W271 Individual Assignment 3

Ren Tu

We will begin by loading libraries required for this assignment.

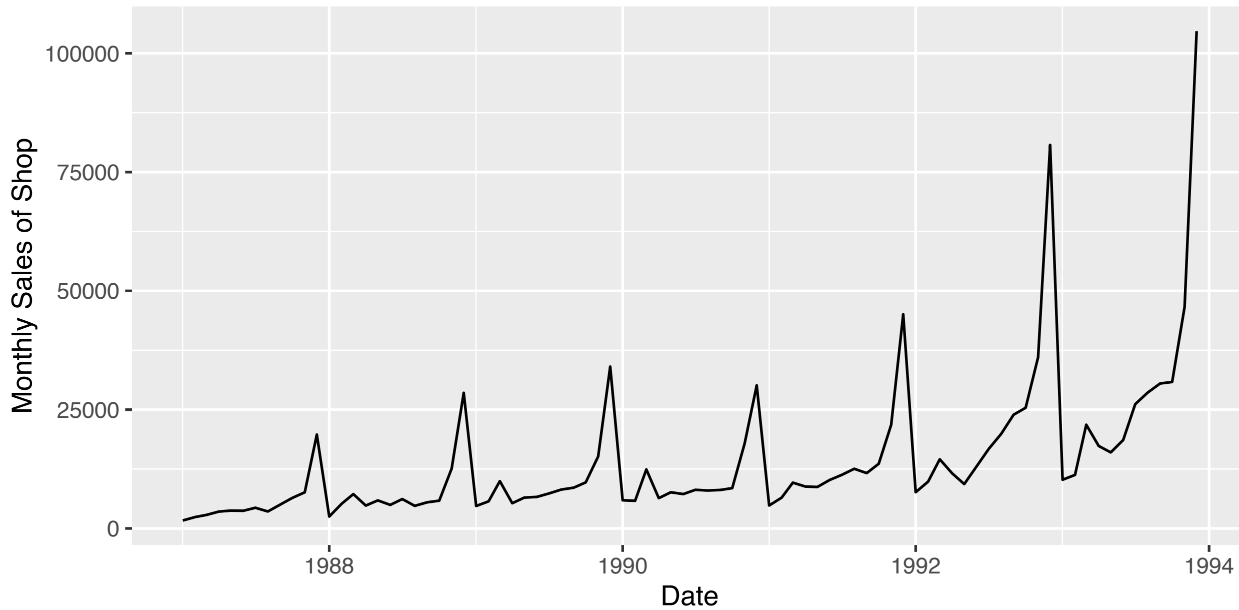
```
library(ggplot2)
library(tsibble)
library(dplyr)
library(lubridate)
library(fable)
library(fabletools)
library(stats)
library(feasts)
library(forecast)
library(fma)
library(lmtest)
library(fpp3)
library(car)
library(vars)
library(tseries)
```

Question 1: Time Series Linear Model

Part 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.

We will begin by loading the data and plotting the time series:

```
### Read in Q1 dataset
shop <- read.csv("Q1.csv")
shop <- shop %>% mutate(time_index=row_number())

### Convert Q1 dataset to tsibble
monthly.interval <- ymd("1987-01-01") + months(seq(from=0, length.out=84, by=1))
obs <- as.numeric(shop$sales)
shop.ts <- tsibble(Date=monthly.interval, Sales = obs, index=Date)
shop.ts <- shop.ts %>% mutate(time_index=row_number())

### Time plot of shop data
shop.ts %>% autoplot(Sales) + labs(x="Date", y="Monthly Sales of Shop")
```



The above time plot shows a gradual upward trend with seasonal spikes that are most pronounced in December of every year. This is consistent with the large influx of visitors that the shop has every Christmas. In addition, there are smaller seasonal spikes around March of every year since 1988 in line with the local surfing festival. Unusual patterns occur in the later years in the dataset in which the sales are on faster increasing trends within the second halves of 1991 through 1993. In addition, the December spikes in 1993 and 1994 are much higher than what the gradual growth of December spikes in prior years would suggest.

Part B: Explain why it is necessary to take logarithms of these data before fitting a model.

Due to the sales numbers in the later years of 1991 to 1993 growing to multiples of the values in the earlier years, taking logarithms of the data can better facilitate models that satisfy the underlying model assumptions such as homoskedasticity and normality of residuals.

Part C: Fit a regression model to the logarithms of these sales data with a linear trend, seasonal dummies and a "surfing festival" dummy variable.

We will create a "surfing festival" dummy variable that has a value of 1 for every March since 1988. Then we will fit a TSLM model with a linear trend and seasonal dummies:

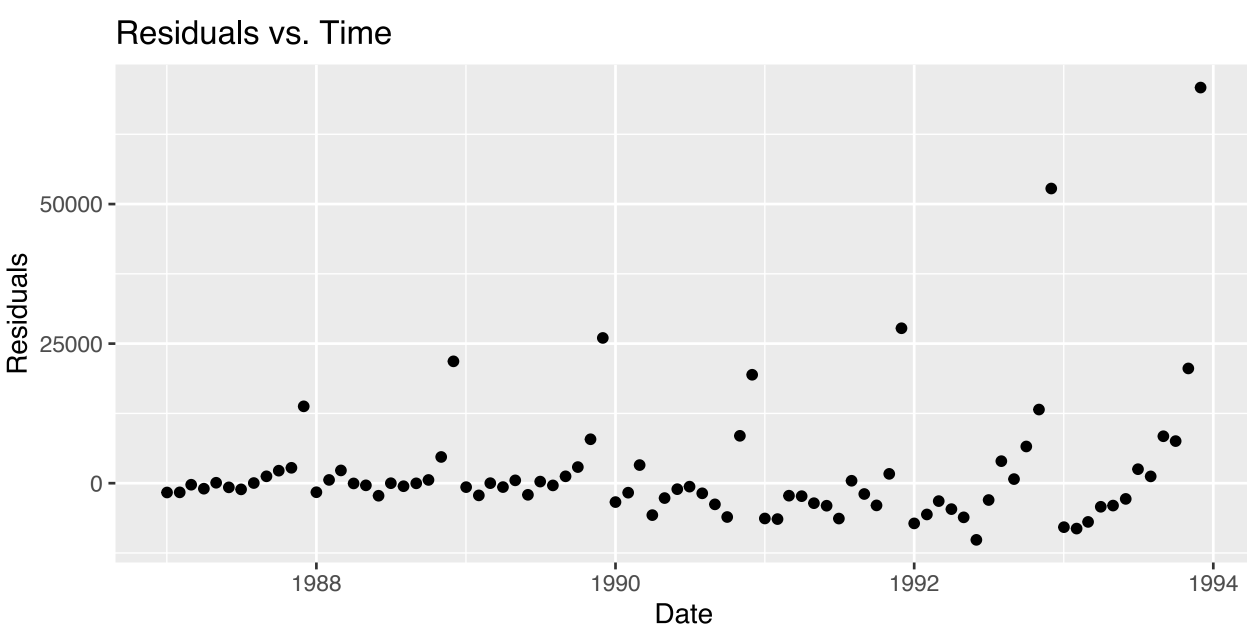
```
### Create surfing festival dummy variable
surf <- append(rep(0,12), rep(c(0,0,1,0,0,0,0,0,0,0,0,0), 6))
shop.ts$surf <- surf

### Fit TSLM model with log of sales as dependent variable
q1.model <- shop.ts %>% model(TSLM(log(Sales) ~ trend() + surf + season(period=12)))
report(q1.model)
```

```
## Series: Sales
## Model: TSLM
## Transformation: log(.x)
##
## Residuals:
##      Min       1Q  Median       3Q      Max
## -0.8370 -0.3064 -0.1000  0.2362  1.4472
##
## Coefficients:
```

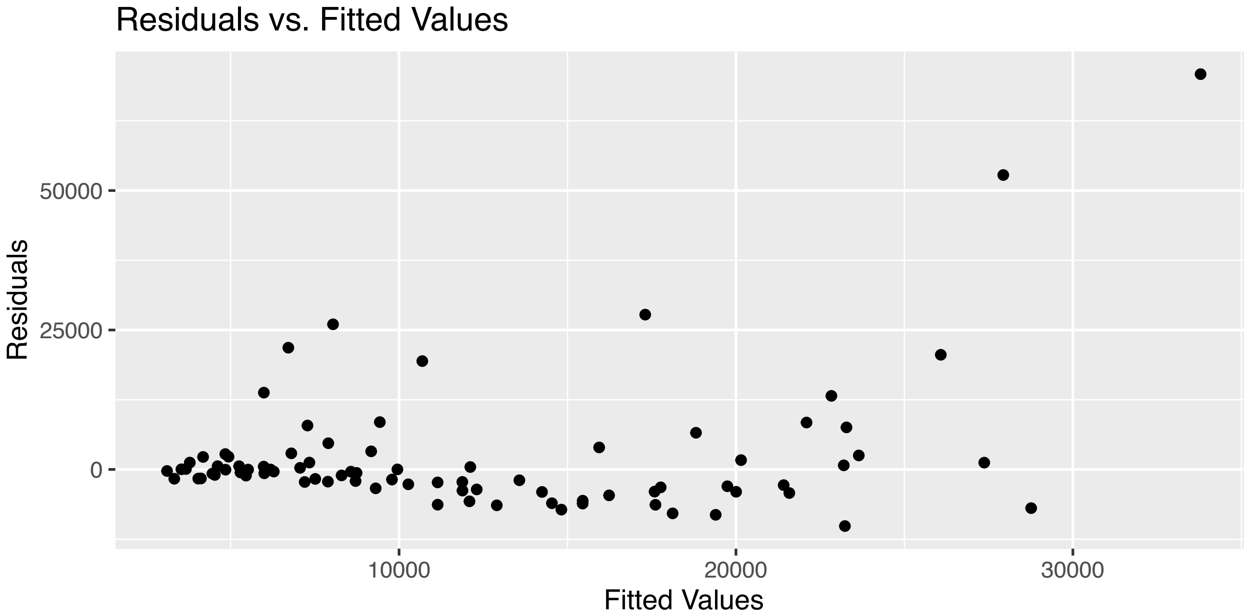
```
##                                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)                     8.278e+00  2.109e-01  39.242  < 2e-16 ***
## trend()                         8.247e-04  7.996e-05  10.313 1.09e-15 ***
## surf                            1.319e-01  2.357e-01   0.560    0.578    
## season(period = 12)season_122  -2.815e-01  2.880e-01  -0.977    0.332    
## season(period = 12)season_123  -1.646e-01  2.680e-01  -0.614    0.541    
## season(period = 12)season_124   1.436e-01  2.871e-01   0.500    0.619    
## season(period = 12)season_125  -2.835e-01  2.766e-01  -1.025    0.309    
## season(period = 12)season_126  -1.680e-01  2.599e-01  -0.646    0.520    
## season(period = 12)season_127   1.769e-01  3.043e-01   0.581    0.563    
## season(period = 12)season_128  -1.808e-01  2.884e-01  -0.627    0.533    
## season(period = 12)season_129  -2.389e-01  2.663e-01  -0.897    0.373    
## season(period = 12)season_1210 -4.389e-02  2.678e-01  -0.164    0.870    
## season(period = 12)season_1211 -2.560e-01  2.871e-01  -0.892    0.376    
## season(period = 12)season_1212  6.626e-02  2.766e-01   0.240    0.811    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.5313 on 70 degrees of freedom
## Multiple R-squared: 0.6187,  Adjusted R-squared: 0.5479 
## F-statistic: 8.737 on 13 and 70 DF,  p-value: 3.041e-10
```

Part D: Plot the residuals against time and against fitted values. Do these plots reveal any problems with the model?

```r
### Residuals vs. Time Plot
q1.model.fitted <- augment(q1.model)
ggplot(data=q1.model.fitted, aes(x=Date, y=.resid)) +
  geom_point() + labs(x="Date", y="Residuals", title="Residuals vs. Time")
```



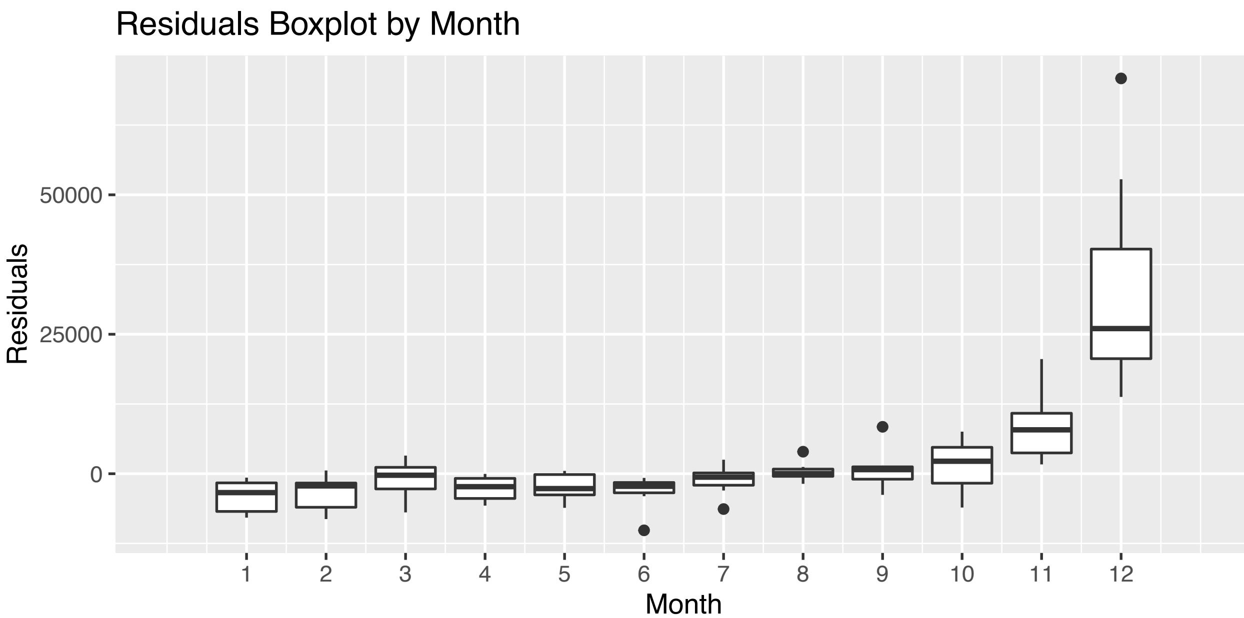

```r
### Residuals vs. Fitted Values Plot
ggplot(data=q1.model.fitted, aes(x=.fitted, y=.resid)) +
  geom_point() + labs(x="Fitted Values", y="Residuals", title="Residuals vs. Fitted Values")
```



The residuals vs. time plot reveal that the residuals have seasonality and are serially correlated. The residuals vs. fitted values plot indicate that the variance of the residuals tend to increase as fitted values increase. These characteristics are not ideal as they violate many of the assumptions of linear regression.

Part E: Do boxplots of the residuals for each month. Does this reveal any problems with the model?

```
### Boxplot
q1.model.fitted$month <- rep(c(1,2,3,4,5,6,7,8,9,10,11,12), 7)
ggplot(data=q1.model.fitted, aes(x=month, y=.resid, group=month)) +
  geom_boxplot() +
  scale_x_continuous(limits=c(0,13), breaks=c(1,2,3,4,5,6,7,8,9,10,11,12)) +
  labs(x="Month", y="Residuals", title="Residuals Boxplot by Month")
```



The boxplots demonstrate that the residuals have a seasonal pattern in which residuals tend to increase from generally negative values in the first 6 months to generally positive values over the latter 6 months. In addition, the magnitude and variance of the residuals grow exponentially in November and December. These reaffirm the serial correlation and heteroskedasticity problems discussed in Part D.

Part F: What do the values of the coefficients tell you about each variable?

Let's take a look at the model coefficients:

```r
### Display model coefficients
report(q1.model)
```

```
## Series: Sales
## Model: TSLM
## Transformation: log(.x)
##
## Residuals:
##      Min       1Q  Median      3Q     Max
## -0.8370 -0.3064 -0.1000  0.2362  1.4472
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      8.278e+00  2.109e-01  39.242  < 2e-16 ***
## trend()                          8.247e-04  7.996e-05  10.313 1.09e-15 ***
## surf                             1.319e-01  2.357e-01   0.560    0.578
## season(period = 12)season_122  -2.815e-01  2.880e-01  -0.977    0.332
## season(period = 12)season_123  -1.646e-01  2.680e-01  -0.614    0.541
## season(period = 12)season_124   1.436e-01  2.871e-01   0.500    0.619
## season(period = 12)season_125  -2.835e-01  2.766e-01  -1.025    0.309
## season(period = 12)season_126  -1.680e-01  2.599e-01  -0.646    0.520
## season(period = 12)season_127   1.769e-01  3.043e-01   0.581    0.563
## season(period = 12)season_128  -1.808e-01  2.884e-01  -0.627    0.533
## season(period = 12)season_129  -2.389e-01  2.663e-01  -0.897    0.373
## season(period = 12)season_1210 -4.389e-02  2.678e-01  -0.164    0.870
## season(period = 12)season_1211 -2.560e-01  2.871e-01  -0.892    0.376
```

```
## season(period = 12)season_1212  6.626e-02  2.766e-01   0.240    0.811
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5313 on 70 degrees of freedom
## Multiple R-squared: 0.6187,  Adjusted R-squared: 0.5479
## F-statistic: 8.737 on 13 and 70 DF, p-value: 3.041e-10
```

The positive coefficient for the trend component suggests that there is a gradual upward trend in the time series. The positive surfing dummy coefficient is consistent with our expectation that the March surfing festival tends to increase visitors and sales. Among the monthly seasonal dummy coefficients, positive values such as April, July and December indicate that these months tend to increase sales relative to the January base level. On the other hand, negative values such as February, May and June indicate that these months tend to decrease sales relative to the January base level.

Part G: What does the Breusch-Godfrey test tell you about your model?

We will conduct a Breusch-Godfrey test at a 5% level of significance:

```
### Breusch-Godfrey test using model residuals vs. time index
bgtest(q1.model.fitted$.resid ~ shop.ts$time_index)
```

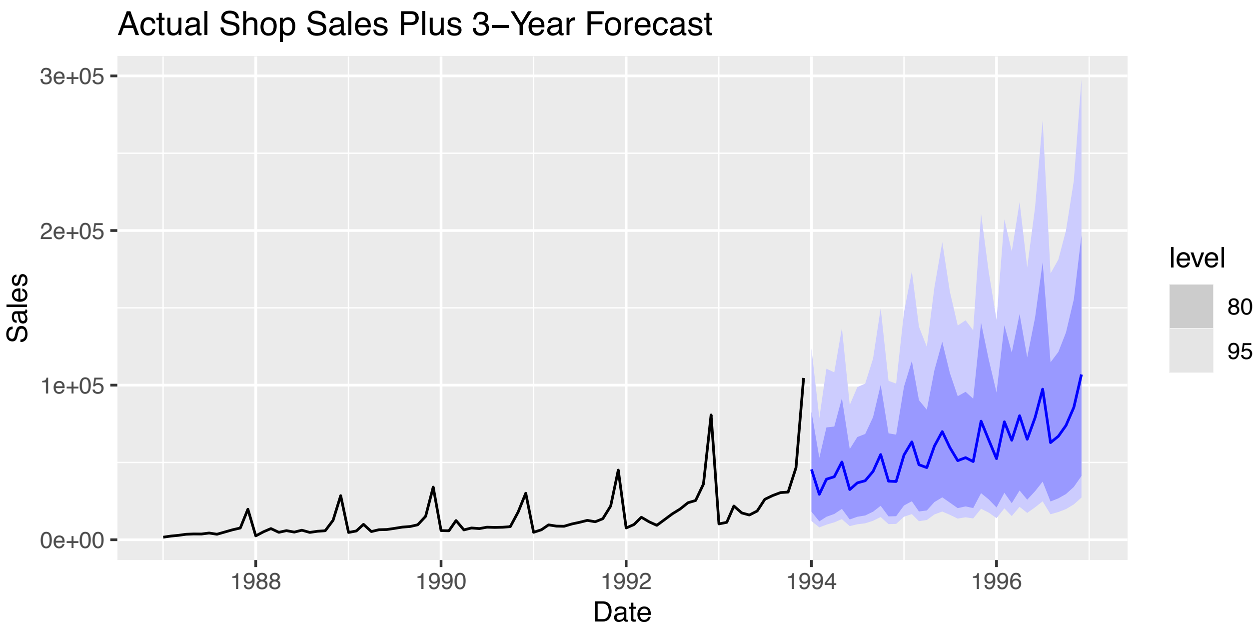
```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  q1.model.fitted$.resid ~ shop.ts$time_index
## LM test = 5.715, df = 1, p-value = 0.01682
```

With a p-value of 0.01682, we reject the null hypothesis of no serial correlation among the residuals. This suggests evidence of the existence of serial correlation in our model residuals.

Part H: Use your regression model to predict the monthly sales for 1994, 1995, and 1996. Produce prediction intervals for each of your forecasts.

We will forecast for three years using our TSLM model object and plot the forecasts and 80%/95% prediction intervals:

```
### Forecast for 1994 to 1996
monthly.interval2 <- ymd("1987-01-01") + months(seq(from=0, length.out=120, by=1))
dummyobs <- rep(0,120)
newdata1 <- tsibble(Date=monthly.interval2, Sales = dummyobs, index=Date)
newdata1 <- newdata1 %>% mutate(time_index=row_number())
surf2 <- append(rep(0,12), rep(c(0,0,1,0,0,0,0,0,0,0,0,0), 9))
newdata1$surf <- surf2
fc.q1 <- q1.model %>% forecast(newdata1)
fc.q1 %>% dplyr::filter(year(Date)>=1994) %>%
  autoplot(shop.ts) + ggtitle("Actual Shop Sales Plus 3-Year Forecast")
```



Part I: Transform your predictions and intervals to obtain predictions and intervals for the raw data

We will extract the raw predictions and intervals from our forecast object in Part H:

```
### Extract prediction estimates and interval values
fc.q1.raw1 <- hilo(fc.q1, 80)
fc.q1.raw.80 <- unpack_hilo(data=fc.q1.raw1, cols=`80%`)
fc.q1.raw2 <- hilo(fc.q1, 95)
fc.q1.raw.95 <- unpack_hilo(data=fc.q1.raw2, cols=`95%`)
q1.pred <- tsibble(Date=monthly.interval2,
                   estimate = round(fc.q1.raw.80$.mean,0),
                   lower80 = round(fc.q1.raw.80$`80%_lower`,0),
                   upper80 = round(fc.q1.raw.80$`80%_upper`,0),
                   lower95 = round(fc.q1.raw.95$`95%_lower`,0),
                   upper95 = round(fc.q1.raw.95$`95%_upper`,0),
                   index=Date) %>% dplyr::filter(year(Date)>=1994)
head(q1.pred, 18)
```

```
## # A tsibble: 18 x 6 [1D]
##    Date       estimate lower80 upper80 lower95 upper95
##    <date>        <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 1994-01-01    45508   18133   82692   12135  123565
##  2 1994-02-01    29380   11905   52994    8019   78682
##  3 1994-03-01    39154   14854   72660    9758  110607
##  4 1994-04-01    40759   16682   73194   11278  108260
##  5 1994-05-01    50289   19972   91509   13349  136910
##  6 1994-06-01    32463   13117   58627    8825   87139
##  7 1994-07-01    36795   14877   66433   10012   98717
##  8 1994-08-01    38181   15705   68413   10638  100996
##  9 1994-09-01    44217   18152   79296   12287  117148
## 10 1994-10-01    55121   21973  100137   14708  149604
## 11 1994-11-01    37908   15132   68827   10133  102775
## 12 1994-12-01    37662   15234   67987   10253  101011
## 13 1995-01-01    54828   22161   99006   14911  147138
```

```
## 14 1995-02-01    63295   24923  115605   16604  173522
## 15 1995-03-01    48533   18302   90293   11996  137758
## 16 1995-04-01    46659   18944   84087   12769  124752
## 17 1995-05-01    60601   24394  109628   16388  163180
## 18 1995-06-01    69962   27434  128010   18248  192445
```

```r
tail(q1.pred, 18)
```

```
## # A tsibble: 18 x 6 [1D]
##    Date       estimate lower80 upper80 lower95 upper95
##    <date>        <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 1995-07-01    59623   24028  107803   16150  160391
##  2 1995-08-01    51165   20458   92830   13709  138528
##  3 1995-09-01    53107   21578   95677   14548  141908
##  4 1995-10-01    50576   20496   91221   13805  135433
##  5 1995-11-01    76727   30140  140282   20062  210754
##  6 1995-12-01    64672   26155  116749   17603  173467
##  7 1996-01-01    52420   20899   95225   13990  142257
##  8 1996-02-01    76315   30399  138687   20342  207257
##  9 1996-03-01    64332   23642  120985   15347  186382
## 10 1996-04-01    80242   31885  145979   21316  218358
## 11 1996-05-01    64981   25949  117961   17380  176116
## 12 1996-06-01    78966   31386  143643   20984  214844
## 13 1996-07-01    97412   37608  179430   24870  271338
## 14 1996-08-01    62830   24746  114742   16488  172211
## 15 1996-09-01    66966   26789  121469   17955  181231
## 16 1996-10-01    73932   29586  134085   19833  200027
## 17 1996-11-01    85644   34173  155526   22882  232273
## 18 1996-12-01   106891   41260  196908   27282  297793
```

Part J: How could you improve these predictions by modifying the model?

We can consider a dummy variable for December to account for the large jumps in sales due to Christmas and reduce the residuals in December. Polynomial terms may provide a better fit to the accelerating sales trend. To address the serial correlation problems, we can consider an ARIMA model that can account for multiple components in the series (trend, seasonal, irregular) and transform the data to a stationary series with much less autocorrelation.

Question 2: Cross Validation

Part A: Define the accuracy measures returned by the "accuracy" function. Explain how the given code calculates these measures using cross-validation.

Here is a listing of the accuracy measures from the *accuracy* function:

ME: Mean error, or the sum of all errors divided by number of forecasts.

RMSE: Root mean squared error. This can be calculated by using the sum of the squared errors divided by the number of forecasts, then taking the square root.

MAE: Mean absolute error, or the sum of absolute values of all errors divided by number of forecasts.

MPE: Mean percentage error, or the sum of percentage deviations from actual values divided by the number of forecasts.

MAPE: Mean absolute percentage error, or the sum of absolute values of percentage deviations from actual values divided by the number of forecasts.

MASE: Mean absolute scaled error, or the mean absolute error of forecasts divided by mean absolute error of a base scale such as a naive forecast.

ACF1: Autocorrelation of errors relative to their first lag.

Let us examine the given code for cross validation:

```r
### Google stock cross validation
google_stock <- gafa_stock %>%
  filter(Symbol == "GOOG") %>%
  mutate(day = row_number()) %>%
  update_tsibble(index = day, regular = TRUE)

google_2015 <- google_stock %>% filter(year(Date) == 2015)

google_2015_train <- google_2015 %>%
  slice(1:(n()-1)) %>%
  stretch_tsibble(.init=3, .step=1)

fc.goog <- google_2015_train %>%
  model(RW(Close ~ drift())) %>%
  forecast(h=1)

fc.goog %>% 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
```

The code first uploads the Google stock data and filters for 2015 dates only. Then a training set is created by removing the final data point from the Google 2015 series and then stretching the remaining data into increasingly large partitions. The stretching would start with the first 3 dates of 2015 (Jan 2, Jan 5, Jan 6), then it would add a 4-date partition from Jan 2 to Jan 7, then it would add a 5-date partition from Jan 2 to Jan 8, and so on. Next, a drift model is estimated using the training set and a 1-date-ahead forecast is done for each partition. Finally, then accuracy function compares the 1-date-ahead forecasts with the actual values from the Google 2015 series to calculate the error measures.

Part B: Obtain Facebook stock data. Use cross-validation to compose the RMSE forecasting accuracy of naive and drift models for the Volume series.

We will first obtain Facebook stock data for 2015.

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

facebook_2015 <- facebook_stock %>% filter(year(Date) == 2015)
```

Now using the Facebook 2015 series, we will evaluate the accuracy of naive and drift models for Volume:

```
### Facebook 2015 cross validation
facebook_2015_train <- facebook_2015 %>%
  slice(1:(n()-1)) %>%
  stretch_tsibble(.init=3, .step=1)

fc.fb.naive <- facebook_2015_train %>%
  model(NAIVE(Volume)) %>%
  forecast(h=1)

fc.fb.drift <- facebook_2015_train %>%
  model(RW(Volume ~ drift())) %>%
  forecast(h=1)

fc.fb.naive %>% accuracy(facebook_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 NAIVE(Volume) FB     Test  -36549. 10713951. 7430153. -6.18  27.4  1.00 -0.139
```

```
fc.fb.drift %>% accuracy(facebook_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(Volume ~ dr~ FB     Test  -194181.  1.08e7 7.50e6 -6.74  27.8  1.01 -0.138
```

The Naive model had an RMSE of 10.7 million, slightly better than the Drift model RMSE of 10.8 million.

Question 3: ARIMA Model

Part A: Produce a time plot of the time series

We will begin by loading the sheep dataset and creating a time plot.

```
### Load sheep dataset and convert into tsibble
sheep <- fma::sheep
sheep.ts <- as_tsibble(sheep)

### Time Plot
sheep.ts %>% autoplot(value) + labs(x="Date", y="Sheep Population")
```

Part B: Assume you decide to fit the following model:

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

Given the differencing terms in the model above, there is 1 degree of differencing. Given the 3 differenced terms for lagged y values, the AR order is 3. Given the lone white noise term, the MA order is 0.

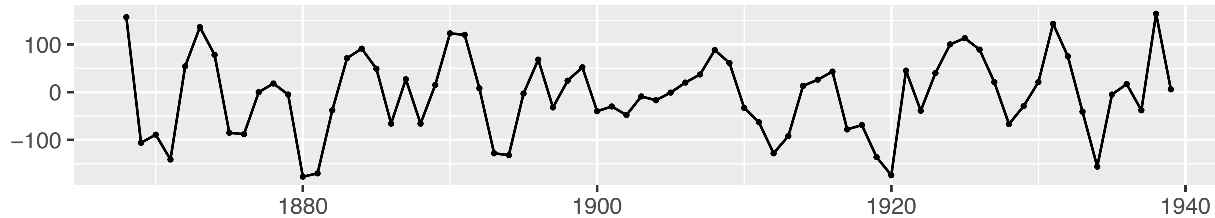
This is an $ARIMA(3,1,0)$ model, which can be expressed in the following notation:

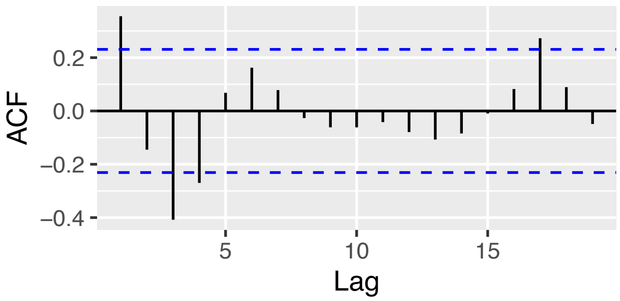
$$(1 - \phi_1 B - \phi_2 B^2 - \phi_3 B^3)(1 - B)y_t = \epsilon_t$$

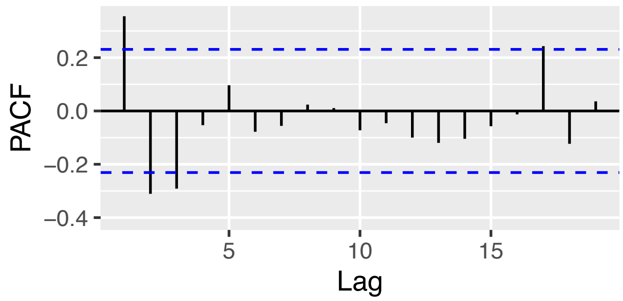
Part C: By examining the ACF and PACF of the differenced data, explain why this model is appropriate

We will look at the ACF and PACF plots of the differenced sheep dataset and see how they fit relative to an $ARIMA(3,1,0)$ model:

```
### Display ACF and PACF
sheep %>% diff() %>% ggtsdisplay()
```







The PACF of the differenced series has significant spikes in the first three lags, which indicates an AR(3) process. Since the MA(q) part of the $ARIMA(3,1,0)$ is zero, this model is equivalent to a differenced AR(3) model, consistent with the PACF spikes.

Part D: Calculate forecasts for the next three years

We will write a function to calculate forecasts based on the equation for our model and the given values of phi(0.42, -0.20, -0.30):

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

```
### Function for our model equation
predict.yt <- function(lag1, lag2, lag3, lag4) {
  phi1 <- 0.42
  phi2 <- -0.20
  phi3 <- -0.30
  epsilon <- 0
  yt <- lag1+phi1*(lag1-lag2)+phi2*(lag2-lag3)+phi3*(lag3-lag4)+epsilon
  yt
}
```

Using our custom function, we can calculate forecasts for the next 3 years (1940-1942):

```
### Forecast for 1940
fc.1940 <- predict.yt(1797, 1791, 1627, 1665)
print(paste("Forecast for 1940: ", round(fc.1940,2)))
```

```
## [1] "Forecast for 1940:  1778.12"
```

```
### Forecast for 1941
fc.1941 <- predict.yt(fc.1940, 1797, 1791, 1627)
print(paste("Forecast for 1941: ", round(fc.1941,2)))
```

```
## [1] "Forecast for 1941:  1719.79"
```

```
### Forecast for 1942
fc.1942 <- predict.yt(fc.1941, fc.1940, 1797, 1791)
print(paste("Forecast for 1942: ", round(fc.1942,2)))
```

```
## [1] "Forecast for 1942:  1697.27"
```

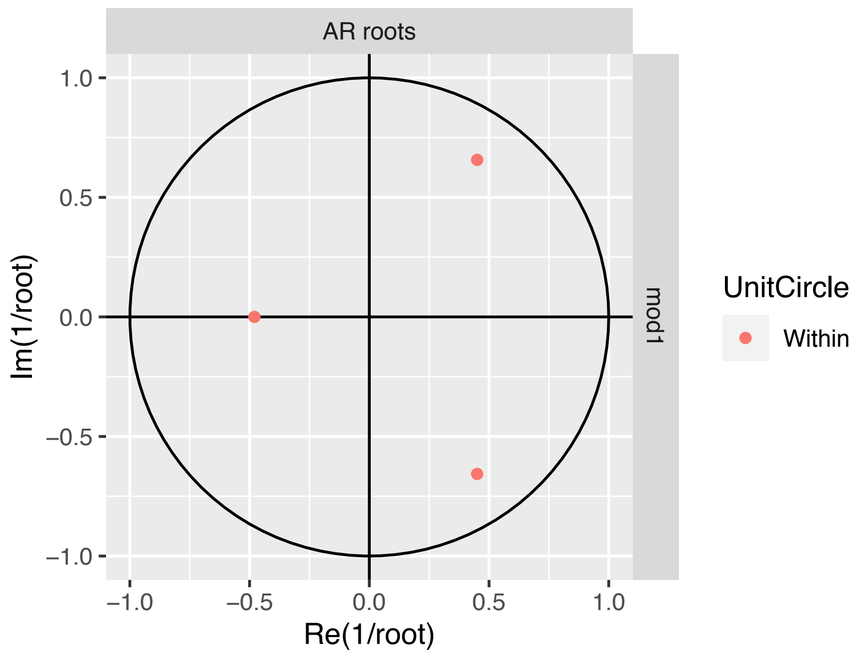
Part E: Find the roots of your model's characteristic equation and explain their significance

We will create our model object and examine model roots via the unit circle:

```
### Estimate ARIMA(3,1,0) model
q3.model <- sheep.ts %>% model(mod1 = ARIMA(value ~ pdq(3,1,0)))

### Look at model roots
glance(q3.model)[["ar_roots"]]
```

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

```
### Check that inverse roots are within unit circle
gg_arma(q3.model)
```



We have three complex AR roots and the inverses of each of these roots fall within the unit circle. This indicates that our $ARIMA(3,1,0)$ model satisfies stationarity conditions and we can have confidence to use this model for forecasting.

Question 4: Model Averaging

Part A: Apply a Holt-Winters model to the ECOMPCTNSA time series data. Compare this model's forecasting performance to that of a seasonal ARIMA model using cross-validation. Compare both of these models to the performance of a simple average of the ARIMA and Holt-Winters models.

We will begin by preparing the dataset.

```
### Read in ECOMPCTNSA dataset
q4 <- read.csv("Q4.csv")

### Convert Q4 dataset to tsibble
interval4 <- dmy(q4$DATE)
obs <- as.numeric(q4$ECOMPCTNSA)
eco.ts <- tsibble(Date=yearquarter(interval4), Eco = obs, index=Date)
eco.ts <- eco.ts %>% mutate(time_index=row_number())
```

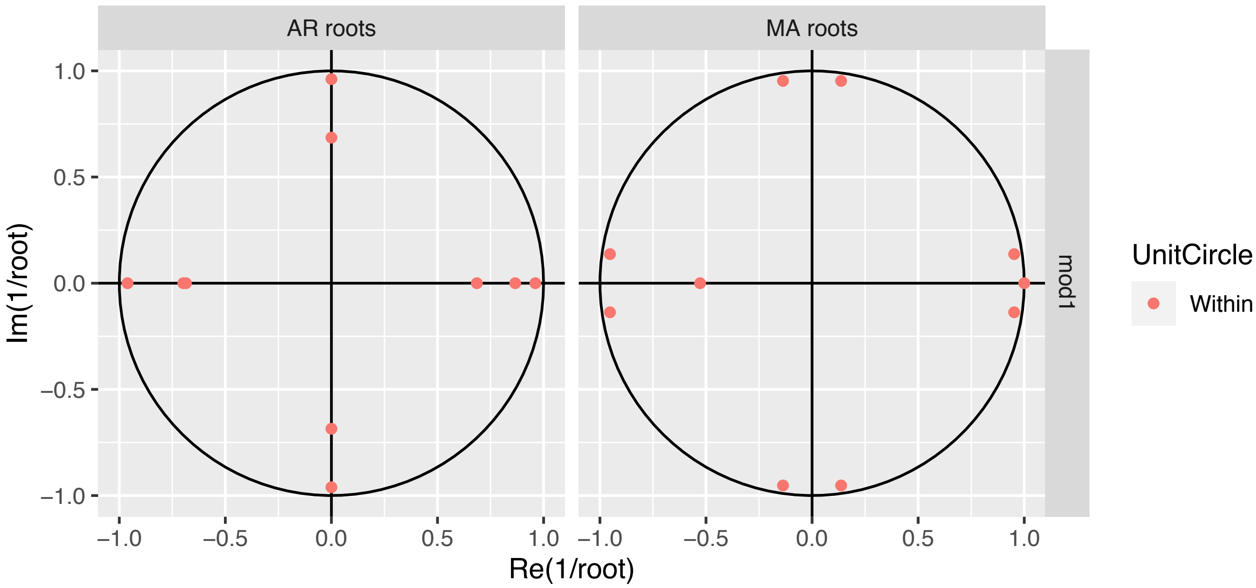
Before we compare the Holt-Winters and ARIMA models, we must choose a specific ARIMA model. We will use the looping method (Ps and Qs up to 2, Ds up to 1) to determine our ARIMA choice:

```
# ### Looping commented out to efficiently generate PDF
# ### Find best ARIMA models with best RMSE using loop method
# q4.arima <- data.frame(p=integer(), d=integer(), q=integer(),
#                         P=integer(), D=integer(), Q=integer(),
#                         RMSE=double(), AICc=double())
#
# for (p in 0:2) {
#   for (d in 0:1) {
#     for (q in 0:2) {
#       for (P in 0:2) {
#         for (D in 0:1) {
#           for (Q in 0:2) {
#             loop.model <- eco.ts %>%
#               model(a1=ARIMA(Eco ~ pdq(p,d,q)+PDQ(P,D,Q)+1))
#             loop.RMSE <- as.numeric(accuracy(loop.model)$RMSE)
#             loop.AICc <- as.numeric(glance(loop.model)$AICc)
#             q4.arima <- q4.arima %>%
#               add_row(p=p, d=d, q=q, P=P, D=D, Q=Q,
#                       RMSE=loop.RMSE, AICc=loop.AICc)
#             print(paste(p,d,q,P,D,Q,loop.RMSE,loop.AICc))
#           }
#         }
#       }
#     }
#   }
# }
#
# ### Best model based on RMSE
# q4.arima[which.min(q4.arima$RMSE), ]
```

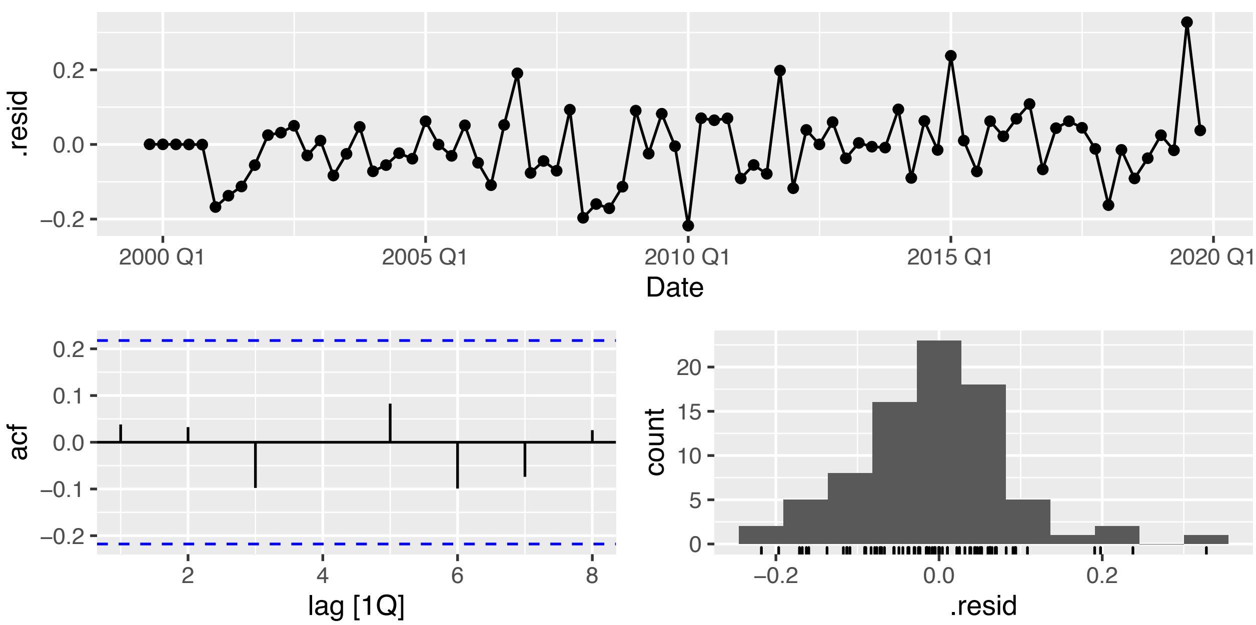
Best ARIMA model based on in-sample RMSE is $ARIMA(2,1,2)(2,1,2)$ with an RMSE of 0.0925. Next we will examine the inverse roots and residuals of our best model and conduct a portmanteau test:

```
### Create ARIMA(2,1,2)(2,1,2) model object
q4.arima.best <- eco.ts %>%
  model(mod1 = ARIMA(Eco ~ pdq(2,1,2)+PDQ(2,1,2)+1))
```

```r
### Check inverse roots all lie within unit circle
q4.arima.best %>% gg_arma()
```




```r
### Evaluate residuals
q4.arima.best %>% gg_tsresiduals(lag_max=8)
```




```r
q4.arima.best %>% augment() %>% features(.resid, ljung_box, lag=8)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 mod1      3.08     0.929
```

The AR and MA inverse roots all lie within the unit circle, so our fitted model satisfies those stationarity and invertibility conditions.

The residual diagnostic plots all point to white noise with no significant spikes in the ACF plot and a normally distributed histogram centered around zero. The Ljung-Box test also indicates that at a 5% level of significance, we fail to reject the null hypothesis of no autocorrelation among residuals. The $ARIMA(2,1,2)(2,1,2)$ model is ready to be used in the Holt-Winters comparisons.

We will do a cross validation by looking at 1-period-ahead forecasts for additive Holt-Winters, multiplicative Holt-Winters, and $ARIMA(2,1,2)(2,1,2)$ models. We will also look at the results of a simple average of those three models:

```
### Convert dataset into training set for cross validation
eco.train <- eco.ts %>%
  slice(1:(n()-1)) %>%
  stretch_tsibble(.init=8, .step=1)

### Cross validation of Holt-Winters, ARIMA, and ensemble average models
fc.q4.cv <- eco.train %>%
  model(
    hw_additive = ETS(Eco ~ error("A") + trend("A") + season("A")),
    hw_multiplicative = ETS(Eco ~ error("M") + trend("A") + season("M")),
    arima_212_212 = ARIMA(Eco ~ pdq(2,1,2)+PDQ(2,1,2)+1)
  ) %>% mutate(model_average = (hw_additive + hw_multiplicative +
                                  arima_212_212)/3) %>% forecast(h=1)

### Display accuracy metrics
fc.q4.cv %>% accuracy(eco.ts)
```

```
## # A tibble: 4 x 9
##   .model            .type     ME  RMSE    MAE   MPE  MAPE  MASE    ACF1
##   <chr>             <chr>  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 arima_212_212     Test  0.0239 0.127 0.0999 0.221  2.03 0.194 -0.0124
## 2 hw_additive       Test  0.0589 0.166 0.118  1.17   2.70 0.228  0.109
## 3 hw_multiplicative Test  0.0727 0.180 0.137  1.46   2.72 0.265 -0.128
## 4 model_average     Test  0.0506 0.145 0.115  0.646  2.16 0.222  0.0564
```

Among the Holt-Winters models, the additive version was better with an RMSE of 0.1657 versus 0.1799 for the multiplicative version. When adding $ARIMA(2,1,2)(2,1,2)$ into the comparison, the ARIMA model performed the best across all accuracy metrics including an RMSE of 0.1270. When considering a simple average of those three models, the ensemble method performed in between the ARIMA and Holt-Winters models with an RMSE of 0.1447. Therefore, based on 1-period-ahead cross validation, the seasonal ARIMA model is the best relative to the Holt-Winters and ensemble average models.

Question 5

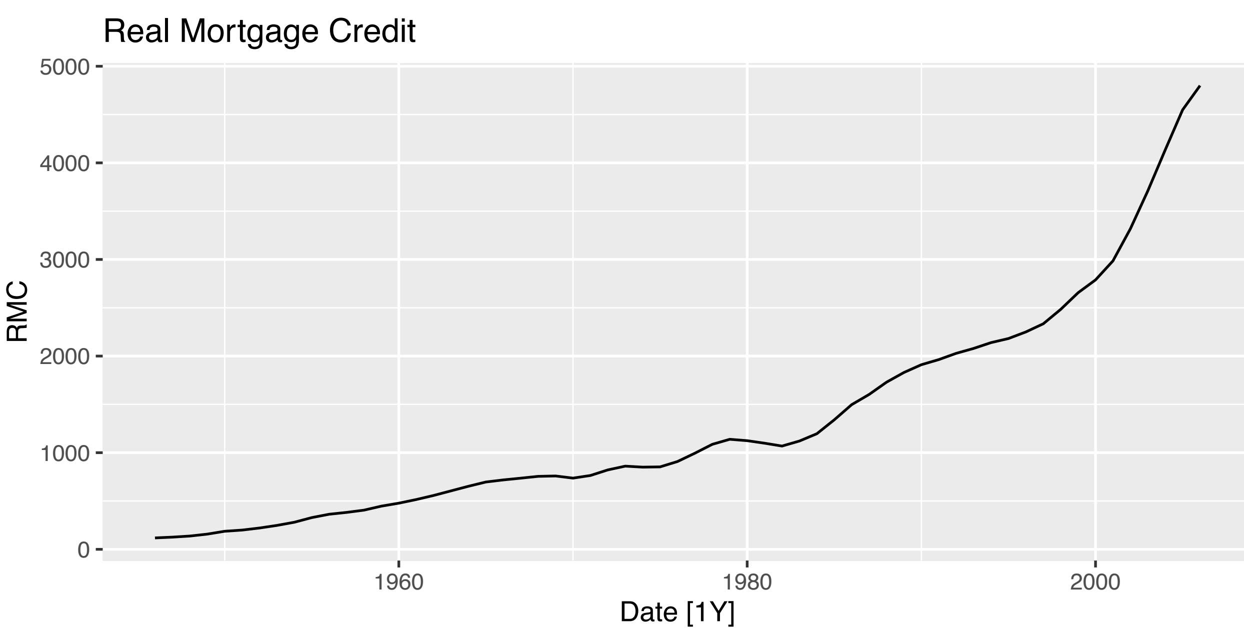
Part A: Conduct an EDA on these data. Develop a VAR model for the period 1946-2003. Forecast the last three years, 2004-2006, conducting model diagnostics. Examine the relative advantages of logarithmic transformations and the use of differences.

First, we will load in the dataset:

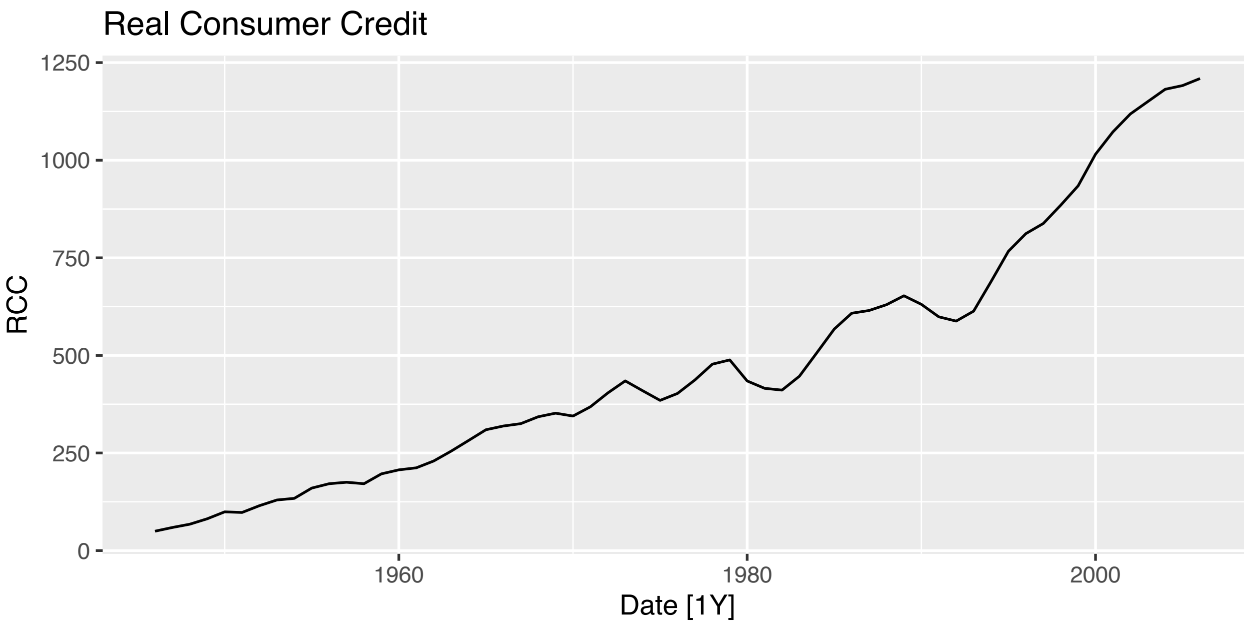
```
### Load dataset for Question 5
q5 <- read.csv("Q5.csv")

### Convert dataset to tsibble
monthly.interval5 <- ymd("1946-01-01") + months(seq(from=0, length.out=61, by=12))
q5.ts <- tsibble(Date=year(monthly.interval5),
                 RMC=q5$RMC, RCC=q5$RCC, RDPI=q5$RDPI, index=Date)
```

We will conduct some EDA, starting with time plots:

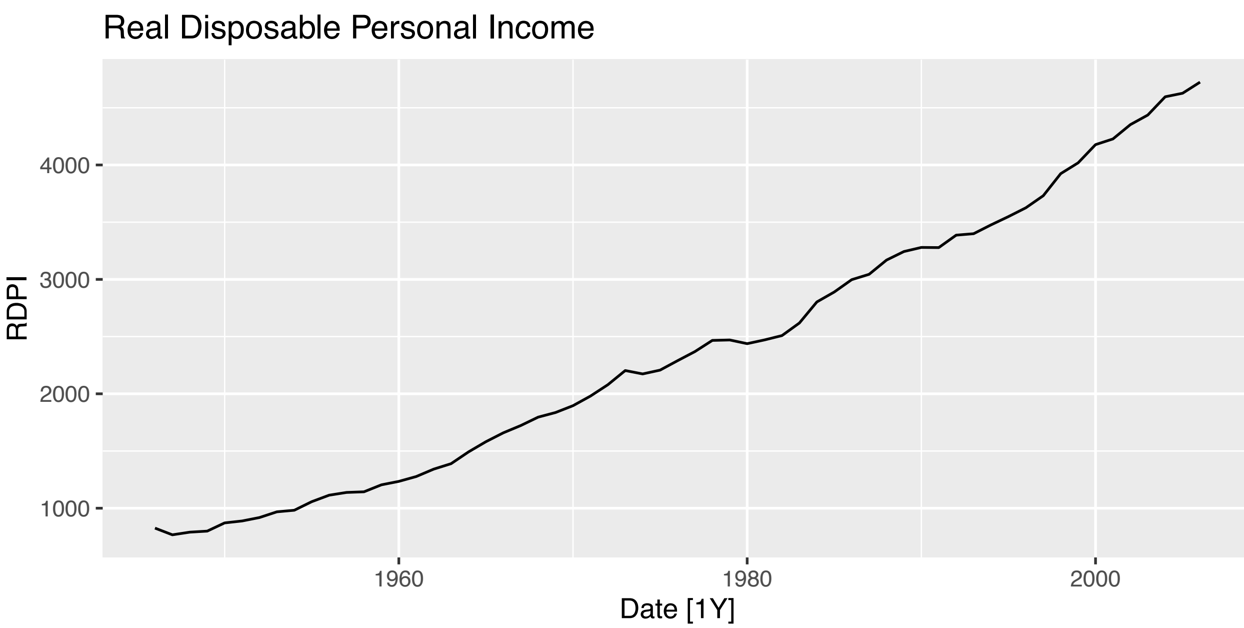
```
### Time plots of RMC, RCC, RDPI
par(mfrow=c(2,2))
q5.ts %>% autoplot(RMC) + ggtitle("Real Mortgage Credit")
```




```
q5.ts %>% autoplot(RCC) + ggtitle("Real Consumer Credit")
```



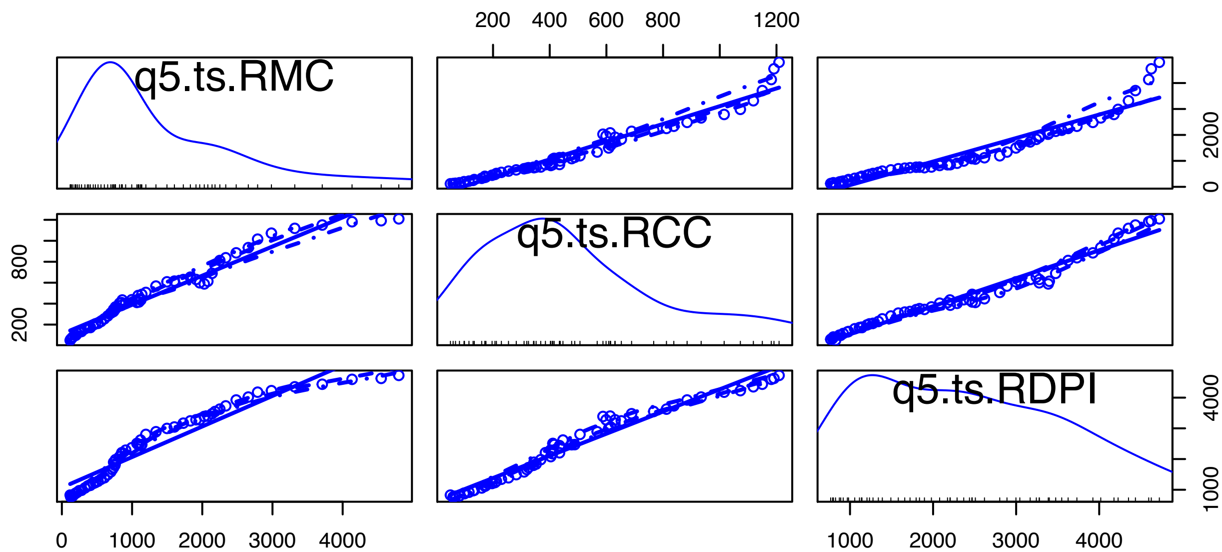

```
q5.ts %>% autoplot(RDPI) + ggtitle("Real Disposable Personal Income")
```



The above time plots show that each of the three economic series have increasing trends from 1946 to 2006. Mortgage credit and consumer credit were growing at faster rates in the later years of the series while RDPI was growing at a steadier rate. All three time plots do not appear to be stationary.

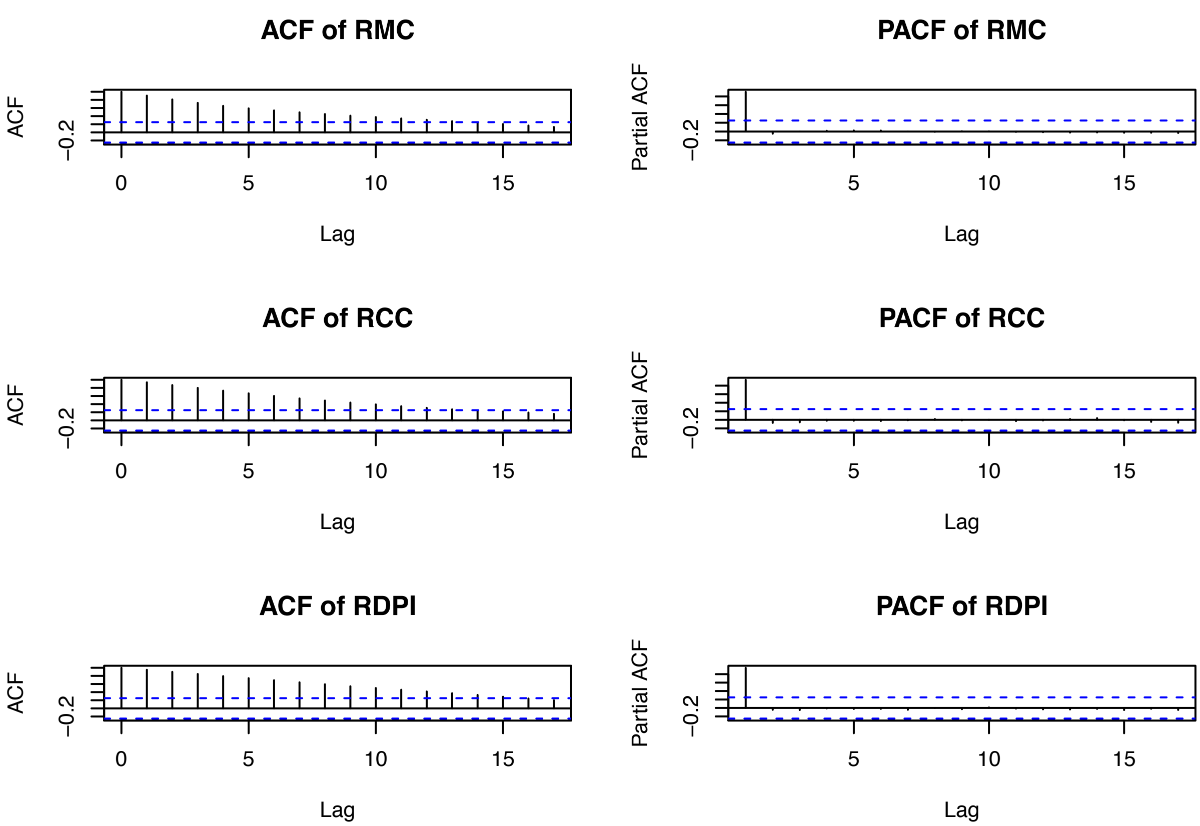
Now let's look at a scatterplot matrix for contemporaneous correlation:

```
### Scatterplot matrix
scatterplotMatrix(~q5.ts$RMC + q5.ts$RCC + q5.ts$RDPI)
```

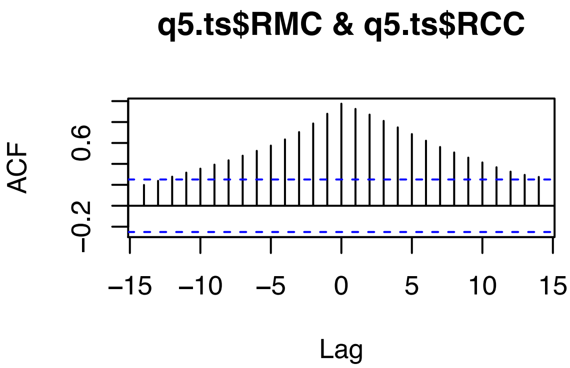


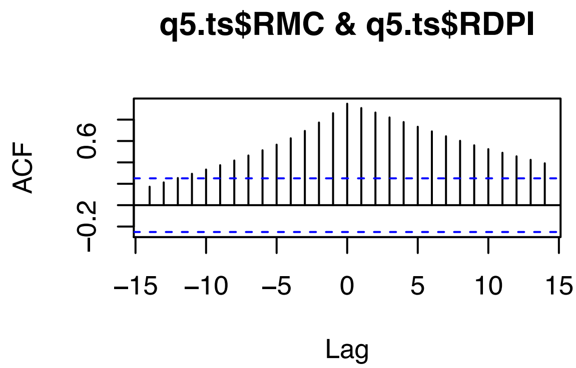
There is strong contemporaneous correlation among the three variables. Now we will examine the ACF/PACF and cross correlation plots.

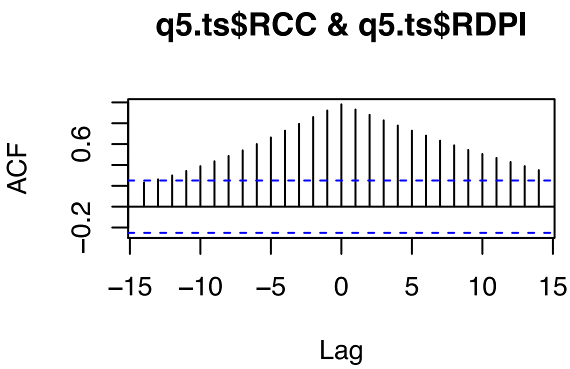
```
### ACF and PACF plots
par(mfrow=c(3,2))
acf(q5.ts$RMC, main="ACF of RMC")
pacf(q5.ts$RMC, main="PACF of RMC")
acf(q5.ts$RCC, main="ACF of RCC")
pacf(q5.ts$RCC, main="PACF of RCC")
acf(q5.ts$RDPI, main="ACF of RDPI")
pacf(q5.ts$RDPI, main="PACF of RDPI")
```



```r
### Cross correlation
par(mfrow=c(2,2))
ccf(q5.ts$RMC, q5.ts$RCC)
ccf(q5.ts$RMC, q5.ts$RDPI)
ccf(q5.ts$RCC, q5.ts$RDPI)
```







The PACF plots show no significant spikes after lag 1, while the gradually declining spikes in the ACF plots indicate a potential AR process. The cross correlation plots show meaningful spikes across many lags for each of the variable pairings.

Now we will conduct the Augmented Dickey-Fuller and Phillips-Ouliaris tests:

```r
### ADF and PO Tests
adf.test(q5.ts$RMC)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts$RMC
## Dickey-Fuller = 1.3586, Lag order = 3, p-value = 0.99
## alternative hypothesis: stationary
```

```r
adf.test(q5.ts$RCC)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts$RCC
## Dickey-Fuller = -0.016205, Lag order = 3, p-value = 0.99
## alternative hypothesis: stationary
```

```r
adf.test(q5.ts$RDPI)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts$RDPI
## Dickey-Fuller = -0.93887, Lag order = 3, p-value = 0.9402
## alternative hypothesis: stationary
```

```r
po.test(q5.ts[ , 2:3])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts[, 2:3]
## Phillips-Ouliaris demeaned = 2.1239, Truncation lag parameter = 0,
## p-value = 0.15
```

```r
po.test(q5.ts[ , c(2,4)])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts[, c(2, 4)]
## Phillips-Ouliaris demeaned = 6.795, Truncation lag parameter = 0,
## p-value = 0.15
```

```r
po.test(q5.ts[ , 3:4])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts[, 3:4]
## Phillips-Ouliaris demeaned = -1.879, Truncation lag parameter = 0,
## p-value = 0.15
```

The high p-values (0.99, 0.99, 0.94) for the ADF tests indicate that we fail to reject the null hypothesis of a unit root. This reaffirms earlier plots that show that the three series are nonstationary.

With p-values of 0.15 for the PO tests, we fail to reject the null hypothesis that variable pairs are not cointegrated.

Now we will look into building a VAR model with both a constant and a trend:

```r
### Create training data and select order for VAR model
q5.train <- q5.ts %>% filter(Date<=2003)
VARselect(q5.train[ , 2:4], lag.max=10, type="both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     10     10      2      9
##
## $criteria
##                   1            2            3            4            5
## AIC(n) 2.129452e+01 2.013628e+01 2.000040e+01 2.007703e+01 2.010043e+01
## HQ(n)  2.151550e+01 2.048984e+01 2.048655e+01 2.069577e+01 2.085175e+01
## SC(n)  2.187927e+01 2.107188e+01 2.128685e+01 2.171433e+01 2.208858e+01
## FPE(n) 1.774520e+09 5.612584e+08 4.975746e+08 5.521629e+08 5.905870e+08
```

```
##                  6            7            8            9           10
## AIC(n) 1.974655e+01 1.965113e+01 1.898699e+01 1.865871e+01 1.844180e+01
## HQ(n)  2.063046e+01 2.066762e+01 2.013607e+01 1.994038e+01 1.985606e+01
## SC(n)  2.208555e+01 2.234098e+01 2.202769e+01 2.205026e+01 2.218421e+01
## FPE(n) 4.427234e+08 4.423172e+08 2.599647e+08 2.250446e+08 2.338332e+08
```

Since the different measures chose different orders, we will prioritize the SC/BIC value, which chose order 2. Now we will estimate a VAR(2) model

```r
### VAR(2) model on training set
q5.var1 <- VAR(q5.train[ , 2:4], p=2, type="both")
summary(q5.var1)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: RMC, RCC, RDPI
## Deterministic variables: both
## Sample size: 56
## Log Likelihood: -768.931
## Roots of the characteristic polynomial:
## 1.157 0.8639 0.8639 0.8374 0.6331 0.0473
## Call:
## VAR(y = q5.train[, 2:4], p = 2, type = "both")
##
##
## Estimation results for equation RMC:
## ====================================
## RMC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##          Estimate Std. Error t value Pr(>|t|)
## RMC.l1   1.799678   0.146932  12.248  < 2e-16 ***
## RCC.l1   0.204995   0.258493   0.793    0.432
## RDPI.l1 -0.025617   0.158550  -0.162    0.872
## RMC.l2  -0.786066   0.146643  -5.360 2.34e-06 ***
## RCC.l2   0.001376   0.253267   0.005    0.996
## RDPI.l2  0.001162   0.142219   0.008    0.994
## const    8.805267  32.526993   0.271    0.788
## trend   -1.312656   3.413383  -0.385    0.702
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 34.44 on 48 degrees of freedom
## Multiple R-Squared: 0.9987,  Adjusted R-squared: 0.9985
## F-statistic:  5114 on 7 and 48 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation RCC:
## ====================================
## RCC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##          Estimate Std. Error t value Pr(>|t|)
## RMC.l1   0.002156   0.087117   0.025    0.980
## RCC.l1   1.472530   0.153262   9.608 9.23e-13 ***
```

```
## RDPI.l1  0.054159   0.094005   0.576    0.567
## RMC.l2   0.058906   0.086945   0.678    0.501
## RCC.l2  -0.659156   0.150163  -4.390 6.21e-05 ***
## RDPI.l2 -0.075895   0.084322  -0.900    0.373
## const   16.277333  19.285373   0.844    0.403
## trend    1.657949   2.023807   0.819    0.417
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 20.42 on 48 degrees of freedom
## Multiple R-Squared: 0.9954,  Adjusted R-squared: 0.9947
## F-statistic:  1469 on 7 and 48 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation RDPI:
## =====================================
## RDPI = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##          Estimate Std. Error t value Pr(>|t|)
## RMC.l1   0.090310   0.182094   0.496 0.622189
## RCC.l1   0.423064   0.320351   1.321 0.192889
## RDPI.l1  0.824607   0.196491   4.197 0.000116 ***
## RMC.l2  -0.070556   0.181735  -0.388 0.699559
## RCC.l2  -0.314406   0.313874  -1.002 0.321513
## RDPI.l2  0.005734   0.176253   0.033 0.974184
## const   93.061946  40.310784   2.309 0.025316 *
## trend    9.071795   4.230214   2.145 0.037082 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 42.68 on 48 degrees of freedom
## Multiple R-Squared: 0.9986,  Adjusted R-squared: 0.9984
## F-statistic:  4983 on 7 and 48 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##         RMC   RCC   RDPI
## RMC  1186.1 406.4  735.6
## RCC   406.4 416.9  642.2
## RDPI  735.6 642.2 1821.6
##
## Correlation matrix of residuals:
##         RMC    RCC   RDPI
## RMC  1.0000 0.5780 0.5004
## RCC  0.5780 1.0000 0.7369
## RDPI 0.5004 0.7369 1.0000
```

```
roots(q5.var1)
```

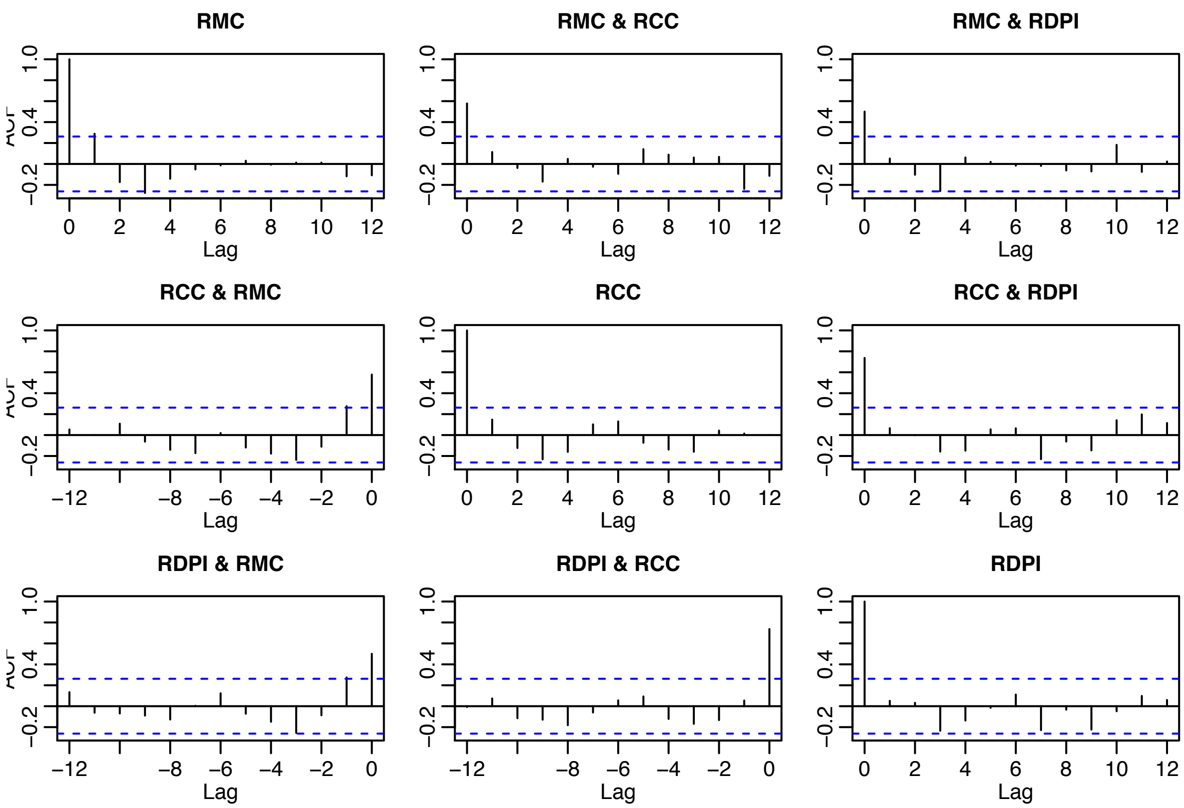
```
## [1] 1.15729313 0.86389207 0.86389207 0.83741311 0.63307489 0.04730256
```

The roots of the above VAR(2) model are not all less than one, indicating that the model may not be a stable process. Let's examine the model residuals:

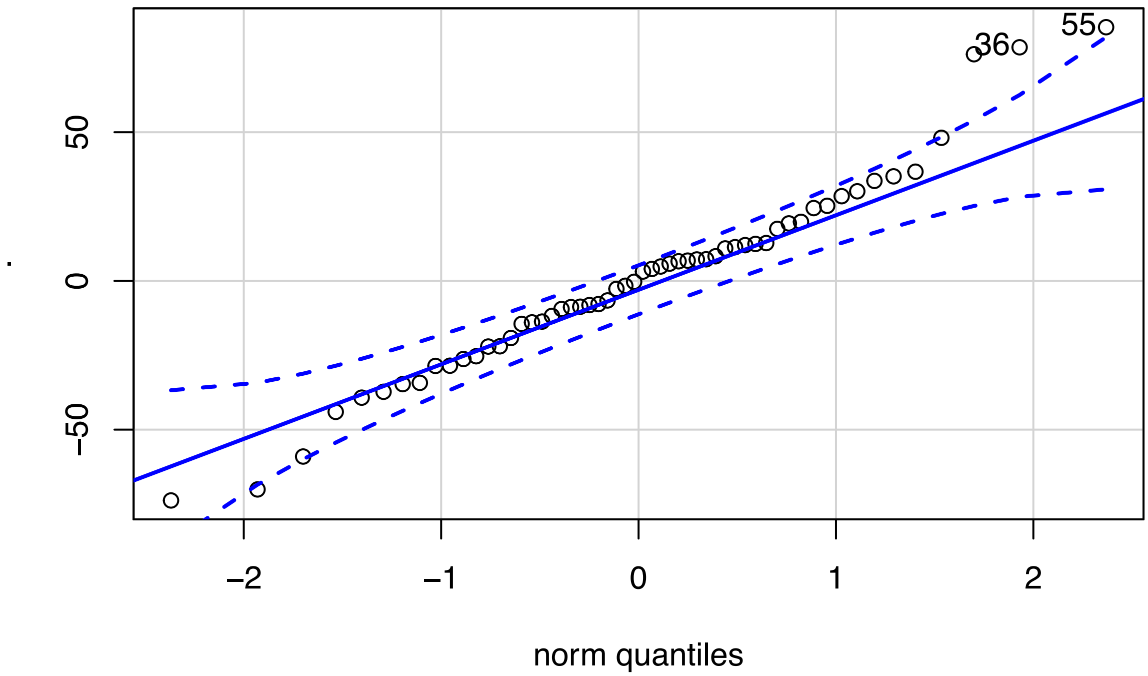
```
### Model residuals diagnostics
serial.test(q5.var1)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object q5.var1
## Chi-squared = 146.38, df = 126, p-value = 0.1035
```

```
q5.var1 %>% resid %>% acf
```

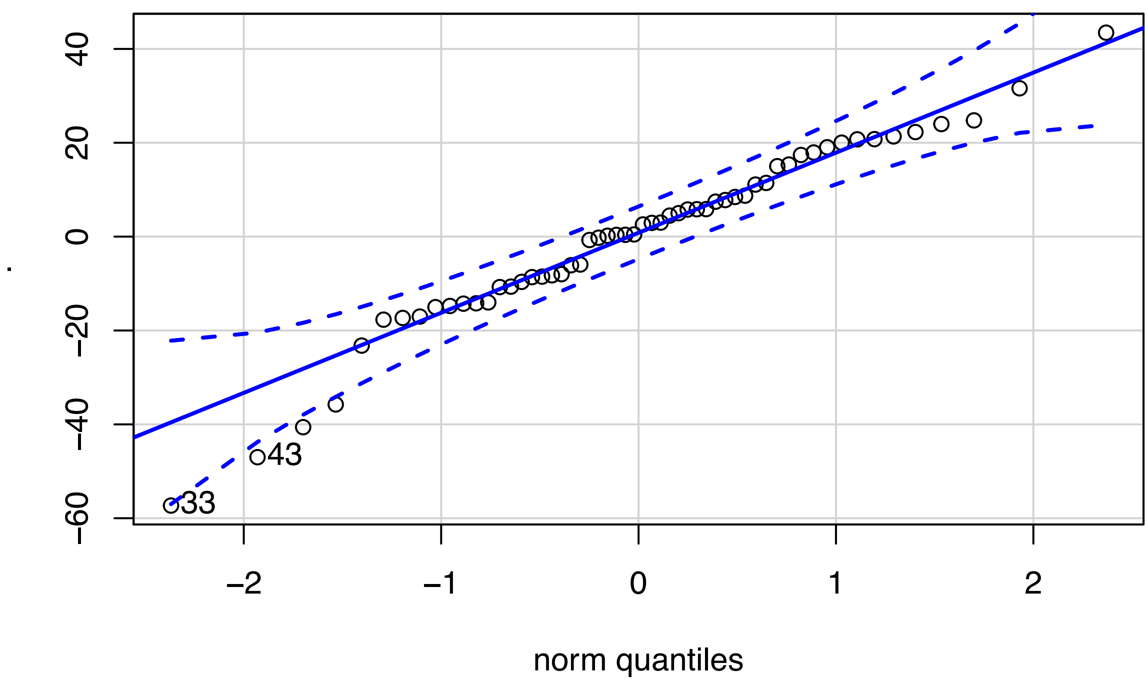



```
q5.var1 %>% resid %>% .[ , "RMC"] %>% qqPlot
```



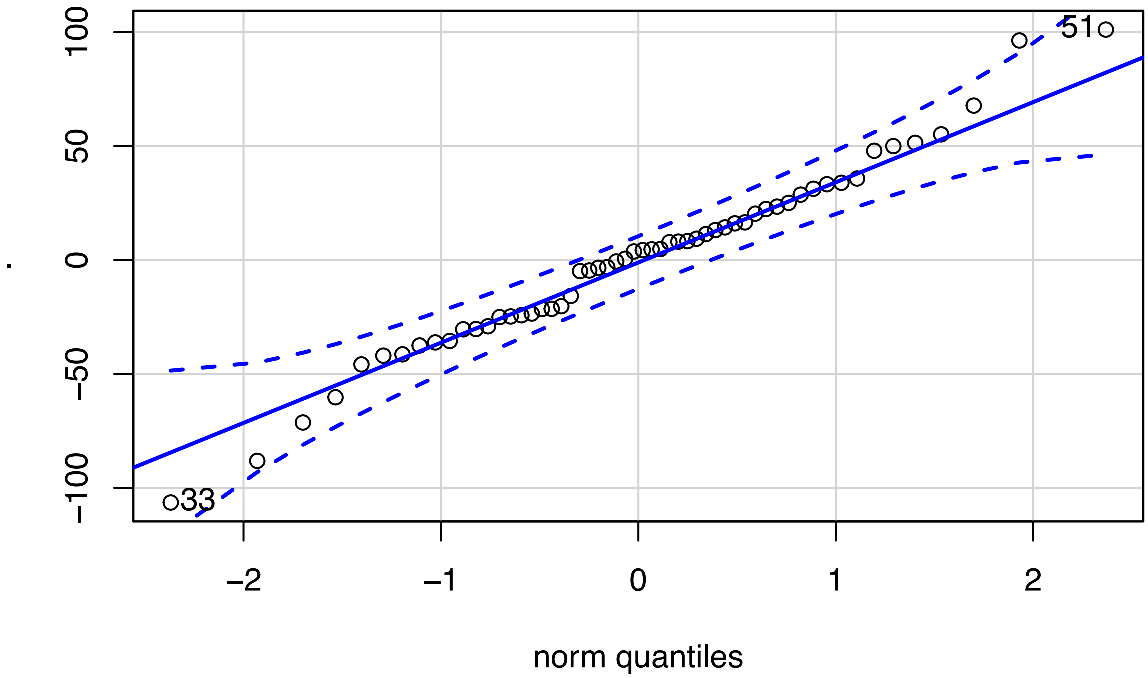

```
## [1] 55 36
```

```
q5.var1 %>% resid %>% .[ , "RCC"] %>% qqPlot
```




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

```
q5.var1 %>% resid %>% .[ , "RDPI"] %>% qqPlot
```




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

We fail to reject the null hypothesis in the serial correlation test, indicating evidence that the residuals are not serially correlated. The ACF plots also show a lack of autocorrelation in residuals with no significant spikes besides at lag 0. The Q-Q plots demonstrate that some residuals fall outside the expected range of a normal distribution.

Overall, the VAR(2) model using the original three series is not ideal given the lack of cointegration, not having all roots below one, and some non-normality in the residuals. We will examine other models that use logarithmic and differencing transformations.

First we will look at logarithmic transformations and conduct ADF/PO tests:

```
### Log transformation
q5.ts.log <- log(q5.ts)
q5.ts.log$Date <- q5.ts$Date

### ADF and PO Tests
adf.test(q5.ts.log$RMC)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts.log$RMC
## Dickey-Fuller = -2.8437, Lag order = 3, p-value = 0.2336
## alternative hypothesis: stationary
```

```
adf.test(q5.ts.log$RCC)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts.log$RCC
## Dickey-Fuller = -3.1309, Lag order = 3, p-value = 0.1174
## alternative hypothesis: stationary
```

```r
adf.test(q5.ts.log$RDPI)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts.log$RDPI
## Dickey-Fuller = -1.3067, Lag order = 3, p-value = 0.8552
## alternative hypothesis: stationary
```

```r
po.test(q5.ts.log[ , 2:3])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts.log[, 2:3]
## Phillips-Ouliaris demeaned = -8.0491, Truncation lag parameter = 0,
## p-value = 0.15
```

```r
po.test(q5.ts.log[ , c(2,4)])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts.log[, c(2, 4)]
## Phillips-Ouliaris demeaned = -5.8731, Truncation lag parameter = 0,
## p-value = 0.15
```

```r
po.test(q5.ts.log[ , 3:4])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts.log[, 3:4]
## Phillips-Ouliaris demeaned = -14.164, Truncation lag parameter = 0,
## p-value = 0.15
```

The log-transformed ADF and PO tests show the same results as the original series with evidence for nonstationary series and variables not being cointegrated.

Now we will look into building a VAR model with both a constant and a trend:

```r
### Create training data and select order for VAR model
q5.log.train <- q5.ts.log %>% filter(Date<=2003)
VARselect(q5.log.train[ , 2:4], lag.max=10, type="both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      3      2      3
##
## $criteria
##                    1             2             3             4             5
## AIC(n) -2.144888e+01 -2.245813e+01 -2.267805e+01 -2.246715e+01 -2.232205e+01
## HQ(n)  -2.122791e+01 -2.210456e+01 -2.219189e+01 -2.184841e+01 -2.157073e+01
## SC(n)  -2.086413e+01 -2.152253e+01 -2.139159e+01 -2.082985e+01 -2.033390e+01
## FPE(n)  4.851282e-10  1.780935e-10  1.451597e-10  1.842308e-10  2.225542e-10
##                    6             7             8             9            10
## AIC(n) -2.254081e+01 -2.258322e+01 -2.295533e+01 -2.302491e+01 -2.300516e+01
## HQ(n)  -2.165689e+01 -2.156672e+01 -2.180625e+01 -2.174323e+01 -2.159090e+01
```

```
## SC(n)  -2.020180e+01 -1.989337e+01 -1.991463e+01 -1.963335e+01 -1.926276e+01
## FPE(n)  1.909717e-10  2.011827e-10  1.583415e-10  1.775434e-10  2.337314e-10
```

Since the different measures chose different orders, we will prioritize the SC/BIC value, which chose order 2. Now we will estimate a VAR(2) model on the log-transformed series

```r
### VAR(2) model on log-transformed training set
q5.var2 <- VAR(q5.log.train[ , 2:4], p=2, type="both")
summary(q5.var2)
```

```
##
## 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 = q5.log.train[, 2:4], 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
```

```r
roots(q5.var2)
```

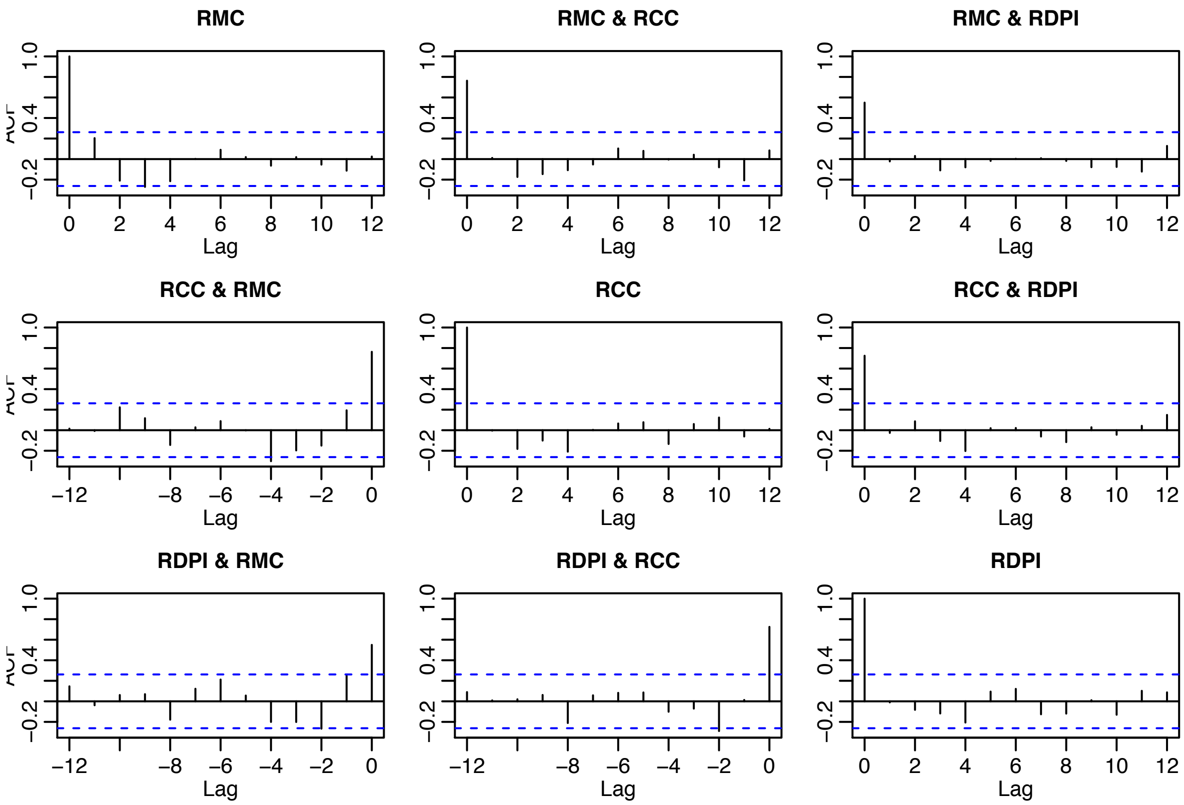
```
## [1] 0.9103601 0.9103601 0.8033969 0.8033969 0.2637261 0.1021632
```

The roots of the above log-transformed VAR(2) model are all less than one, which is an improvement over the original series' model. Let's examine the model residuals:

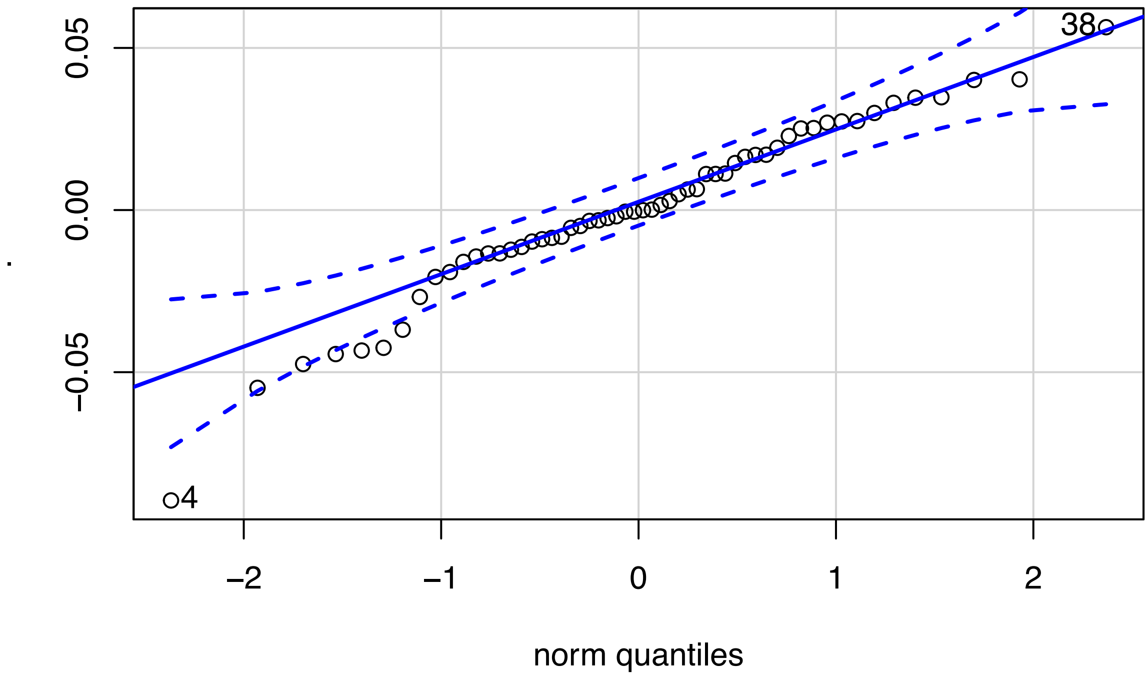
```r
### Model residuals diagnostics
serial.test(q5.var2)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object q5.var2
## Chi-squared = 114.87, df = 126, p-value = 0.7519
```

```r
q5.var2 %>% resid %>% acf
```

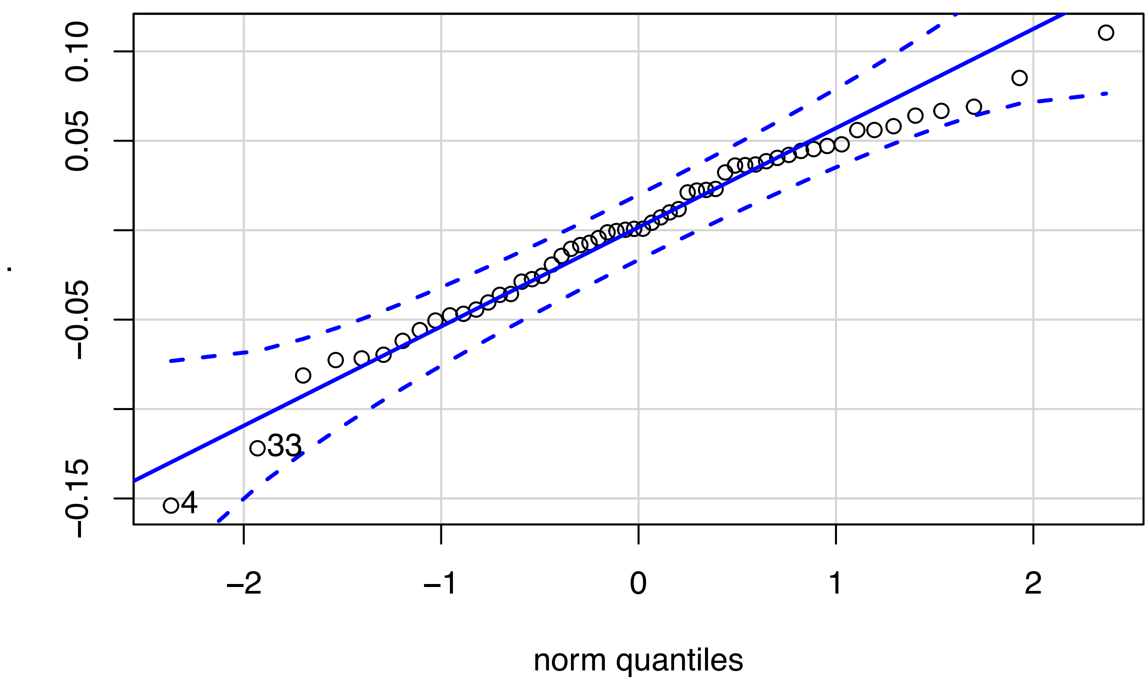



```r
q5.var2 %>% resid %>% .[ , "RMC"] %>% qqPlot
```



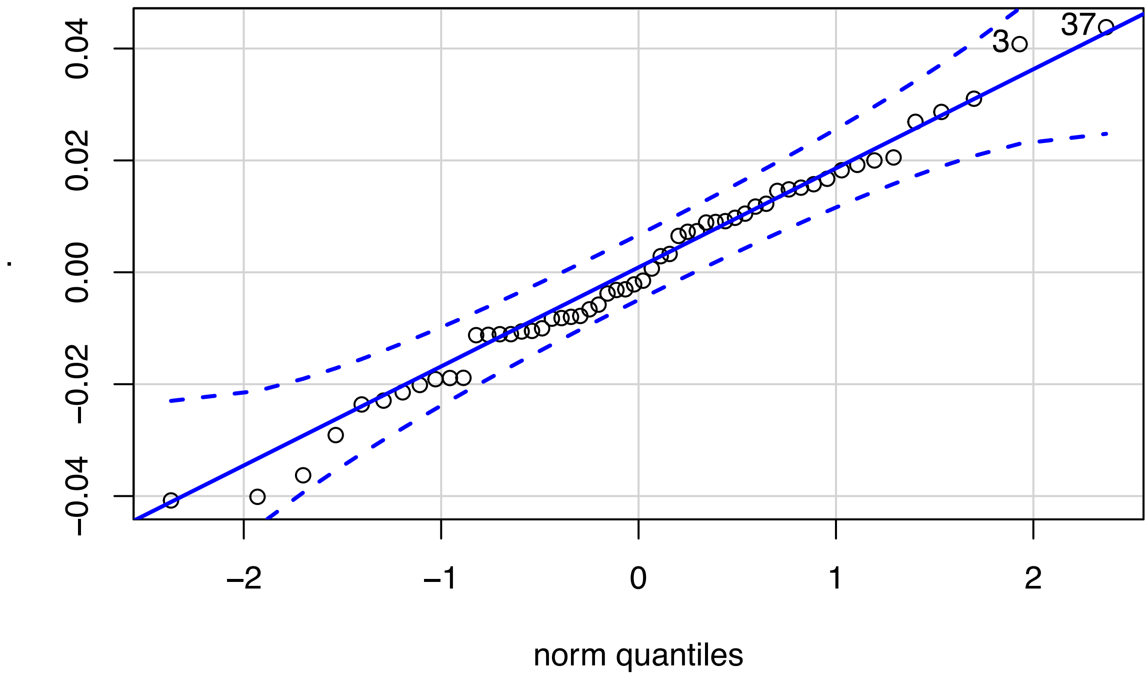

```
## [1]  4 38
```

```
q5.var2 %>% resid %>% .[ , "RCC"] %>% qqPlot
```




```
## [1]  4 33
```

```
q5.var2 %>% resid %>% .[ , "RDPI"] %>% qqPlot
```




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

We get some improvement over the original series' model. The serial test and ACF plots show evidence for residuals not being serially correlated and the Q-Q plots demonstrate that fewer points fall outside the expected normality range. In totality, the VAR(2) model with log transformation is better due to having all roots less than one and more normality in the residuals.

Now we will look at differencing transformations and conduct ADF/PO tests:

```
### Differencing transformations
q5.ts.diff <- q5.ts %>% mutate(RMC_d = difference(RMC))
q5.ts.diff <- q5.ts.diff %>% mutate(RCC_d = difference(RCC))
q5.ts.diff <- q5.ts.diff %>% mutate(RDPI_d = difference(RDPI))
q5.ts.diff <- q5.ts.diff[2:61, c(1,5,6,7)]

### ADF and PO Tests
adf.test(q5.ts.diff$RMC_d)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts.diff$RMC_d
## Dickey-Fuller = -2.8096, Lag order = 3, p-value = 0.2475
## alternative hypothesis: stationary
```

```
adf.test(q5.ts.diff$RCC_d)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts.diff$RCC_d
## Dickey-Fuller = -3.7792, Lag order = 3, p-value = 0.02569
## alternative hypothesis: stationary
```

```r
adf.test(q5.ts.diff$RDPI_d)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  q5.ts.diff$RDPI_d
## Dickey-Fuller = -3.9824, Lag order = 3, p-value = 0.01647
## alternative hypothesis: stationary
```

```r
po.test(q5.ts.diff[ , 2:3])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts.diff[, 2:3]
## Phillips-Ouliaris demeaned = -5.3464, Truncation lag parameter = 0,
## p-value = 0.15
```

```r
po.test(q5.ts.diff[ , c(2,4)])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts.diff[, c(2, 4)]
## Phillips-Ouliaris demeaned = -11.638, Truncation lag parameter = 0,
## p-value = 0.15
```

```r
po.test(q5.ts.diff[ , 3:4])
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  q5.ts.diff[, 3:4]
## Phillips-Ouliaris demeaned = -32.593, Truncation lag parameter = 0,
## p-value = 0.01
```

For the differenced series, we can reject the ADF tests for the RCC and RDPI variables, suggesting evidence of those series being stationary. The PO test also had one different result from the original series with a p-value of 0.01 between RCC and RDPI, indicating that those two variables are cointegrated.

Now we will look into building a VAR model with both a constant and a trend:

```r
### Create training data and select order for VAR model
q5.diff.train <- q5.ts.diff %>% filter(Date<=2003)
VARselect(q5.diff.train[ , 2:4], lag.max=10, type="both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      9      2      9
##
## $criteria
##                   1            2            3            4            5
## AIC(n) 2.062688e+01 2.027101e+01 2.040542e+01 2.051008e+01 2.071817e+01
## HQ(n)  2.084908e+01 2.062653e+01 2.089426e+01 2.113224e+01 2.147365e+01
## SC(n)  2.121735e+01 2.121576e+01 2.170446e+01 2.216340e+01 2.272578e+01
## FPE(n) 9.103178e+08 6.426077e+08 7.472713e+08 8.544290e+08 1.102690e+09
##                   6            7            8            9           10
## AIC(n) 2.079923e+01 2.052184e+01 1.968258e+01 1.910453e+01 1.916961e+01
```

```
## HQ(n)  2.168802e+01 2.154395e+01 2.083802e+01 2.039328e+01 2.059168e+01
## SC(n)  2.316112e+01 2.323801e+01 2.275304e+01 2.252927e+01 2.294863e+01
## FPE(n) 1.283176e+09 1.076513e+09 5.368592e+08 3.679610e+08 5.196645e+08
```

Since the different measures chose different orders, we will prioritize the SC/BIC value, which chose order 2. Now we will estimate a VAR(2) model on the differenced series

```r
### VAR(2) model on differenced training set
q5.var3 <- VAR(q5.diff.train[ , 2:4], p=2, type="both")
summary(q5.var3)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: RMC_d, RCC_d, RDPI_d
## Deterministic variables: both
## Sample size: 55
## Log Likelihood: -756.486
## Roots of the characteristic polynomial:
## 0.7686 0.7521 0.7521 0.6008 0.6008 0.5441
## Call:
## VAR(y = q5.diff.train[, 2:4], p = 2, type = "both")
##
##
## Estimation results for equation RMC_d:
## ======================================
## RMC_d = RMC_d.l1 + RCC_d.l1 + RDPI_d.l1 + RMC_d.l2 + RCC_d.l2 + RDPI_d.l2 + const + trend
##
##           Estimate Std. Error t value Pr(>|t|)
## RMC_d.l1   1.32808    0.16089   8.255 1.06e-10 ***
## RCC_d.l1  -0.01999    0.30079  -0.066  0.94729
## RDPI_d.l1 -0.12716    0.14930  -0.852  0.39870
## RMC_d.l2  -0.55609    0.20162  -2.758  0.00826 **
## RCC_d.l2   0.29042    0.28996   1.002  0.32167
## RDPI_d.l2 -0.06609    0.13106  -0.504  0.61644
## const     -3.50856   11.21943  -0.313  0.75588
## trend      0.92451    0.38718   2.388  0.02102 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 33.65 on 47 degrees of freedom
## Multiple R-Squared: 0.8315,  Adjusted R-squared: 0.8064
## F-statistic: 33.13 on 7 and 47 DF,  p-value: 4.162e-16
##
##
## Estimation results for equation RCC_d:
## ======================================
## RCC_d = RMC_d.l1 + RCC_d.l1 + RDPI_d.l1 + RMC_d.l2 + RCC_d.l2 + RDPI_d.l2 + const + trend
##
##           Estimate Std. Error t value Pr(>|t|)
## RMC_d.l1   0.15580    0.09075   1.717   0.0926 .
## RCC_d.l1   0.77378    0.16965   4.561 3.65e-05 ***
## RDPI_d.l1 -0.08078    0.08421  -0.959   0.3423
## RMC_d.l2  -0.26729    0.11372  -2.350   0.0230 *
```

```
## RCC_d.l2  -0.35417    0.16355  -2.166   0.0355 *
## RDPI_d.l2  0.03167    0.07392   0.428   0.6703
## const      3.32683    6.32801   0.526   0.6015
## trend      0.54132    0.21838   2.479   0.0168 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 18.98 on 47 degrees of freedom
## Multiple R-Squared:  0.58,   Adjusted R-squared: 0.5174
## F-statistic: 9.271 on 7 and 47 DF,  p-value: 3.728e-07
##
##
## Estimation results for equation RDPI_d:
## =======================================
## RDPI_d = RMC_d.l1 + RCC_d.l1 + RDPI_d.l1 + RMC_d.l2 + RCC_d.l2 + RDPI_d.l2 + const + trend
##
##           Estimate Std. Error t value Pr(>|t|)
## RMC_d.l1   0.51550    0.18338   2.811 0.007176 **
## RCC_d.l1   0.04119    0.34282   0.120 0.904881
## RDPI_d.l1 -0.09588    0.17016  -0.563 0.575816
## RMC_d.l2  -0.89706    0.22980  -3.904 0.000301 ***
## RCC_d.l2   0.03292    0.33049   0.100 0.921079
## RDPI_d.l2  0.26034    0.14938   1.743 0.087899 .
## const     30.76337   12.78742   2.406 0.020126 *
## trend      1.37028    0.44129   3.105 0.003219 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 38.35 on 47 degrees of freedom
## Multiple R-Squared: 0.4034,  Adjusted R-squared: 0.3146
## F-statistic:  4.54 on 7 and 47 DF,  p-value: 0.0006199
##
##
##
## Covariance matrix of residuals:
##         RMC_d RCC_d RDPI_d
## RMC_d  1132.2 280.6  466.6
## RCC_d   280.6 360.2  462.4
## RDPI_d  466.6 462.4 1470.8
##
## Correlation matrix of residuals:
##         RMC_d  RCC_d RDPI_d
## RMC_d  1.0000 0.4393 0.3616
## RCC_d  0.4393 1.0000 0.6353
## RDPI_d 0.3616 0.6353 1.0000
```

```r
roots(q5.var3)
```

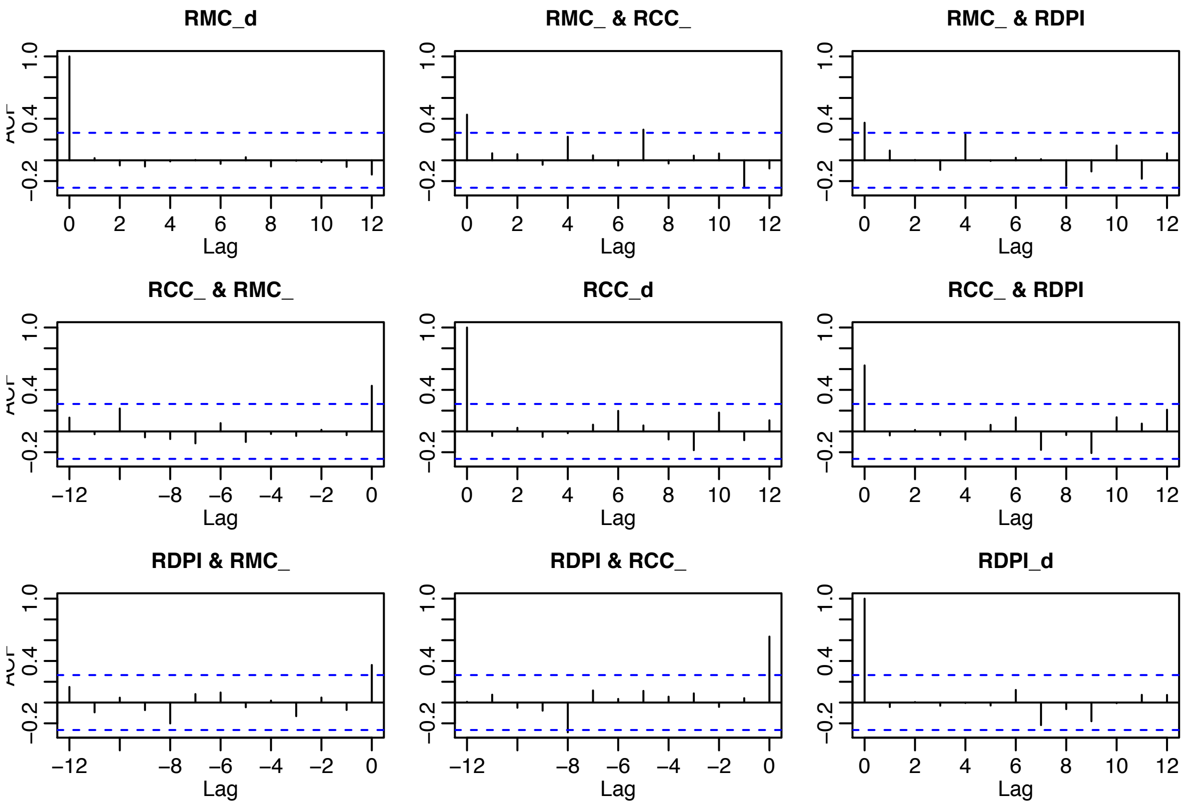
```
## [1] 0.7685702 0.7520791 0.7520791 0.6008421 0.6008421 0.5441126
```

The roots of the above differenced VAR(2) model are all less than one, which is an improvement over the original series' model. Let's examine the model residuals:

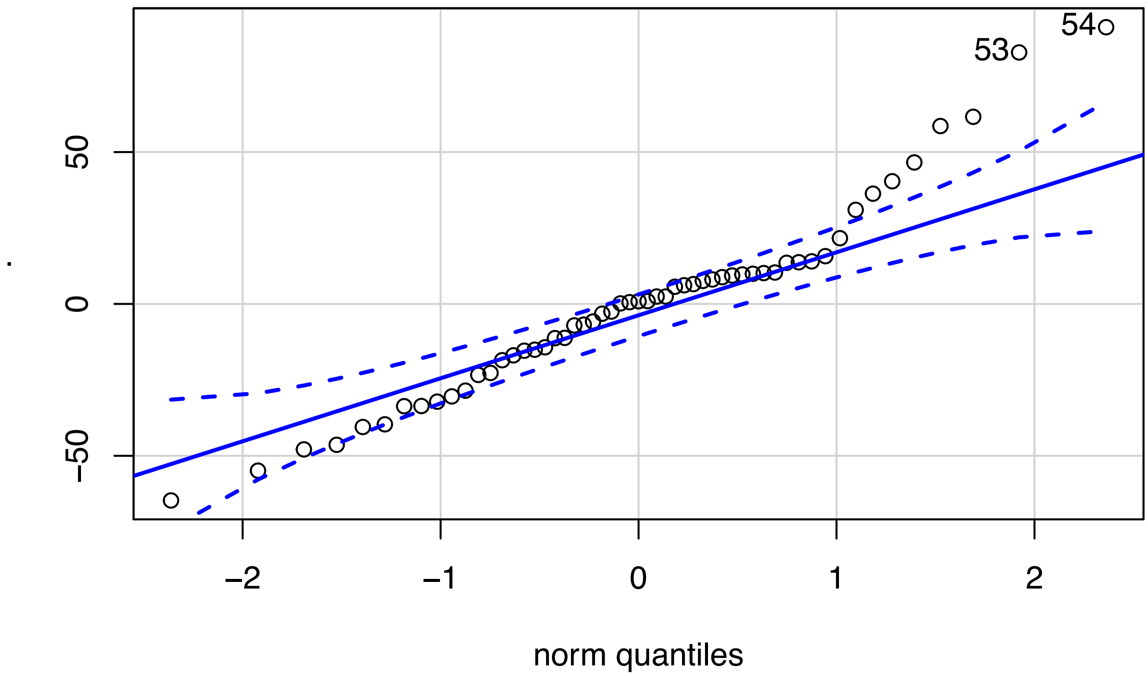
```r
### Model residuals diagnostics
serial.test(q5.var3)
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object q5.var3
## Chi-squared = 118.68, df = 126, p-value = 0.6656
```

```r
q5.var3 %>% resid %>% acf
```

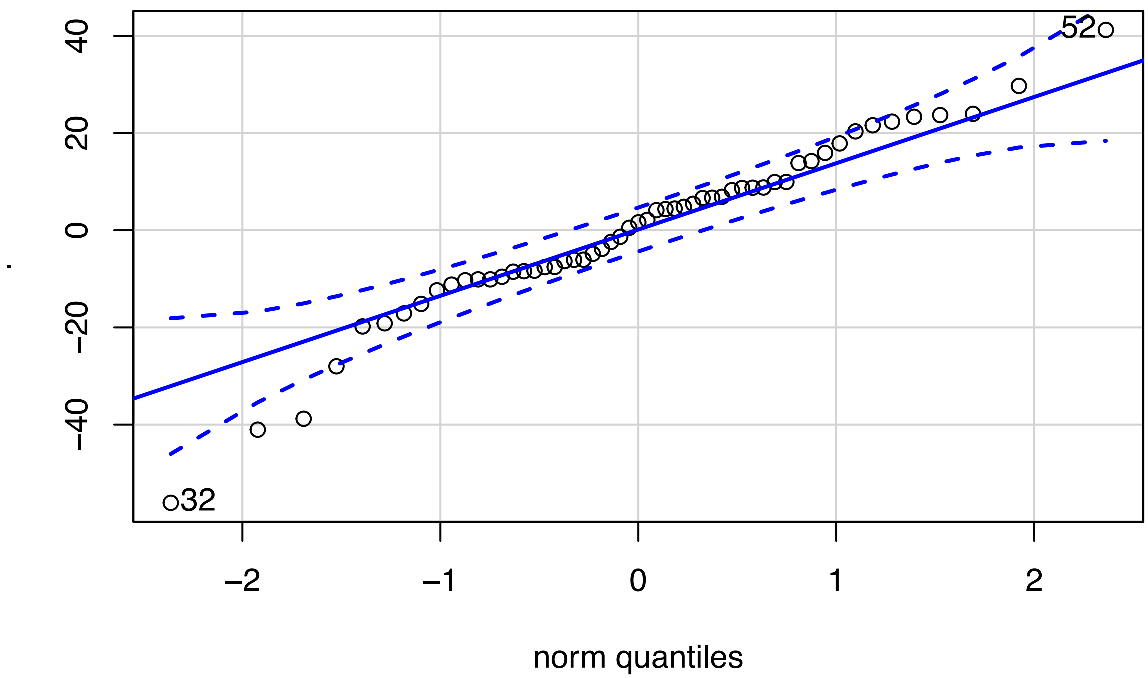



```r
q5.var3 %>% resid %>% .[ , "RMC_d"] %>% qqPlot
```



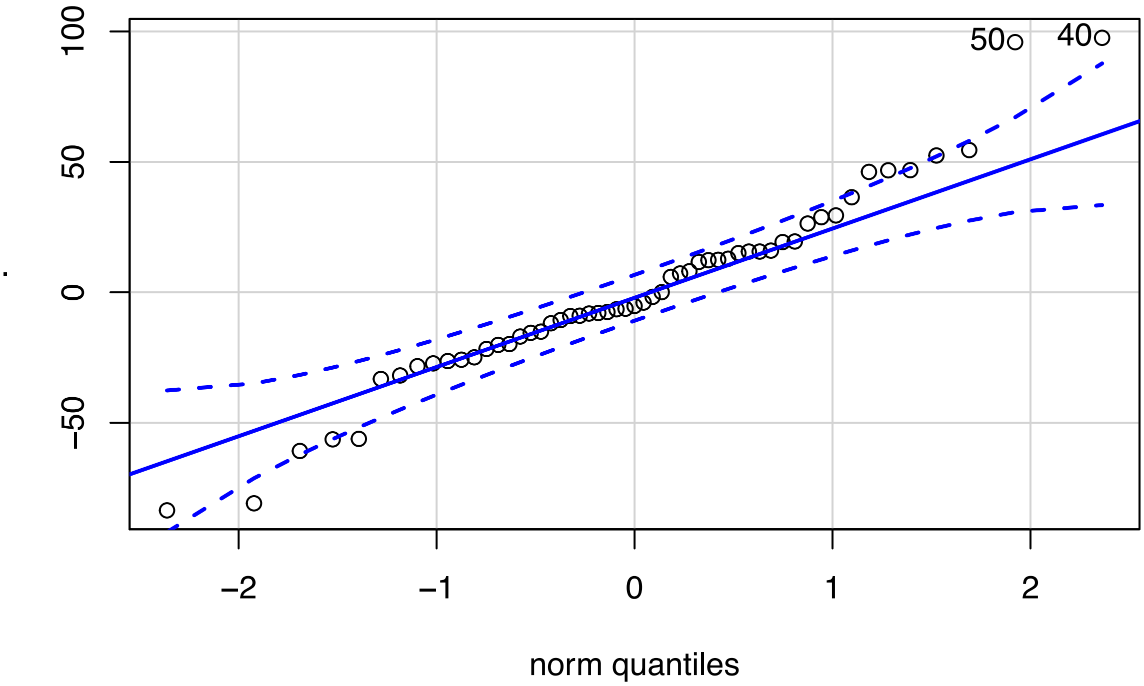

```
## [1] 54 53
```

```
q5.var3 %>% resid %>% .[ , "RCC_d"] %>% qqPlot
```




```
## [1] 32 52
```

```
q5.var3 %>% resid %>% .[ , "RDPI_d"] %>% qqPlot
```




```
## [1] 40 50
```

The above residual diagnostics do not show improvement over the original series' model. The serial test and ACF plots show evidence for residuals not being serially correlated. The Q-Q plots still indicate that some points fall outside the expected normality range.

In totality, while differencing did introduce more cointegration between variables, the logarithmic transformation offered more improvements.

Therefore we will use the log-transformed VAR(2) model as our best model for forecasting for the years 2004-2006:

```r
### Log VAR(2) forecast over next 3 years, 2004-2006
q5.fc <- q5.var2 %>% predict(n.ahead = 3, ci = 0.95)

### Transform back to raw data scale through exponentiation
q5.fc.RMC <- as.data.frame(cbind(c(2004,2005,2006),
                                 exp(q5.fc$fcst$RMC)[,1:3],
                                 q5.ts[59:61, 2]))
colnames(q5.fc.RMC) <- c("Year","Forecast_RMC","Lower_RMC",
                         "Upper_RMC","Actual_RMC")

q5.fc.RCC <- as.data.frame(cbind(c(2004,2005,2006),
                                 exp(q5.fc$fcst$RCC)[,1:3],
                                 q5.ts[59:61, 3]))
colnames(q5.fc.RCC) <- c("Year","Forecast_RCC","Lower_RCC",
                         "Upper_RCC","Actual_RCC")

q5.fc.RDPI <- as.data.frame(cbind(c(2004,2005,2006),
                                  exp(q5.fc$fcst$RDPI)[,1:3],
                                  q5.ts[59:61, 4]))
colnames(q5.fc.RDPI) <- c("Year","Forecast_RDPI","Lower_RDPI",
                          "Upper_RDPI","Actual_RDPI")
```

```r
### Display forecast results
q5.fc.RMC
```

```
##   Year Forecast_RMC Lower_RMC Upper_RMC Actual_RMC
## 1 2004     4081.289  3856.601  4319.067     4133.6
## 2 2005     4417.526  4011.427  4864.736     4548.5
## 3 2006     4711.611  4146.228  5354.091     4799.5
```

```r
q5.fc.RCC
```

```
##   Year Forecast_RCC Lower_RCC Upper_RCC Actual_RCC
## 1 2004     1179.709  1058.528  1314.763     1181.7
## 2 2005     1221.403  1039.799  1434.724     1191.2
## 3 2006     1273.733  1053.449  1540.080     1209.2
```

```r
q5.fc.RDPI
```

```
##   Year Forecast_RDPI Lower_RDPI Upper_RDPI Actual_RDPI
## 1 2004      4571.420   4394.949   4754.977      4595.9
## 2 2005      4703.384   4449.143   4972.155      4626.8
## 3 2006      4840.662   4522.385   5181.340      4723.8
```

Overall these forecasts for 2004-2006 perform quite well when comparing to actual values. For each of the RMC, RCC and RDPI actual values, they are relatively close to the point estimate forecasts and are solidly within the 95% prediction intervals.