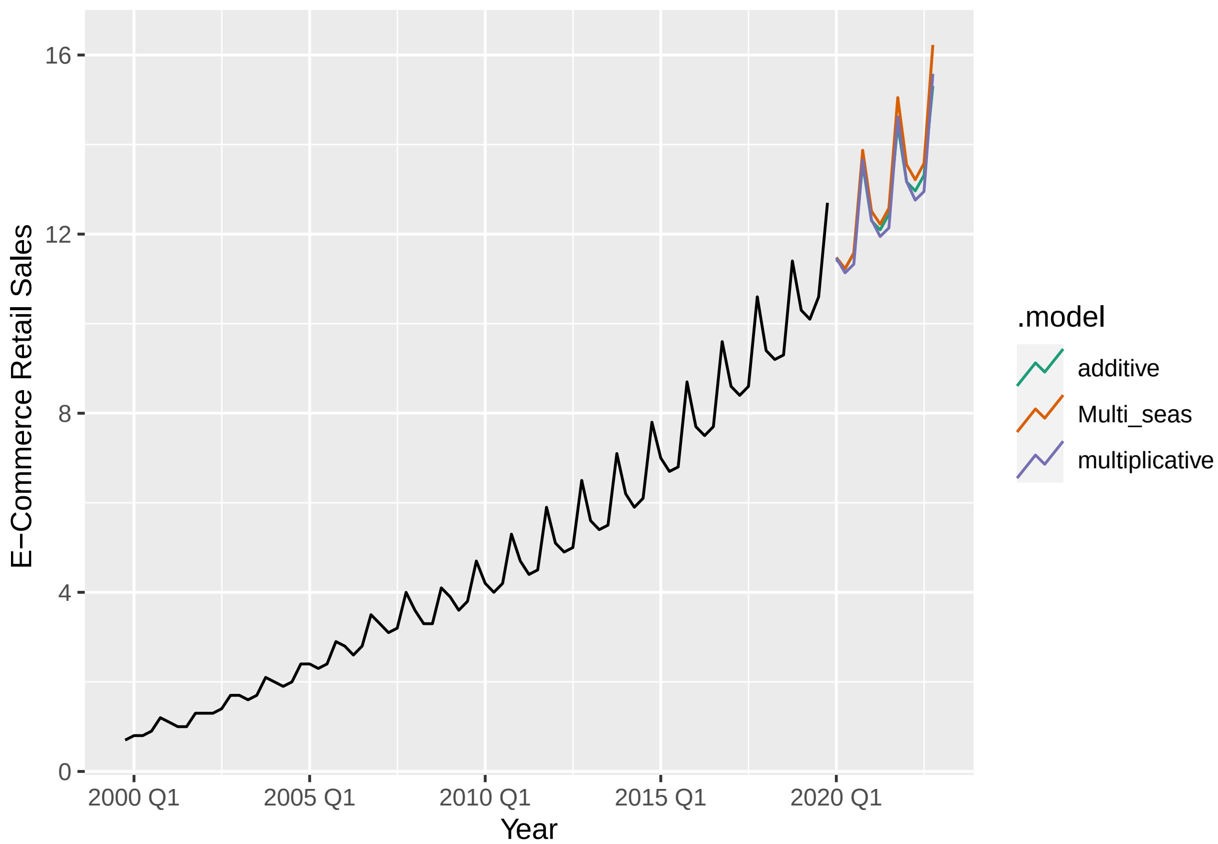
We define and compare three models using the Holt-Winters method:

Additive: ETS(A, A, A) - All three components are additive

Mutli_seas: ETS(A, A, M) - The seasonal component is multiplicative while other compenents remain additive

Multiplicative: ETS(M, A, M) - The error and seasonal components are multiplicative while the trend compenent remains additive

```
# using ETS() function from fable
ecom_fit1 <- ecom %>% model(additive = ETS(value ~ error("A") +
    trend("A") + season("A")), Multi_seas = ETS(value ~ error("A") +
    trend("A") + season("M")), multiplicative = ETS(value ~ error("M") +
    trend("A") + season("M")))
ecom_fc1 <- ecom_fit1 %>% forecast(h = "3 years")

ecom_fc1 %>% autoplot(ecom, level = NULL) + xlab("Year") + ylab("E-Commerce Retail Sales ") +
    scale_color_brewer(type = "qual", palette = "Dark2")
```




```
ecom_fit1 %>% accuracy()
```

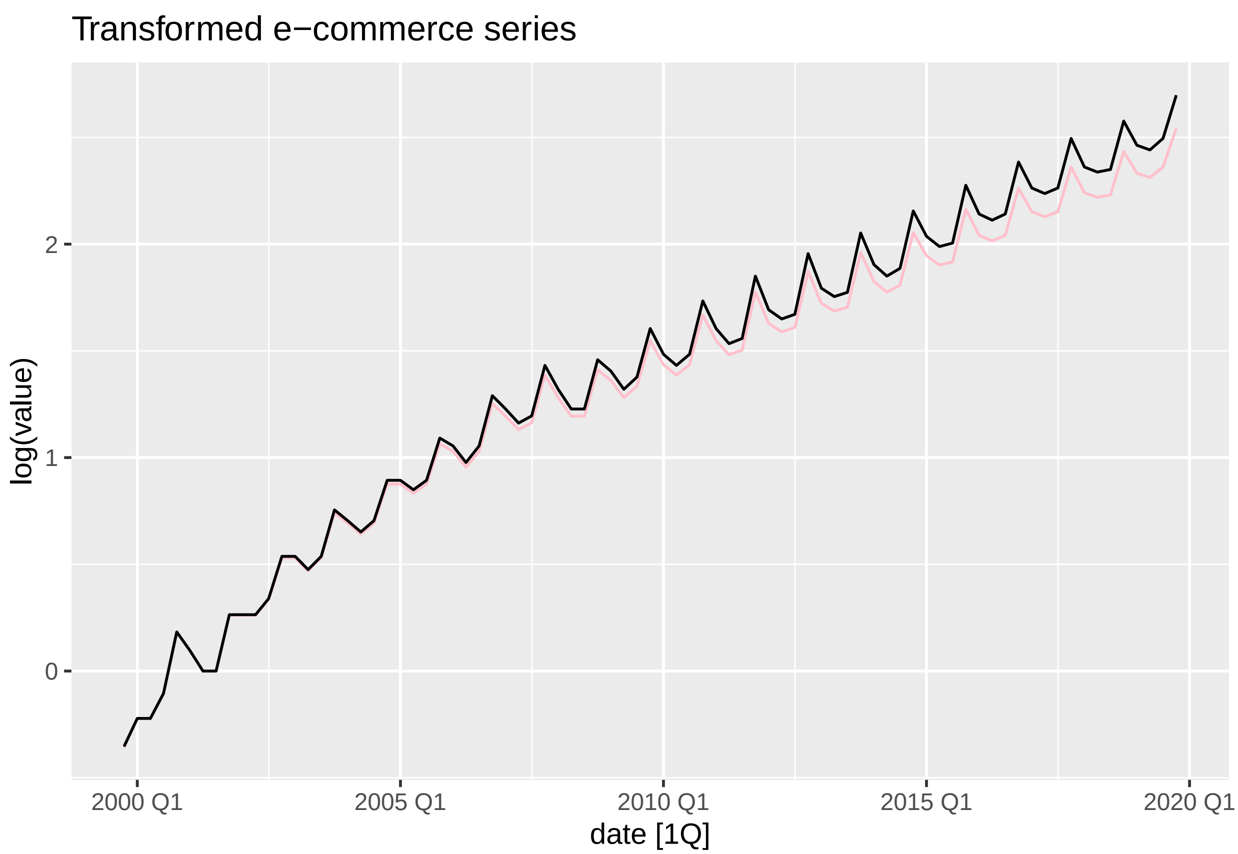
```
## # A tibble: 3 x 9
##   .model         .type        ME  RMSE    MAE    MPE  MAPE  MASE     ACF1
##   <chr>          <chr>     <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl>    <dbl>
## 1 additive       Training 0.0190 0.149 0.114  -1.30   4.58 0.218  0.229
## 2 Multi_seas     Training 0.0212 0.111 0.0834  0.232  2.59 0.158  0.153
## 3 multiplicative Training 0.0284 0.122 0.0896  0.244  2.35 0.170 -0.00773
```

All three models predict very similar values for next 3 years, and their accuracy measures are also similar. We can see the “Multi_seas” model performs slightly better than the other two since it has 6 out of 7 measures better than the “addtive” model and 5 out of 7 measures better than the “multiplicative” model. Therefore, we select the “Multi_seas” model for further comparison with ARIMA model.

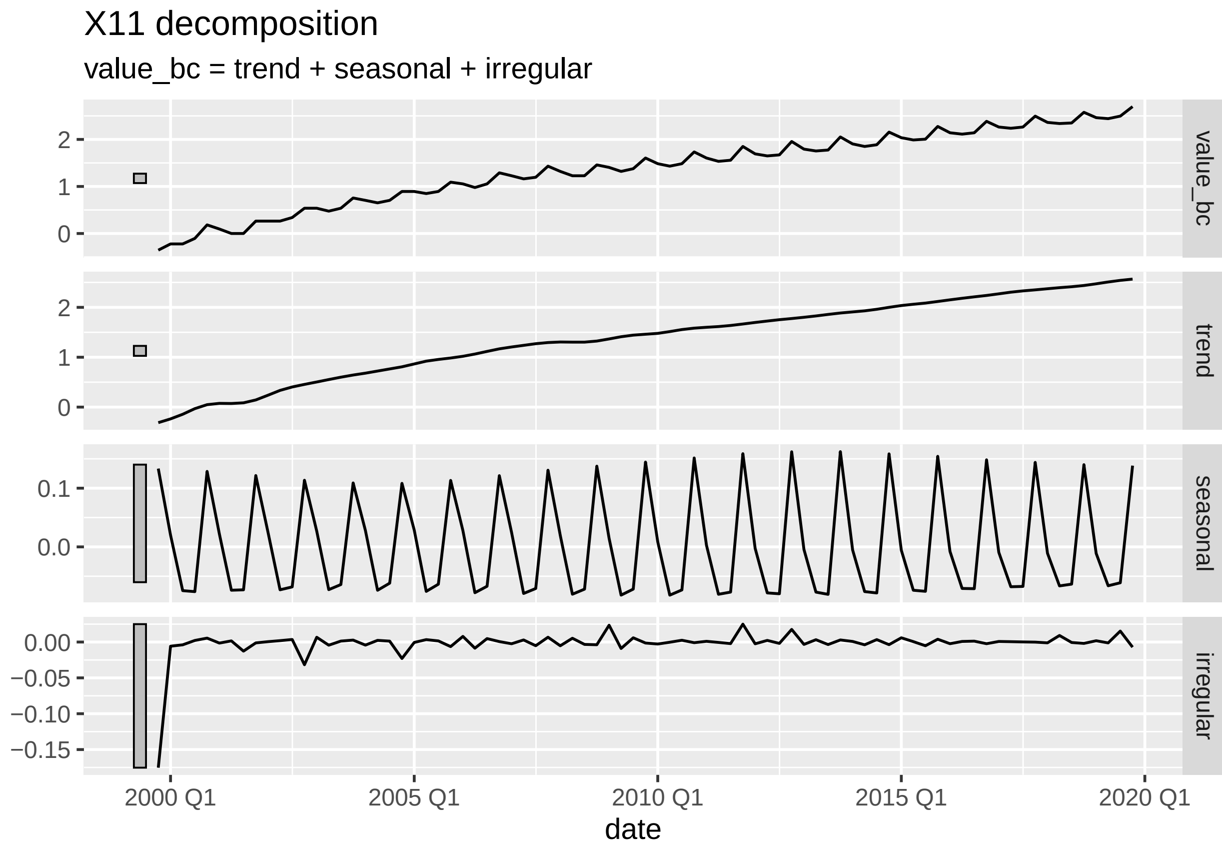
Following the live session exercise, I determine that a log or box-cox transformation is needed to stablize the variance, and the two methods are very similar in this case. Hence, we apply box-cox transformation and then apply first and seasonal differencing. We also filter the anomalous period prior to 2003 for only regular seasonal data.

```
lambda <- ecom %>% features(value, features = guerrero) %>% pull(lambda_guerrero)
ecom <- ecom %>% mutate(value_bc = box_cox(value, lambda = lambda))
```

```
ecom %>% autoplot(log(value), color = "pink") + autolayer(ecom,
    value_bc) + ggtitle("Transformed e-commerce series")
```



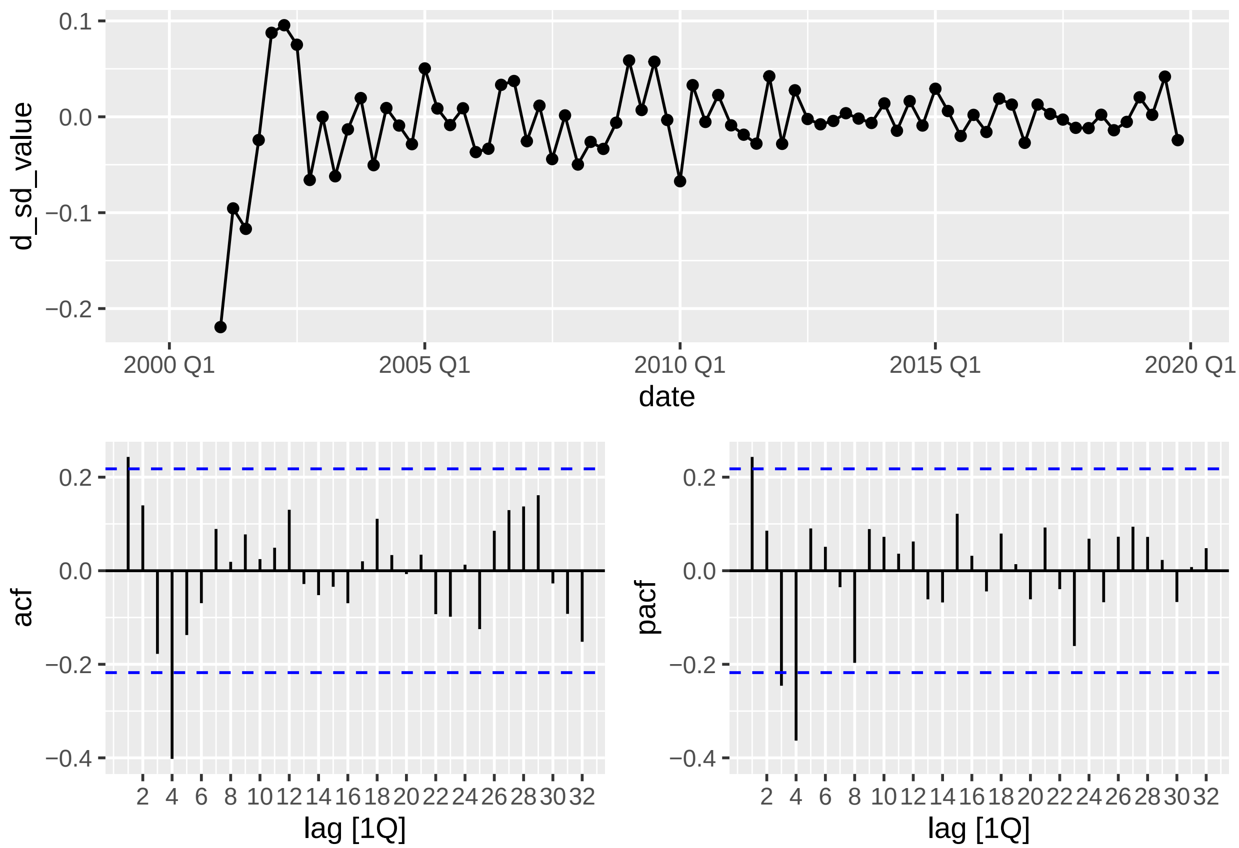

```
ecom %>% model(x11 = feasts:::X11(value_bc, type = "additive")) %>%
    components() %>% autoplot()
```



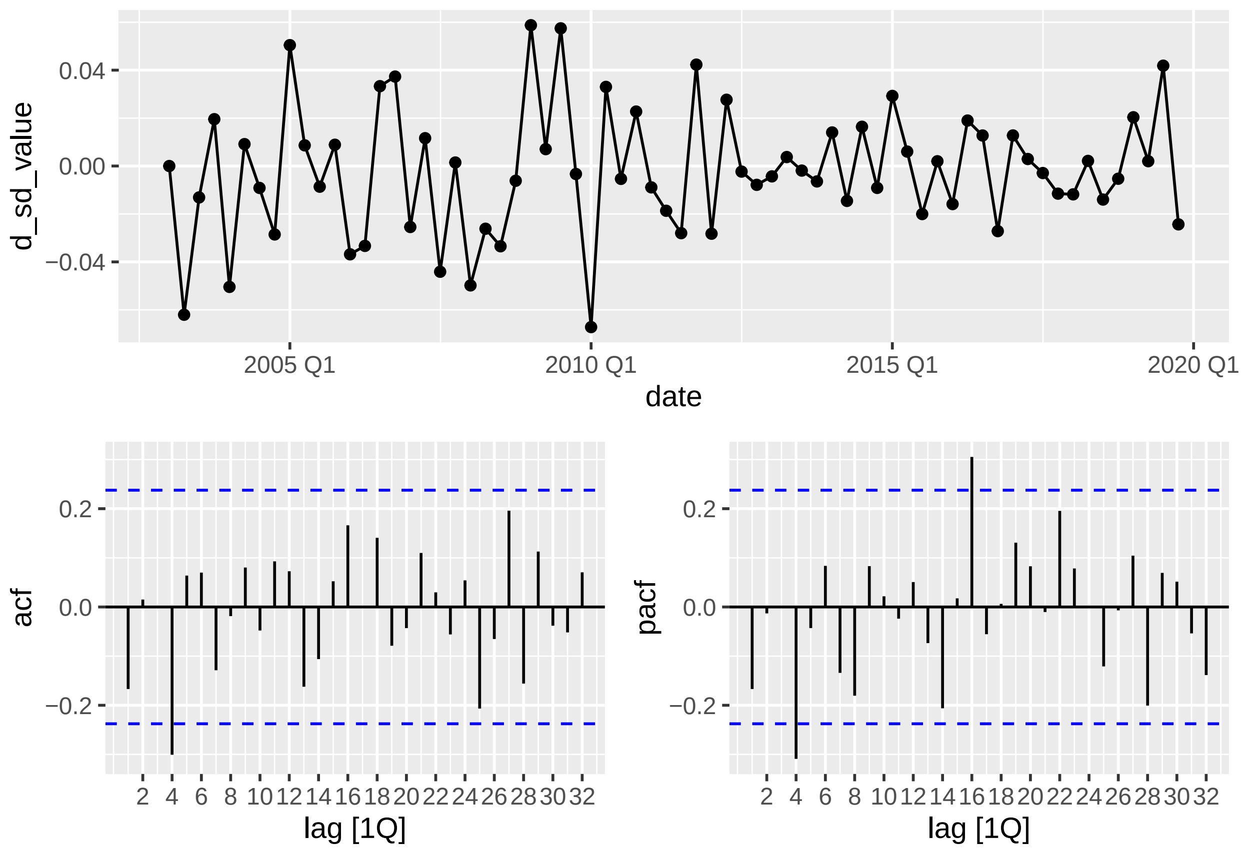

```
ecom <- ecom %>% mutate(d_sd_value = difference(difference(value_bc,
    4)))
ecom %>% gg_tsdisplay(y = d_sd_value, plot = "partial", lag_max = 32)
```

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

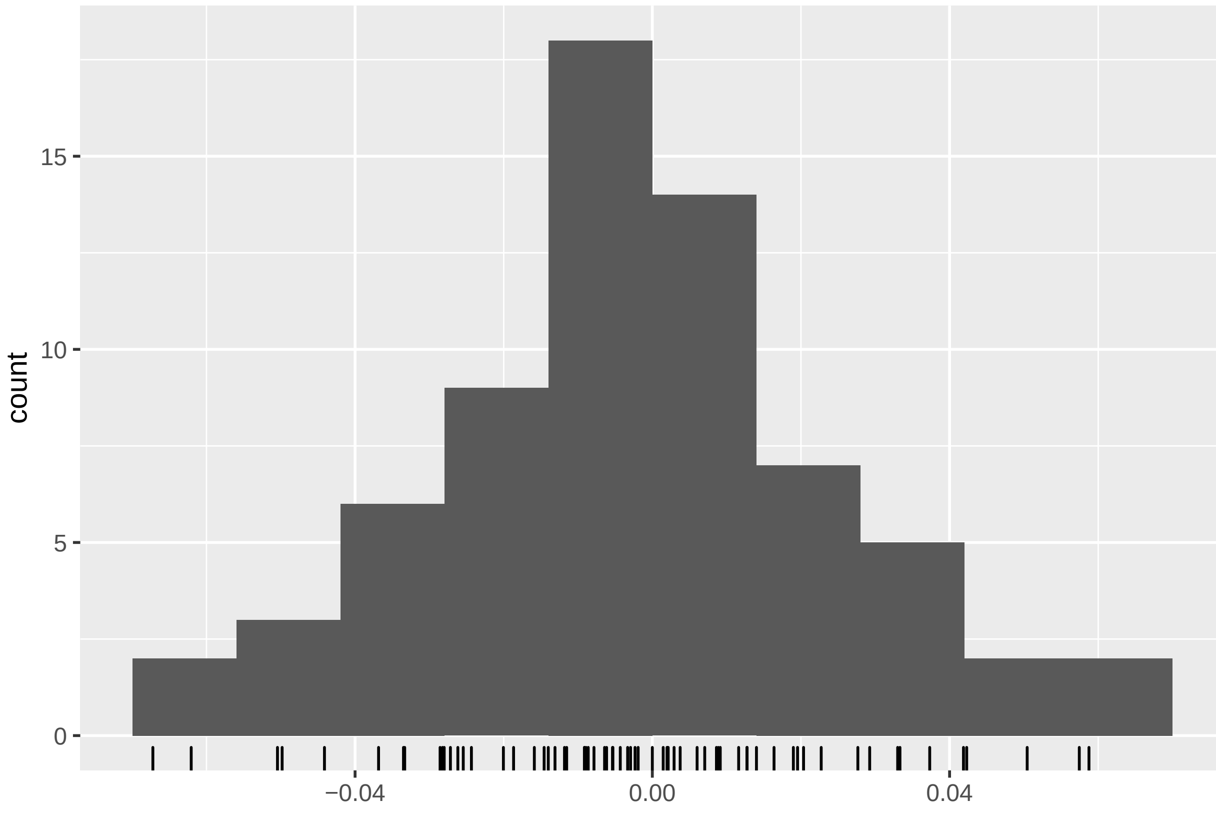
```
## Warning: Removed 5 rows containing missing values (geom_point).
```




```r
ecom03 <- ecom %>% filter_index("2003 Q1" ~ .)
ecom03 %>% gg_tsdisplay(y = d_sd_value, plot = "partial", lag_max = 32)
```




```
ecom03$d_sd_value %>% gghistogram()
```



The correlograms of the first and seasonal-differenced series show significant spikes in ACF and PACF at lag 4, potentially suggesting a seasonal MA(1) and/or AR(1) component. There may be other components but let's first compare an $ARIMA(0,1,0)(0,1,1)_4$, an $ARIMA(0,1,0)(1,1,0)_4$ and an $ARIMA(0,1,0)(1,1,1)_4$, using in-sample and pseudo-out-of-sample accuracy comparisons.

We split the data into training and test sets, taking the final two years as a test set period for pseudo-out-of-sample forecasting performance.

```
ecom.training <- ecom03 %>% filter_index(~"2017 Q4")
ecom.test <- ecom03 %>% filter_index("2018 Q1" ~ .)
```

```
results <- data.frame(p = integer(), q = integer(), P = integer(),
    Q = integer(), AICc = double())
for (p in 0:4) {
    for (q in 0:4) {
        for (P in 0:4) {
            for (Q in 0:4) {
                tryCatch({
                  mod <- ecom.training %>% model(ARIMA(box_cox(value,
                    lambda = lambda) ~ 0 + pdq(p, 1, q) + PDQ(P,
                    1, Q)))

                  if (has_name(glance(mod), "AICc")) {
                  }
```

```
                  results <- results %>% add_row(p = p, q = q,
                    P = P, Q = Q, AICc = as.numeric(glance(mod)$AICc))
                  # print(paste(p, q, P, Q, as.numeric(glance(mod)$AICc)))
                }, error = function(e) {
                  # print(paste('error encountered for', p, q, P, Q))
                })
            }
        }
    }
}
results[which.min(results$AICc), ]
```

```
##    p q P Q      AICc
## 27 0 1 0 1 -246.2127
```

Though we found the model with the lowest AICc is $ARIMA(0,1,1)(0,1,1)_4$ through loop from lag 0 to 4 for each of p, q, P, and Q, we should still compare the accuracy measures of the model against the other two models we proposed.

```
models <- ecom.training %>% model(mod1 = ARIMA(box_cox(value,
    lambda = lambda) ~ pdq(0, 1, 0) + PDQ(0, 1, 1, period = 4)),
    mod2 = ARIMA(box_cox(value, lambda = lambda) ~ pdq(0, 1,
        0) + PDQ(1, 1, 0, period = 4)), mod3 = ARIMA(box_cox(value,
        lambda = lambda) ~ pdq(0, 1, 0) + PDQ(1, 1, 1, period = 4)),
    mod4 = ARIMA(box_cox(value, lambda = lambda) ~ pdq(0, 1,
        1) + PDQ(0, 1, 1, period = 4)))
models %>% dplyr::select(mod1) %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,0)(0,1,1)[4]
## Transformation: box_cox(.x, lambda = lambda)
##
## Coefficients:
##          sma1
##       -0.3191
## s.e.   0.1276
##
## sigma^2 estimated as 0.0006277:  log likelihood=125.02
## AIC=-246.03   AICc=-245.8   BIC=-242.02
```

```
models %>% dplyr::select(mod2) %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,0)(1,1,0)[4]
## Transformation: box_cox(.x, lambda = lambda)
```

```
##
## Coefficients:
##          sar1
##       -0.2636
## s.e.   0.1312
##
## sigma^2 estimated as 0.0006414:  log likelihood=124.49
## AIC=-244.99   AICc=-244.76   BIC=-240.97
```

```
models %>% dplyr::select(mod3) %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,0)(1,1,1)[4]
## Transformation: box_cox(.x, lambda = lambda)
##
## Coefficients:
##         sar1     sma1
##       0.0783  -0.3851
## s.e.  0.3542   0.3162
##
## sigma^2 estimated as 0.0006391:  log likelihood=125.04
## AIC=-244.08   AICc=-243.61   BIC=-238.06
```

```
models %>% dplyr::select(mod4) %>% report()
```

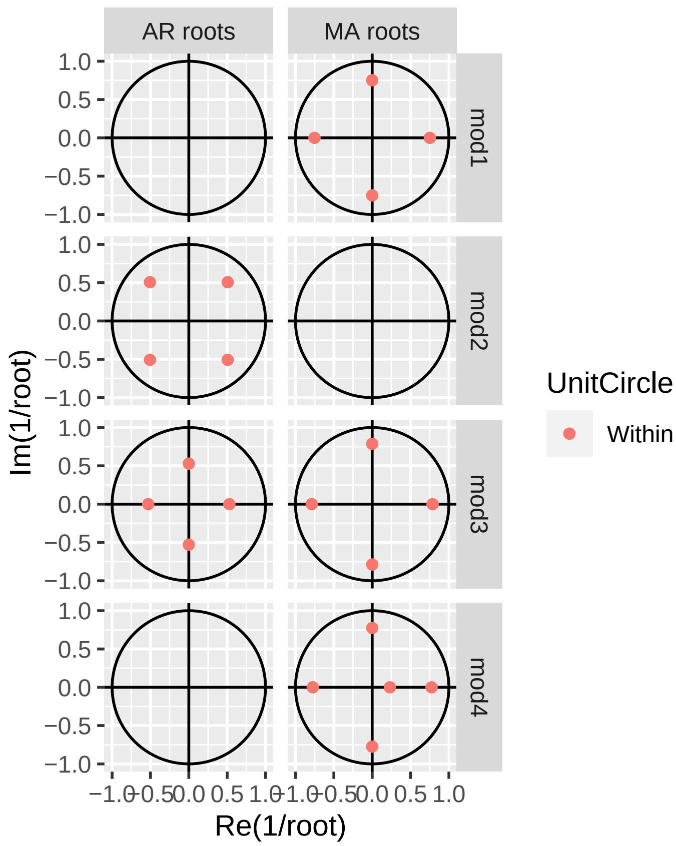
```
## Series: value
## Model: ARIMA(0,1,1)(0,1,1)[4]
## Transformation: box_cox(.x, lambda = lambda)
##
## Coefficients:
##           ma1     sma1
##       -0.2308  -0.3595
## s.e.   0.1414   0.1267
##
## sigma^2 estimated as 0.0006075:  log likelihood=126.34
## AIC=-246.68   AICc=-246.21   BIC=-240.66
```

```
# The `glance()` function applied to a set of ARIMA models
# shows the variance of residuals (sigma2), the
# log-likelihood (log_lik), information criterion (AIC, AICc,
# BIC) and the characteristic roots (ar_roots and ma_roots).

glance(models)
```

```
## # A tibble: 4 x 8
```

```
##    .model   sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##    <chr>     <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 mod1   0.000628    125. -246. -246. -242. <cpl [0]> <cpl [4]>
## 2 mod2   0.000641    124. -245. -245. -241. <cpl [4]> <cpl [0]>
## 3 mod3   0.000639    125. -244. -244. -238. <cpl [4]> <cpl [4]>
## 4 mod4   0.000608    126. -247. -246. -241. <cpl [0]> <cpl [5]>
```

```
# Inverse roots all lie within the unit circle
gg_arma(models)
```

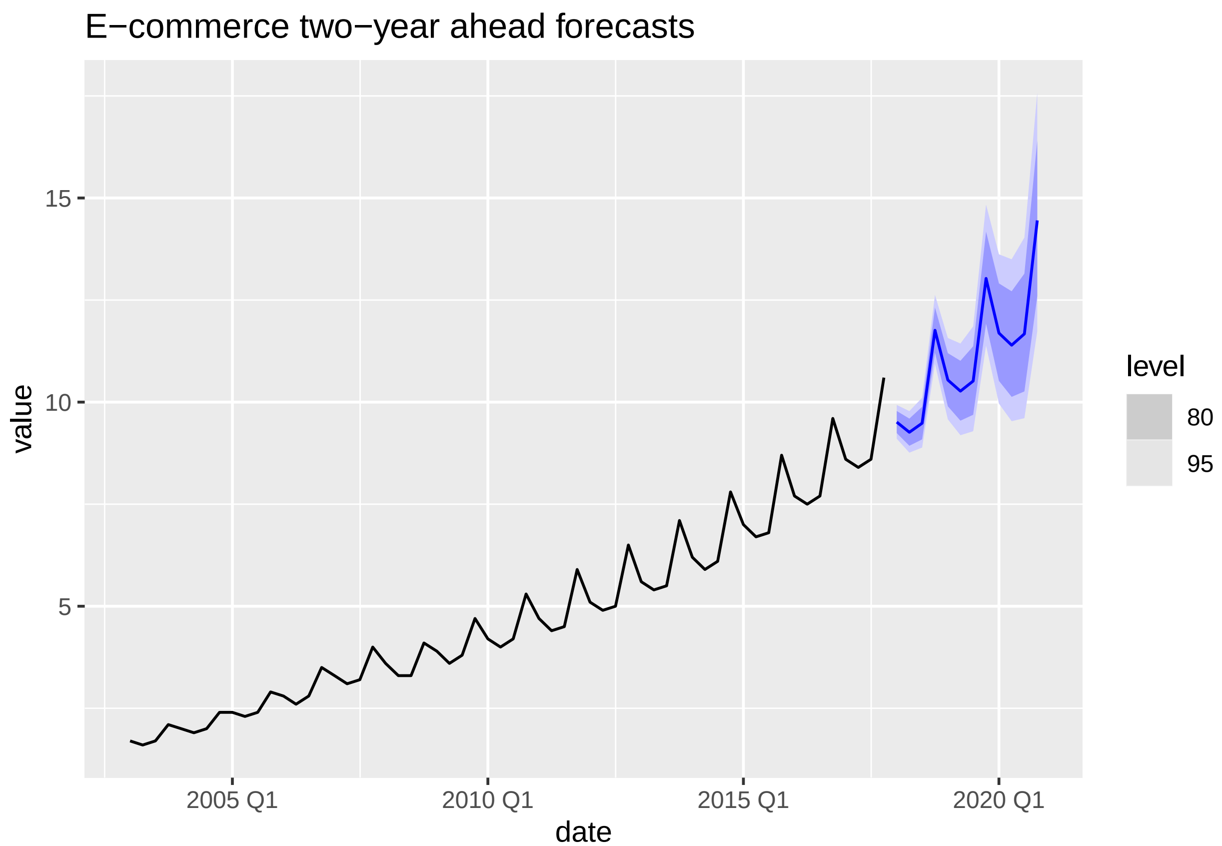



```
models %>% accuracy()
```

```
## # A tibble: 4 x 9
##    .model .type          ME   RMSE    MAE    MPE  MAPE  MASE   ACF1
##    <chr>  <chr>       <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 mod1   Training -0.00559 0.101  0.0767 -0.159  1.69 0.148 -0.302
## 2 mod2   Training -0.00505 0.101  0.0771 -0.140  1.71 0.148 -0.290
## 3 mod3   Training -0.00568 0.101  0.0764 -0.162  1.68 0.147 -0.303
## 4 mod4   Training -0.00651 0.0969 0.0750 -0.203  1.66 0.144 -0.132
```

We found that the third model $ARIMA(0,1,0)(1,1,1)_4$ has a lower AICc and BIC. All four models have their roots within unit circle and have similar AICc, BIC and accuracy measures. The fourth model has a marginally better RMSE, MAE, MAPE, MASE and ACF1. Hence we proceed to forecast using the fourth model.

```
ecom_fit2 <- models %>% dplyr::select(mod4)
ecom_fc2 <- ecom_fit2 %>% forecast(h = "3 years")
ecom_fc2 %>% autoplot(ecom.training) + ggtitle("E-commerce two-year ahead forecasts")
```



We assess Pseudo-out-of-sample performance by applying the accuracy function to the test set.

```
ecom_fc2 %>% accuracy(ecom.test)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as
## 4 observations are missing between 2020 Q1 and 2020 Q4
```

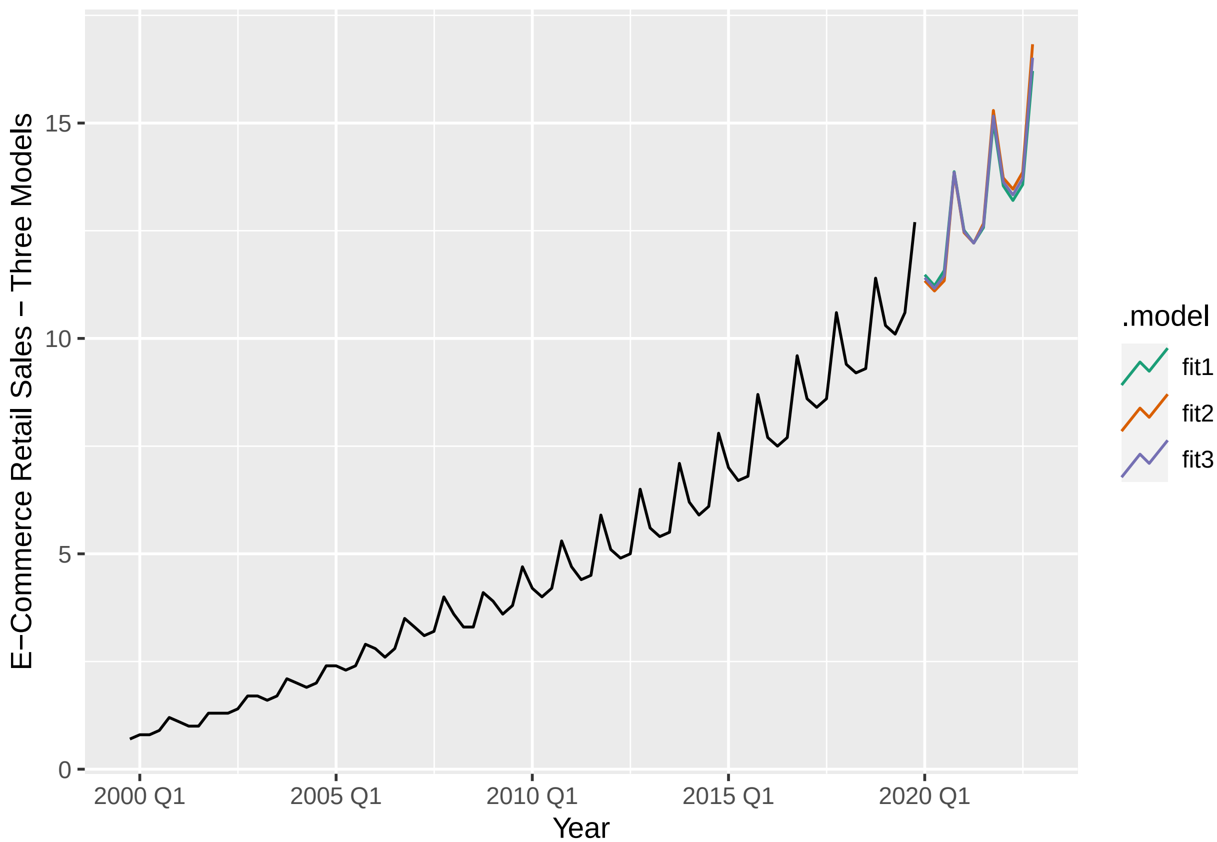
```
## # A tibble: 1 x 9
##   .model .type     ME  RMSE   MAE   MPE  MAPE  MASE   ACF1
##   <chr>  <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 mod4   Test  -0.171 0.218 0.192 -1.60  1.79   NaN -0.132
```

Now we create the third model using a simple average of the selected ARIMA model and Holt-Winters model.

```r
ecom_fit <- ecom %>% model(fit1 = ETS(value ~ error("A") + trend("A") +
    season("M")), fit2 = ARIMA(box_cox(value, lambda = lambda) ~
    pdq(0, 1, 0) + PDQ(1, 1, 1, period = 4))) %>% mutate(fit3 = (fit1 +
    fit2)/2)

ecom_fit %>% accuracy()
```

```
## # A tibble: 3 x 9
##   .model .type          ME  RMSE    MAE    MPE  MAPE  MASE   ACF1
##   <chr>  <chr>       <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 fit1   Training  0.0212  0.111 0.0834  0.232  2.59 0.158  0.153
## 2 fit2   Training -0.0182  0.124 0.0922 -0.881  2.35 0.175 -0.299
## 3 fit3   Training  0.00149 0.108 0.0830 -0.324  2.29 0.158 -0.117
```

```r
ecom_fc <- ecom_fit %>% forecast(h = "3 years")
ecom_fc %>% autoplot(ecom, level = NULL) + xlab("Year") + ylab("E-Commerce Retail Sales - Three
    scale_color_brewer(type = "qual", palette = "Dark2")
```



First we compare the forecasting performance between the Holt-Winters model (fit1) and the ARIMA model (fit2). The two models have similar accuracy measures and forecast values. Some accuracy measures of the two models are in different directions. the ARIMA model has negative ME, MPE and ACF1 whereas the Holt-Winters model has positive values of these measures. In absolute value term, the Holt-Winters model has 5 out of 7 accuracy measures better than the ARIMA model.

The forecast values from the third model (fit3) are the average of the first two models. The model's accuracy measures are not neccessarily the average of those of the first two models. Its RMSE, MAE, MAPE, MASE and ACF1 in absolute values are smaller than those of the first two models. In addition, its ME, MPE and ACF1 are close to the average of those of the first two models since ME, MPE and ACF1 of the first models are in different directions and somewhat offset each other.

Therefore, the average model (fit3) is the best model.

Question 5 (2 points):

Vector autoregression

Annual values for real mortgage credit (RMC), real consumer credit (RCC) and real disposable personal income (RDPI) for the period 1946-2006 are recorded in `Q5.csv`. All of the observations are measured in billions of dollars, after adjustment by the Consumer Price Index (CPI). Conduct an EDA on these data 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.

```r
args(VAR)
```

```
## function (y, p = 1, type = c("const", "trend", "both", "none"),
##     season = NULL, exogen = NULL, lag.max = NULL, ic = c("AIC",
##         "HQ", "SC", "FPE"))
## NULL
```

```r
args(VARselect)
```

```
## function (y, lag.max = 10, type = c("const", "trend", "both",
##     "none"), season = NULL, exogen = NULL)
## NULL
```

```r
# Load data
econ <- read.csv("Q5.csv")
econ <- ts(econ[, 2:4], start = c(1946, 1), end = c(2006, 1),
    frequency = 1)

# Inspect data
str(econ)
```

```
##  Time-Series [1:61, 1:3] from 1946 to 2006: 118 126 137 157 187 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "RMC" "RCC" "RDPI"
```

```r
head(econ)
```

```
## Time Series:
## Start = 1946
## End = 1951
## Frequency = 1
##         RMC  RCC  RDPI
## 1946 117.9 49.7 826.2
```

```
## 1947 126.0 59.2 767.7
## 1948 137.3 67.6 790.9
## 1949 157.1 81.5 800.0
## 1950 186.7 99.2 871.8
## 1951 198.8 97.7 888.5
```

```r
tail(econ)
```

```
## Time Series:
## Start = 2001
## End = 2006
## Frequency = 1
##         RMC    RCC   RDPI
## 2001 2985.0 1072.6 4227.4
## 2002 3317.3 1118.5 4352.5
## 2003 3709.0 1150.1 4436.1
## 2004 4133.6 1181.7 4595.9
## 2005 4548.5 1191.2 4626.8
## 2006 4799.5 1209.2 4723.8
```

```r
# Aggregate statistics
summary(econ)
```

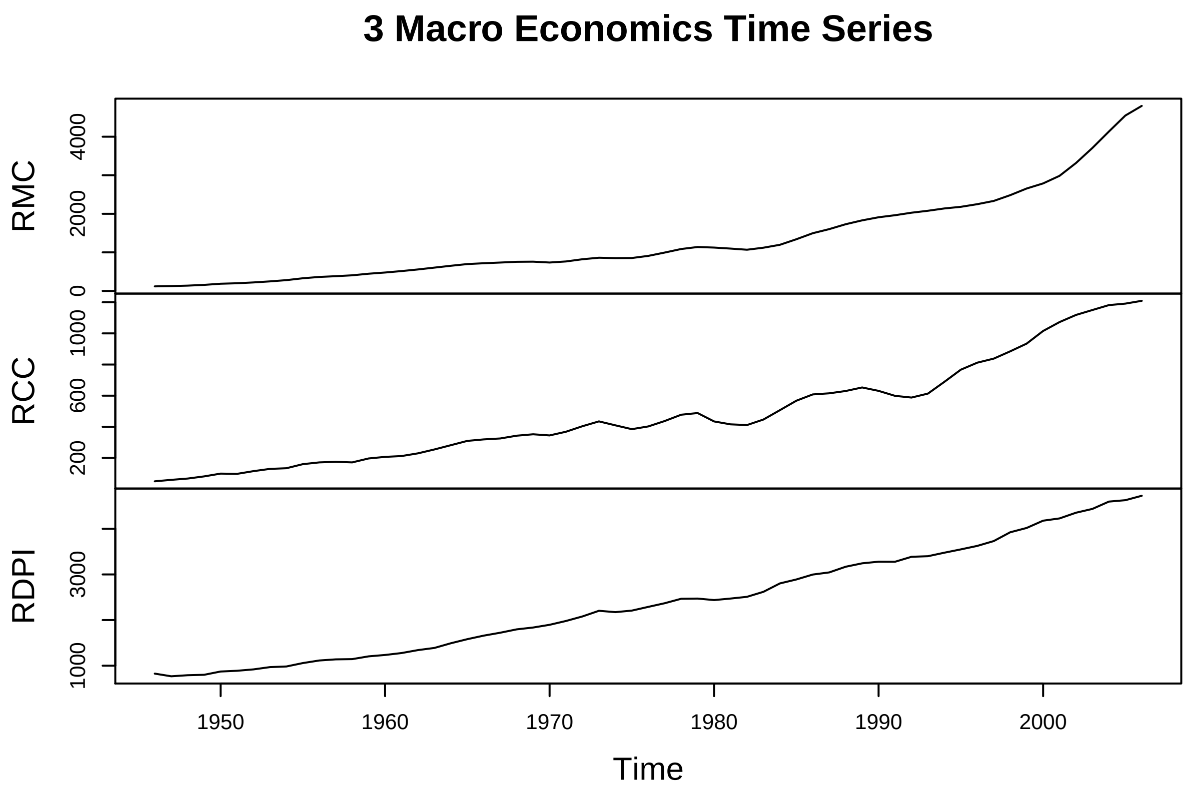
```
##       RMC               RCC               RDPI
##  Min.   : 117.9   Min.   :  49.7   Min.   : 767.7
##  1st Qu.: 515.1   1st Qu.: 212.0   1st Qu.:1276.9
##  Median : 908.6   Median : 411.1   Median :2289.1
##  Mean   :1327.9   Mean   : 479.9   Mean   :2392.2
##  3rd Qu.:1963.5   3rd Qu.: 629.9   3rd Qu.:3279.1
##  Max.   :4799.5   Max.   :1209.2   Max.   :4723.8
```

```r
# No missing data
```

Since the data is alredy "cleaned" and is stored in a time series object, we can proceed to EDA

The data contains three annual macroeconomic times series, real mortgage credit (RMC), real consumer credit (RCC) and real disposable personal income (RDPI), for the period 1946-2006. All three times series have a similar upward trend, but RDPI has a more linear trend whereas RCC and RMC have a more exponential trend.

```r
plot.ts(econ, main = "3 Macro Economics Time Series")
```




```r
tsplot <- function(series) {
    autoplot(series)
}
for (k in 1:ncol(econ)) {
    tsplot(econ[, k])
}
```

Figure 5 shows that the three time series have almost perfect postive correlation with each other.

Looking at Figures 6 - 8, we observed that the three time series share similar characteristics of the histogram, ACF and PACF. Each time series is concentrated on the left side and does not follow a Normal distribution. The ACF gradually drops and becomes insignificant after lag 10 while the PACF sharply cuts off after lag 1.

Figures 9 - 11 show that the three cross-correlograms have a mountain shape with peak at lag 0. These indicate that the most dominant cross correlations amongst the three variables occurs at lag 0.

```r
# Scatterplot Matrix, which displays the contemporaneous
# correlation
scatterplotMatrix(~econ[, 1] + econ[, 2] + econ[, 3])
title("Contemporaneous Correlation of the 3 Macroeconomic Series ")
```

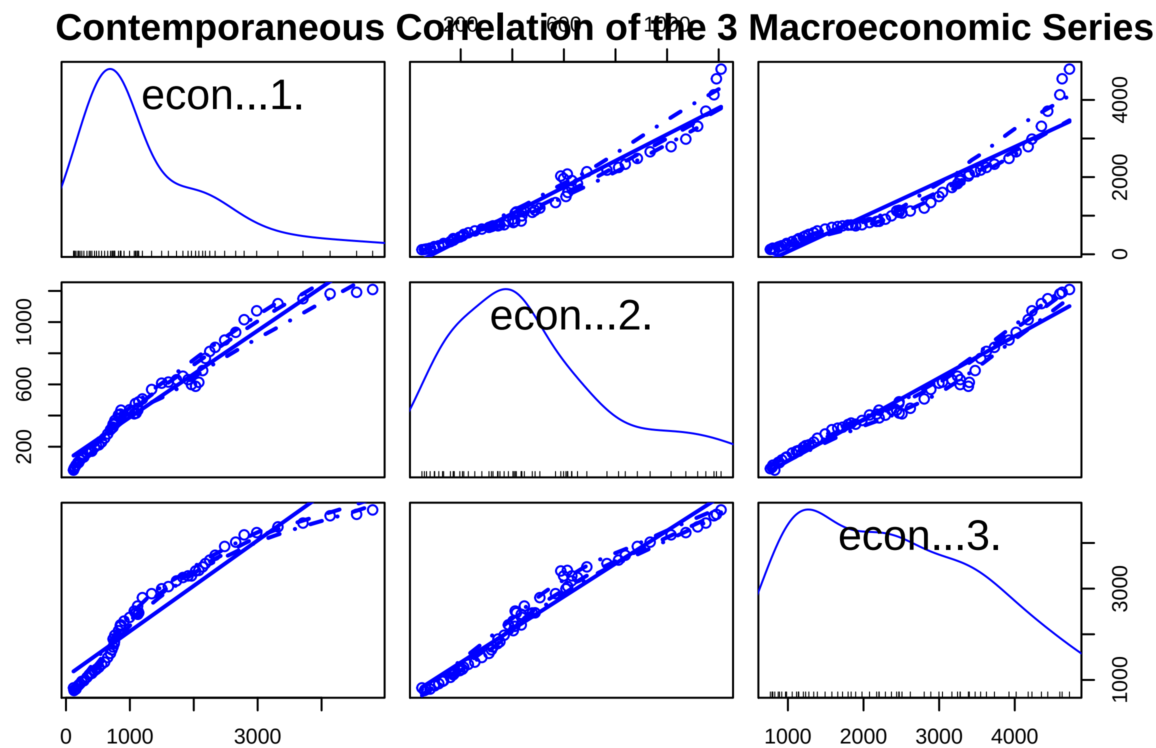


Figure 5: Contemporaneous correlation of the macroeconomic time series

```r
# Time series plot, ACF and PACF of each of the individual
# series

tsplot <- function(series, title) {
    par(mfrow = c(2, 2))
    hist(series, main = "")
    title(title)
    plot.ts(series, main = "")
    title(title)
    acf(series, main = "")
    title(paste("ACF", title))
    pacf(series, main = "")
    title(paste("PACF", title))
}
tsplot(econ[, 1], "Real Mortage Credit")
```

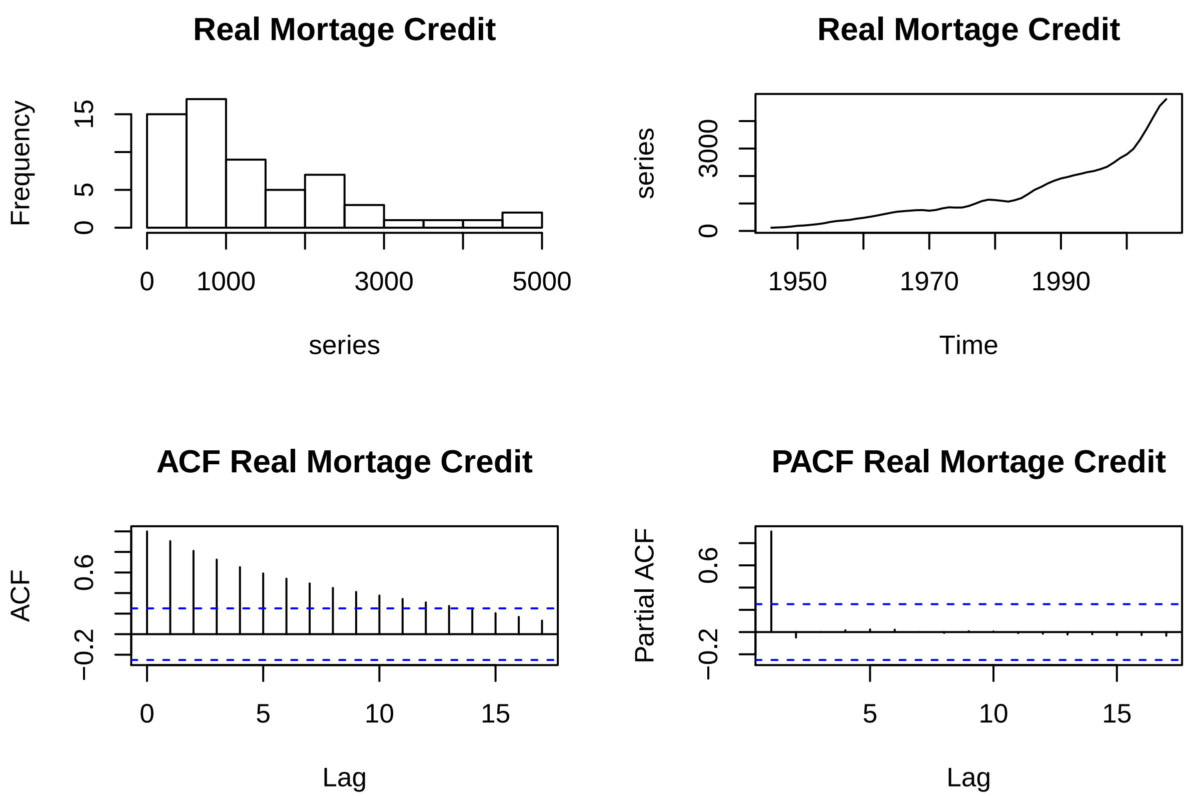


Figure 6: Time series plot, ACF and PACF of each of the macroeconomic time series

```r
tsplot(econ[, 2], "Real Consumer Credit")
```

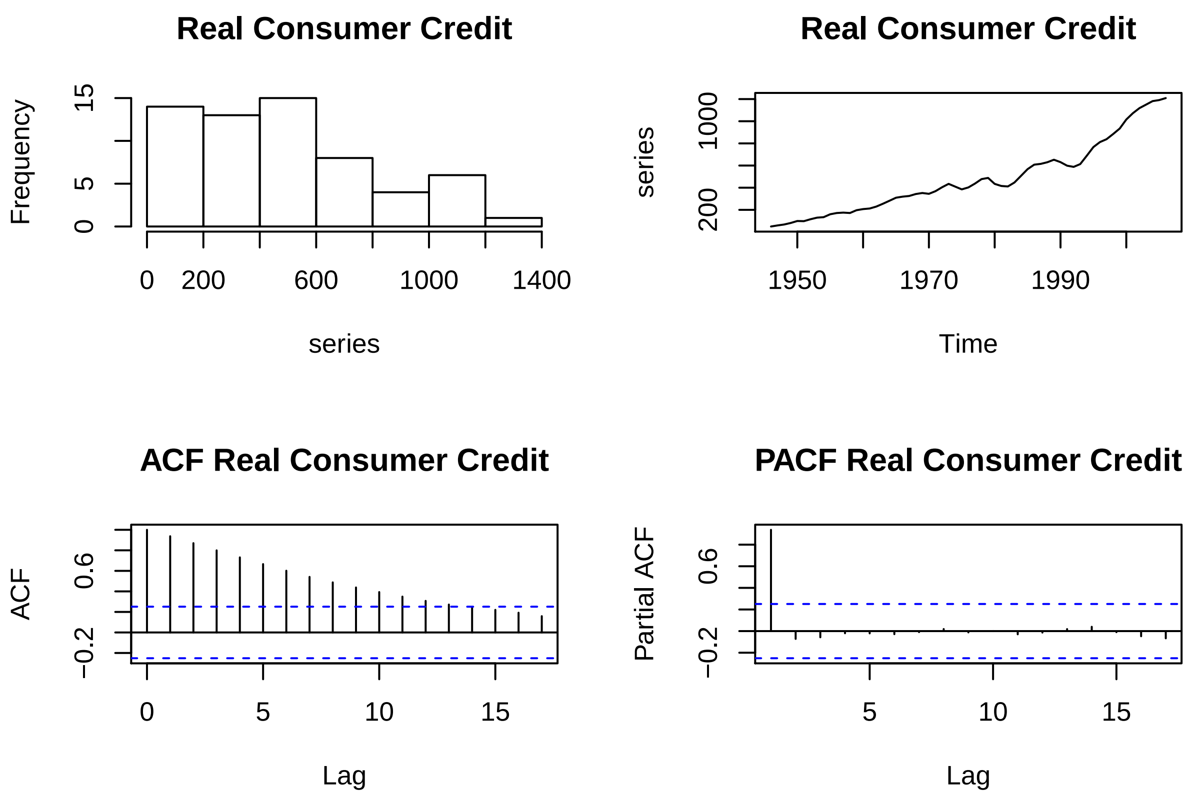


Figure 7: Time series plot, ACF and PACF of each of the macroeconomic time series

```r
tsplot(econ[, 3], "Real Disposable Personal Income")
```

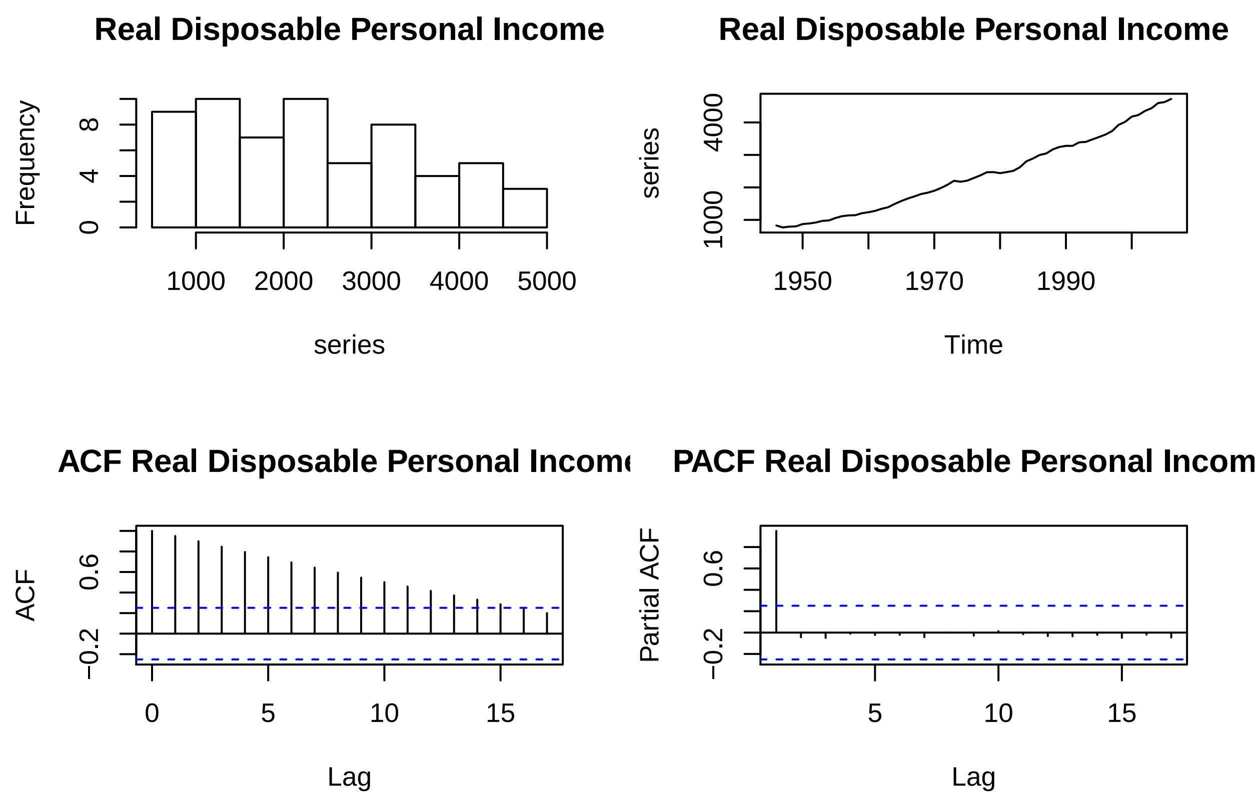


Figure 8: Time series plot, ACF and PACF of each of the macroeconomic time series

```r
# Correlation and Cross-correlation
par(mfrow = c(1, 1))

corrfunc <- function(series1, series2) {
    cat("Correlation Matrix: ", cor(series1, series2))
    ccf(series1, series2)
}

for (i in 1:3) {
    for (j in 1:3) {
        if (i != j & j > i) {
            corrfunc(econ[, i], econ[, j])
        }
    }
}
```

```
## Correlation Matrix:  0.9756163
```

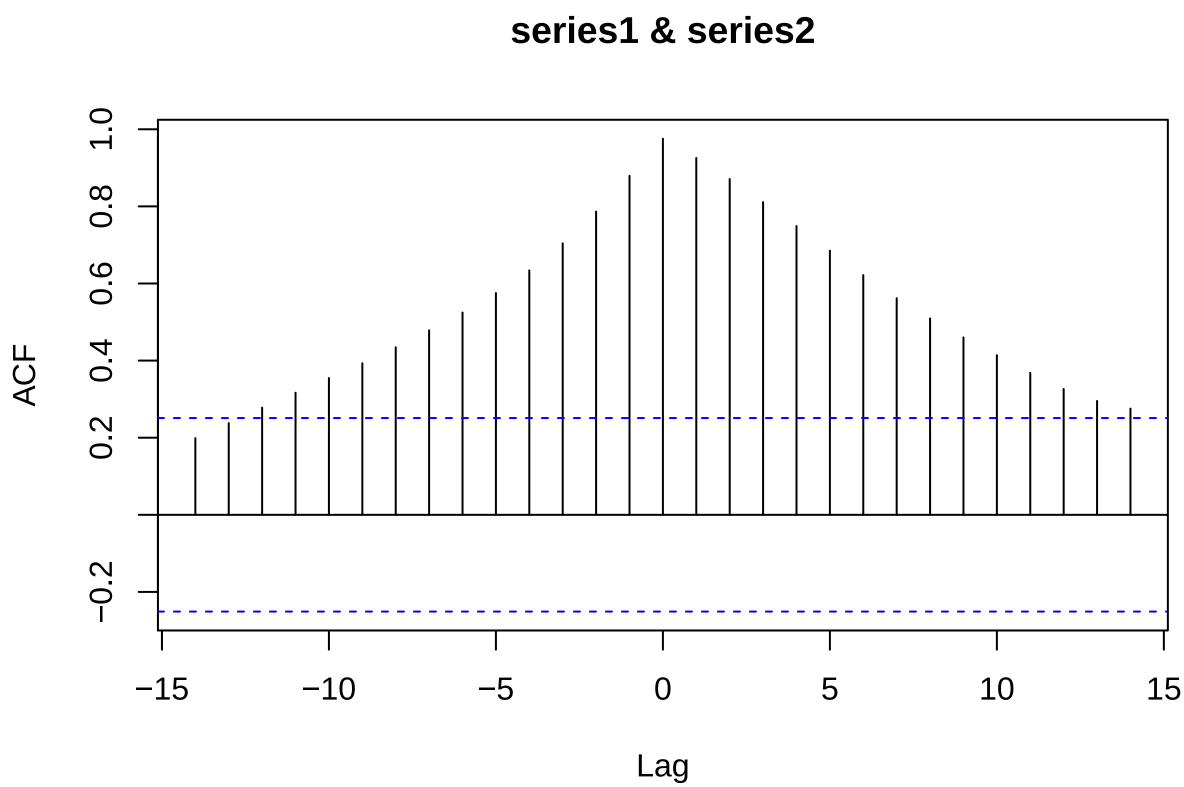


Figure 9: Correlation and Cross-correlation

```
## Correlation Matrix:  0.9491886
```

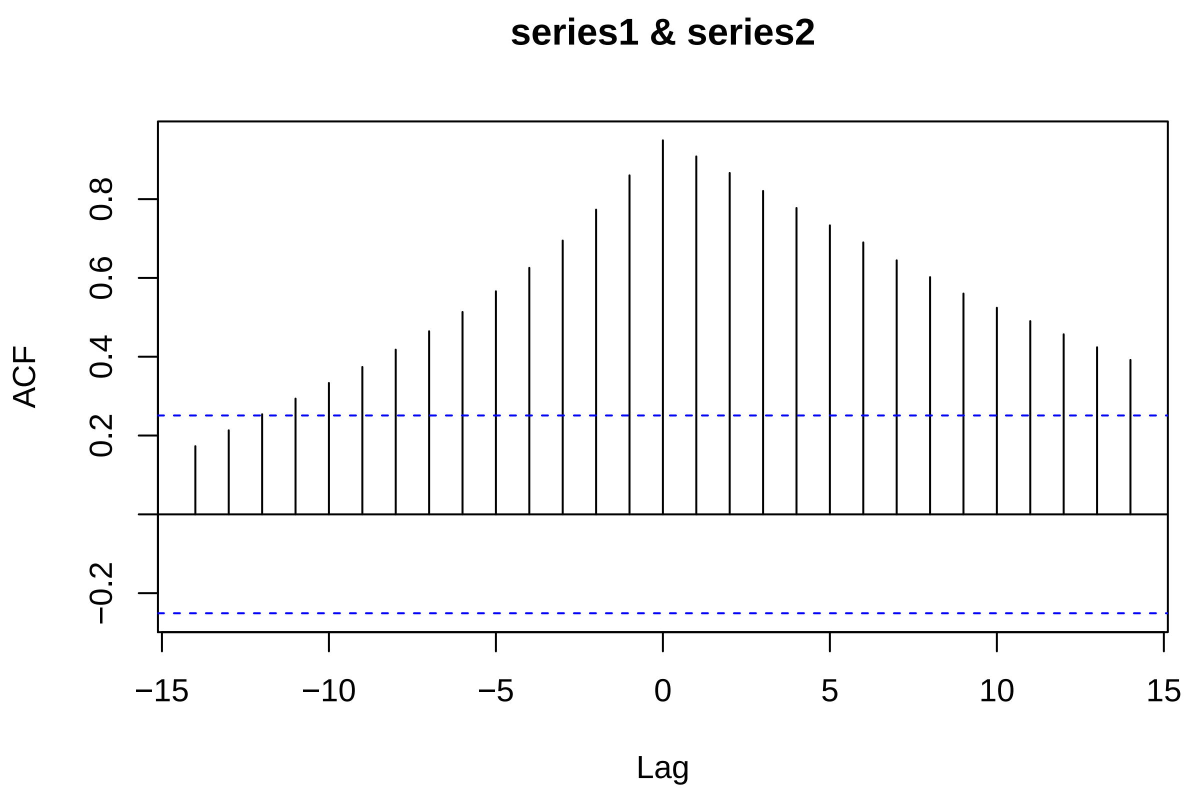


Figure 10: Correlation and Cross-correlation

```
## Correlation Matrix:  0.9815842
```

```r
# Split the data into training and test sets
econ.training <- window(econ, start = c(1946, 1), end = c(2003,
    1))
econ.test <- window(econ, start = c(2004, 1), end = c(2006, 1))
```

Select optimal number of lags

```r
VARselect(diff(econ.training), lag.max = 8, type = "both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      8      2      1      8
```

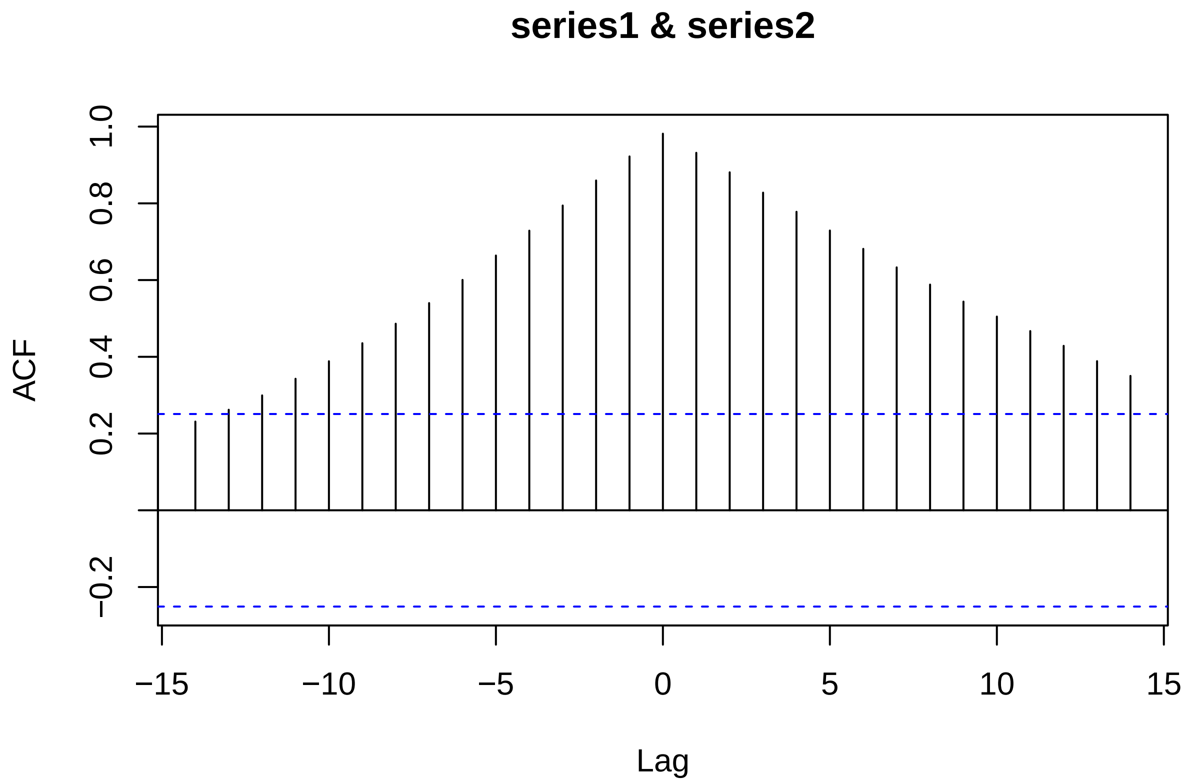


Figure 11: Correlation and Cross-correlation

```
##
## $criteria
##                 1            2            3            4            5
## AIC(n) 2.051498e+01 2.017357e+01 2.029174e+01 2.039790e+01 2.061409e+01
## HQ(n)  2.073470e+01 2.052512e+01 2.077512e+01 2.101311e+01 2.136114e+01
## SC(n)  2.109411e+01 2.110017e+01 2.156582e+01 2.201946e+01 2.258313e+01
## FPE(n) 8.137154e+08 5.822507e+08 6.648507e+08 7.585881e+08 9.811272e+08
##                 6            7            8
## AIC(n) 2.067009e+01 2.038718e+01 1.962541e+01
## HQ(n)  2.154897e+01 2.139789e+01 2.076795e+01
## SC(n)  2.298661e+01 2.305117e+01 2.263688e+01
## FPE(n) 1.103269e+09 9.078251e+08 4.792414e+08
```

Based on SC, we select VAR(1) to be the best model.

```r
econ.fit1 <- VAR(diff(econ.training), p = 1, type = "both")
summary(econ.fit1)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: RMC, RCC, RDPI
## Deterministic variables: both
## Sample size: 56
## Log Likelihood: -788.365
## Roots of the characteristic polynomial:
## 0.9672 0.6396 0.01387
## Call:
## VAR(y = diff(econ.training), p = 1, type = "both")
##
##
## Estimation results for equation RMC:
## ====================================
## RMC = RMC.l1 + RCC.l1 + RDPI.l1 + const + trend
##
##         Estimate Std. Error t value Pr(>|t|)
## RMC.l1   0.99427    0.11072   8.980 4.43e-12 ***
## RCC.l1   0.16706    0.24695   0.676    0.502
## RDPI.l1 -0.07555    0.13600  -0.555    0.581
## const   -7.48845   10.35627  -0.723    0.473
## trend    0.55032    0.37612   1.463    0.150
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 35.2 on 51 degrees of freedom
## Multiple R-Squared: 0.8017,  Adjusted R-squared: 0.7861
```

```
## F-statistic: 51.54 on 4 and 51 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation RCC:
## ====================================
## RCC = RMC.l1 + RCC.l1 + RDPI.l1 + const + trend
##
##         Estimate Std. Error t value Pr(>|t|)
## RMC.l1  -0.07310    0.06748  -1.083 0.283831
## RCC.l1   0.59866    0.15052   3.977 0.000221 ***
## RDPI.l1  0.02898    0.08289   0.350 0.728031
## const   -0.67322    6.31206  -0.107 0.915481
## trend    0.37555    0.22924   1.638 0.107527
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 21.45 on 51 degrees of freedom
## Multiple R-Squared: 0.4195,  Adjusted R-squared: 0.374
## F-statistic: 9.213 on 4 and 51 DF,  p-value: 1.11e-05
##
##
## Estimation results for equation RDPI:
## =====================================
## RDPI = RMC.l1 + RCC.l1 + RDPI.l1 + const + trend
##
##         Estimate Std. Error t value Pr(>|t|)
## RMC.l1  -0.03294    0.13549  -0.243   0.8089
## RCC.l1   0.28531    0.30220   0.944   0.3496
## RDPI.l1  0.02779    0.16642   0.167   0.8681
## const   29.49591   12.67329   2.327   0.0240 *
## trend    1.04062    0.46027   2.261   0.0281 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 43.07 on 51 degrees of freedom
## Multiple R-Squared: 0.1962,  Adjusted R-squared: 0.1331
## F-statistic: 3.111 on 4 and 51 DF,  p-value: 0.02292
##
##
##
## Covariance matrix of residuals:
##         RMC   RCC   RDPI
## RMC  1238.8 359.4  684.4
## RCC   359.4 460.2  602.4
## RDPI  684.4 602.4 1855.1
##
```

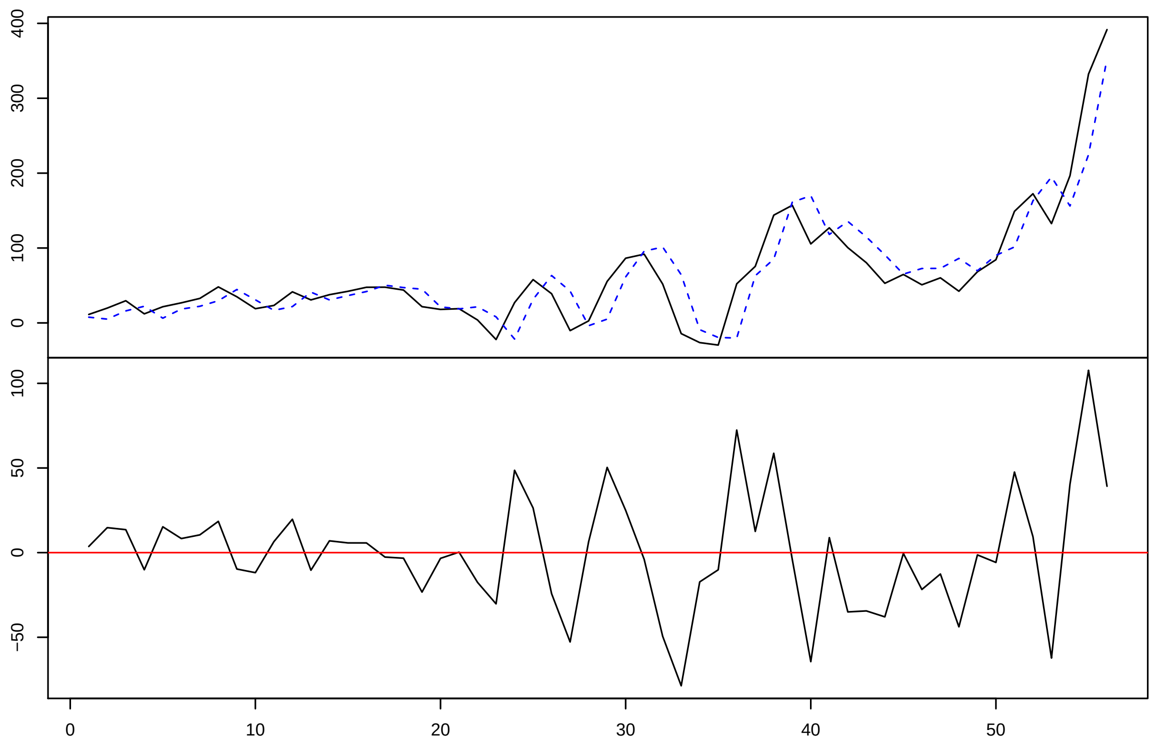
```
## Correlation matrix of residuals:
##         RMC    RCC   RDPI
## RMC  1.0000 0.4761 0.4515
## RCC  0.4761 1.0000 0.6520
## RDPI 0.4515 0.6520 1.0000
```

```r
names(econ.fit1)
```

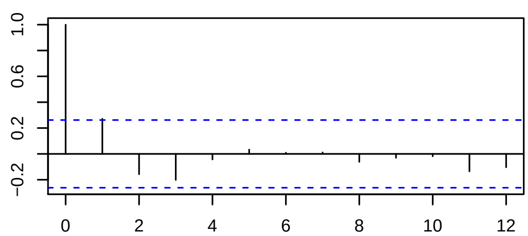
```
##  [1] "varresult"    "datamat"      "y"            "type"         "p"
##  [6] "K"            "obs"          "totobs"       "restrictions" "call"
```

```r
par(mar = rep(2, 4))
plot(econ.fit1)
```

Diagram of fit and residuals for RMC



ACF Residuals



PACF Residuals

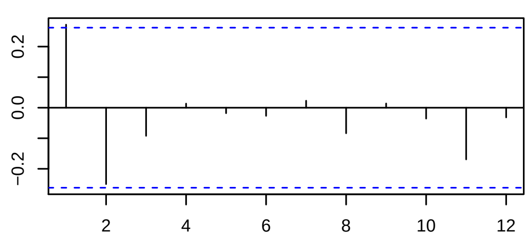
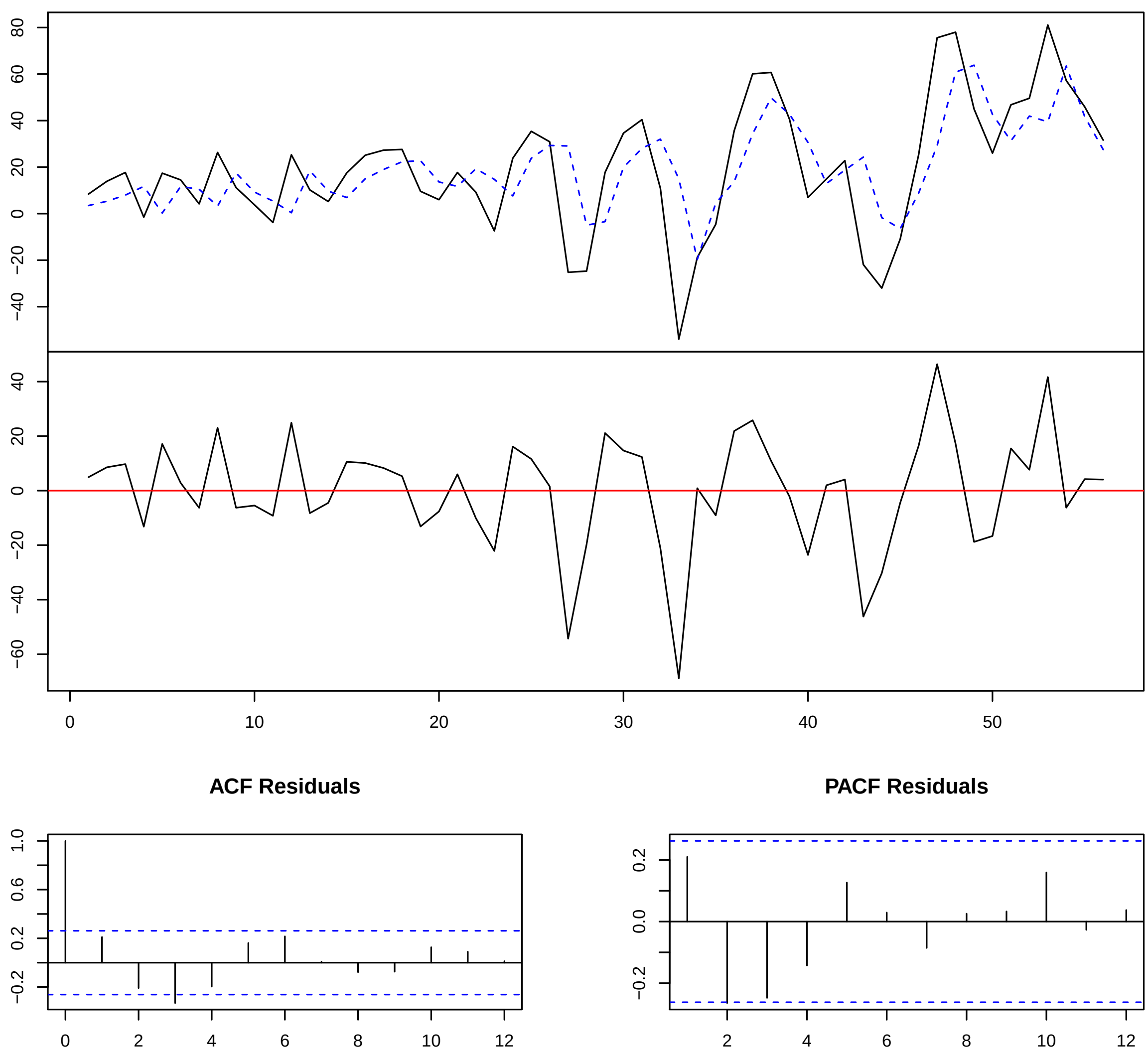
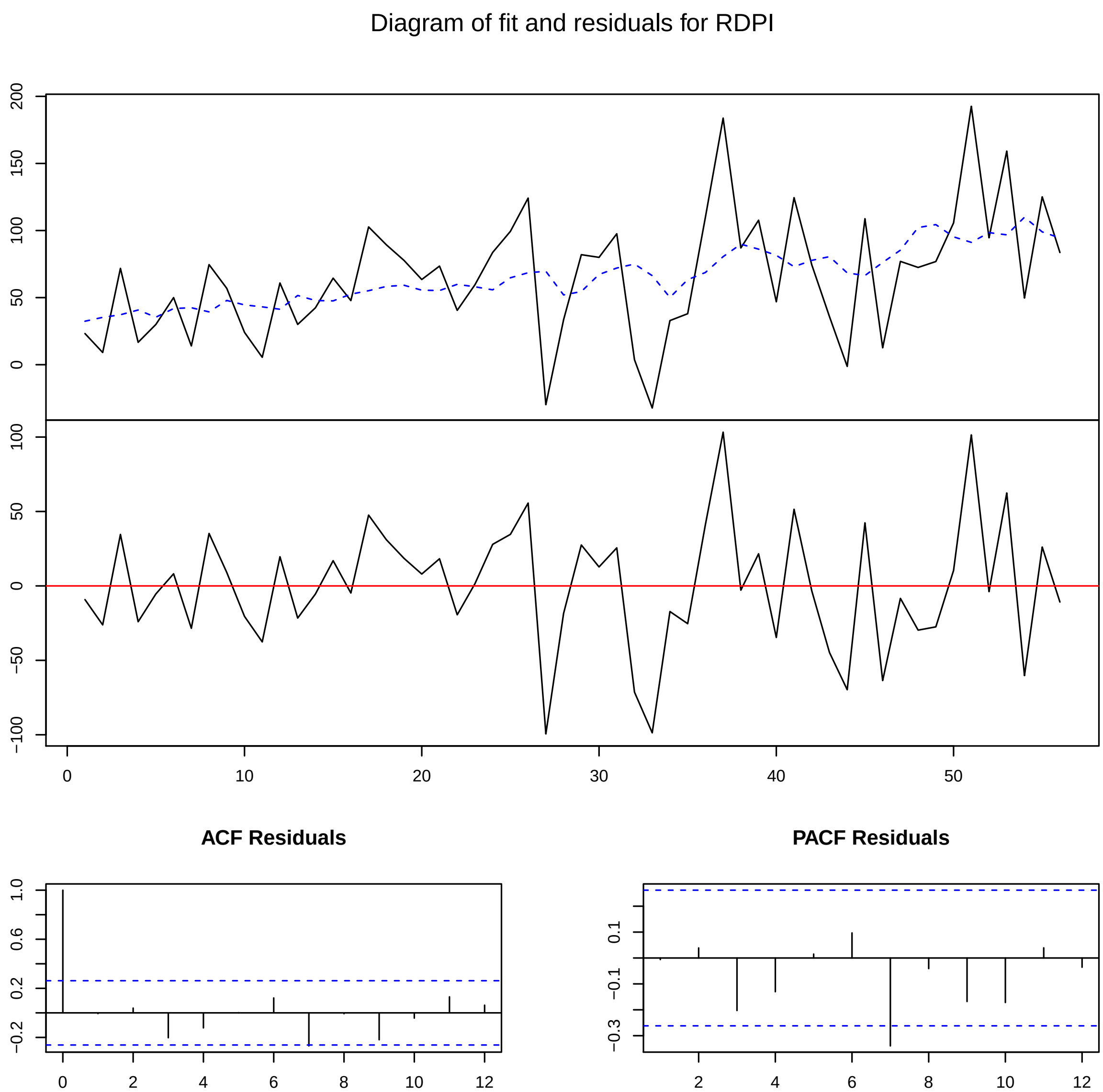


Diagram of fit and residuals for RCC






```r
roots(econ.fit1)
```

```
## [1] 0.96723533 0.63960604 0.01387363
```

The fitted values are fairly close to the actual values but somewhat lag behind the actual values by about 1 year. All roots of the model are within unity. The ACF and PACF do not present significance after log 0. However, we can see the residuals are not stable from the above residual plot vs time. the residual variance increases as time passes. This indicates that the use of difference on the macroeconomic data can not stablize the residual variance.

Select optimal number of lags for the log-transformed data

```r
VARselect(log(econ.training), lag.max = 8, type = "both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      3      2      3
##
## $criteria
##                     1             2             3             4             5
## AIC(n) -2.143489e+01 -2.233391e+01 -2.255731e+01 -2.233523e+01 -2.219648e+01
## HQ(n)  -2.121645e+01 -2.198442e+01 -2.207676e+01 -2.172362e+01 -2.145381e+01
## SC(n)  -2.086128e+01 -2.141614e+01 -2.129537e+01 -2.072913e+01 -2.024622e+01
## FPE(n)  4.918374e-10  2.014276e-10  1.633100e-10  2.089052e-10  2.494094e-10
##                     6             7             8
## AIC(n) -2.208235e+01 -2.215229e+01 -2.227366e+01
## HQ(n)  -2.120862e+01 -2.114750e+01 -2.113781e+01
## SC(n)  -1.978792e+01 -1.951370e+01 -1.929091e+01
## FPE(n)  2.960607e-10  2.996453e-10  2.974946e-10
```

Based on SC, we select VAR(2) to be the best model for the log-transformed data.

```r
econ.fit2 <- VAR(log(econ.training), p = 2, type = "both")
summary(econ.fit2)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: RMC, RCC, RDPI
## Deterministic variables: both
## Sample size: 56
## Log Likelihood: 399.378
## Roots of the characteristic polynomial:
## 0.9104 0.9104 0.8034 0.8034 0.2637 0.1022
## Call:
## VAR(y = log(econ.training), 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   1.474590   0.155961   9.455 1.54e-12 ***
## RCC.l1  -0.012961   0.107042  -0.121 0.904129
```

```
## RDPI.l1 -0.219918   0.244412  -0.900 0.372727
## RMC.l2  -0.597011   0.142006  -4.204 0.000114 ***
## RCC.l2   0.097296   0.092093   1.056 0.296029
## RDPI.l2  0.006335   0.219556   0.029 0.977099
## const    1.714383   0.622538   2.754 0.008293 **
## trend    0.008861   0.002760   3.211 0.002363 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.02889 on 48 degrees of freedom
## Multiple R-Squared: 0.9989,  Adjusted R-squared: 0.9988
## F-statistic:  6502 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  -0.208760   0.298524  -0.699   0.4877
## RCC.l1   1.143391   0.204889   5.581 1.09e-06 ***
## RDPI.l1  0.133284   0.467829   0.285   0.7769
## RMC.l2   0.235405   0.271814   0.866   0.3908
## RCC.l2  -0.269670   0.176274  -1.530   0.1326
## RDPI.l2 -0.322939   0.420252  -0.768   0.4460
## const    1.769765   1.191599   1.485   0.1440
## trend    0.009015   0.005282   1.707   0.0943 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0553 on 48 degrees of freedom
## Multiple R-Squared: 0.9946,  Adjusted R-squared: 0.9939
## F-statistic:  1272 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  -0.029665   0.108427  -0.274    0.786
## RCC.l1   0.092244   0.074418   1.240    0.221
## RDPI.l1  0.823465   0.169921   4.846 1.36e-05 ***
## RMC.l2   0.023460   0.098726   0.238    0.813
## RCC.l2  -0.043651   0.064025  -0.682    0.499
## RDPI.l2  0.048943   0.152640   0.321    0.750
```

```
## const    0.701800   0.432802   1.622    0.111
## trend    0.001874   0.001919   0.977    0.334
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.02009 on 48 degrees of freedom
## Multiple R-Squared: 0.9986,  Adjusted R-squared: 0.9985
## F-statistic:  5063 on 7 and 48 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##            RMC       RCC      RDPI
## RMC  0.0008347 0.0012198 0.0003191
## RCC  0.0012198 0.0030582 0.0008056
## RDPI 0.0003191 0.0008056 0.0004035
##
## Correlation matrix of residuals:
##         RMC    RCC   RDPI
## RMC  1.0000 0.7635 0.5500
## RCC  0.7635 1.0000 0.7252
## RDPI 0.5500 0.7252 1.0000
```

```
names(econ.fit2)
```

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

```
plot(econ.fit2)
```

Diagram of fit and residuals for RMC

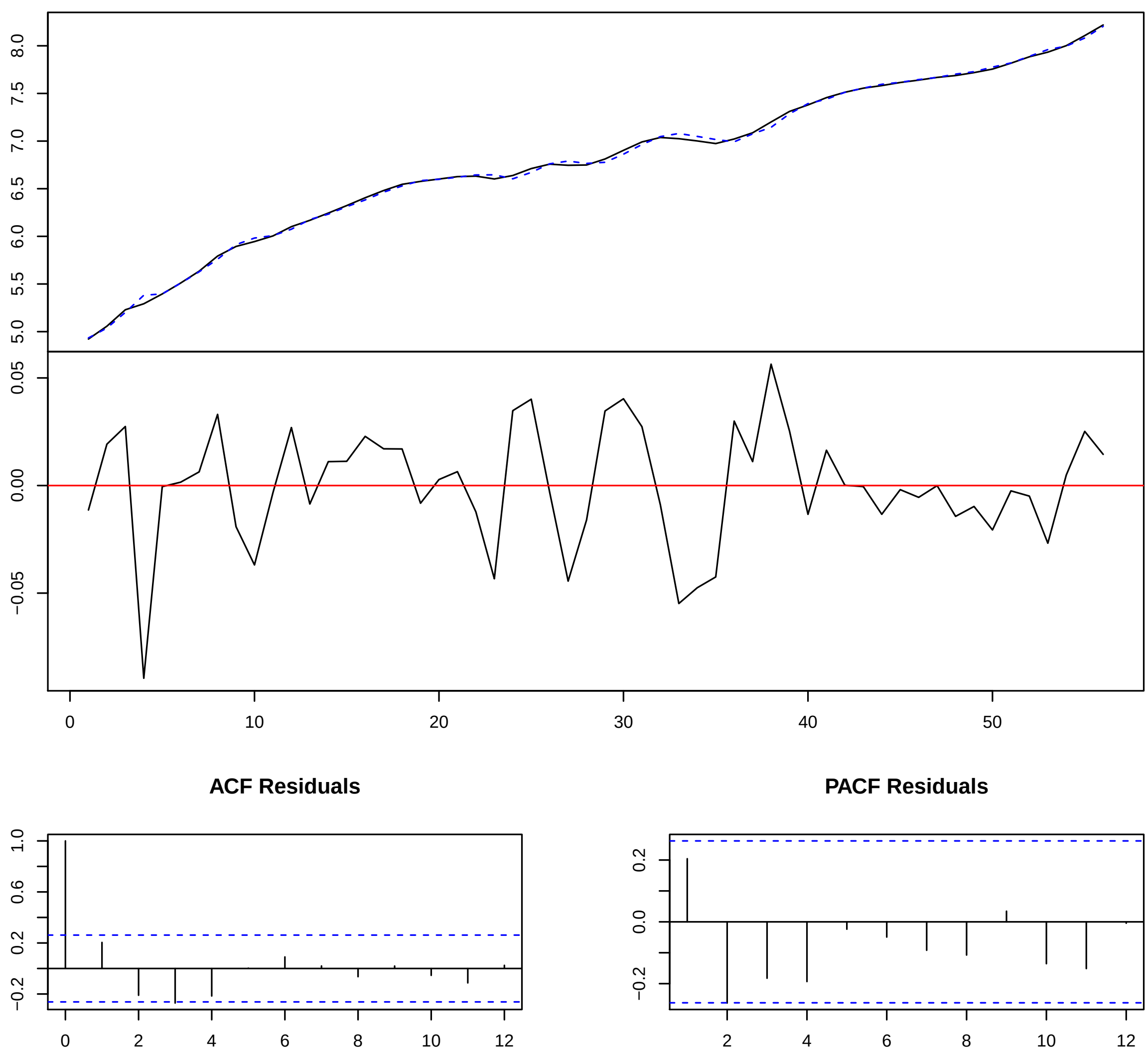
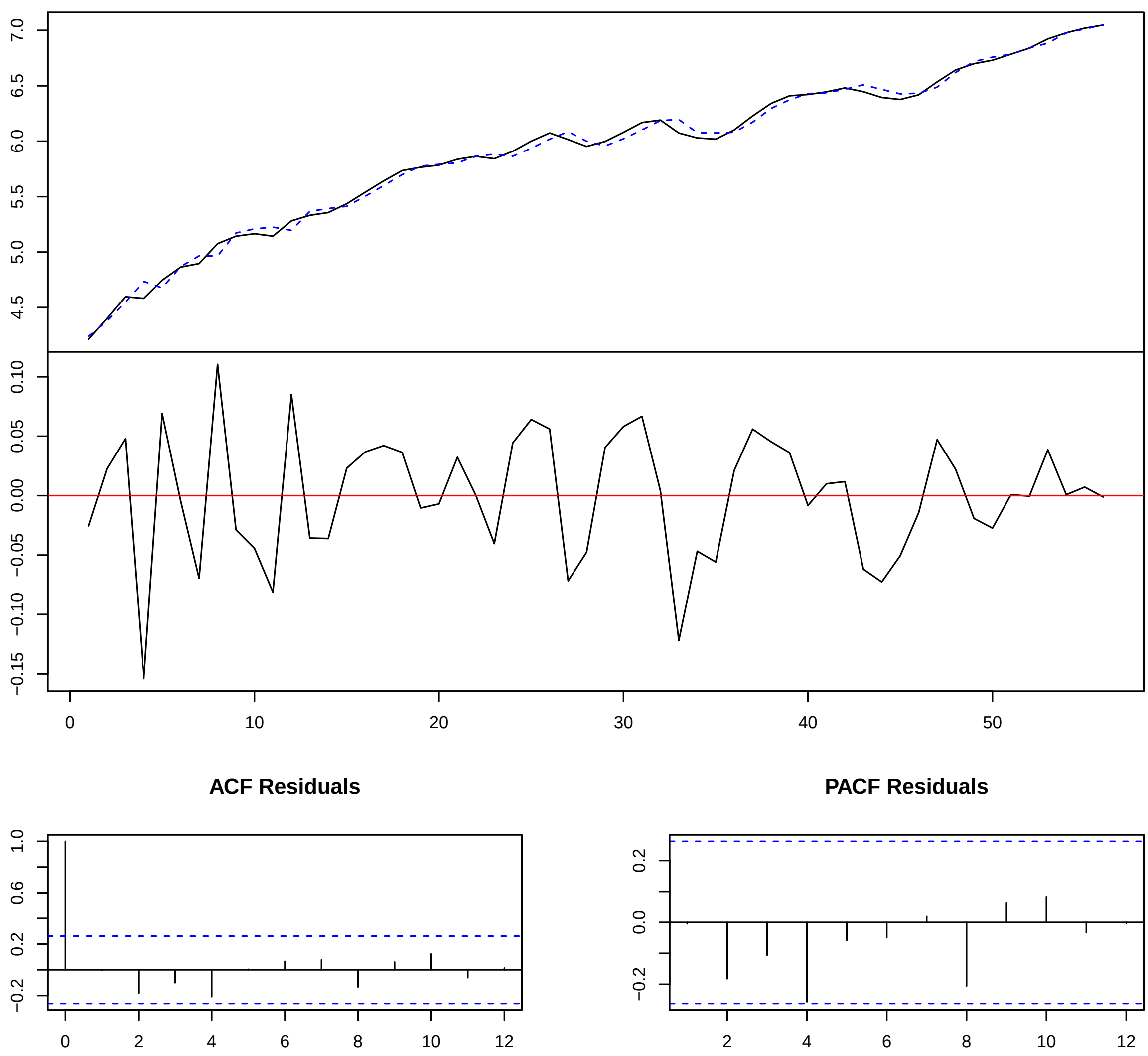
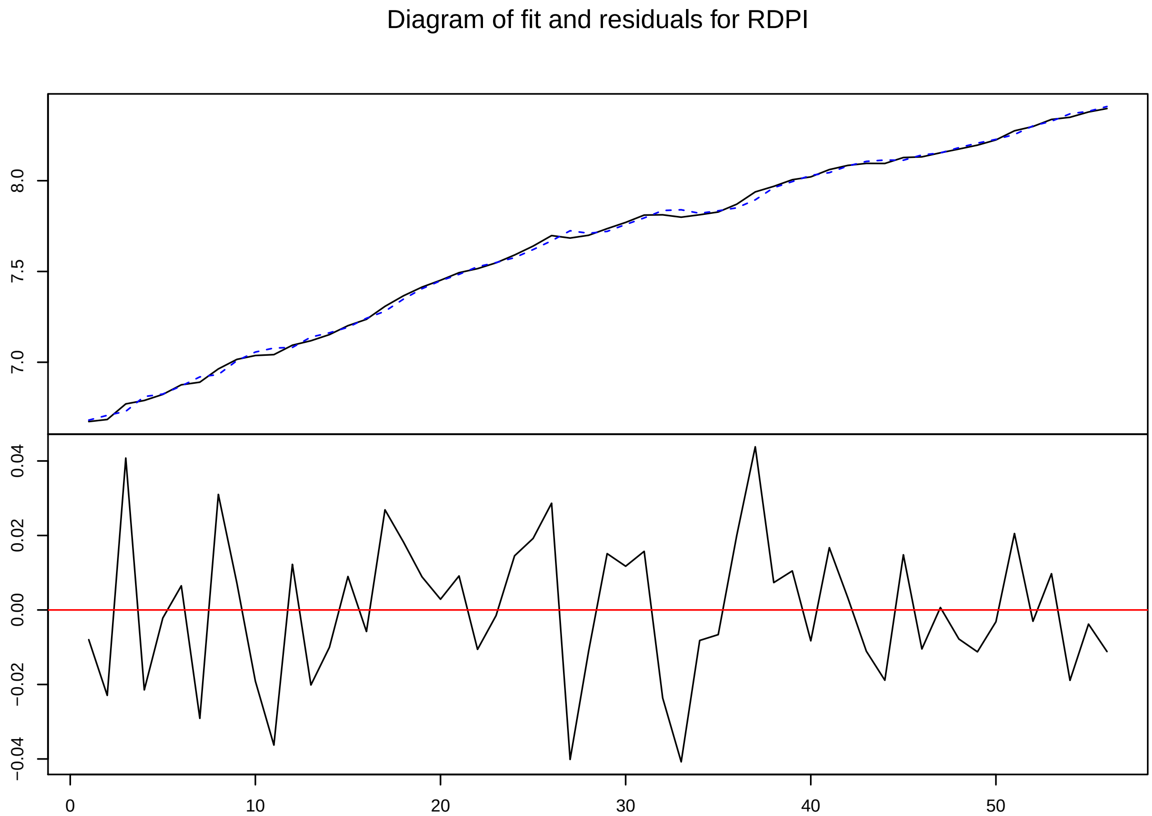
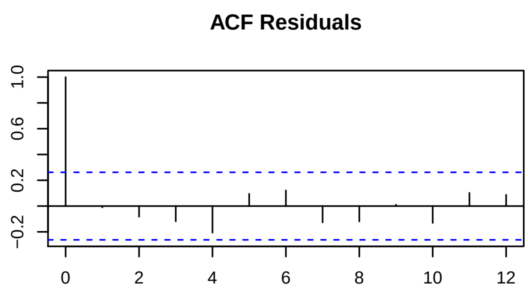
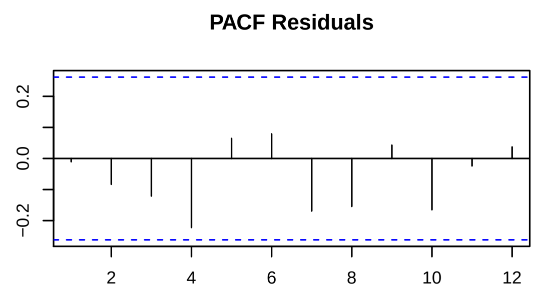


Diagram of fit and residuals for RCC










```r
roots(econ.fit2)
```

```
## [1] 0.9103601 0.9103601 0.8033969 0.8033969 0.2637261 0.1021632
```

The fitted values from this model are closer to the actual values than the values provided by the previous model using the original data. All roots of the model are within unity. The ACF and PACF do not present significance after log 0. We can see the residual variance is now stable across time period and homoscedasticity presents. This indicated the model is a better fit than the first model because the log transformation can stablize the residual variance whereas the use of difference cannot.

Diagnostic Testing for model 1

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

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

```r
econ.fit1.norm
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object econ.fit1
## Chi-squared = 17.488, df = 6, p-value = 0.007647
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object econ.fit1
## Chi-squared = 3.4219, df = 3, p-value = 0.331
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object econ.fit1
## Chi-squared = 14.066, df = 3, p-value = 0.002816
```

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

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object econ.fit1
## Chi-squared = 119.35, df = 99, p-value = 0.08012
```

```r
# plot(econ.fit1.ptasy)

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

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

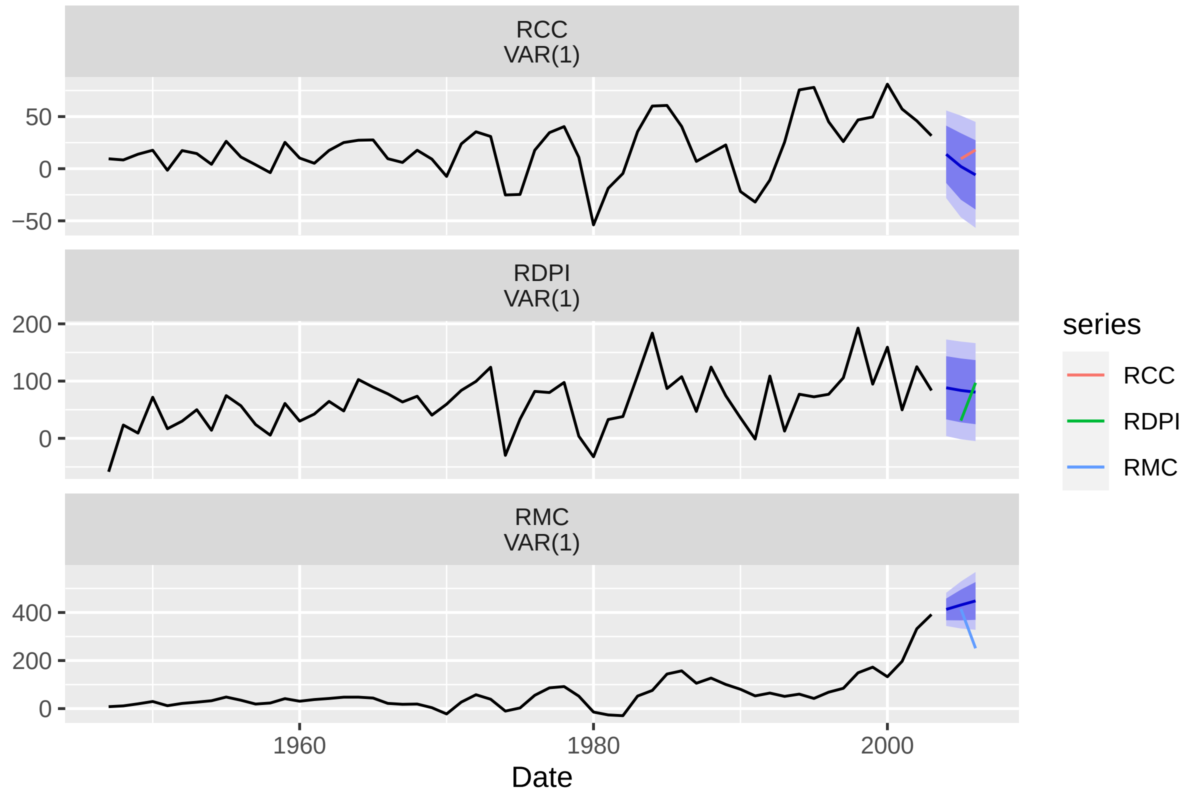
```r
econ.fit1.arch
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object econ.fit1
## Chi-squared = 256.75, df = 180, p-value = 0.00015
```

Forecast by model 1

```r
econ_fc1 <- forecast(econ.fit1, h = 3)
econ_fc1 %>% autoplot() + autolayer(diff(econ.test)) + xlab("Date")
```

```
## For a multivariate timeseries, specify a seriesname for each timeseries. Defaulting to colu
```



From the results of the diagnostic testing for model 1, we know that:

- JB test: we can conclude that the data are not from a normal distribution as the null hypothesis is rejected. The skewness being zero is not rejected as the p-value is large, but the excess kurtosis being zero is rejected.
- Portmanteau test: The null hypothesis is not rejected that no residual autocorrelations.
- ARCH test: the null hypothesis is not rejected and thus the multivariate time series is homoscedastic.

The forecast values are quite far away from the actual values and do not even present a same trend as the actual values.

Diagnostic Testing for model 2

```r
# Test of normality:
econ.fit2.norm <- normality.test(econ.fit2, multivariate.only = TRUE)
names(econ.fit2.norm)
```

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

```r
econ.fit2.norm
```

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object econ.fit2
## Chi-squared = 7.5677, df = 6, p-value = 0.2715
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object econ.fit2
## Chi-squared = 4.8739, df = 3, p-value = 0.1813
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
##
## data:  Residuals of VAR object econ.fit2
## Chi-squared = 2.6939, df = 3, p-value = 0.4413
```

```
# Test of no serial correlation:
econ.fit2.ptasy <- serial.test(econ.fit2, lags.pt = 12, type = "PT.asymptotic")
econ.fit2.ptasy
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object econ.fit2
## Chi-squared = 95.507, df = 90, p-value = 0.3257
```

```
# plot(econ.fit1.ptasy)

# Test of the absence of ARCH effect:
econ.fit2.arch <- arch.test(econ.fit2)
names(econ.fit2.arch)
```

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

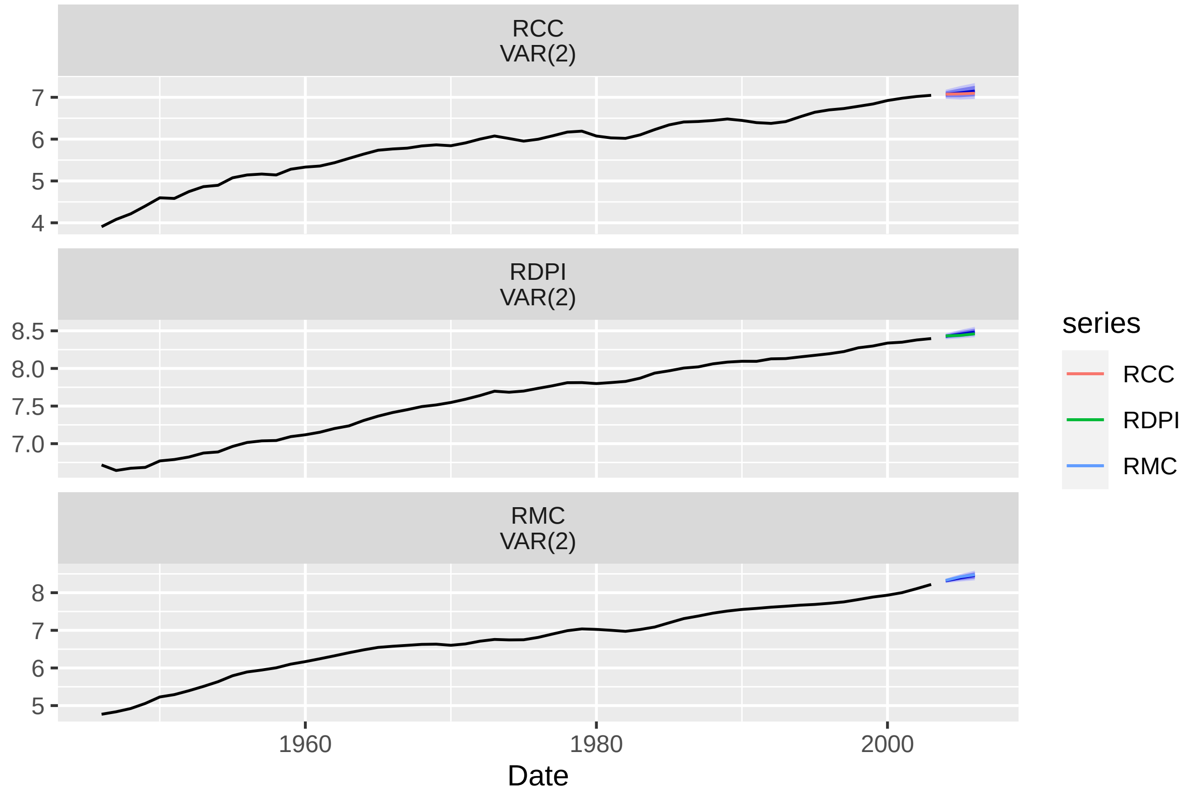
```
econ.fit2.arch
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object econ.fit2
## Chi-squared = 182.63, df = 180, p-value = 0.4312
```

Forecast by model 2

```
econ_fc2 <- forecast(econ.fit2, h = 3)
econ_fc2 %>% autoplot() + autolayer(log(econ.test)) + xlab("Date")
```

```
## For a multivariate timeseries, specify a seriesname for each timeseries. Defaulting to colu
```



From the results of the diagnostic testing for model 2, we know that:

- JB test: the null hypothesis is not rejected that the data are from a normal distribution. The skewness being zero is not rejected and the excess kurtosis being zero is not rejected as the p-value is large.
- Portmanteau test: The null hypothesis is not rejected that no residual autocorrelations, and thus the residuals are homoscedastic.
- ARCH test: the null hypothesis is not rejected and thus the multivariate time series is homoscedastic.

Unlike the first model which uses differenced values, the second model forecasts log-transformed values extremely close to actual logged values.