Shishir Agarwal - W271 Assignment 3

Due 11:59pm Pacific Time Sunday April 11 2021

```r
rm(list = ls())
knitr::opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)
# Load Libraries
library(ggplot2)
library(GGally)
library(stargazer)
library(tidyverse)
library(patchwork)
library(tsibble)
library(fable)
library(fpp2)
library(fpp3)

library(car)
library(dplyr)
library(Hmisc)

library(forecast)
library(astsa)
library(xts)
library(vars)
library(zoo)
library(tseries)
library(tsibble)
```

```r
setwd("/home/jovyan/r_bridge/student_work/shagarwa/Assignment#3")
options(scipen=999)
```

Question 1 (2.5 points)

Time Series Linear Model

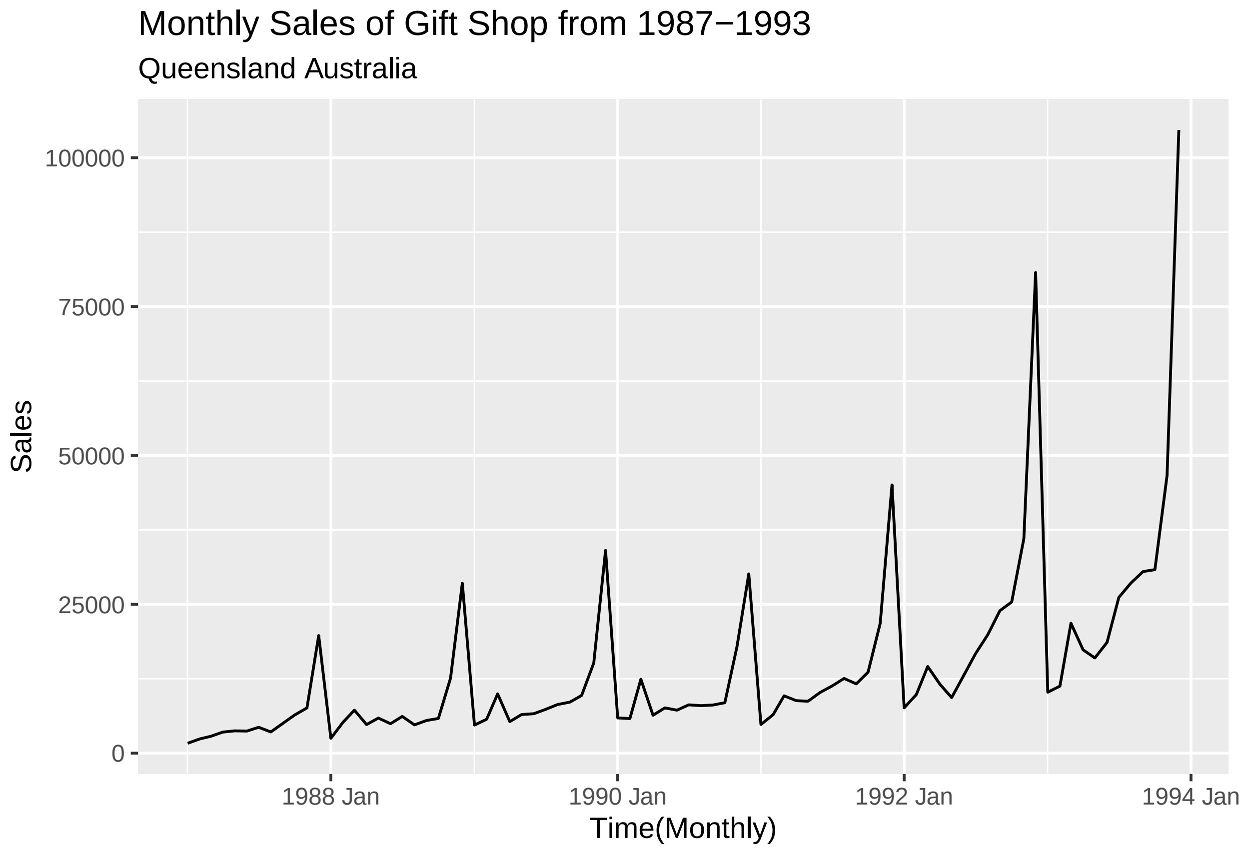
The data set `Q1.csv` concerns the monthly sales figures of a shop which opened in January 1987 and sells gifts, souvenirs, and novelties. The shop is situated on the wharf at a beach resort town in Queensland, Australia. The sales volume varies with the seasonal population of tourists. There is a large 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.

```
# Read the monthly sales data as a dataframe
ss.df <- read.csv("Q1.csv", header=TRUE, sep=",")
# Convert the dataframe into ts object
ss.ts <- ts(ss.df$sales, frequency = 12, start = c(1987,1), end = c(1993,12))
# Convert the dataframe into tsibble object
ss.tsibble <- tsibble(month = yearmonth(ss.df$X), sales = ss.df$sales, index = month)
#Quick EDA
#plot(aggregate(ss.ts))
#monthplot(ss.ts, phase = cycle(ss.ts))
#boxplot(ss.ts ~ cycle(ss.ts))
```

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.

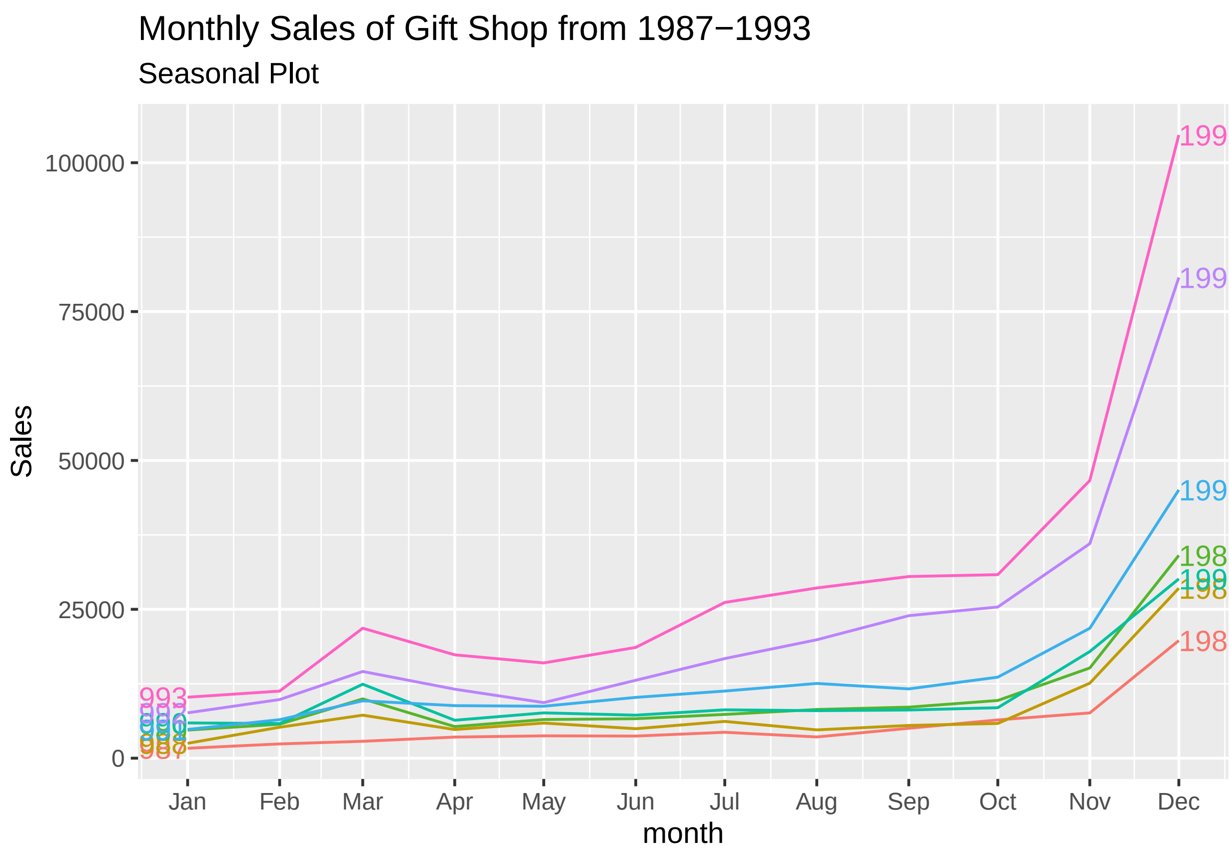
From the time plot we notice the monthly sales is trending upwards and it is seasonal in nature. The monthly sales consistently peaks in December and is lowest in January. In month of March we see a little bump in sales compared to Feb every year and most probably it is due to the festival. Also, there is persistent drop in year-to-year sales between year 1990 and year 1991 for month of January and March. Similarly, there is persistent drop in year-to-year sales between year 1989 and year 1990 for month of Aug, Sept, Oct, Dec. Lastly, the fluctuations between Dec and Jan keeps increasing with every passing year execpt bewteen 1990-1991.

```
#Time Plot of Data
ss.tsibble %>%
  autoplot() +
  labs(
    title = "Monthly Sales of Gift Shop from 1987-1993",
    subtitle = "Queensland Australia",
    y = "Sales",
    x = "Time(Monthly)"
    )
```

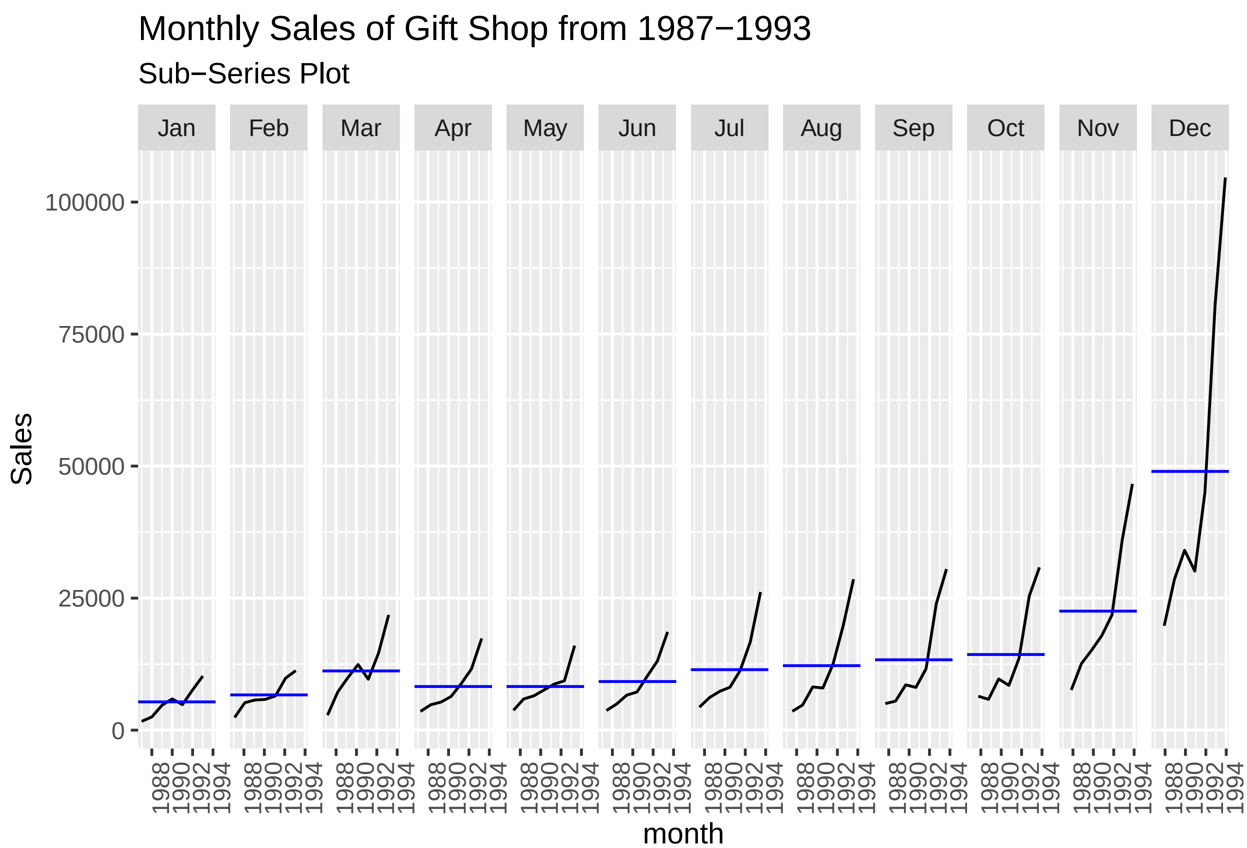



```
#Time Plot of Data for 2 Years
ss.tsibble %>%
  filter(year(month) > 1991) %>%
  autoplot() +
  labs(
    title = "Monthly Sales of Gift Shop from 1992-1994",
    subtitle = "Queensland Australia",
    y = "Sales",
    x = "Time(Monthly)"
    )
```


```
#Seasonal Time Plot of Data
ss.tsibble %>%
  gg_season(sales, labels = "both") +
  labs(
    title = "Monthly Sales of Gift Shop from 1987-1993",
    subtitle = "Seasonal Plot",
    y = "Sales"
    )
```




```
#Sub Series Time Plot of Data
ss.tsibble %>%
  gg_subseries(sales) +
  labs(
    title = "Monthly Sales of Gift Shop from 1987-1993",
    subtitle = "Sub-Series Plot",
    y = "Sales"
    )
```



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

Because we see fluctuation between Jan and Dec sales keeps increasing with every year, we take the log to reduce the amount of variance in our analysis. It is unrealistic to assume the variation will continue to grow at a same pace. Also, we know the value of sales will be a positive number more than zero and taking a log helps us with better forecasting. Lastly, by taking the log we are still able to interpret the regression results in a meaningful manner.

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.

```r
#create a dummy variable for the surfing festival
surf <- ifelse(test = cycle(ss.ts) == 3, yes = 1, no = 0)
#the surfing festival in 1988 did not happen
surf[3] <- 0

#fit the model using fable
ss.fit.TSLM <- ss.tsibble %>%
  model(TSLM(log(sales) ~ trend() + season() + surf))
report(ss.fit.TSLM)
```

```
## Series: sales
## Model: TSLM
## Transformation: log(sales)
##
```

```
## 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 < 0.0000000000000002 ***
## trend()        0.0220198  0.0008268  26.634 < 0.0000000000000002 ***
## 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       0.0000588067270 ***
## season()year6  0.4488283  0.0957647   4.687       0.0000132668325 ***
## season()year7  0.6104545  0.0958039   6.372       0.0000000170771 ***
## season()year8  0.5879644  0.0958503   6.134       0.0000000453365 ***
## season()year9  0.6693299  0.0959037   6.979       0.0000000013630 ***
## season()year10 0.7473919  0.0959643   7.788       0.0000000000448 ***
## season()year11 1.2067479  0.0960319  12.566 < 0.0000000000000002 ***
## season()year12 1.9622412  0.0961066  20.417 < 0.0000000000000002 ***
## surf           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: < 0.000000000000000222
```

```r
#fit the model using forecast
ss.fit.tslm.log <- tslm(ss.ts ~ trend + season + surf, lambda = 0)
summary(ss.fit.tslm.log)
```

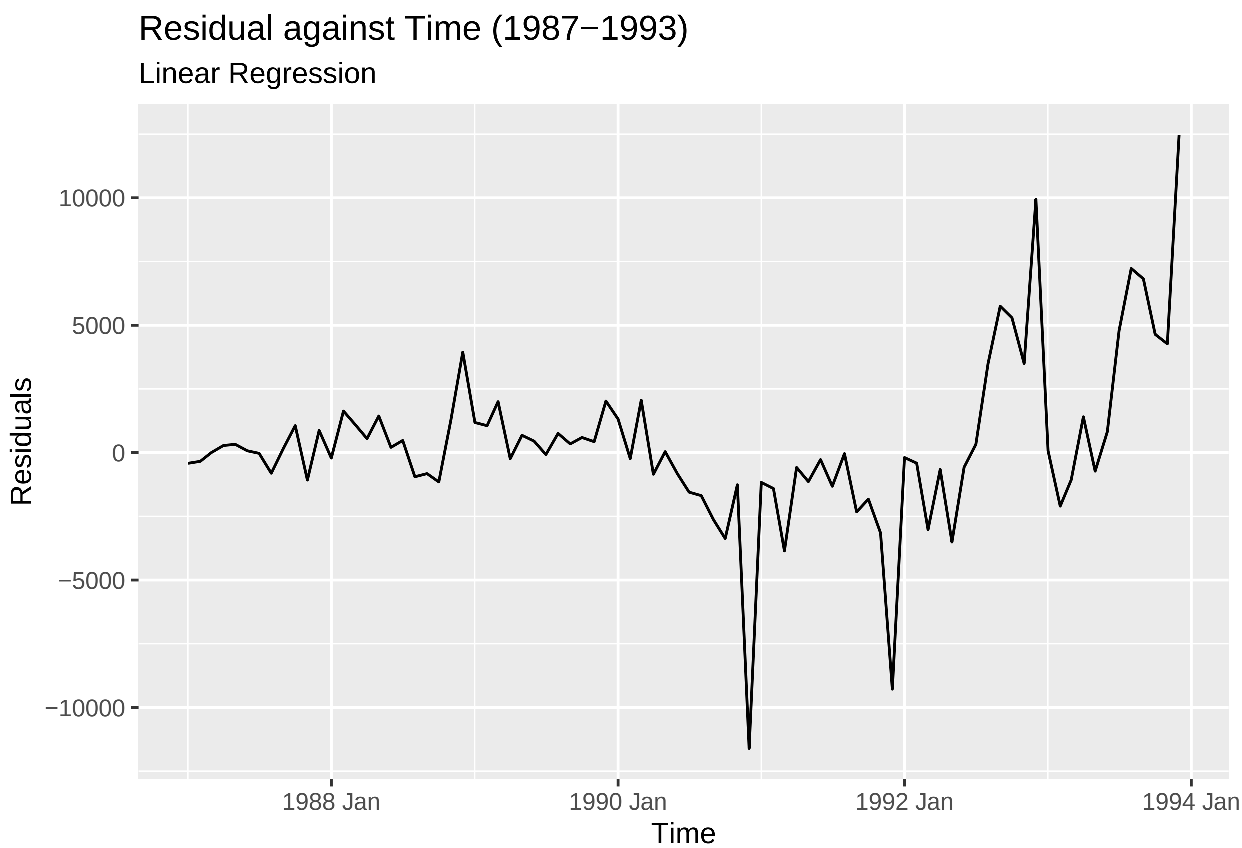
```
##
## Call:
## tslm(formula = ss.ts ~ trend + season + surf, lambda = 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33673 -0.12757  0.00257  0.10911  0.37671
##
## Coefficients:
##              Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 7.6196670  0.0742471 102.626 < 0.0000000000000002 ***
## trend       0.0220198  0.0008268  26.634 < 0.0000000000000002 ***
## season2     0.2514168  0.0956790   2.628             0.010555 *
## season3     0.2660828  0.1934044   1.376             0.173275
## season4     0.3840535  0.0957075   4.013             0.000148 ***
## season5     0.4094870  0.0957325   4.277       0.0000588067270 ***
## season6     0.4488283  0.0957647   4.687       0.0000132668325 ***
```

```
## season7     0.6104545  0.0958039   6.372       0.0000000170771 ***
## season8     0.5879644  0.0958503   6.134       0.0000000453365 ***
## season9     0.6693299  0.0959037   6.979       0.0000000013630 ***
## season10    0.7473919  0.0959643   7.788       0.0000000000448 ***
## season11    1.2067479  0.0960319  12.566 < 0.0000000000000002 ***
## season12    1.9622412  0.0961066  20.417 < 0.0000000000000002 ***
## surf        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: < 0.00000000000000022
```

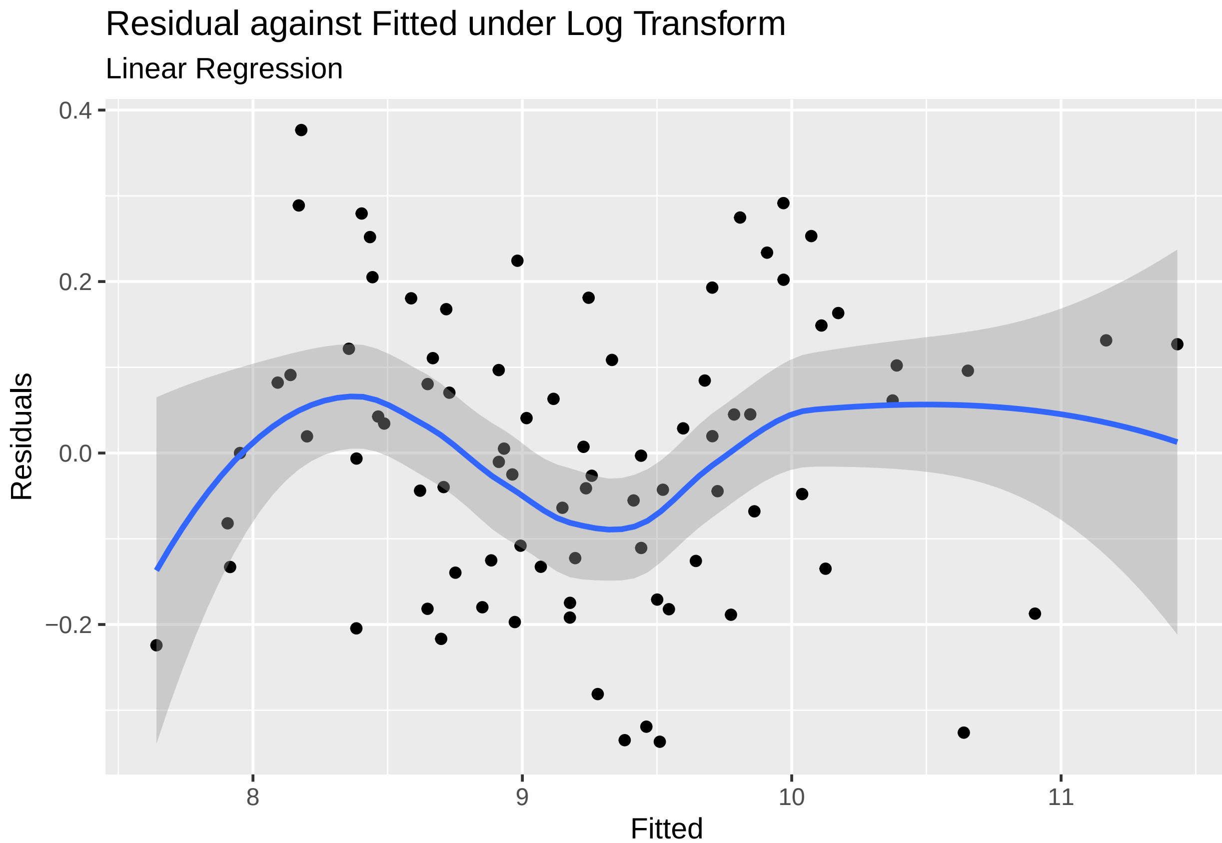
d) Plot the residuals against time and against the fitted values. Do these plots reveal any problems with the model?

Yes, both the plots show violation of zero conditional mean and homoskedasticity. When we plot residuals against time, we see a persistent pattern instead of white noise. For example, we see a pattern between 1988-1990 which is repeated between 1992-1994. When we plot residuals against fitted values we see zero conditional mean is violated and the variance is not constant across the fitted values. Thus, we notice the primary assumptions of linear regression are violated.

```r
#Introspect the Fitted Model
#augment(ss.fit.TSLM)

#Plot Residuals Against Time
augment(ss.fit.TSLM) %>%
  autoplot(.resid) +
  labs(x = "Time", y = "Residuals") +
  labs(
    title = "Residual against Time (1987-1993)",
    subtitle = "Linear Regression"
    )
```



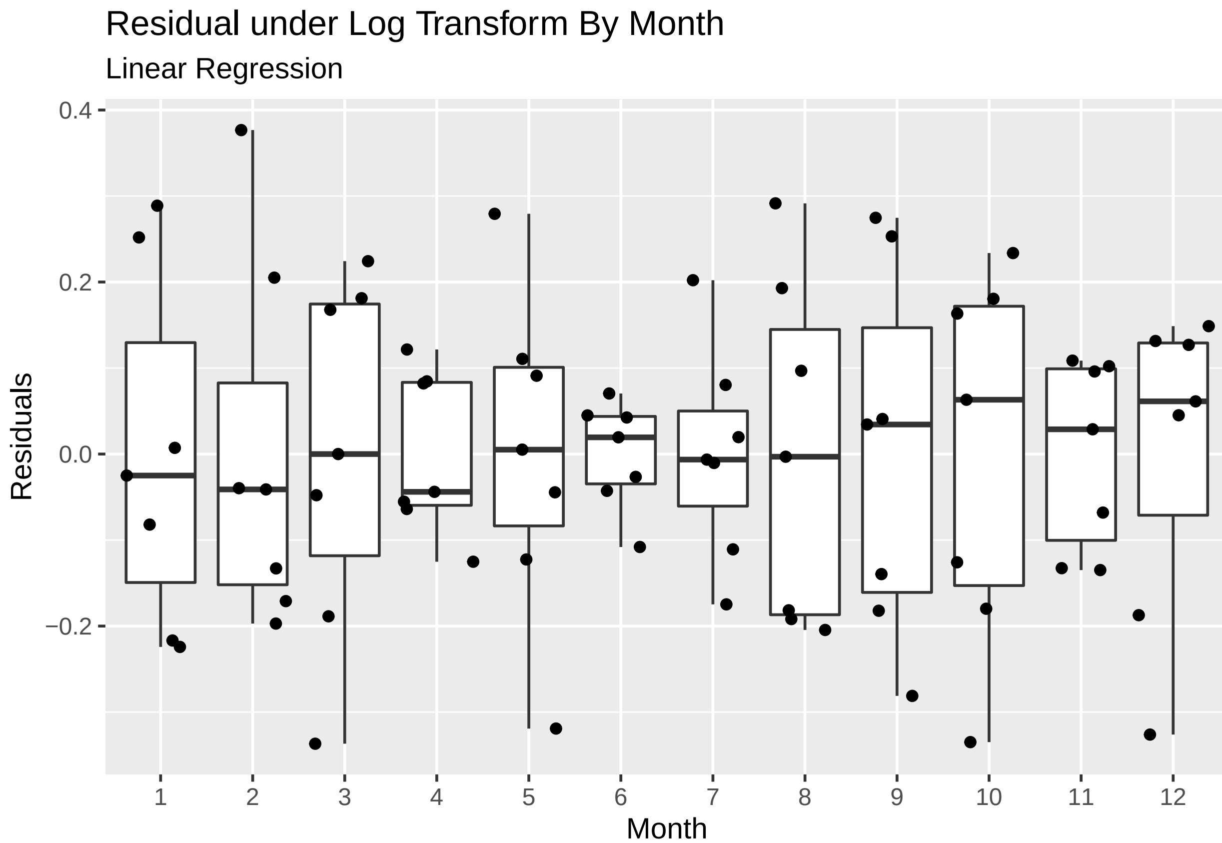

```r
#Plot Residual Against Fitted under log Transformation
augment(ss.fit.TSLM) %>%
  ggplot(aes(x = log(.fitted), y = .innov)) +
  geom_point() +
  geom_smooth(method = "loess") +
  labs(x = "Fitted", y = "Residuals") +
  labs(
    title = "Residual against Fitted under Log Transform",
    subtitle = "Linear Regression"
    )
```



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

Yes, the boxplot shows variance of residuals is not constant from month to month. Also, we notice seasonality. Thus, it confirms our doubt on the validity of the linear regression model and its suitability for inferencing.

```
augment(ss.fit.TSLM) %>%
  mutate(Monthly = factor(month(month))) %>%
  ggplot(aes(x = Monthly, y = .innov)) +
  geom_boxplot() +
  geom_jitter() +
  labs(x = "Month", y = "Residuals") +
  labs(
    title = "Residual under Log Transform By Month",
    subtitle = "Linear Regression"
    )
```



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

The values of the coefficients cannot be trusted for inference since few key assumptions of linear regression are violated. However we do notice there is a positive uptrend (trend is positively correlated) in sales month-over-month. Also, we notice strong seasonality and on average sales in other months are higher compared to sales in January (base month). Thus, we notice the coefficients for all the seasonal dummy variables are positively correlated. Also, we notice on average sales in any given month are higher than sales of preceding month except in August. Thus, we notice and increasing trend in the seasonal coefficients. Lastly, we notice the month of March in itself is not significant however the dummy variable $surf$ is. This shows the festival makes a difference. In general the results are in-line with our observations in the time series plot. We also notice, this model has a high R^2 values, we could still use the model for predicting and forecasting.

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

The low p-value means we reject the null hypothesis of no serial correlation. This, tells us there is serial correlation remaining in the residuals and it has not been eliminated. This means we can still use our model for predicting and forecasting however the predicting interval will be wider due to serial correlation.

```
lmtest::bgtest(ss.fit.tslm.log)
```

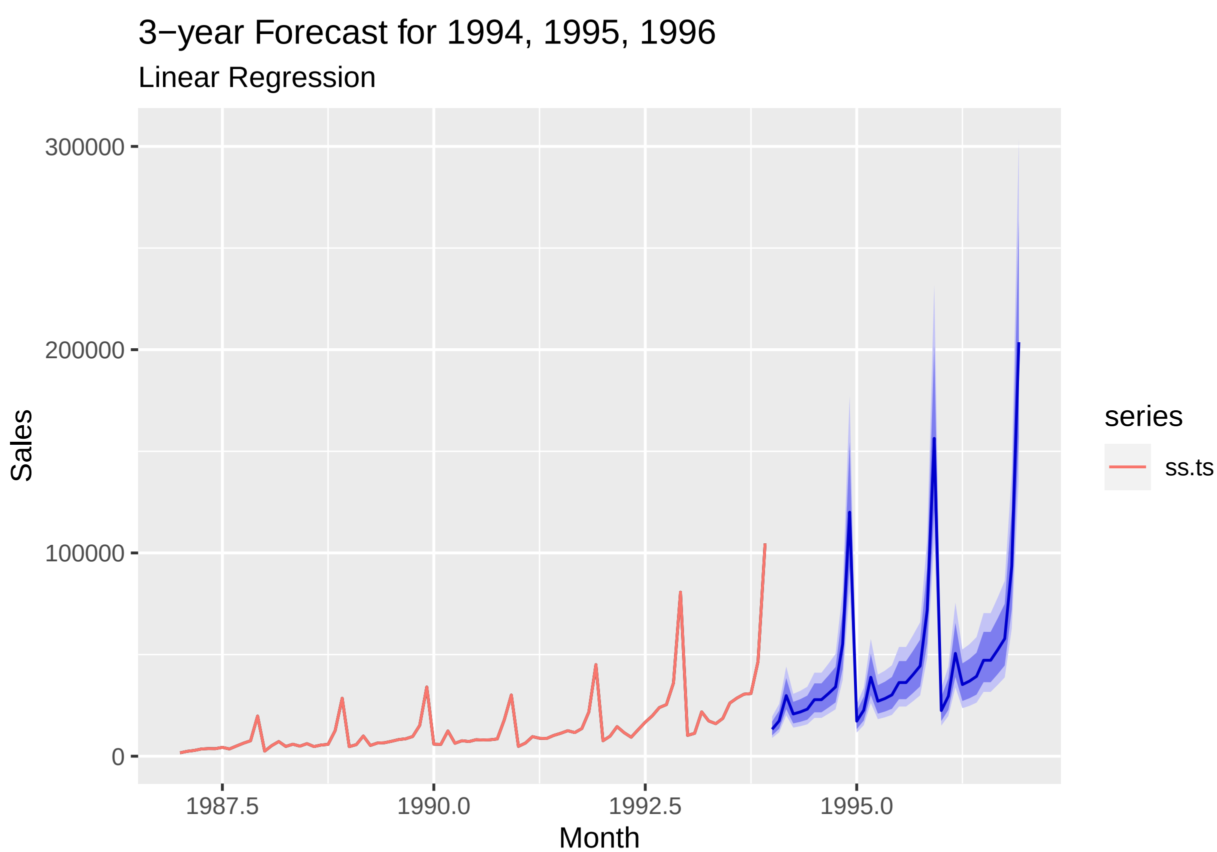
```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
```

```
## data:  ss.fit.tslm.log
## LM test = 25.031, df = 1, p-value = 0.0000005642
```

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.

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

```r
surf <- rep(c(0,0,1,0,0,0,0,0,0,0,0,0),3)
newdata.df <- data.frame(surf=surf)
surf_forecast <- forecast(ss.fit.tslm.log, h = 36, new_data = newdata.df)
surf_forecast %>%
  autoplot() +
  autolayer(ss.ts, sales) +
  labs(x = "Month", y = "Sales") +
  labs(
    title = "3-year Forecast for 1994, 1995, 1996",
    subtitle = "Linear Regression"
    )
```



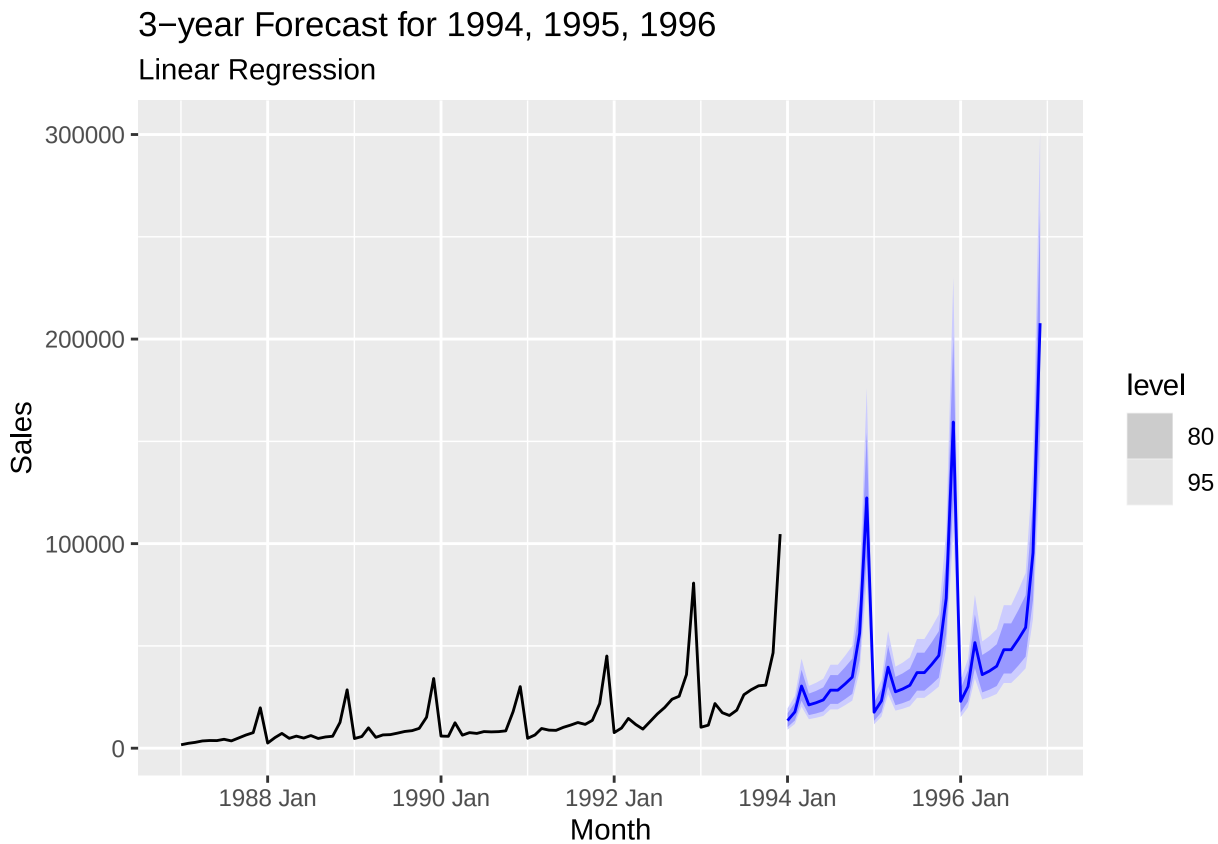

```r
surf_forecast
```

```
##          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
```

```r
surf_forecast_scenarios <- scenarios(
  "March Festival" = new_data(ss.tsibble, 36) %>%
    mutate(surf = rep(c(0,0,1,0,0,0,0,0,0,0,0,0),3)),
  names_to = "Scenario"
)
surf_forecast_TSLM <- forecast(ss.fit.TSLM, new_data = surf_forecast_scenarios)
surf_forecast_TSLM %>%
  autoplot() +
  autolayer(ss.tsibble, sales) +
  labs(x = "Month", y = "Sales") +
  labs(
    title = "3-year Forecast for 1994, 1995, 1996",
    subtitle = "Linear Regression"
```

```
    )
```




```
surf_forecast_TSLM
```

```
## # A fable: 36 x 6 [1M]
## # Key:     Scenario, .model [1]
##    Scenario     .model                          month            sales  .mean  surf
##    <chr>        <chr>                           <mth>           <dist>  <dbl> <dbl>
##  1 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Jan t(N(9.5, 0.038)) 13498.     0
##  2 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Feb t(N(9.8, 0.038)) 17742.     0
##  3 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Mar  t(N(10, 0.039)) 30397.     1
##  4 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Apr t(N(9.9, 0.038)) 21171.     0
##  5 March Festi~ TSLM(log(sales) ~ trend(~ 1994 May  t(N(10, 0.038)) 22200.     0
##  6 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Jun  t(N(10, 0.038)) 23605.     0
##  7 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Jul  t(N(10, 0.038)) 28363.     0
##  8 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Aug  t(N(10, 0.038)) 28350.     0
##  9 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Sep  t(N(10, 0.038)) 31437.     0
## 10 March Festi~ TSLM(log(sales) ~ trend(~ 1994 Oct  t(N(10, 0.038)) 34747.     0
## # ... with 26 more rows
```

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

There are number of ways to improve the model. One of the ways to improving the predictions would be to use the Auto Regressive or Moving Average models that exploit the serial correlation within the time series model. Thus, we will explore using the SARIMA model to capture the stochastic process that is being used to generate the time series and use the model to improve the predictions.

Question 2 (2.5 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:

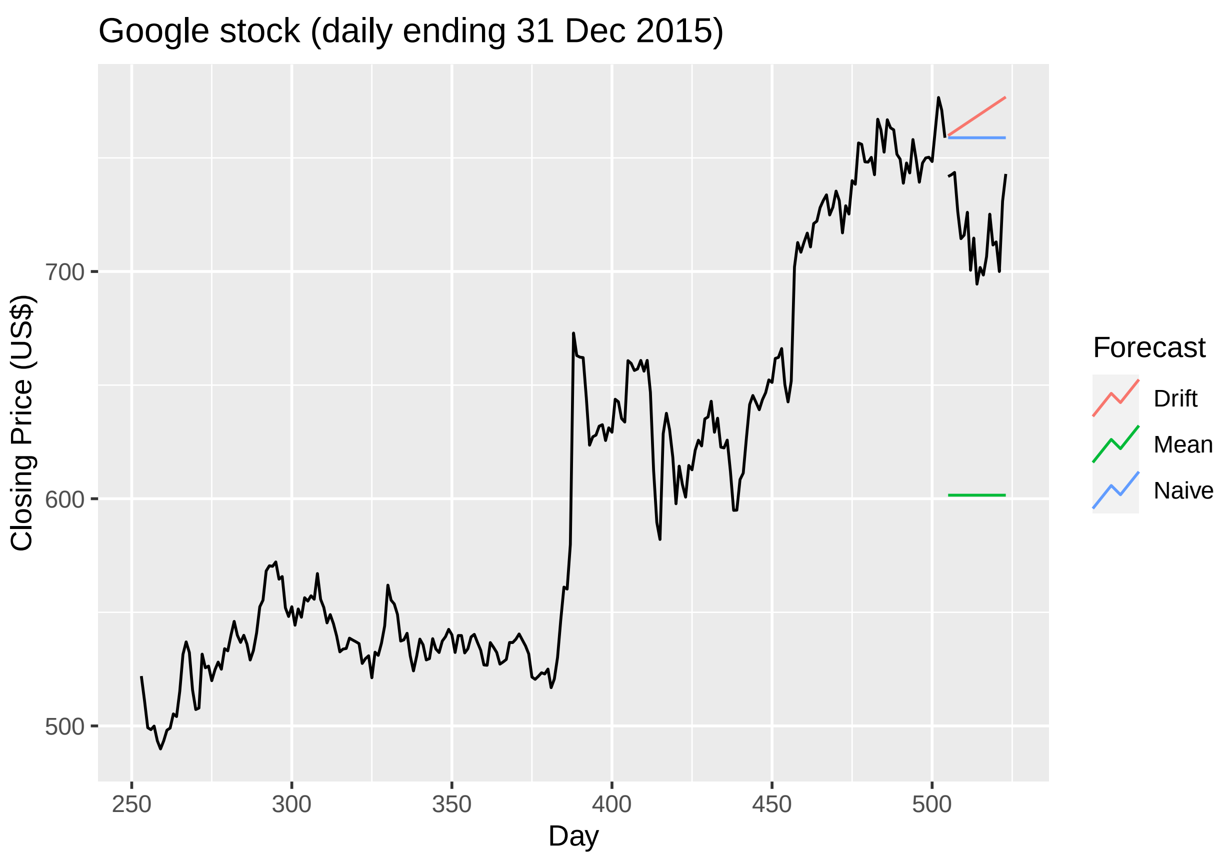
- `MEAN()`: the *average method*, forecasting all future values to be equal to the mean of the historical data
- `NAIVE()`: the *naive method*, forecasting all future values to be equal to the value of the latest observation
- `RW()`: the *drift method*, forecasting all future values to continue following 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.

```
library(fpp3)
#library(tidyverse)
#library(lubridate)
#library(tsibble)
#library(fable)

# 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),
    Naive = 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)
```

```r
# 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:

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

```
## # A tibble: 3 x 11
##   .model Symbol .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift  GOOG   Test  -49.8  53.1  49.8 -6.99  6.99  7.84  5.60 0.604
## 2 Mean   GOOG   Test  117.  118.  117.  16.2  16.2  18.4  12.4  0.496
## 3 Naive  GOOG   Test  -40.4  43.4  40.4 -5.67  5.67  6.36  4.58 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
```
# 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 11
##   .model              Symbol .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>               <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 RW(Close ~ drif~ GOOG   Test  0.726  11.3  7.26 0.112  1.19  1.02  1.01 0.0985
```

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

A time series cross-validation procedure uses series of test sets, each consisting of a single observation. The corresponding training set consists only of observations that occurred prior to the observation that forms the test set. Since it is not possible to obtain a reliable forecast based on a small training set, the earliest observations are not considered as test sets. For example, we could start with a training set of length 3 and increase the size of successive training set by 1. The forecast accuracy is computed by averaging over the test sets. The accuracy measure calculates forecasting error by taking the difference between the observed value and the predicted value on a test data set and averaging it for cross-validation. It calculates following errors

- ME (Mean Error)
- RMSE (Root Mean Square Error)
- MAE (Mean Absolute Error)
- MPE (Mean Percentage Error)
- MAPE (Mean Absolute Percentage Error)
- MASE (Mean Absolute Scaled Error)
- RMSSE (Root Mean Squared Scaled Error)
- ACF1 (First Coefficient of Autocorrelation Function)

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.

```
# Create training data using 2015 stock data
fb_2015 <- facebook_stock %>% filter(year(Date) == 2015)

# Train the model using 2015 stock data
```

```r
fb_fit <- fb_2015 %>%
  model(
    Naive = NAIVE(Volume),
    Drift = RW(Volume ~ drift())
  )

# Create Test Data using 2016 stock data
facebook_stock_2016 <- facebook_stock %>%
  filter(yearmonth(Date) == yearmonth("2016 Jan"))

# Using 2015 Data, Forecast for 2016
fb_fc <- fb_fit %>% forecast(facebook_stock_2016)

# Calculate Accuracy against the 2016 Stock Test Data
accuracy(fb_fc, facebook_stock)
```

```
## # A tibble: 2 x 11
##   .model Symbol .type        ME      RMSE      MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr>  <chr>     <dbl>     <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift  FB     Test   23407624. 30410093.  2.34e7  49.4  49.4  2.28  1.98 0.391
## 2 Naive  FB     Test   23412453. 30414687.  2.34e7  49.4  49.4  2.28  1.98 0.391
```

```r
# Calculate Accuracy against the 2015 Stock Training Data
fb_fit %>% accuracy()
```

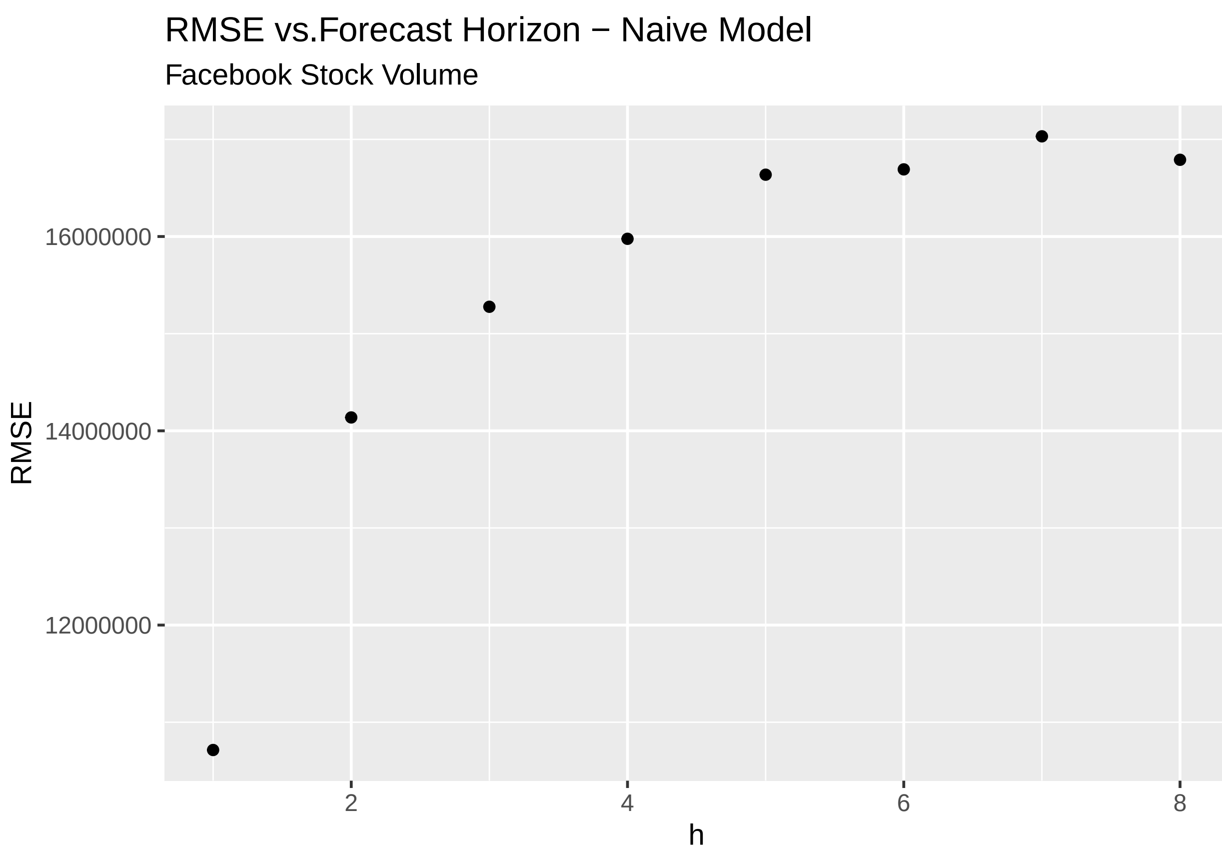
```
## # A tibble: 2 x 11
##   Symbol .model .type          ME     RMSE     MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>  <chr>  <chr>       <dbl>    <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 FB     Naive  Traini~  4.83e+ 2  1.07e7  7.41e6 -6.00  27.3  1     1     -0.139
## 2 FB     Drift  Traini~ -3.86e-10  1.07e7  7.41e6 -6.00  27.3  1.00  1.00 -0.139
```

```r
# Time series cross-validation accuracy
fb_2015_tr <- fb_2015 %>%
  slice(1:(n()-1)) %>%
  stretch_tsibble(.init = 3, .step = 1)

fc <- fb_2015_tr %>%
  model(
    Naive_Vol = NAIVE(Volume),
    Drift_Vol = RW(Volume ~ drift())
    ) %>%
  forecast(h=1)

fc %>% accuracy(fb_2015)
```

```
## # A tibble: 2 x 11
##   .model    Symbol .type      ME     RMSE     MAE   MPE  MAPE  MASE RMSSE   ACF1
##   <chr>     <chr>  <chr>   <dbl>    <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 Drift_V~  FB     Test  -194181.  1.08e7  7.50e6 -6.74  27.8  1.01  1.01 -0.138
```

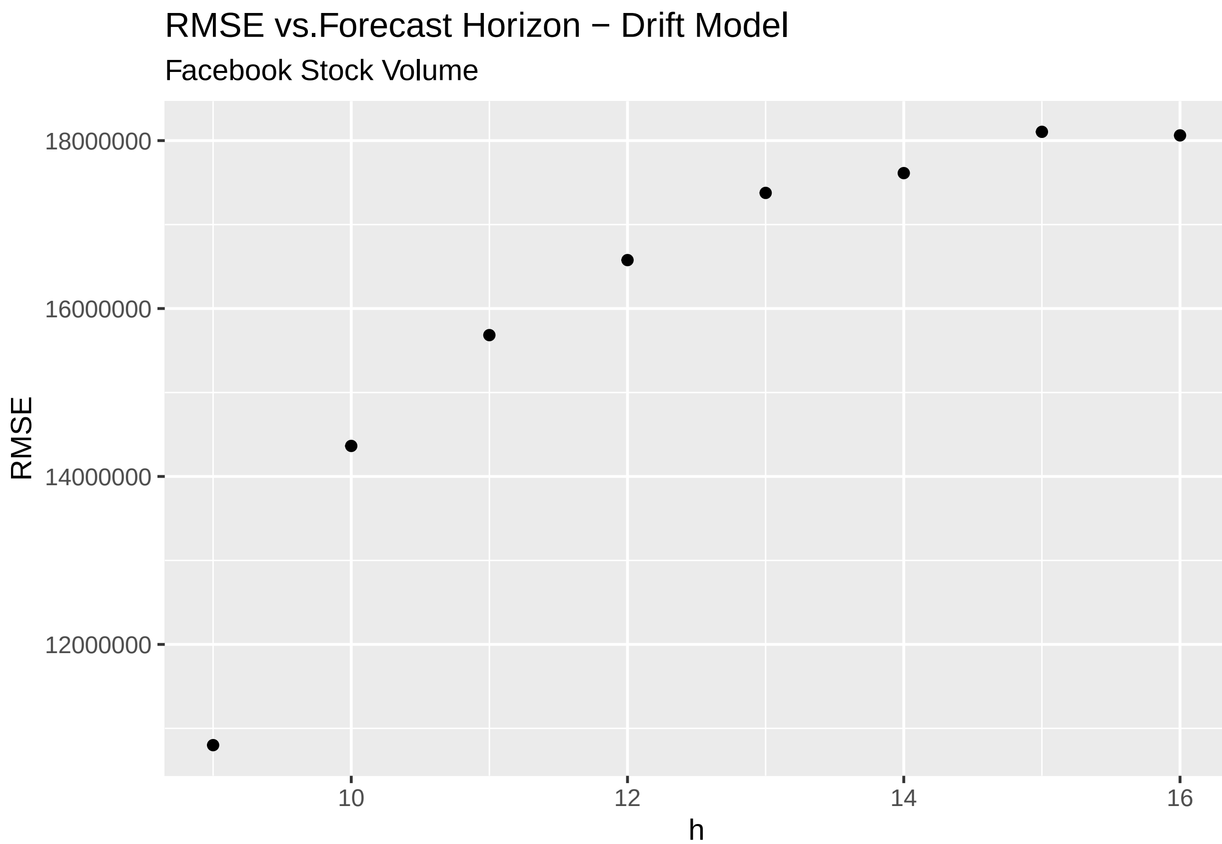
```
## 2 Naive_V~ FB      Test    -36549.    1.07e7  7.43e6 -6.18   27.4   1.00   1.00 -0.139
```

```r
fc <- fb_2015_tr %>%
  model(
    Naive_Vol = NAIVE(Volume),
    Drift_Vol = RW(Volume ~ drift())
    ) %>%
  forecast(h = 8) %>%
  group_by(.id) %>%
  mutate(h = row_number()) %>%
  ungroup()

fc %>%
  filter(.model == "Naive_Vol") %>%
  accuracy(facebook_stock, by = c("h", ".model")) %>%
  ggplot(aes(x = h, y = RMSE)) +
  geom_point() +
  labs(
    title = "RMSE vs.Forecast Horizon - Naive Model",
    subtitle = "Facebook Stock Volume"
    )
```




```r
fc %>%
  filter(.model == "Drift_Vol") %>%
```

```r
accuracy(facebook_stock, by = c("h", ".model")) %>%
ggplot(aes(x = h, y = RMSE)) +
geom_point() +
labs(
  title = "RMSE vs.Forecast Horizon - Drift Model",
  subtitle = "Facebook Stock Volume"
  )
```



Question 3 (2.5 points):

ARIMA model

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

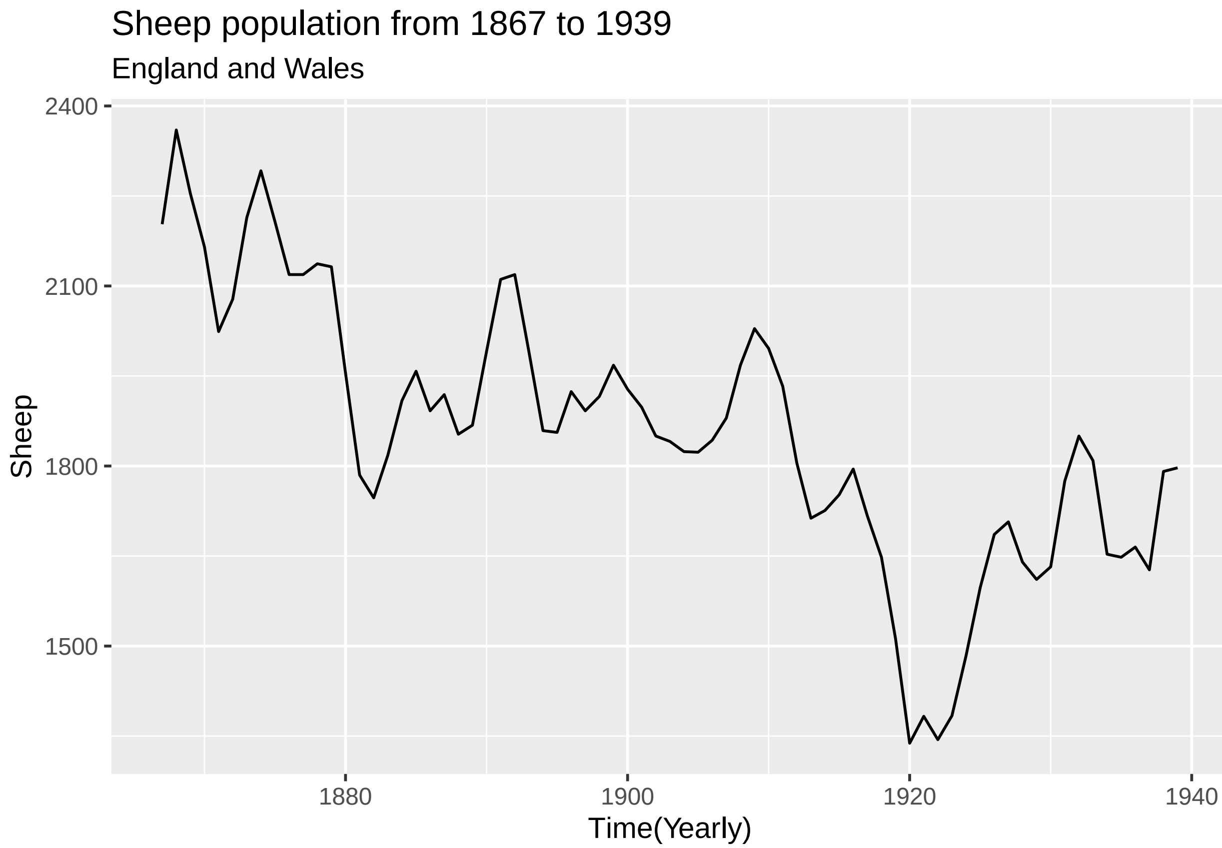
```
#install.packages('fma')
library(fma)
head(fma::sheep)
```

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

```
sheep.ts <- fma::sheep
sheep.tsibble <- as_tsibble(sheep.ts)
```

a) Produce a time plot of the time series.

```
#Time Plot of Data
sheep.ts %>%
  autoplot() +
  labs(
    title = "Sheep population from 1867 to 1939",
    subtitle = "England and Wales",
    y = "Sheep",
    x = "Time(Yearly)"
    )
```



b) Assume you decide to fit the following model:

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

where ϵ_t is a white noise series.

What sort of ARIMA model is this (i.e., what are p, d, and q)?

ARIMA(3,1,0)

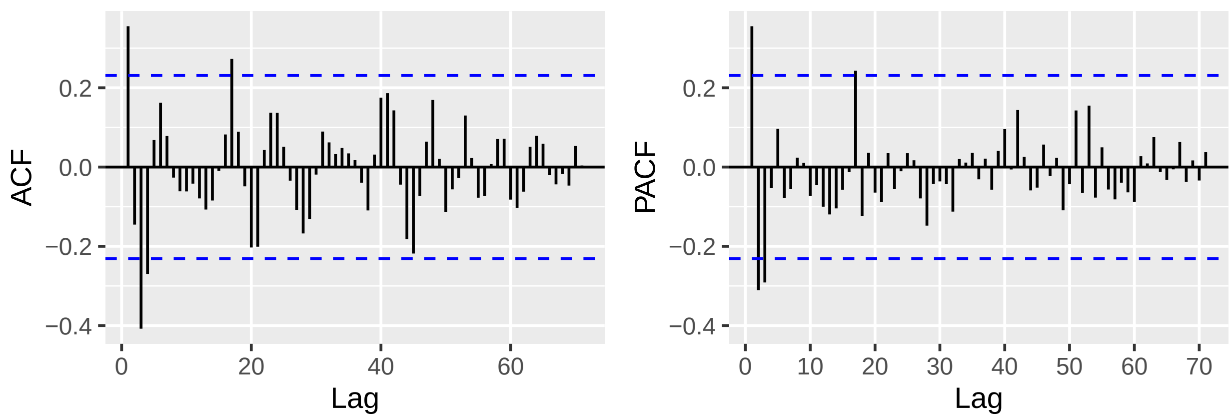
Express this ARIMA model using backshift operator notation.

$$(1 - B)[1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3]$$

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

The model is appropriate because with 1 differencing we get a PACF which cuts of after 3 which shows it is a AR model with 3. Also, ACF model dampens slowly without providing conclusive evidence on MA model.

```
#Time Plot of Data
sheep.ts %>% diff() %>% ggtsdisplay(lag.max = 144)
```



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).

$$y_{1940} = y_{1939} + \phi_1(y_{1939} - y_{1938}) + \phi_2(y_{1938} - y_{1937}) + \phi_3(y_{1937} - y_{1936}) + \epsilon_t$$
$$y_{1940} = 1797 + 0.42(1797 - 1791) + (-0.2)(1791 - 1627) + (-0.3)(1627 - 1665)$$
$$y_{1940} = 1778.12$$
$$y_{1941} = y_{1940} + \phi_1(y_{1940} - y_{1939}) + \phi_2(y_{1939} - y_{1938}) + \phi_3(y_{1938} - y_{1937}) + \epsilon_t$$
$$y_{1941} = 1778.12 + 0.42(1778.12 - 1797) + (-0.2)(1797 - 1791) + (-0.3)(1791 - 1627)$$
$$y_{1941} = 1719.79$$
$$y_{1942} = y_{1941} + \phi_1(y_{1941} - y_{1940}) + \phi_2(y_{1940} - y_{1939}) + \phi_3(y_{1939} - y_{1938}) + \epsilon_t$$

$$y_{1942} = 1719.79 + 0.42(1719.79 - 1778.12) + (-0.2)(1778.12 - 1797) + (-0.3)(1797 - 1791)$$

$$y_{1942} = 1697.27$$

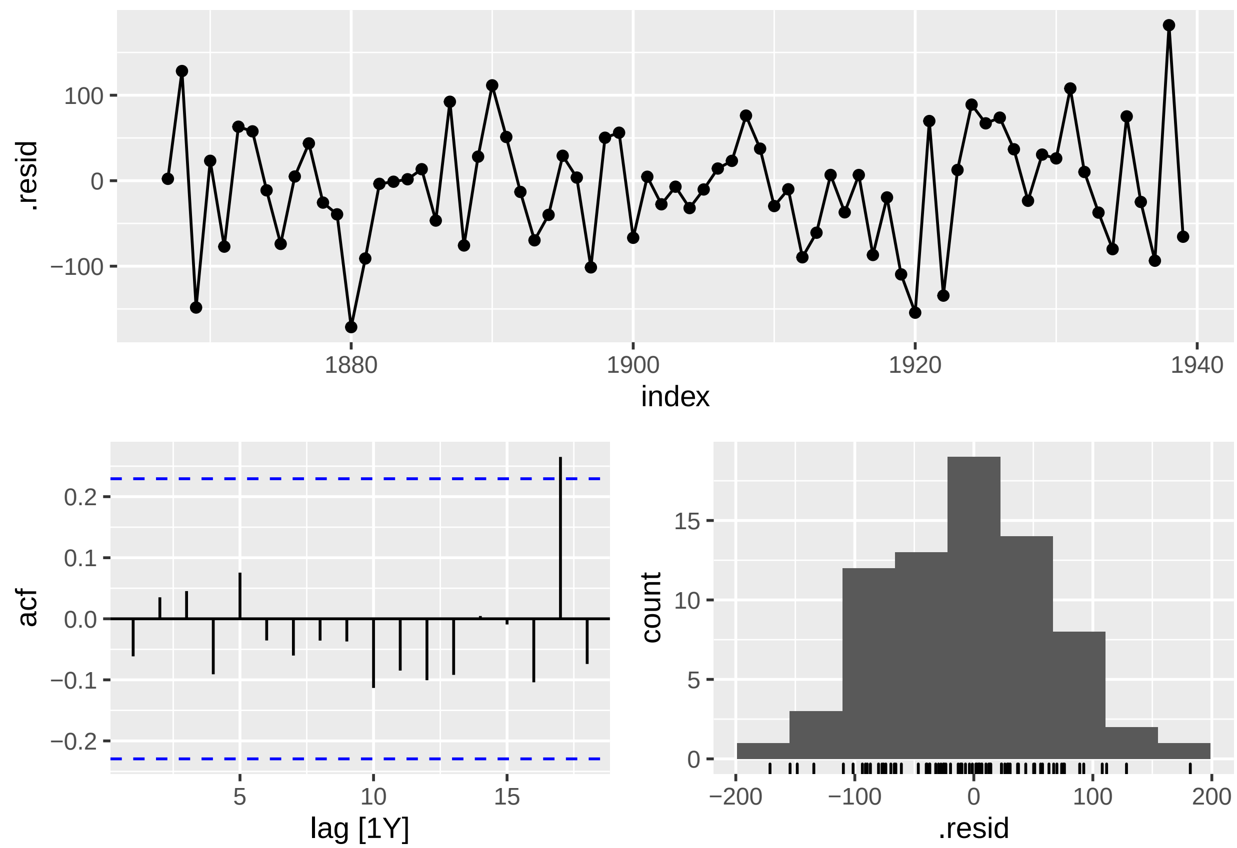
e) Find the roots of your model's characteristic equation and explain their significance.

For the model to provide a valid forecast, it is important for us to meet the stationarity condition. The stationarity condition requires complex roots of model's characteristic equation to lie outside a unit circle. In other words the mod value of roots needs to be greater than 1. In this specific case for ARIMA(3,1,0) we find the mod value of roots to be 1.2557172.0836451.255717. Since mod value of all roots is greater than 1 we can safely assume stationarity for our AR model.

```r
sheep_model <-  sheep.tsibble %>% model(arima1 = ARIMA(value ~ pdq(3,1,0)))

# model properties
sheep_model %>% report(fit)
```

```
## 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
```

```r
# residual characteristics
sheep_model %>% gg_tsresiduals()
```



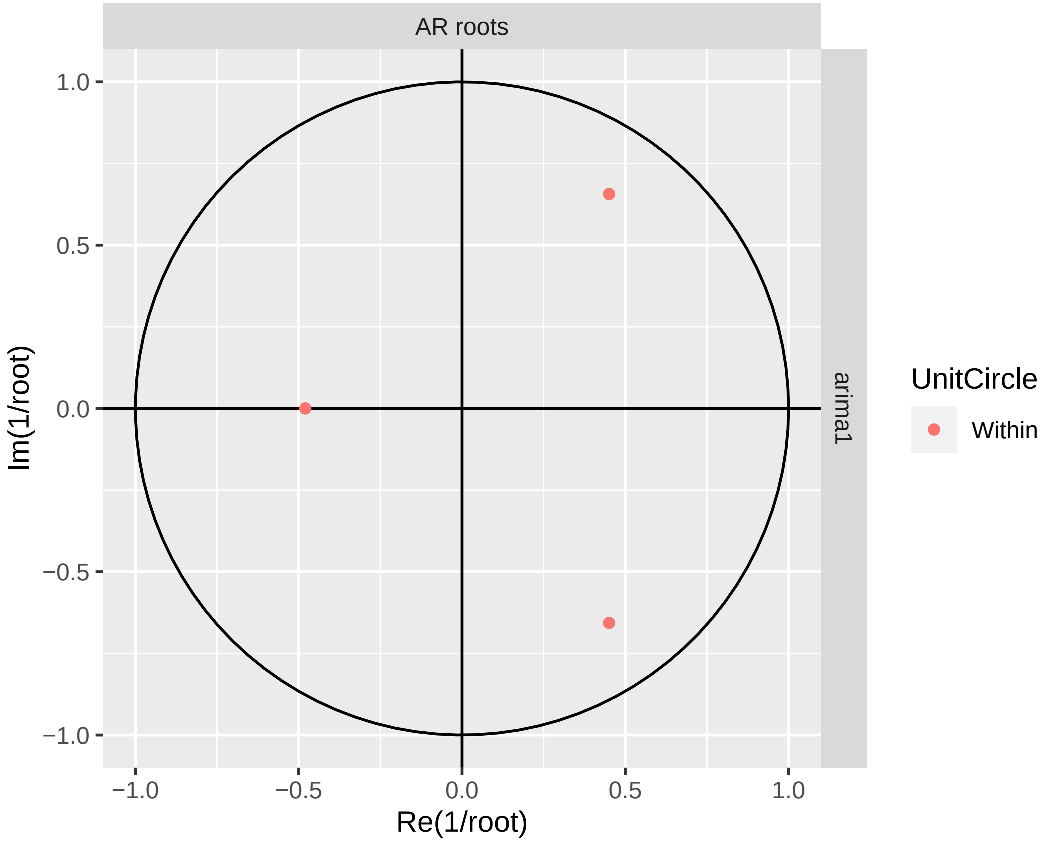

```r
# test for autocorrelaton of residuals
augment(sheep_model) %>% features(.resid, ljung_box)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 arima1   0.288     0.592
```

```r
# model roots (one real, two complex)
glance(sheep_model)[['ar_roots']]
```

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

```r
# inverse roots within unit circle
gg_arma(sheep_model)
```




```r
# modulus of roots exceed unity
Mod(polyroot(c(1, -coef(sheep_model)[['estimate']])))
```

```
## [1] 1.255717 2.083645 1.255717
```

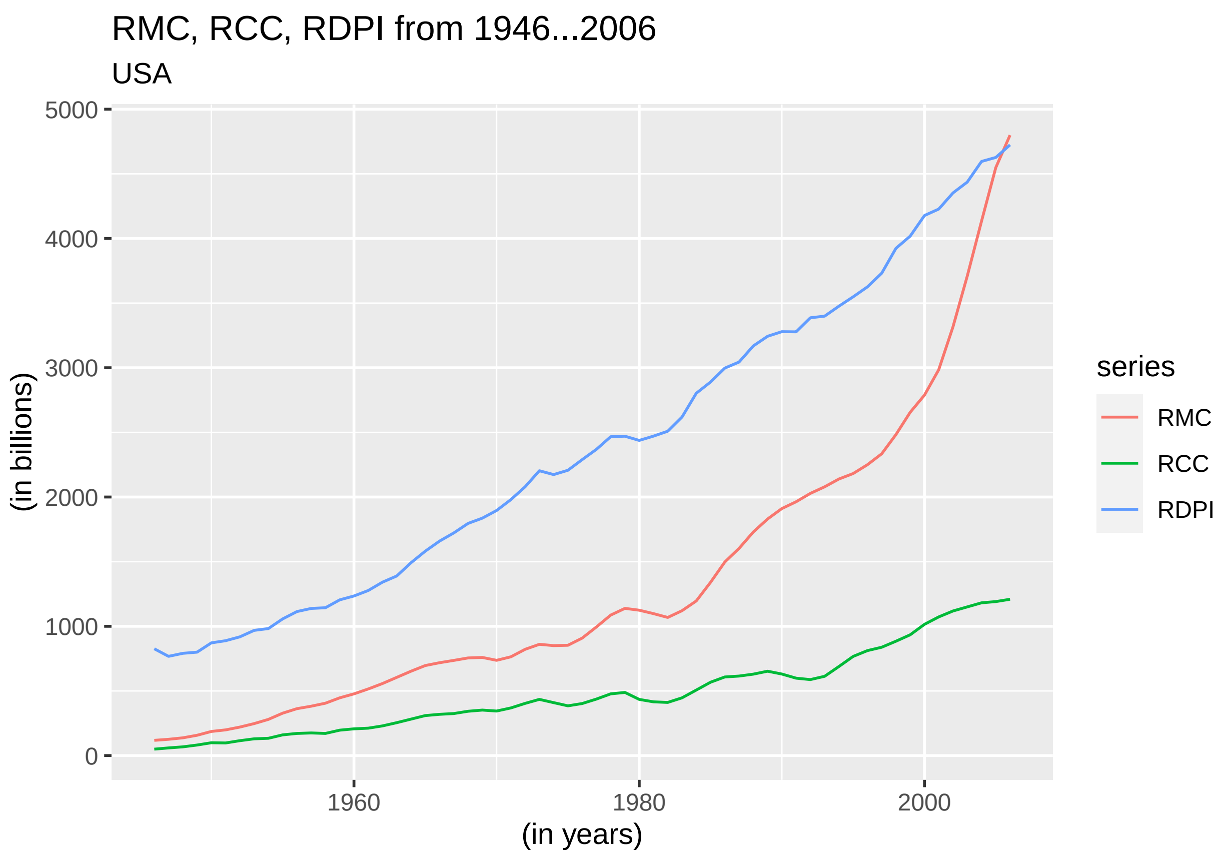
```r
#sheep_model %>% forecast(h = 8)
```

Question 4 (2.5 points):

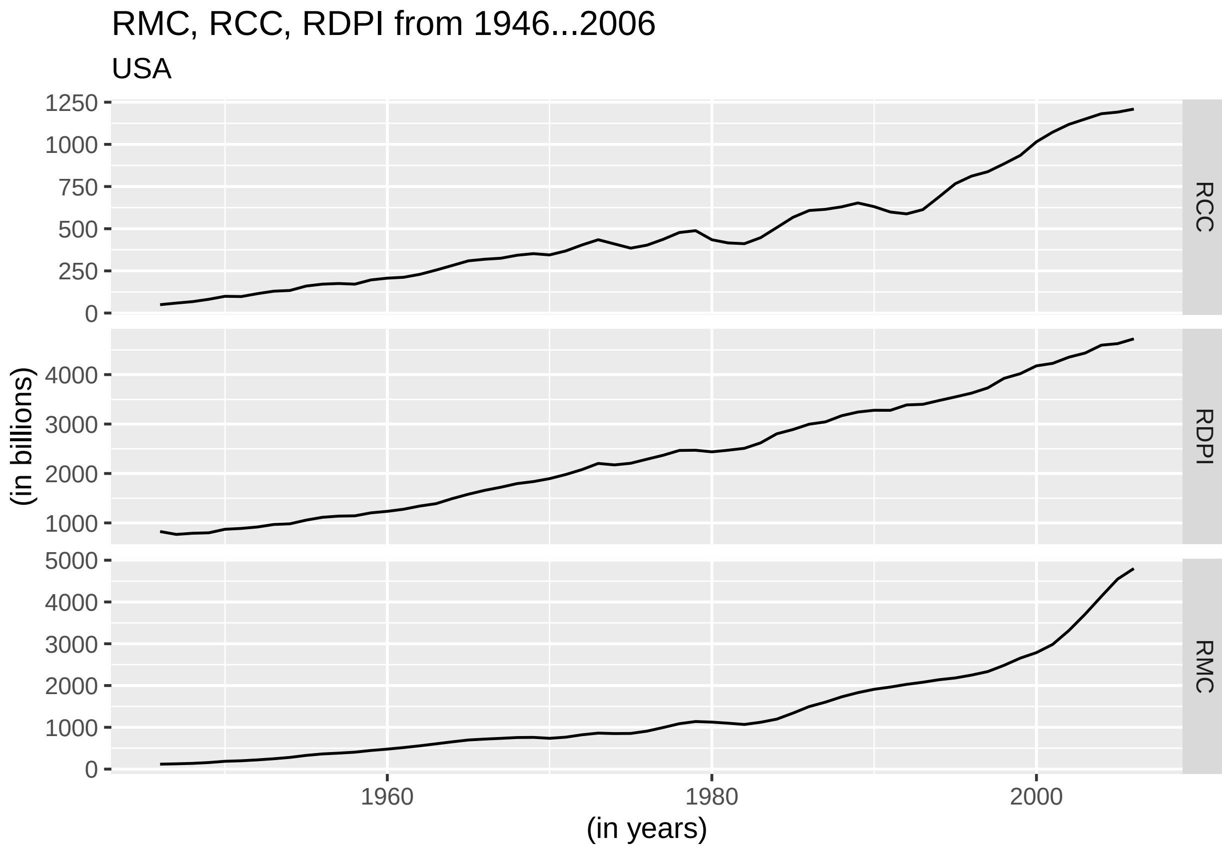
Vector autoregression

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.

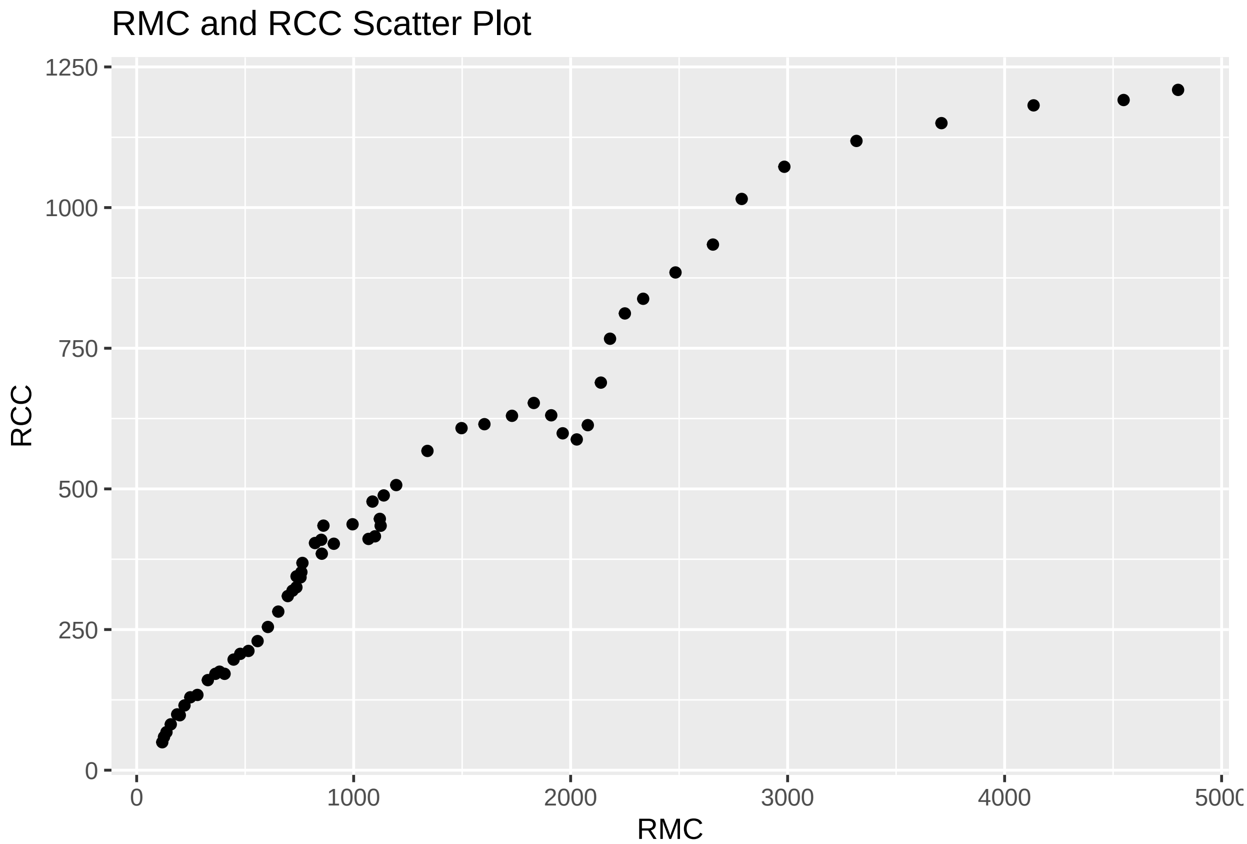
```r
# Read the monthly sales data as a dataframe and create ts objects
credit.df <- read.csv("Q4.csv", header=TRUE, sep=",")

credit.ts <- ts(credit.df[, 2:4], start = c(1946), end = c(2006))
credit.tsibble <- as_tsibble(credit.df, index = Year)

credit.tsibble.wide <- as_tsibble(credit.ts, pivot_longer = FALSE)
credit.tsibble.long <- as_tsibble(credit.ts, pivot_longer = TRUE)

rmc.ts <- ts(credit.df$RMC, start = c(1946), end = c(2006))
rcc.ts <- ts(credit.df$RCC, start = c(1946), end = c(2006))
rdpi.ts <- ts(credit.df$RDPI, start = c(1946), end = c(2006))

#Quick EDA
credit.ts %>% autoplot() +
    labs(
    title = "RMC, RCC, RDPI from 1946-2006",
    subtitle = "USA",
    y = "(in billions)",
    x = "(in years)"
    )
```




```r
credit.tsibble.long %>%
  ggplot(aes(x = index, y = value, group = key)) +
  geom_line() +
  facet_grid(vars(key), scales = "free_y") +
  labs(
    title = "RMC, RCC, RDPI from 1946-2006",
    subtitle = "USA",
    y = "(in billions)",
    x = "(in years)"
    )
```



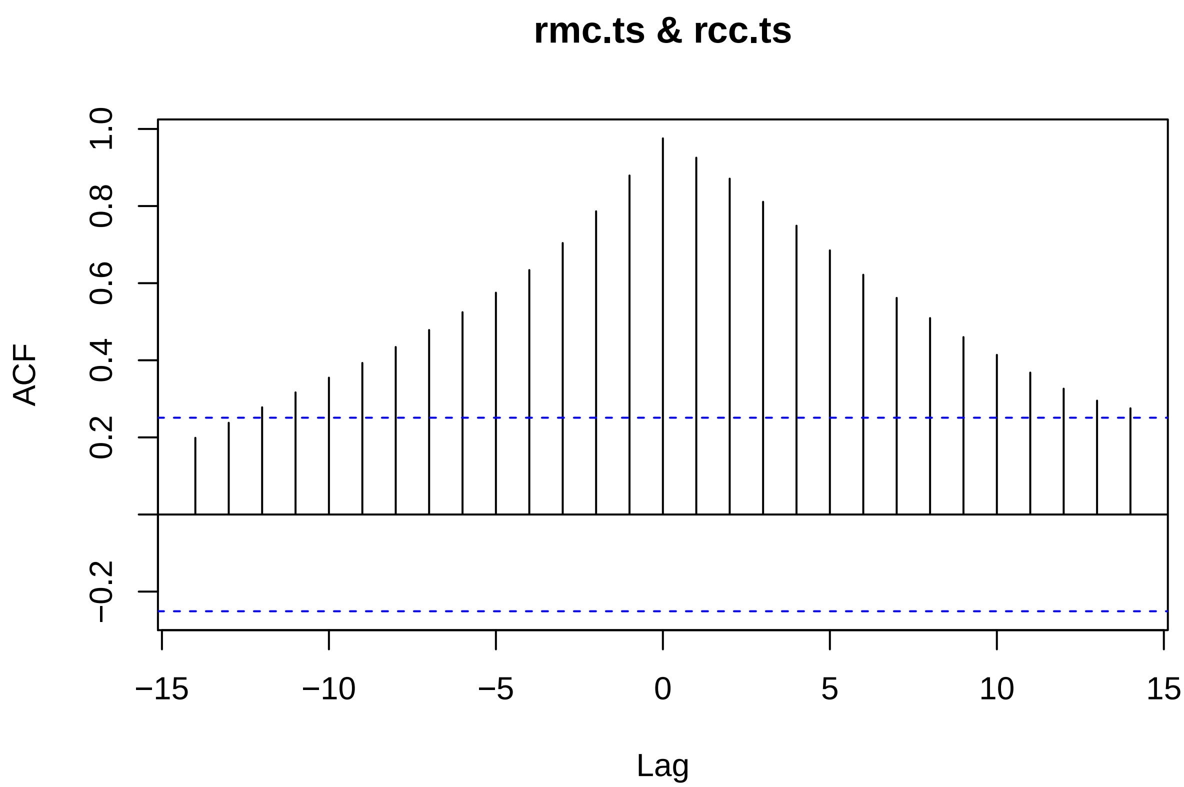

```
qplot(RMC, RCC, data = credit.tsibble, main = "RMC and RCC Scatter Plot")
```




```
cor(rmc.ts, rcc.ts)
```

```
## [1] 0.9756163
```

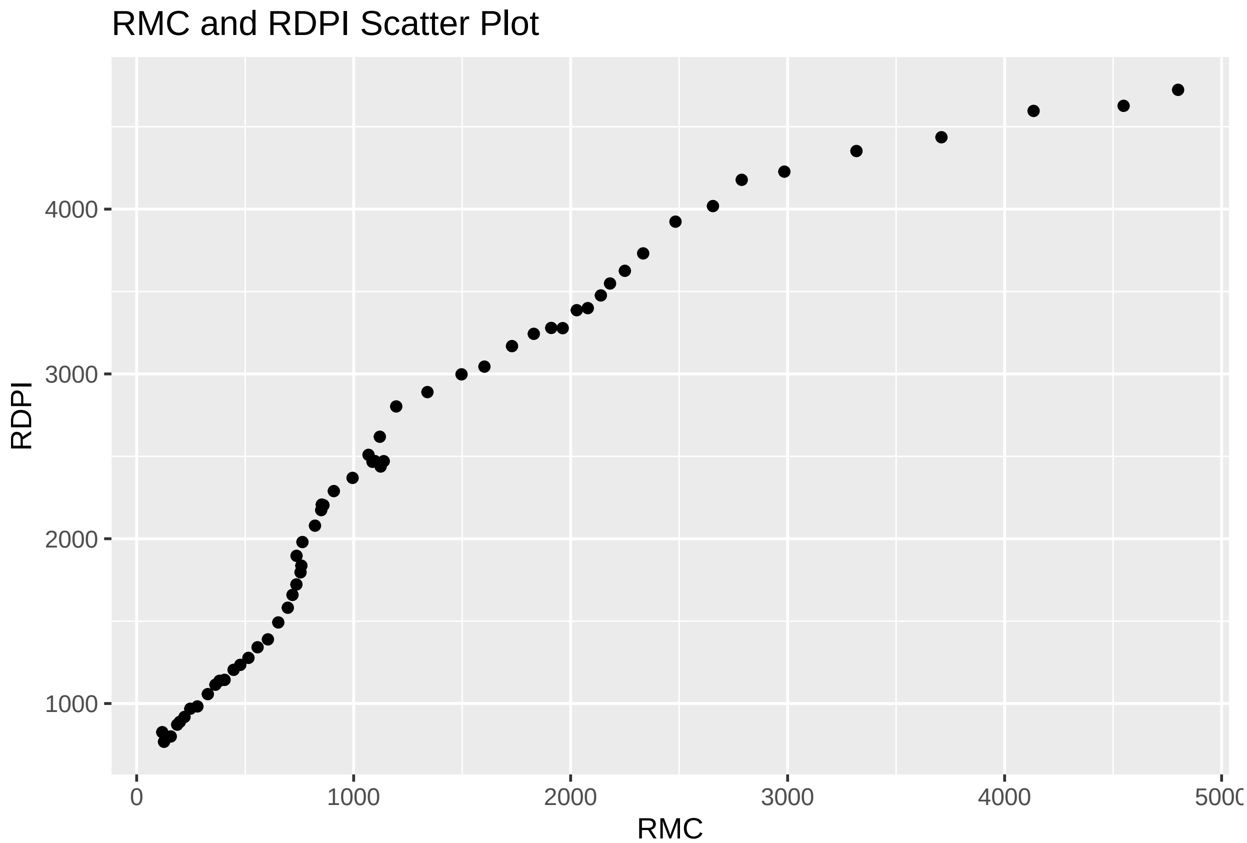
```
ccf(rmc.ts, rcc.ts)
```




```r
summary(lm(rcc.ts ~ rmc.ts))
```

```
##
## Call:
## lm(formula = rcc.ts ~ rmc.ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -236.602  -49.642    7.701   58.625  131.636
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 110.465325  14.159087   7.802       0.000000000118 ***
## rmc.ts        0.278224   0.008149  34.143 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71.33 on 59 degrees of freedom
## Multiple R-squared:  0.9518, Adjusted R-squared:  0.951
## F-statistic:  1166 on 1 and 59 DF,  p-value: < 0.00000000000000022
```

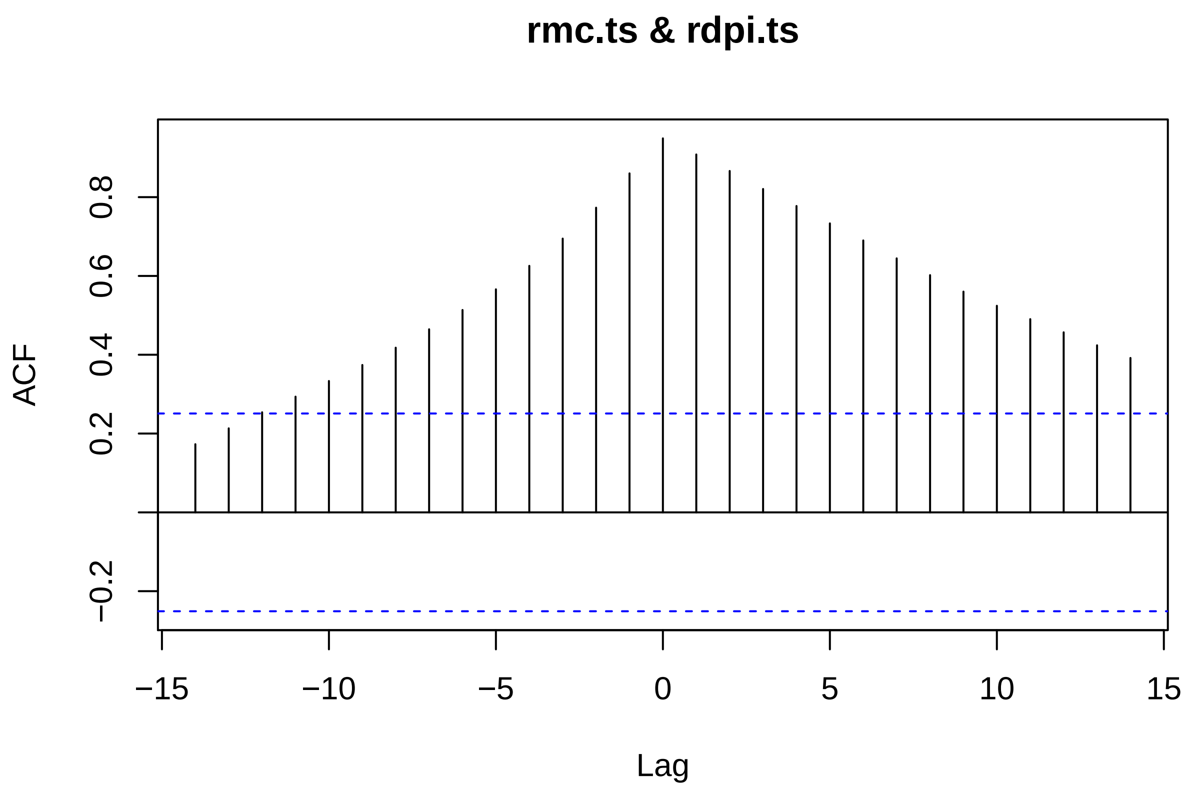
```r
qplot(RMC, RDPI, data = credit.tsibble, main = "RMC and RDPI Scatter Plot")
```




```r
cor(rmc.ts, rdpi.ts)
```

```
## [1] 0.9491886
```

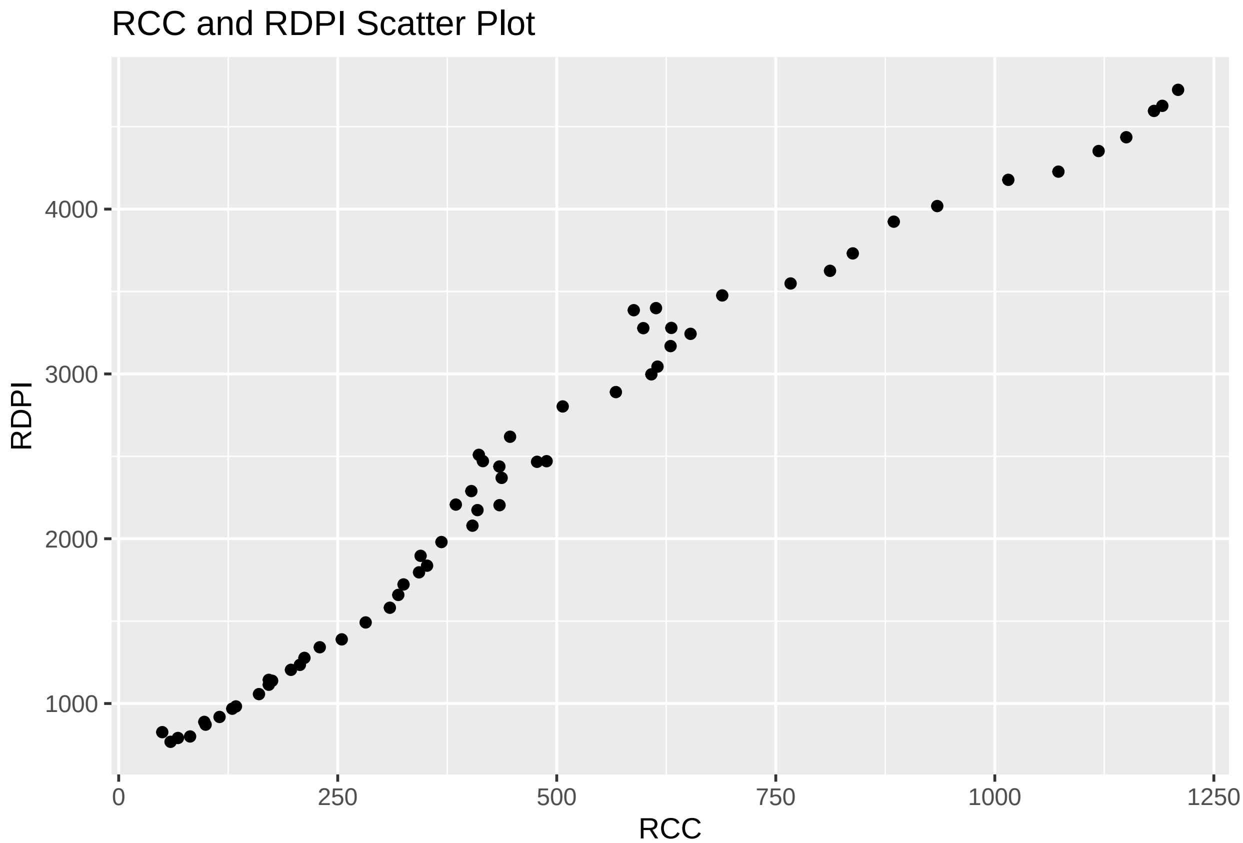
```r
ccf(rmc.ts, rdpi.ts)
```




```r
summary(lm(rdpi.ts ~ rmc.ts))
```

```
##
## Call:
## lm(formula = rdpi.ts ~ rmc.ts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1123.7  -317.8   148.8   307.4   541.5
##
## Coefficients:
##               Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 1070.61231   74.64935   14.34 <0.0000000000000002 ***
## rmc.ts         0.99530    0.04296   23.17 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 376 on 59 degrees of freedom
## Multiple R-squared:  0.901,  Adjusted R-squared:  0.8993
## F-statistic: 536.7 on 1 and 59 DF,  p-value: < 0.00000000000000022
```

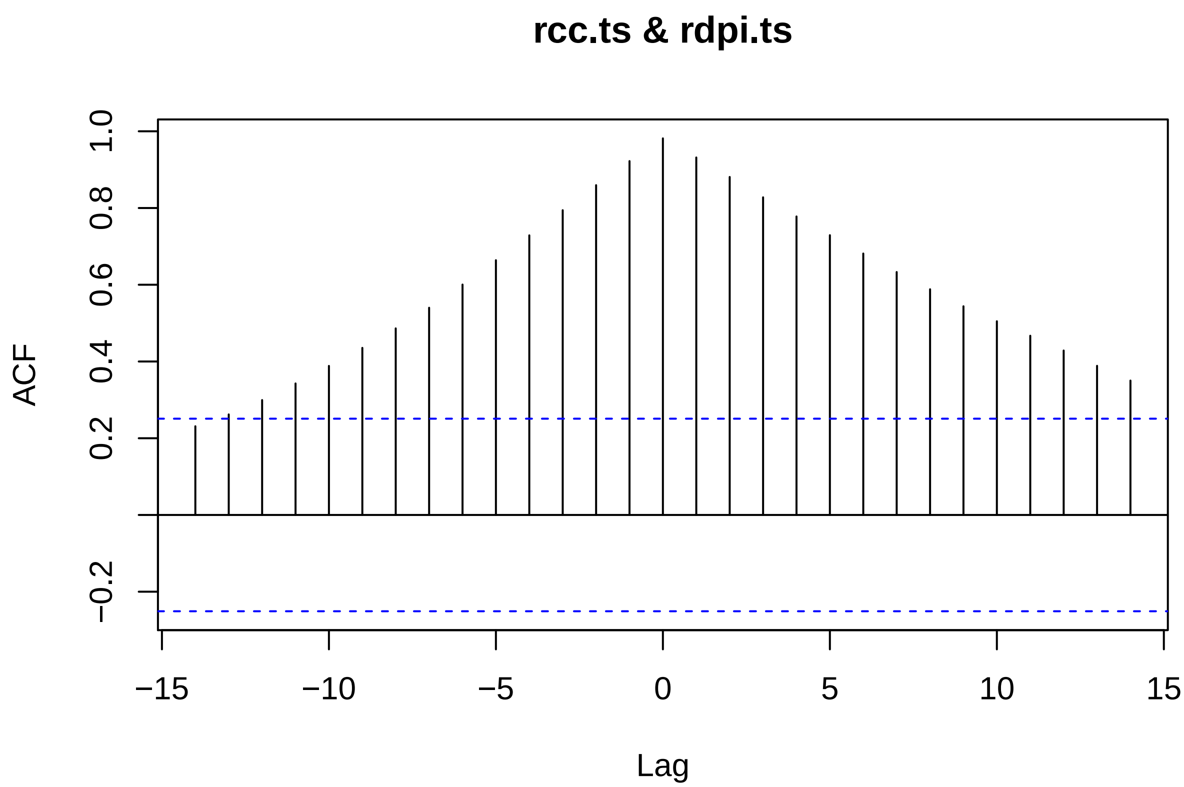
```r
qplot(RCC, RDPI, data = credit.tsibble, main = "RCC and RDPI Scatter Plot")
```




```
cor(rcc.ts, rdpi.ts)
```

```
## [1] 0.9815842
```

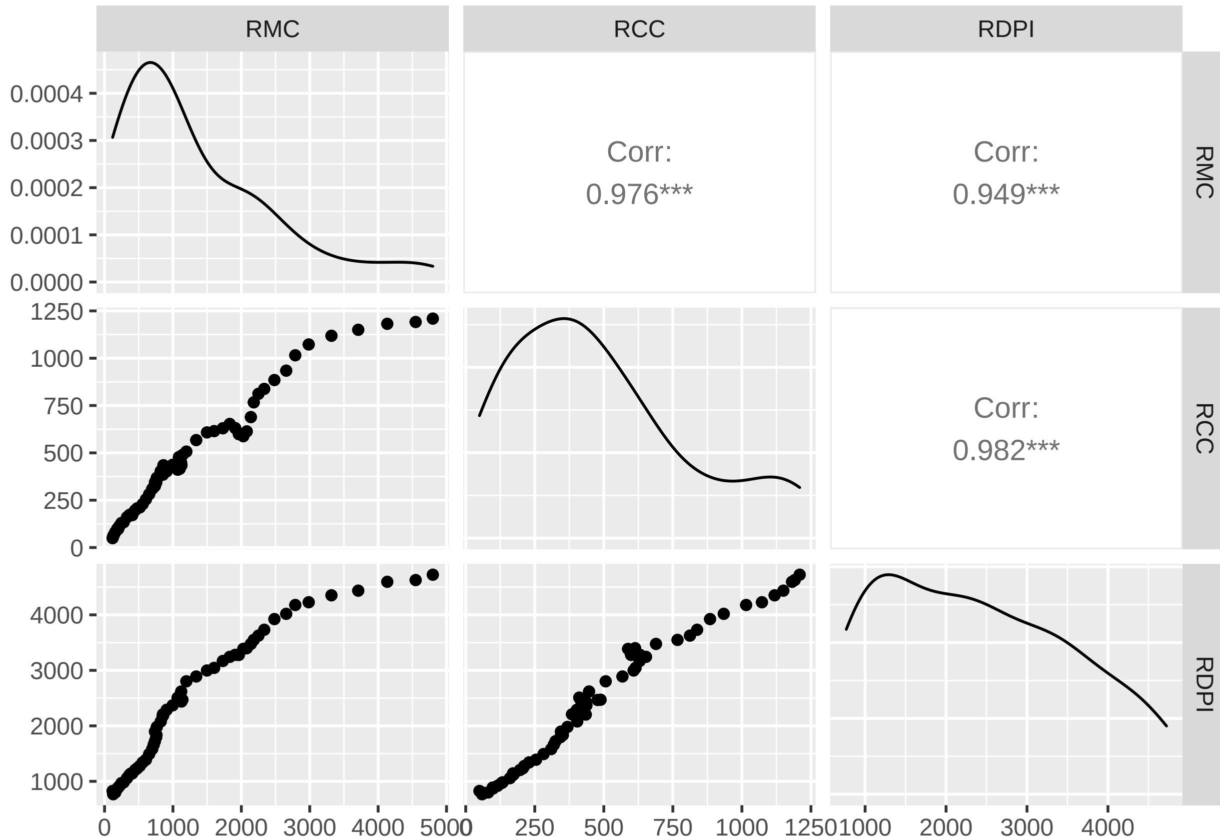
```
ccf(rcc.ts, rdpi.ts)
```



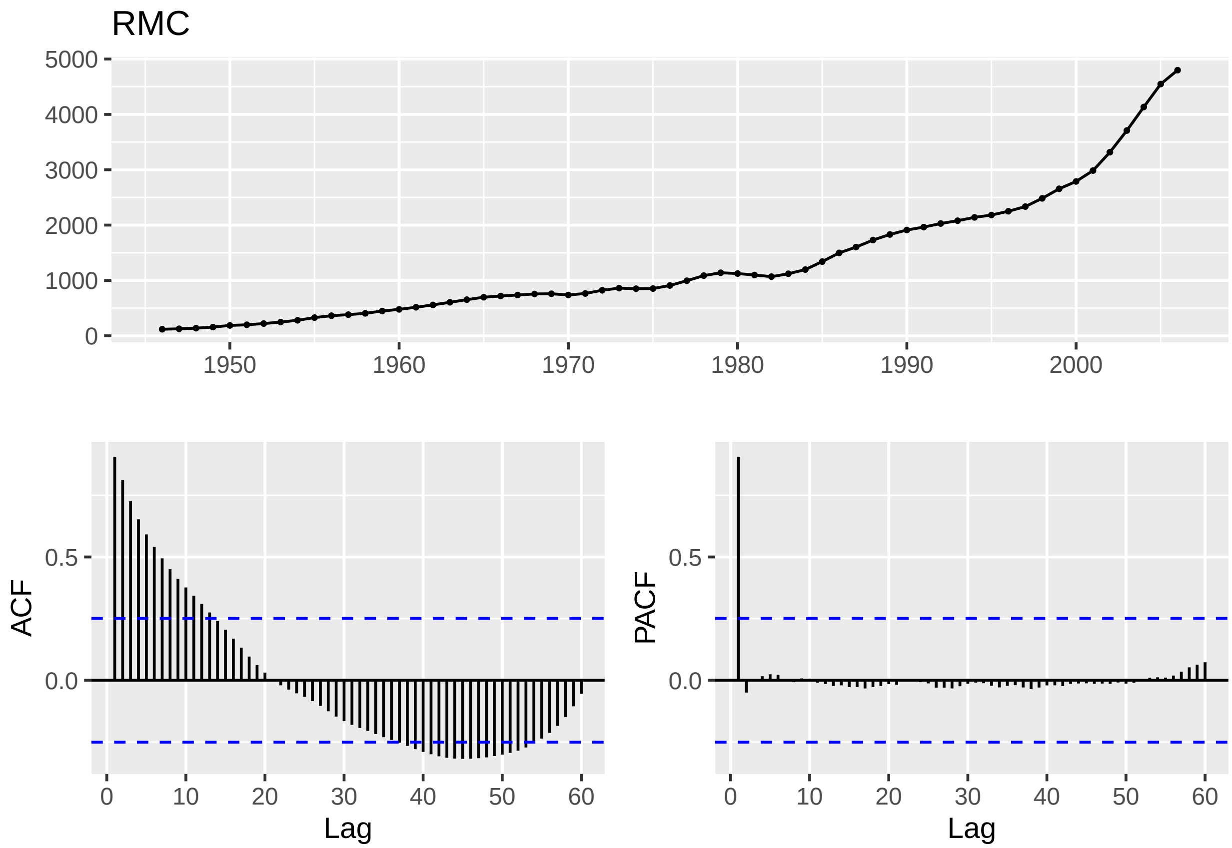

```r
summary(lm(rdpi.ts ~ rcc.ts))
```

```
##
## Call:
## lm(formula = rdpi.ts ~ rcc.ts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -374.99 -157.06  -38.34  158.14  604.61
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 660.13929   52.72622   12.52 <0.0000000000000002 ***
## rcc.ts        3.60921    0.09145   39.47 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 228.3 on 59 degrees of freedom
## Multiple R-squared:  0.9635, Adjusted R-squared:  0.9629
## F-statistic:  1558 on 1 and 59 DF,  p-value: < 0.00000000000000022
```

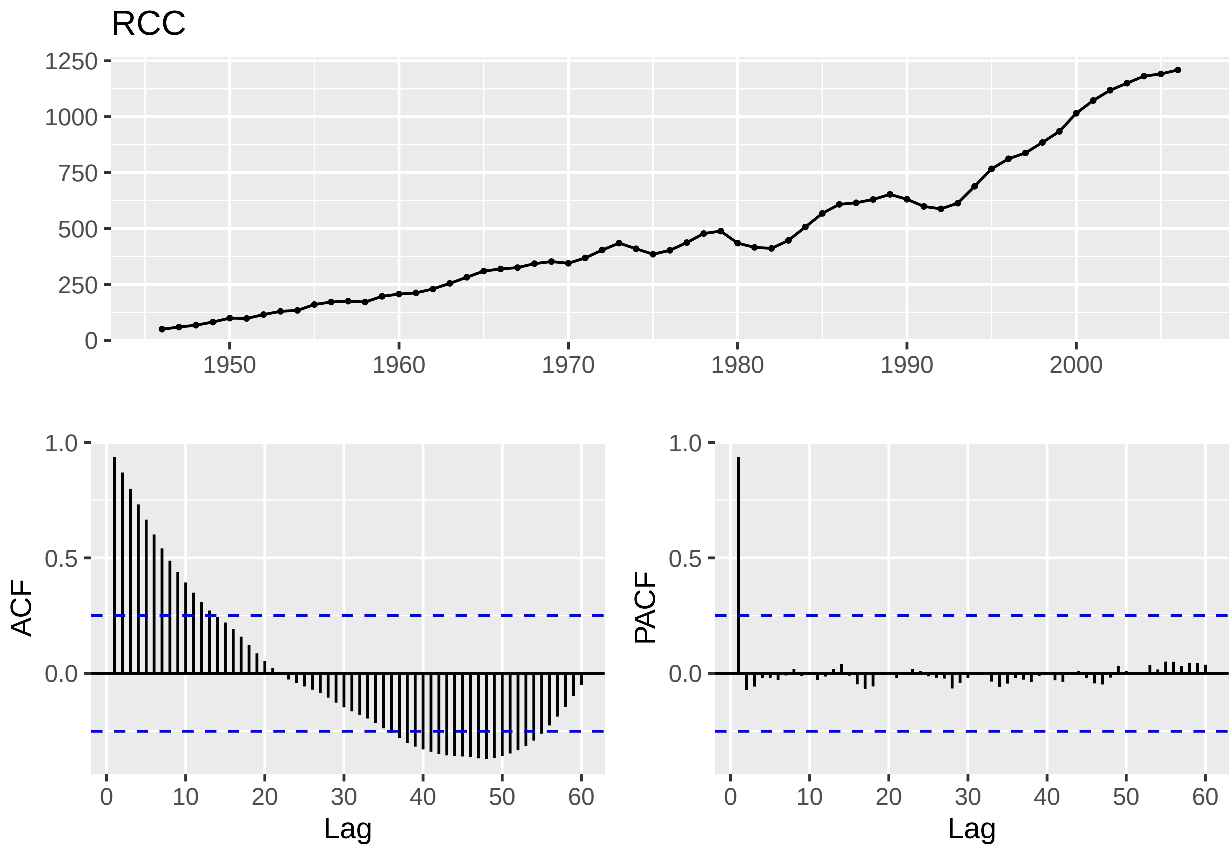
```r
credit.tsibble[,2:4] %>% GGally::ggpairs()
```



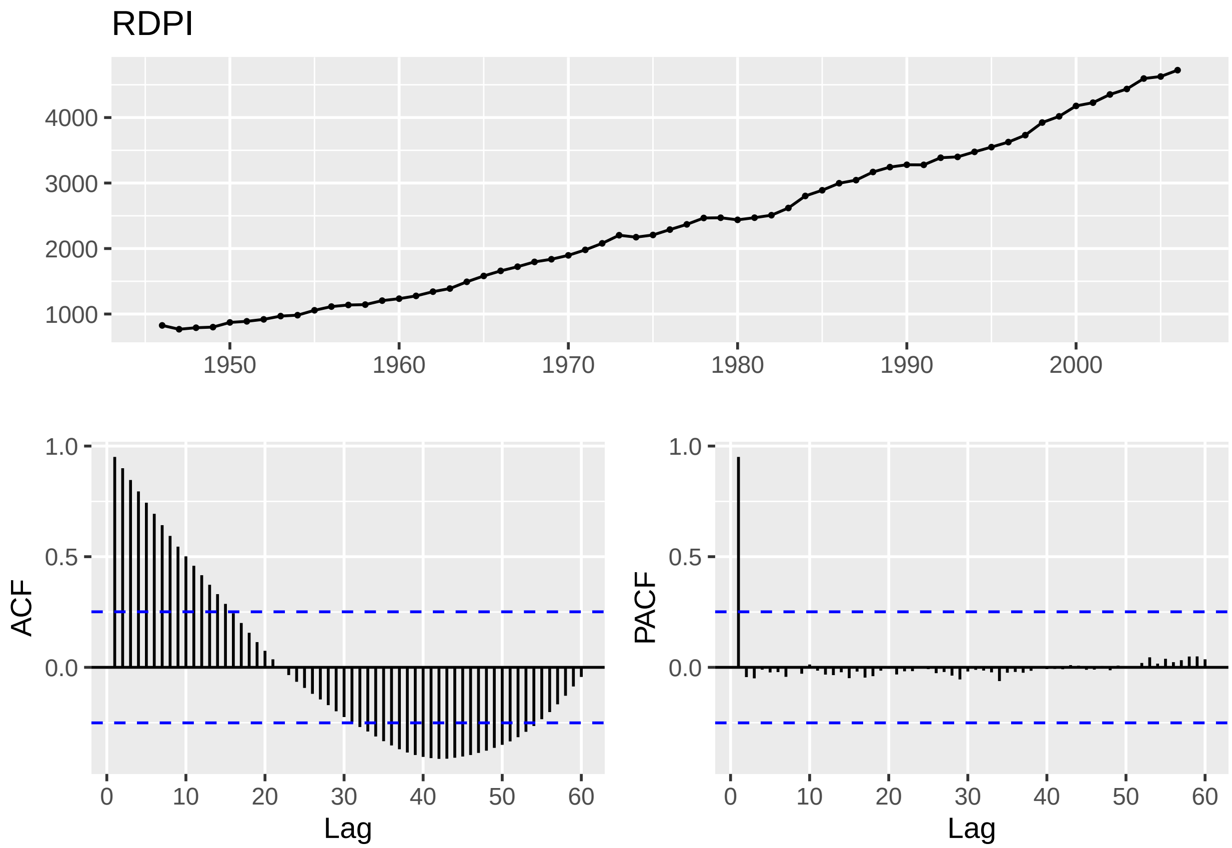

```
rmc.ts %>% ggtsdisplay(lag.max = 144, main = "RMC")
```




```r
rcc.ts %>% ggtsdisplay(lag.max = 144, main = "RCC")
```




```
rdpi.ts %>% ggtsdisplay(lag.max = 144, main = "RDPI")
```




```r
# ADF Test (Each is not stationary)
adf.test(credit.tsibble$RMC)
```

```
## Warning in adf.test(credit.tsibble$RMC): p-value greater than printed p-value

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

```r
adf.test(credit.tsibble$RCC)
```

```
## Warning in adf.test(credit.tsibble$RCC): p-value greater than printed p-value

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

```r
adf.test(credit.tsibble$RDPI)
```

```
##
```

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

```
# PO Test (The series are not co-integrated)
po.test(cbind(rmc.ts, rcc.ts))
```

```
## Warning in po.test(cbind(rmc.ts, rcc.ts)): p-value greater than printed p-value

##
##  Phillips-Ouliaris Cointegration Test
##
## data:  cbind(rmc.ts, rcc.ts)
## Phillips-Ouliaris demeaned = 2.1239, Truncation lag parameter = 0,
## p-value = 0.15
```

```
po.test(cbind(rmc.ts, rdpi.ts))
```

```
## Warning in po.test(cbind(rmc.ts, rdpi.ts)): p-value greater than printed p-value

##
##  Phillips-Ouliaris Cointegration Test
##
## data:  cbind(rmc.ts, rdpi.ts)
## Phillips-Ouliaris demeaned = 6.795, Truncation lag parameter = 0,
## p-value = 0.15
```

```
po.test(cbind(rcc.ts, rdpi.ts))
```

```
## Warning in po.test(cbind(rcc.ts, rdpi.ts)): p-value greater than printed p-value

##
##  Phillips-Ouliaris Cointegration Test
##
## data:  cbind(rcc.ts, rdpi.ts)
## Phillips-Ouliaris demeaned = -1.879, Truncation lag parameter = 0,
## p-value = 0.15
```

```
po.test(credit.tsibble)
```

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

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

```
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 = -0.61932, Truncation lag parameter = 0,
## p-value = 0.15
```

```r
# Select the lag parameter based on SC, In our case p = 2
VARselect(credit.ts, lag.max = 8, type = "both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      8      8      2      8
##
## $criteria
##                      1               2               3               4
## AIC(n)        21.46434        20.30492        20.02378        20.02587
## HQ(n)         21.67878        20.64802        20.49554        20.62629
## SC(n)         22.02197        21.19713        21.25056        21.58723
## FPE(n) 2101742418.92761 662739198.46684 506044326.78074 517413441.17604
##                     5               6               7               8
## AIC(n)       20.19057        20.11386        19.76309        19.24910
## HQ(n)        20.91966        20.97161        20.74950        20.36418
## SC(n)        22.08651        22.34438        22.32819        22.14878
## FPE(n) 629835999.55711 611586724.65094 460546913.09506 302181439.36216
```

```r
#Select Model
credit.var <- VAR(credit.ts, p = 2, type = "both")
summary(credit.var)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: RMC, RCC, RDPI
## Deterministic variables: both
## Sample size: 59
## Log Likelihood: -819.208
## Roots of the characteristic polynomial:
## 1.052 0.9147 0.9147 0.8403 0.6232 0.141
## Call:
## VAR(y = credit.ts, p = 2, type = "both")
##
##
## Estimation results for equation RMC:
## ====================================
## RMC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##          Estimate Std. Error t value             Pr(>|t|)
```

```
## RMC.l1    1.68342    0.13105  12.846 < 0.0000000000000002 ***
## RCC.l1    0.33188    0.26853   1.236               0.222    
## RDPI.l1   0.06852    0.16879   0.406               0.686    
## RMC.l2   -0.74294    0.14502  -5.123          0.00000466 ***
## RCC.l2   -0.01137    0.27301  -0.042               0.967    
## RDPI.l2  -0.02714    0.15236  -0.178               0.859    
## const   -30.28938   32.58617  -0.930               0.357    
## trend    -3.81748    3.65732  -1.044               0.302    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## 
## Residual standard error: 38.47 on 51 degrees of freedom
## Multiple R-Squared: 0.999,   Adjusted R-squared: 0.9988 
## F-statistic:  7096 on 7 and 51 DF,  p-value: < 0.00000000000000022
## 
## 
## Estimation results for equation RCC: 
## ==================================== 
## RCC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend 
## 
##         Estimate Std. Error t value            Pr(>|t|)    
## RMC.l1  -0.04280    0.06938  -0.617               0.540    
## RCC.l1   1.55176    0.14216  10.915 0.00000000000000598 ***
## RDPI.l1  0.03879    0.08936   0.434               0.666    
## RMC.l2   0.08378    0.07677   1.091               0.280    
## RCC.l2  -0.70568    0.14453  -4.883 0.00001074279726294 ***
## RDPI.l2 -0.04598    0.08066  -0.570               0.571    
## const    7.54643   17.25118   0.437               0.664    
## trend    1.21727    1.93619   0.629               0.532    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## 
## Residual standard error: 20.37 on 51 degrees of freedom
## Multiple R-Squared: 0.9964,  Adjusted R-squared: 0.9959 
## F-statistic:  2009 on 7 and 51 DF,  p-value: < 0.00000000000000022
## 
## 
## Estimation results for equation RDPI: 
## ===================================== 
## RDPI = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend 
## 
##         Estimate Std. Error t value Pr(>|t|)    
## RMC.l1   0.08686    0.14649   0.593 0.555845    
## RCC.l1   0.53623    0.30017   1.786 0.079982 .  
## RDPI.l1  0.73800    0.18868   3.911 0.000272 ***
## RMC.l2  -0.08889    0.16211  -0.548 0.585849    
```

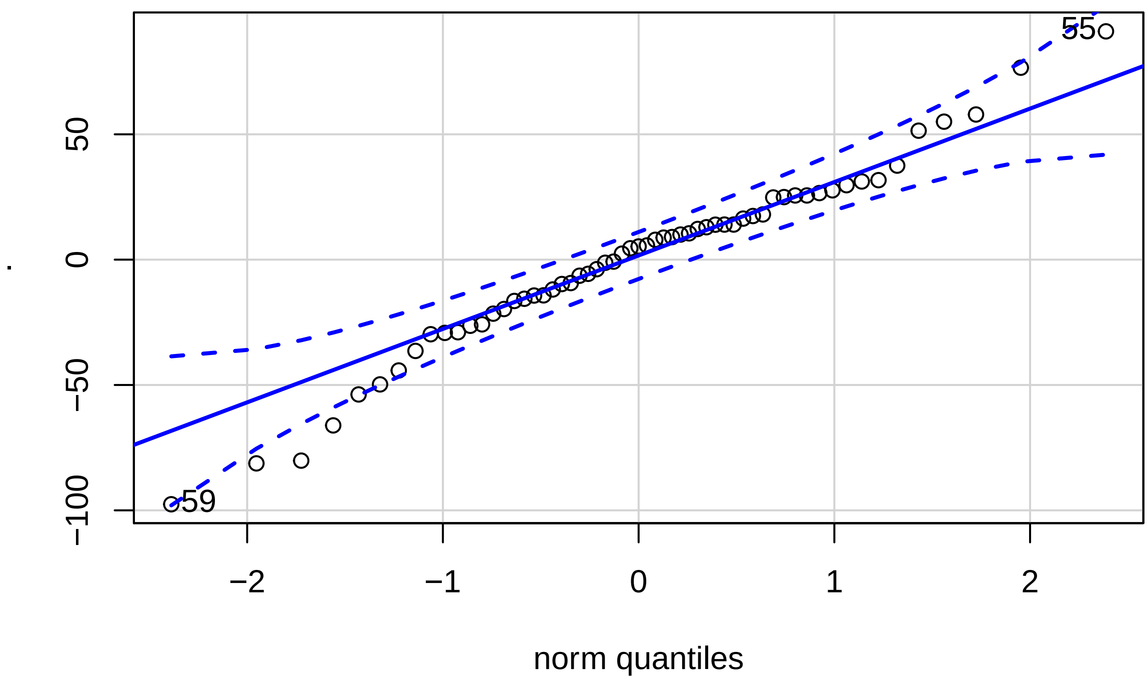
```
## RCC.l2  -0.37914    0.30518  -1.242 0.219783
## RDPI.l2  0.09435    0.17032   0.554 0.582009
## const   89.58746   36.42576   2.459 0.017346 *
## trend    9.26399    4.08826   2.266 0.027726 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 43.01 on 51 degrees of freedom
## Multiple R-Squared: 0.9988,  Adjusted R-squared: 0.9986
## F-statistic:  6092 on 7 and 51 DF,  p-value: < 0.0000000000000022
##
##
##
## Covariance matrix of residuals:
##         RMC   RCC RDPI
## RMC  1480.3 425.7  729
## RCC   425.7 414.9  651
## RDPI  729.0 651.0 1850
##
## Correlation matrix of residuals:
##         RMC    RCC   RDPI
## RMC  1.0000 0.5432 0.4405
## RCC  0.5432 1.0000 0.7431
## RDPI 0.4405 0.7431 1.0000
```

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

```
## $JB
##
##  JB-Test (multivariate)
##
## data:  Residuals of VAR object credit.var
## Chi-squared = 40.106, df = 6, p-value = 0.0000004342
##
##
## $Skewness
##
##  Skewness only (multivariate)
##
## data:  Residuals of VAR object credit.var
## Chi-squared = 8.4757, df = 3, p-value = 0.03714
##
##
## $Kurtosis
##
##  Kurtosis only (multivariate)
```

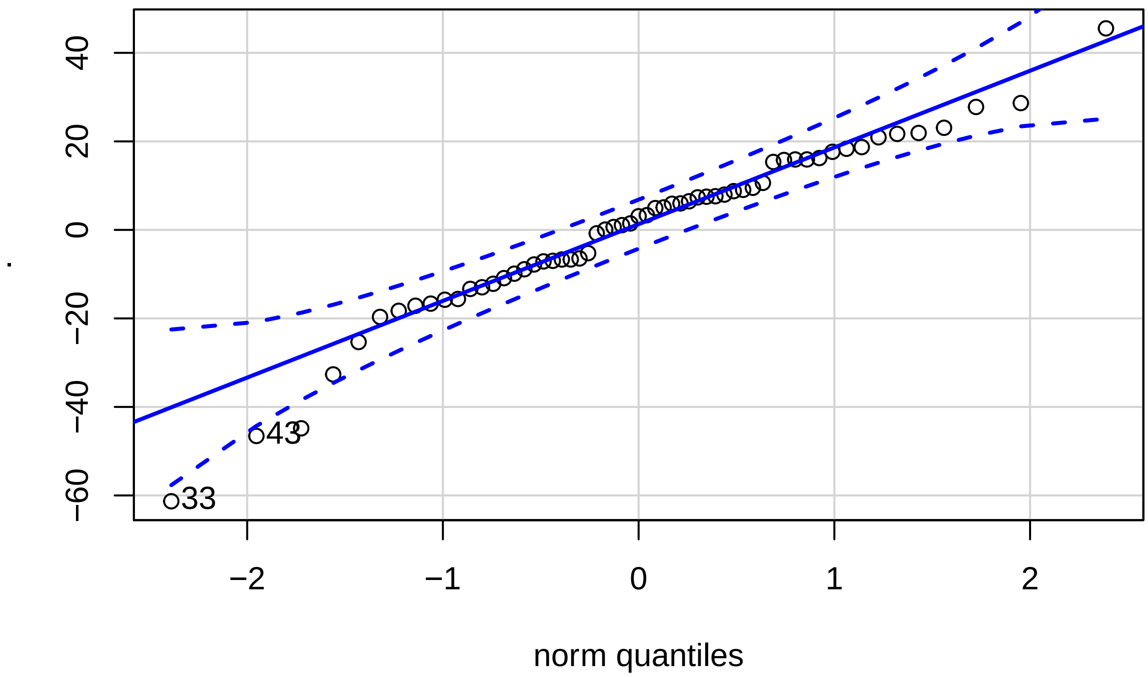
```
##
## data:  Residuals of VAR object credit.var
## Chi-squared = 31.63, df = 3, p-value = 0.0000006262
```

```
credit.var %>% resid %>% .[, "RMC"] %>% qqPlot
```



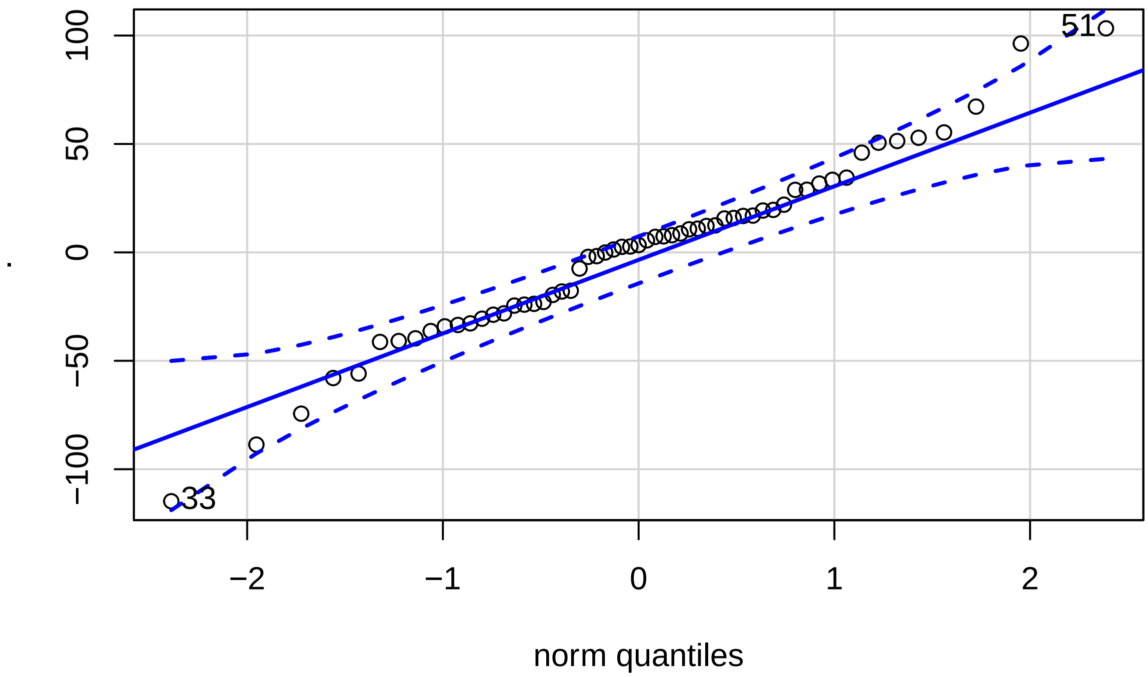

```
## [1] 59 55
```

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




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

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




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

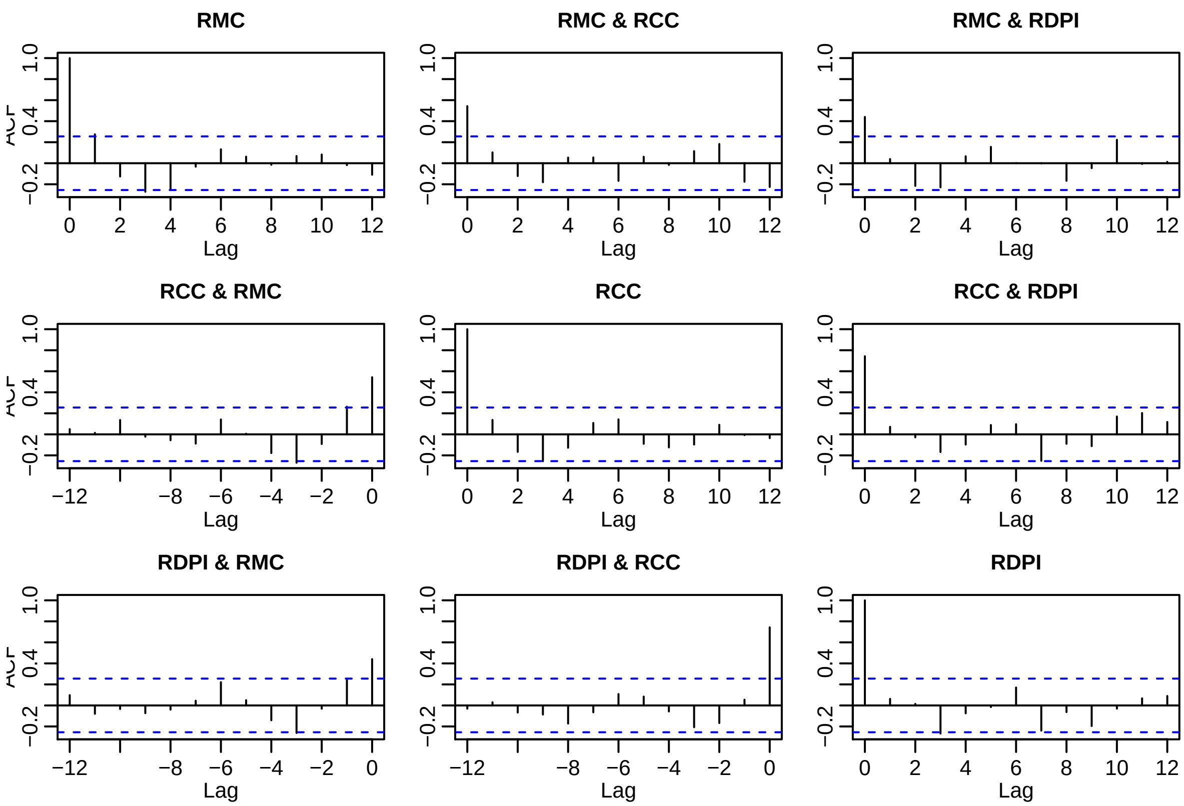
```r
# Test of no serial correlation:
credit.var.ptasy <- serial.test(credit.var, lags.pt = 12, type = "PT.asymptotic")
credit.var.ptasy
```

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

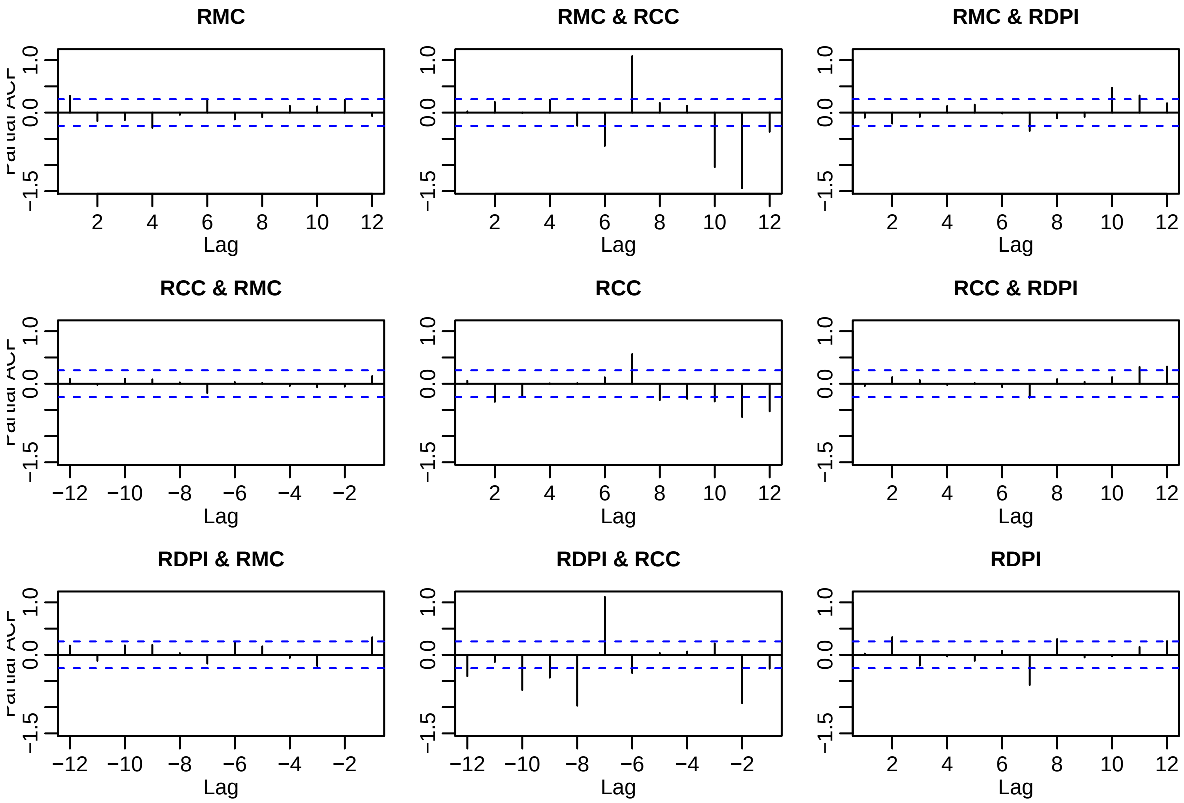
```r
# Test of the absence of ARCH effect:
credit.var.arch <- arch.test(credit.var)
credit.var.arch
```

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

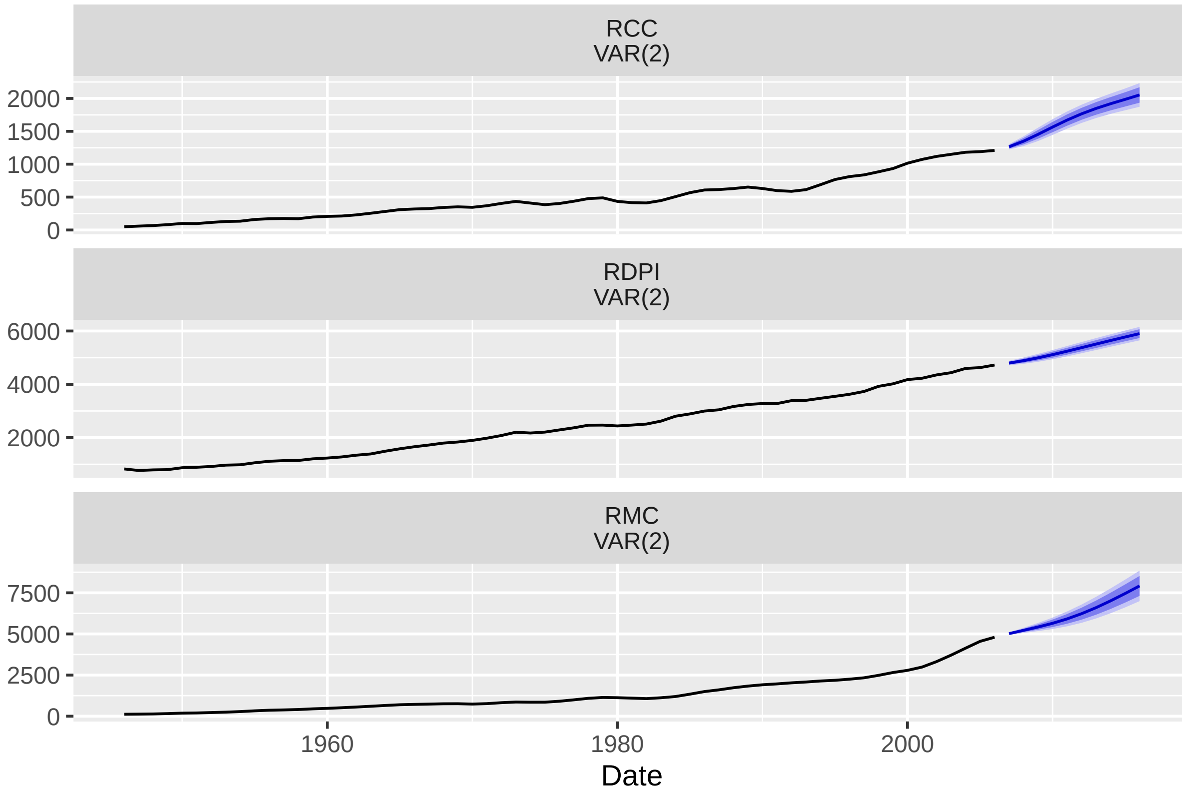
```r
credit.var %>% resid %>% acf
```



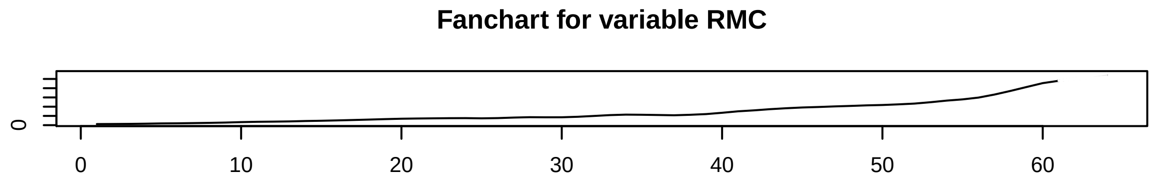

```r
credit.var %>% resid %>% pacf
```

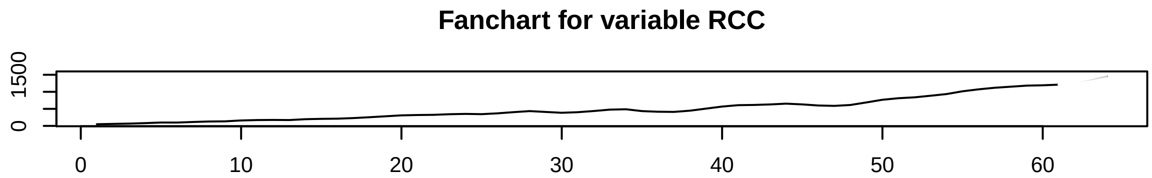


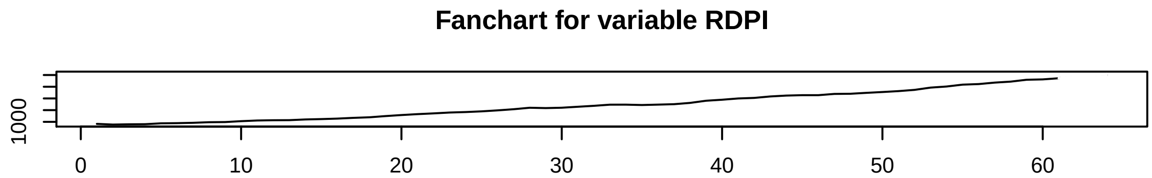

```r
forecast(credit.var) %>%
  autoplot() + xlab("Date")
```




```r
credit.var %>% predict(n.ahead = 3, ci = 0.95) %>% fanchart()
```


Fanchart for variable RMC
0
0
10
20
30
40
50
60


Fanchart for variable RCC
0
1500
0
10
20
30
40
50
60


Fanchart for variable RDPI
1000
0
10
20
30
40
50
60