# Vector AR, ARIMA, Holt-Winters, Harmonic Regression

*Stone Jiang*

This report will explore 4 techniques in time series modeling and forecasting: 1. Vector Autoregressive Models (Vector AR), 2. ARIMA models, 3. Holt-Winters, 4. Harmonic Regression applied on various datasets. Each data set used will be accompanied with an exploratory data analysis. Each fitted models will be accompanied with a brief theoretical formulation, and an extensive exploration of modeling approaches and decisions, along with forecasting performance (pseudo-out-of-sample testing) and residual analysis.

Vector AR models leverage the linear dependencies between multiple time series in order to conduct forecasts on the joint outcome of 3 related series. Harmonic regression is a technique that captures seasonality with a single Fourier series, and advantageous to seasonal dummy variables for data sets with high seasonality (such as weekly data that we will use). We will also explore the relative advantages of ARIMA versus Holt-Winters, two popular methods for modeling behaviors of time series data.

### Vector autoregression

Three time series are provided for the period 1946-2006 are recorded in `vector_ar.csv`. Data provided as part of coursework in W271 (see note at the bottom of the report). We will develop a VAR model for these data for the period 1946-2003, and then forecast the last three years, 2004-2006, and examine the relative advantages of a logarithmic transform and the use of differences.

```
library(vars)
```

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

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

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

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

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

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

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

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

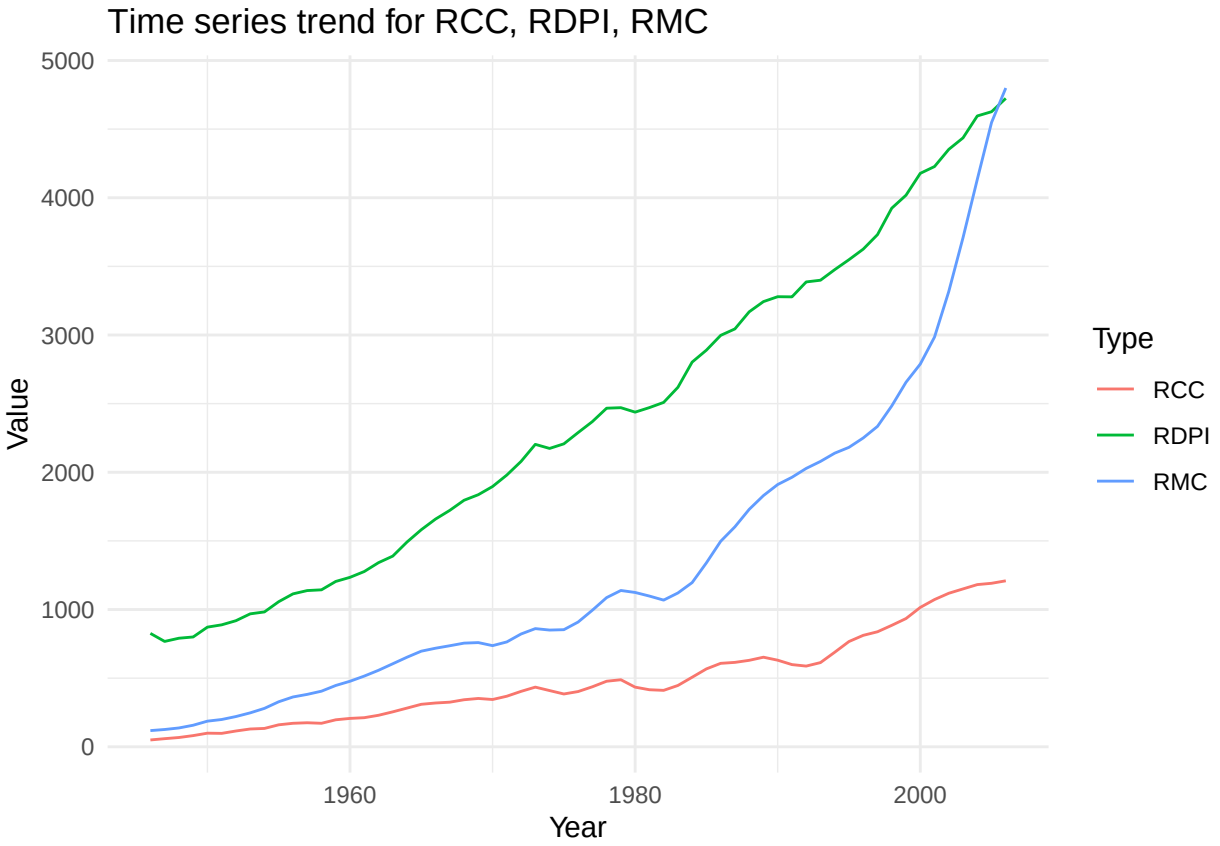
```
##
## Attaching package: 'vars'
```

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

```
dat10 <- read.csv("data/vector_ar.csv")
dat.train <- dat10 %>% dplyr::filter(Year < 2004) %>% dplyr::select(-Year) %>%
    ts(start = 1946)
dat.test <- dat10 %>% dplyr::filter(Year >= 2004) %>% dplyr::select(-Year) %>%
    ts(start = 2004)
```

First, we plot a trend to see how each series is evolving:

```
dat10.plot <- gather(dat10, Type, Value, -Year)
ggplot(dat = dat10.plot, aes(x = Year, y = Value, color = Type)) +
    geom_line() + ggtitle("Time series trend for RCC, RDPI, RMC") +
    theme_minimal()
```




We clearly have an upward trend in all three series, with no indication of seasonality. The growth in RMC looks potentially exponential, which will would problematic for VAR model without transformations. Since the series is not stationary and is increasing with time, we will set type = 'trend' in the VAR model in order to account for the increase. In practice, we will use type = 'both' in order to allow for a constant as well. Also, to compare the relative benefits of log/differencing on forecast error, we will compare by RMSE defined for each series as (for 3 point ahead forecast):

$$RMSE = \sqrt{\frac{1}{3}\Sigma_{t=k}^{3}(y_{2003+k} - \hat{y}_{2003+k})^2}$$

To generate our first a var model on the raw data, we first use VARselect to select the number of lags based on AIC and BIC:

```
VARselect(dat.train, type = "both")[["selection"]]
```

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

It seems like AIC prefers a large number of lags (10), and BIC a smaller number of lags (2). We will fit both and use forecasting accuracy to select between the 2. The model we are trying to fit is (with p equal to number of lags)

$$RMC_t = \sum_{i=1}^{p} \beta_{1i} RMC_{t-p+i} + \sum_{i=1}^{p} \beta_{2i} RCC_{t-p+i} + \sum_{i=1}^{p} \beta_{3i} RDPI_{t-p+i} + const$$

$$RCC_t = \sum_{=1}^{p} \beta_{1j} RMC_{t-p+j} + \sum_{j=1}^{p} \beta_{2j} RCC_{t-p+j} + \sum_{j=1}^{p} \beta_{3j} RDPI_{t-p+j} + const$$

$$RDPI_t = \sum_{k=1}^{p} \beta_{1k} RMC_{t-p+k} + \sum_{k=1}^{p} \beta_{2k} RCC_{t-p+k} + \sum_{k=1}^{p} \beta_{3k} RDPI_{t-p+k} + const$$

We fit the following var1 models with p = 2 (selected by BIC) and p = 10 (selected by AIC)

```
var10 <- vars::VAR(dat.train, p = 10, type = "both")
var2 <- vars::VAR(dat.train, p = 2, type = "both")

var10
```

```
##
## VAR Estimation Results:
## ========================
##
## Estimated coefficients for equation RMC:
## ========================================
## Call:
## RMC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + RMC.l3 + RCC.l3 + RDPI.l3 + RI
##
##        RMC.l1        RCC.l1       RDPI.l1        RMC.l2        RCC.l2
##  1.892734e+00 -1.861092e+00  9.267282e-02 -6.555880e-01  2.491481e-01
##       RDPI.l2        RMC.l3        RCC.l3       RDPI.l3        RMC.l4
## -8.919840e-02 -2.672758e-01 -2.364739e-01 -1.288365e-01 -2.183587e-02
##        RCC.l4       RDPI.l4        RMC.l5        RCC.l5       RDPI.l5
## -3.086720e-01  2.544513e-02  2.929934e-01 -5.821315e-01 -1.453554e-01
##        RMC.l6        RCC.l6       RDPI.l6        RMC.l7        RCC.l7
##  1.199983e-05 -2.639525e-01  3.771325e-02 -4.912752e-01  1.672444e+00
##       RDPI.l7        RMC.l8        RCC.l8       RDPI.l8        RMC.l9
## -1.192111e-01 -6.090053e-01  4.244403e-01 -4.267259e-01  1.332929e+00
##        RCC.l9       RDPI.l9       RMC.l10       RCC.l10      RDPI.l10
## -7.816139e-01 -1.525639e-01  6.969409e-01 -1.517449e+00  4.316926e-01
##         const         trend
##  2.413427e+02  3.555718e+01
##
##
## Estimated coefficients for equation RCC:
## ========================================
## Call:
## RCC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + RMC.l3 + RCC.l3 + RDPI.l3 + RI
##
##        RMC.l1        RCC.l1       RDPI.l1        RMC.l2        RCC.l2
##   0.382911225   0.087502882   0.116051146  -0.369003136  -0.084287097
##       RDPI.l2        RMC.l3        RCC.l3       RDPI.l3        RMC.l4
```

```
##  -0.095335431   0.043965168  -0.365974449  -0.001078826   0.062245326
##        RCC.l4       RDPI.l4        RMC.l5        RCC.l5       RDPI.l5
##   0.114747094  -0.125855259  -0.003865344  -0.174769460  -0.019417475
##        RMC.l6        RCC.l6       RDPI.l6        RMC.l7        RCC.l7
##  -0.060276728   0.118963910   0.094005390  -0.320534960   0.813495703
##       RDPI.l7        RMC.l8        RCC.l8       RDPI.l8        RMC.l9
##  -0.239367965   0.343294033  -0.637149584   0.022516105   0.501477653
##        RCC.l9       RDPI.l9       RMC.l10       RCC.l10      RDPI.l10
##  -0.103780337  -0.078117107  -0.011344286  -0.124198322   0.056503935
##         const         trend
## 149.255774331  15.621847237
##
##
## Estimated coefficients for equation RDPI:
## =========================================
## Call:
## RDPI = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + RMC.l3 + RCC.l3 + RDPI.l3 + R
##
##        RMC.l1        RCC.l1       RDPI.l1        RMC.l2        RCC.l2
##   0.759722866  -1.649725176   0.820860120  -1.057436974   0.909171803
##       RDPI.l2        RMC.l3        RCC.l3       RDPI.l3        RMC.l4
##  -0.070168173  -0.049638435  -0.456894303  -0.008759545   0.553332804
##        RCC.l4       RDPI.l4        RMC.l5        RCC.l5       RDPI.l5
##  -0.542388671  -0.004724648  -0.292459666   0.265468946  -0.116999749
##        RMC.l6        RCC.l6       RDPI.l6        RMC.l7        RCC.l7
##   0.345683123  -1.000637426   0.283305692  -0.495340025   2.138535761
##       RDPI.l7        RMC.l8        RCC.l8       RDPI.l8        RMC.l9
##  -0.571761946  -0.190913597  -1.969300425   0.270999021   1.426067672
##        RCC.l9       RDPI.l9       RMC.l10       RCC.l10      RDPI.l10
##   0.295528346  -0.340390252  -0.142262273  -0.908268744   0.185177819
##         const         trend
## 215.908175776  45.627710582
```

```
var2
```

```
##
## VAR Estimation Results:
## =======================
##
## Estimated coefficients for equation RMC:
## ========================================
## Call:
## RMC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##       RMC.l1       RCC.l1      RDPI.l1       RMC.l2       RCC.l2
##  1.799677538  0.204995390 -0.025617402 -0.786066159  0.001376232
##      RDPI.l2        const        trend
##  0.001161900  8.805267095 -1.312655695
##
```

```
##
## Estimated coefficients for equation RCC:
## =======================================
## Call:
## RCC = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##       RMC.l1       RCC.l1      RDPI.l1       RMC.l2       RCC.l2
##  0.002156404  1.472530177  0.054159186  0.058906327 -0.659156366
##      RDPI.l2        const        trend
## -0.075894860 16.277332551  1.657949117
##
##
## Estimated coefficients for equation RDPI:
## =======================================
## Call:
## RDPI = RMC.l1 + RCC.l1 + RDPI.l1 + RMC.l2 + RCC.l2 + RDPI.l2 + const + trend
##
##       RMC.l1       RCC.l1      RDPI.l1       RMC.l2       RCC.l2
##  0.090310260  0.423063977  0.824607213 -0.070556012 -0.314406489
##      RDPI.l2        const        trend
##  0.005733662 93.061945542  9.071795156
```

We then check whether the residuals are correlated up to 10 lags (as Hyndman recommends) at a significance level of 0.05. The null hypothesis in each case is that the residuals are not serially correlated:

```
serial.test(var2, lags.pt = 10, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var2
## Chi-squared = 92.824, df = 72, p-value = 0.04988
```

```
serial.test(var10, lags.pt = 10, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var10
## Chi-squared = 79.389, df = 0, p-value < 2.2e-16
```

Since p-value $<<$ 0.01 for the var10 model, we reject H0 that the residuals are not serially correlated. There is strong statistical evidence of serial corrlation. While we also reject H0 for the p = 2 model, the p-value is very close to 0.05 so we only have marginal evidence.

We will also test for the presence of the ARCH effect (whether there is constant variance). The null hypothesis is that the residuals of a VAR model are homoscedastic:

```
# Test of the absence of ARCH effect:
arch.test(var2)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var2
## Chi-squared = 229.03, df = 180, p-value = 0.00786
```

```
arch.test(var10)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object var10
## Chi-squared = 183.22, df = 180, p-value = 0.4193
```

Since p-value $< 0.05$ for the var2 model, we reject H0 that the residual are homoskedastic and conclude that we do not have constant variance in the residual series. The var10 model have a high p-value $> 0.05$, so we cannot reject homoskedasticity.

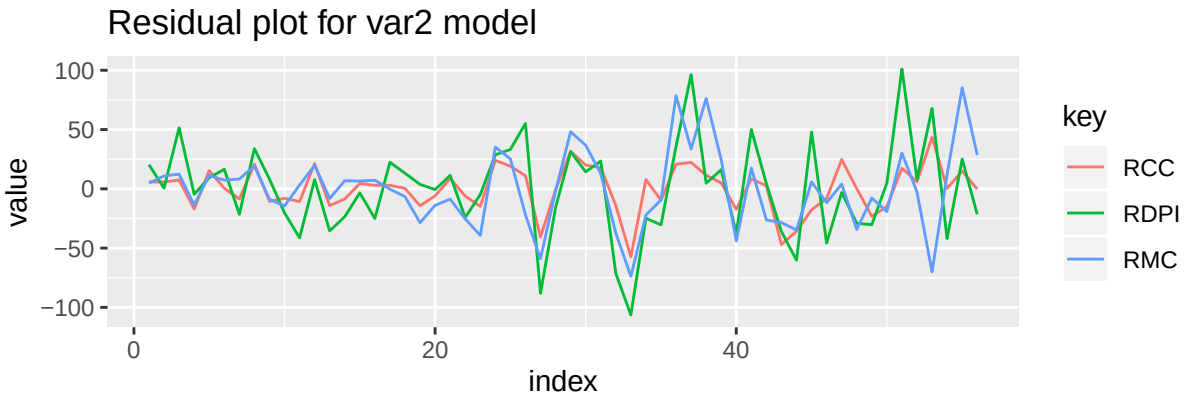
The residual plots are given below:

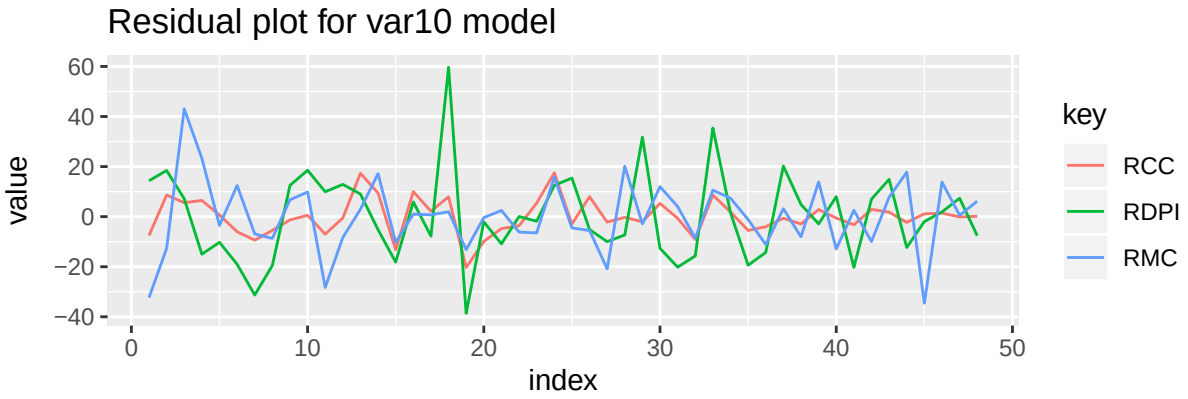
```
library(patchwork)
```

```
##
## Attaching package: 'patchwork'
```

```
## The following object is masked from 'package:MASS':
##
##     area
```

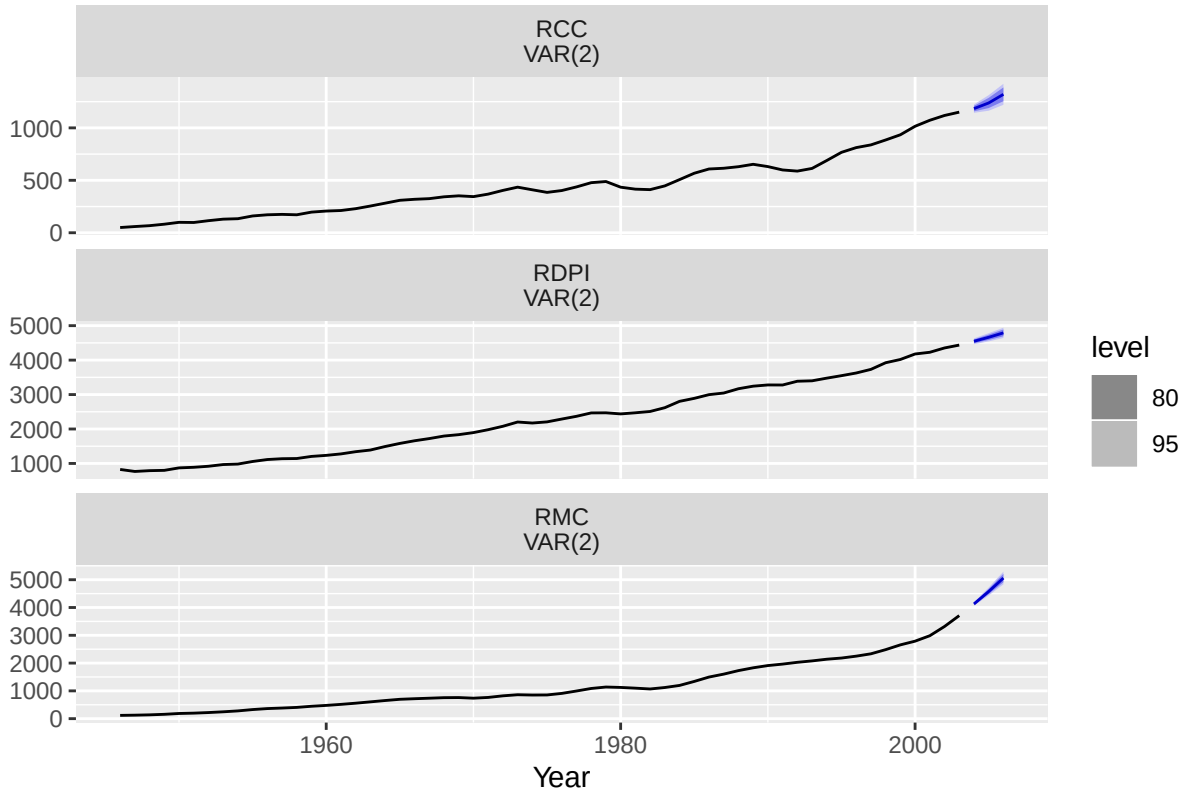
```
residuals2 <- resid(var2) %>% data.frame()
residuals10 <- resid(var10) %>% data.frame()
residuals2$index <- seq(1:length(residuals2$RMC))
residuals10$index <- seq(1:length(residuals10$RMC))

residuals2 <- residuals2 %>% gather(key, value, -index)
residuals10 <- residuals10 %>% gather(key, value, -index)

p2 <- ggplot(data = residuals2, aes(x = index, y = value, color = key)) +
    geom_line() + ggtitle("Residual plot for var2 model")

p10 <- ggplot(data = residuals10, aes(x = index, y = value, color = key)) +
    geom_line() + ggtitle("Residual plot for var10 model")
p2/p10
```




Even though the ARCH test rejected homoskedasticity for the var2 model's residuals, visually, they don't look too different from the var10, except for the larger indices. We will use the var2 model for forecasting, since the residual series is not serially correlated, and since 2 was selected by BIC (the more common criteria for selecting lag order of VAR models):

```
fc <- forecast::forecast(var2, h = 3)
fc %>% forecast::autoplot() + xlab("Year")
```




We now look at accuracy of this model:

```
acc_var <- accuracy(fc, dat.test, d = 1, D = 0)
acc_var
```

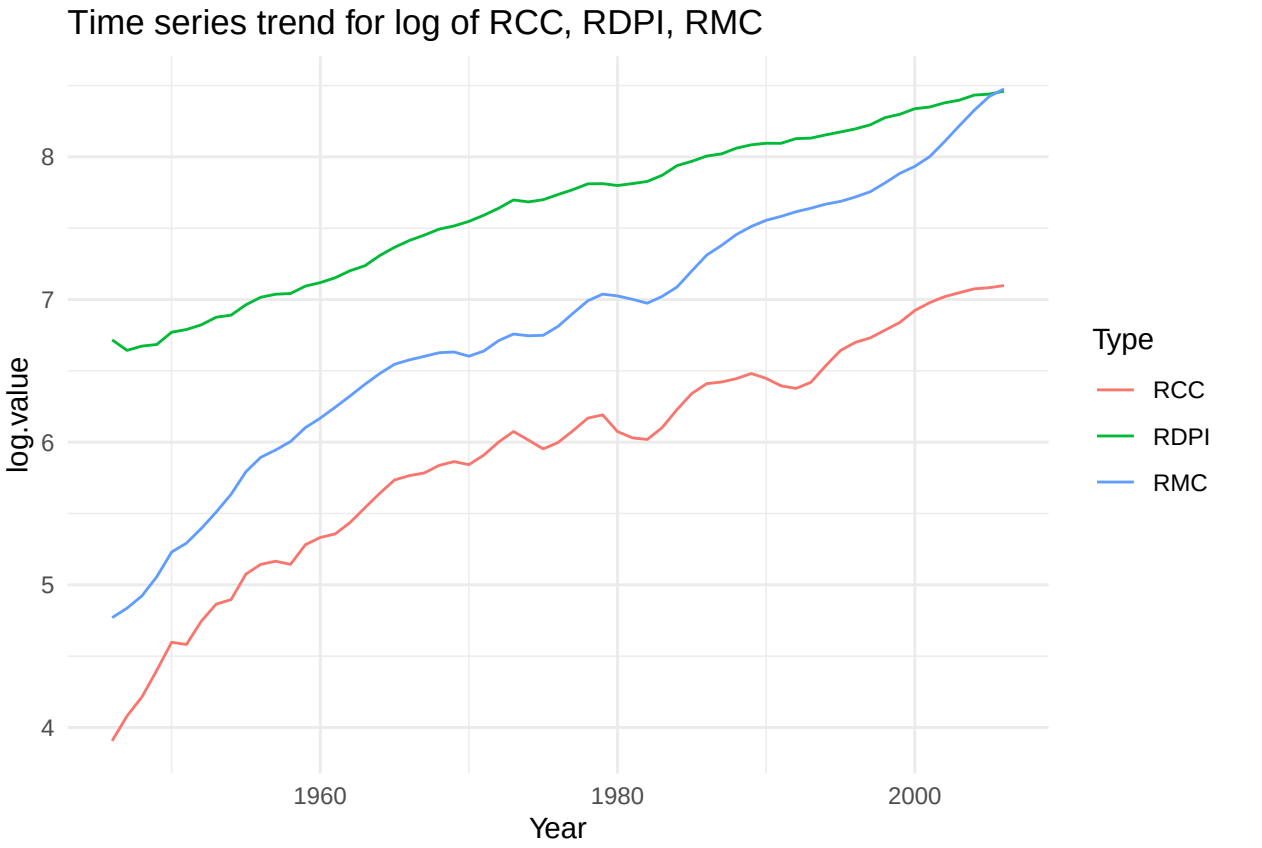
```
##                              ME      RMSE      MAE         MPE     MAPE
## RMC Training set  -9.642615e-15  31.88448 24.01119  0.30706173 2.708963
## RMC Test set      -9.379043e+01 151.22820 97.88001 -1.95767179 2.056607
## RCC Training set   1.776357e-15  18.90443 14.48732  0.12576790 4.337421
## RCC Test set      -5.314409e+01  69.44812 53.14409 -4.41564096 4.415641
## RDPI Training set -2.030370e-15  39.51452 30.24944  0.03232463 1.441354
## RDPI Test set     -1.801096e+01  52.13611 50.54349 -0.37699836 1.084858
##                         MASE         ACF1 Theil's U
## RMC Training set  0.3604999  0.288408579        NA
## RMC Test set      1.4695537 -0.105525013 0.5031875
## RCC Training set  0.5469447  0.147481416        NA
## RCC Test set      2.0063672 -0.007376625 5.9059470
## RDPI Training set 0.4474188  0.051621252        NA
## RDPI Test set     0.7475878 -0.044199689 0.7470147
```

The RMSE is highest for RMC, but on the scale of the values, reasonable for RCC and RDPI.

We perform the same procedure for log transformed data. First we plot the series:

```
dat.train.log <- log(dat.train)
dat.test.log <- log(dat.test)
dat10.plot$log.value <- log(dat10.plot$Value)
ggplot(dat = dat10.plot, aes(x = Year, y = log.value, color = Type)) +
```

```
    geom_line() + ggtitle("Time series trend for log of RCC, RDPI, RMC") +
    theme_minimal()
```




One advantage here is that the growth in RMC now looks more linear, and the growth in the other two series are also roughly linear. Running a VAR model with the trend option might produce a better fit.

```
VARselect(dat.train.log, type = "both")[["selection"]]
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      3      2      3
```

```
# Fit 2 (BIC) and 9 (AIC)
log.var2 <- vars::VAR(dat.train.log, p = 2, type = "both")
log.var9 <- vars::VAR(dat.train.log, p = 9, type = "both")

serial.test(log.var2, lags.pt = 10, type = "PT.asymptotic")
```

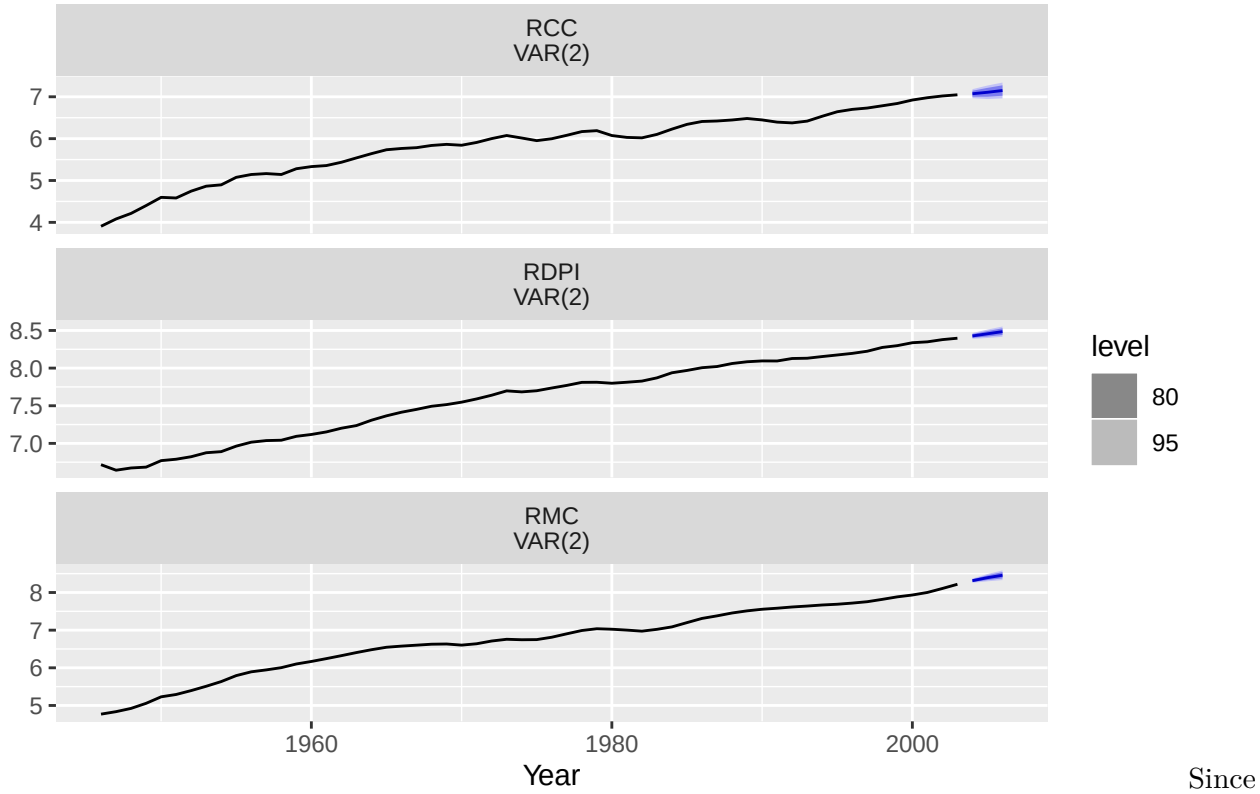
```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object log.var2
## Chi-squared = 80.201, df = 72, p-value = 0.2376
```

```
serial.test(log.var9, lags.pt = 10, type = "PT.asymptotic")
```

```
##
```

```
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object log.var9
## Chi-squared = 84.659, df = 9, p-value = 1.91e-14
```

```
# ONLY the BIC model passes the serial correlation test, so
# we will use this one for forecasting
fc.log <- forecast::forecast(log.var2, h = 3)
fc.log %>% forecast::autoplot() + xlab("Year")
```




Since the BIC model passes the serial correlation test, this is already a potentially better model. We will now also perform an Arch Test:

```
arch.test(log.var2)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object log.var2
## Chi-squared = 182.63, df = 180, p-value = 0.4312
```

Since p-value $> 0.05$, we fail to reject H0 that the residuals are homoskedastic. The plot of the residuals is given below, and these tests shows that the log transformed series provided a better in sample fit.

```
residuals2 <- resid(log.var2) %>% data.frame()
residuals2$index <- seq(1:length(residuals2$RMC))
```

```
residuals2 <- residuals2 %>% gather(key, value, -index)

p2 <- ggplot(data = residuals2, aes(x = index, y = value, color = key)) +
    geom_line() + ggtitle("Residual plot for log var2 model")
p2
```


In order to compare RMSEs for the log model, we will calculate this manually.

```
RMC <- exp(data.frame(fc.log$forecast$RMC)) - exp(data.frame(dat.test.log)$RMC)
RCC <- exp(data.frame(fc.log$forecast$RCC)) - exp(data.frame(dat.test.log)$RCC)
RDPI <- exp(data.frame(fc.log$forecast$RDPI)) - exp(data.frame(dat.test.log)$RDPI)

# RMSE of the model:
log.rmc.rmse <- sqrt(mean(RMC$Point.Forecast^2))
log.rcc.rmse <- sqrt(mean(RCC$Point.Forecast^2))
log.rdpi.rmse <- sqrt(mean(RDPI$Point.Forecast^2))

# non transformed fit
non.test.acc <- acc_var %>% data.frame() %>% dplyr::select(RMSE)
master.results <- cbind(c(log.rmc.rmse, log.rcc.rmse, log.rdpi.rmse),
    c(non.test.acc[2, ], non.test.acc[4, ], non.test.acc[6, ])) %>%
    data.frame
colnames(master.results) <- c("log transformed var model RMSE",
    "raw data var model RMSE")
```

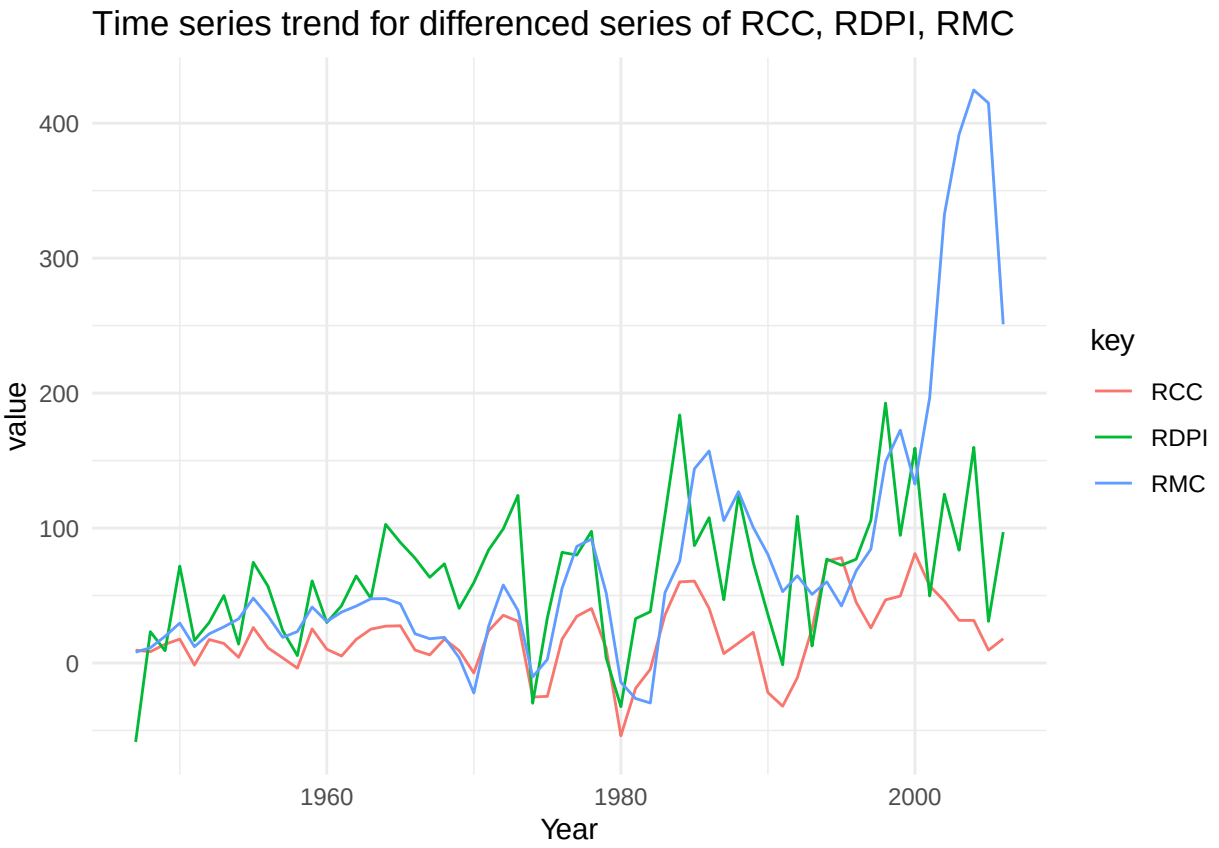
```
rownames(master.results) <- c(rownames(non.test.acc)[2], rownames(non.test.acc)[4],
    rownames(non.test.acc)[6])

master.results
```

```
##                log transformed var model RMSE raw data var model RMSE
## RMC Test set                         95.94286                151.22820
## RCC Test set                         41.15310                 69.44812
## RDPI Test set                        81.89669                 52.13611
```

The log transformed model has much better RMSE for the 3 year test set for the first two series. Given this and the in-sample fit, the log-transformed series is likely better.

We will now generate the model for the differenced series.

```
dat.diff.train <- diff(dat.train)
dat.diff.test <- diff(dat.test)

dat10.ts <- ts(dat10, start = 1946)
dat10.ts.diff <- dat10.ts %>% diff()

dat10.ts.plot <- data.frame(dat10.ts.diff) %>% gather(key, value,
    -Year)
dat10.ts.plot$Year <- dat10$Year[-1]

ggplot(dat = dat10.ts.plot, aes(x = Year, y = value, color = key)) +
    geom_line() + ggtitle("Time series trend for differenced series of RCC, RDPI, RMC") +
    theme_minimal()
```




While the RCC and RDPI series appear to be mean stationary, the RMC has a large amplitude motion toward the end of the series, signifying the rate of increase decreased during this later period. It is unlikely the VAR model will be able to fit this well. Since there appears to be non-stationarity for RMC, we will again fit with type = both.

```
VARselect(dat.diff.train, type = "both")[["selection"]]
```

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

```
# Fit 2 and 9
diff.var2 <- vars::VAR(dat.diff.train, p = 2, type = "both")
diff.var9 <- vars::VAR(dat.diff.train, p = 9, type = "both")

serial.test(diff.var2, lags.pt = 10, type = "PT.asymptotic")
```

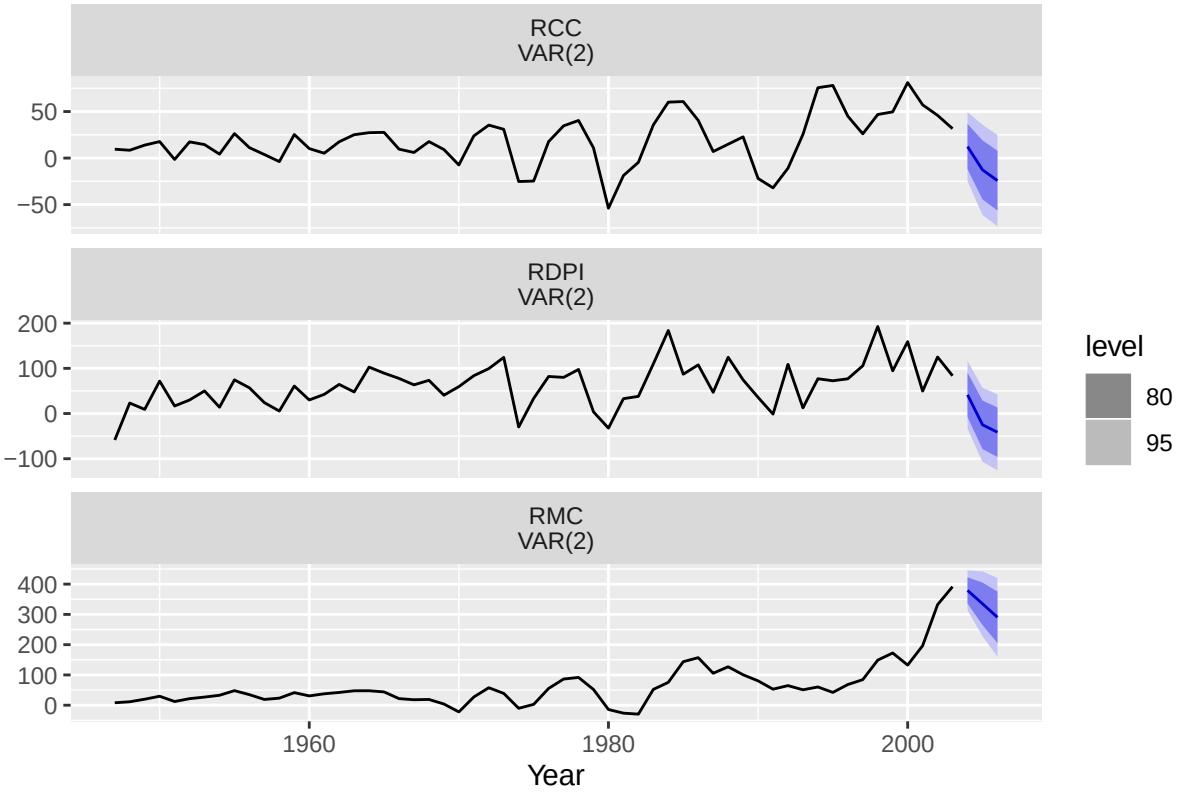
```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object diff.var2
## Chi-squared = 74.282, df = 72, p-value = 0.4038
```

```
serial.test(diff.var9, lags.pt = 10, type = "PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
```

```
## data:  Residuals of VAR object diff.var9
## Chi-squared = 71.598, df = 9, p-value = 7.395e-12
```

```
# We'll pick 2 from BIC as it's the only that that passes
fc.diff <- forecast::forecast(diff.var2, h = 3)
fc.diff %>% forecast::autoplot() + xlab("Year")
```




Checking homoskedasticity:

```
arch.test(diff.var2)
```

```
##
##  ARCH (multivariate)
##
## data:  Residuals of VAR object diff.var2
## Chi-squared = 238.99, df = 180, p-value = 0.002139
```

We reject H0 that the series are homoskedastic. The log model appears better in this respect. We will check the residuals plot to determine in-sample fit.

```
residuals2 <- resid(diff.var2) %>% data.frame()
residuals2$index <- seq(1:length(residuals2$RMC))

residuals2 <- residuals2 %>% gather(key, value, -index)

p2 <- ggplot(data = residuals2, aes(x = index, y = value, color = key)) +
    geom_line() + ggtitle("Residual plot for log var2 model")
p2
```


The residual series appears to be mean stationary for all 3 as the test suggested, although the variance tends to increase toward later years.

To calculate the forecast accuracy, we will generate this manually.

```
# Forecast on the differenced series
RMC.diff.forecast <- fc.diff$forecast$RMC$mean
RCC.diff.forecast <- fc.diff$forecast$RCC$mean
RDPI.diff.forecast <- fc.diff$forecast$RDPI$mean

# Final values of the original series
RMC.prev <- dat.train[dim(dat.train)[1], 1] %>% as.numeric()
RCC.prev <- dat.train[dim(dat.train)[1], 2] %>% as.numeric()
RDPI.prev <- dat.train[dim(dat.train)[1], 3] %>% as.numeric()

# Forecast on the original series
fc.diff.orig <- matrix(nrow = 3, ncol = 3)

for (i in seq(1, 3)) {
    fc.diff.orig[i, 1] <- RMC.prev <- RMC.diff.forecast[i] +
        RMC.prev
    fc.diff.orig[i, 2] <- RCC.prev <- RCC.diff.forecast[i] +
        RCC.prev
    fc.diff.orig[i, 3] <- RDPI.prev <- RDPI.diff.forecast[i] +
        RDPI.prev
```

```
}

results <- data.frame(fc.diff.orig)
colnames(results) <- c("RMC", "RCC", "RDPI")
rownames(results) <- time(dat.test)
results
```

```
##           RMC      RCC     RDPI
## 2004 4088.335 1162.431 4477.536
## 2005 4423.478 1149.745 4452.657
## 2006 4713.853 1125.511 4411.175
```

We will now calculate the accuracy:

```
sq <- (results - data.frame(dat.test))^2
results.RMC <- sqrt(mean(sq$RMC))
results.RCC <- sqrt(mean(sq$RCC))
results.RDPI <- sqrt(mean(sq$RDPI))
master.results$"differenced series var model RMSE" <- c(results.RMC,
    results.RCC, results.RDPI)
master.results
```

```
##               log transformed var model RMSE raw data var model RMSE
## RMC Test set                         95.94286                151.22820
## RCC Test set                         41.15310                 69.44812
## RDPI Test set                        81.89669                 52.13611
##               differenced series var model RMSE
## RMC Test set                            91.31411
## RCC Test set                            55.05671
## RDPI Test set                          217.61612
```

Based on the RMSE on the test set, while the RMSE for RMC is slightly better for the differenced model compared to the log, the RCC and RDPI are both much worse. RDPI is much worse than the model built on the original series. This is likely because the differenced model predicts RDPI to decrease, while in reality the change was positive. Both models tend to outperform the original raw series on RMC and RCC, but worse on RDPI. The log transformed model is still the best model out of the three given that the average RMSE across all three series is better.

*

*Harmonic regression**

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

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

and so on.

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

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

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

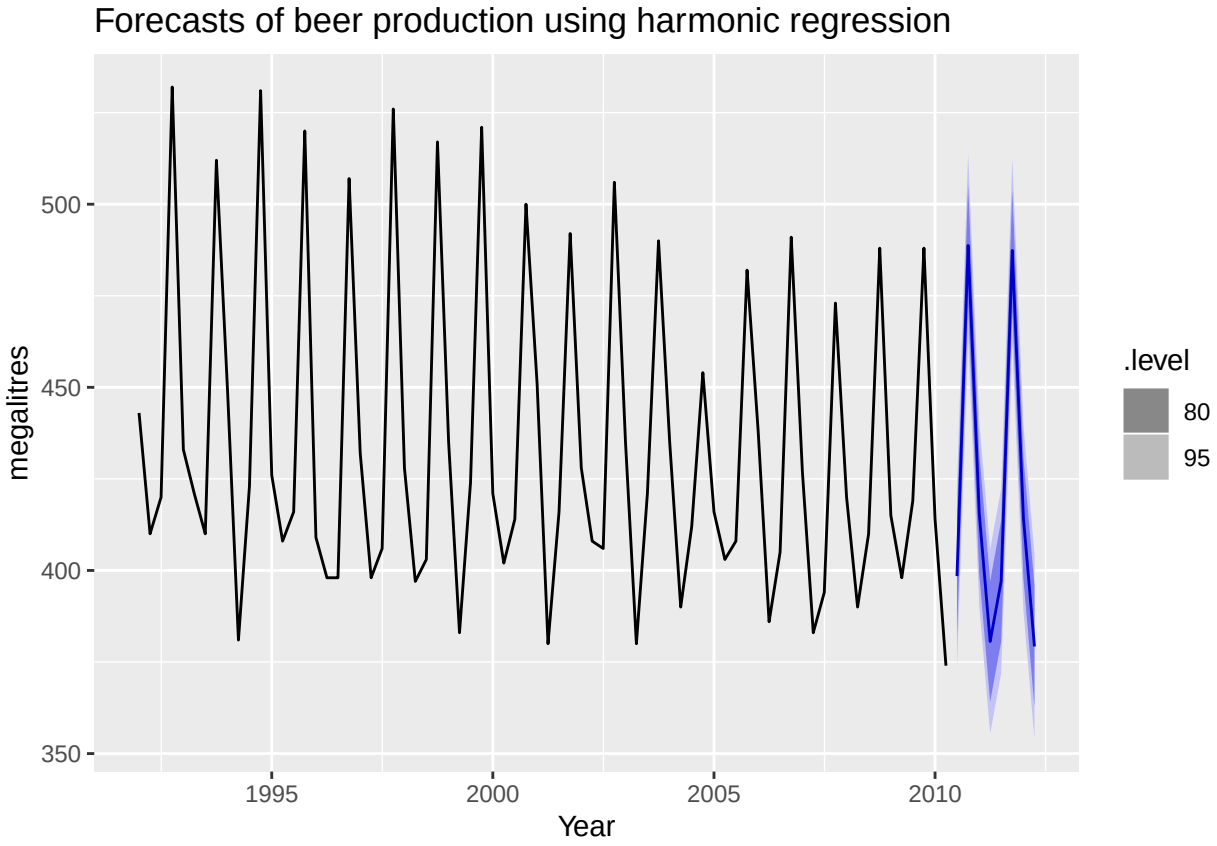
For example, Australian beer production data as provided by fpp3 can be modelled like this:

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

```
## Series: Beer
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -42.9029  -7.5995  -0.4594   7.9908  21.7895
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        446.87920    2.87321 155.533  < 2e-16 ***
## trend()             -0.34027    0.06657  -5.111 2.73e-06 ***
## fourier(K = 2)C1_4   8.91082    2.01125   4.430 3.45e-05 ***
## fourier(K = 2)S1_4 -53.72807    2.01125 -26.714  < 2e-16 ***
## fourier(K = 2)C2_4 -13.98958    1.42256  -9.834 9.26e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.23 on 69 degrees of freedom
## Multiple R-squared: 0.9243,  Adjusted R-squared: 0.9199
```

```
## F-statistic: 210.7 on 4 and 69 DF, p-value: < 2.22e-16
```

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




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

First, we fit a harmonic regression with trend to the data by selecting the appropriate number of Fourier terms to include by minimizing the AIC value. We will scan up to m/2 = 26 for K, the order of Fourier terms.

```
gas.fit <- us_gasoline %>% filter(year(Week) <= 2004)

# generate matrix to store values
aic.scan <- matrix(nrow = 26, ncol = 2)

for (K in seq(1, 26)) {
    fourier_gas <- gas.fit %>% model(TSLM(Barrels ~ trend() +
        fourier(K = K))) %>% glance()
    aic.scan[K, 1] <- K
    aic.scan[K, 2] <- fourier_gas$AIC
}
aic.scan <- data.frame(aic.scan)
```

```
colnames(aic.scan) <- c("K", "AIC")

# Get the top 5 values
aic.scan[order(aic.scan$AIC), ] %>% head(5)
```

```
##     K       AIC
## 7   7 -1888.219
## 12 12 -1886.599
## 11 11 -1886.573
## 8   8 -1885.649
## 6   6 -1884.218
```

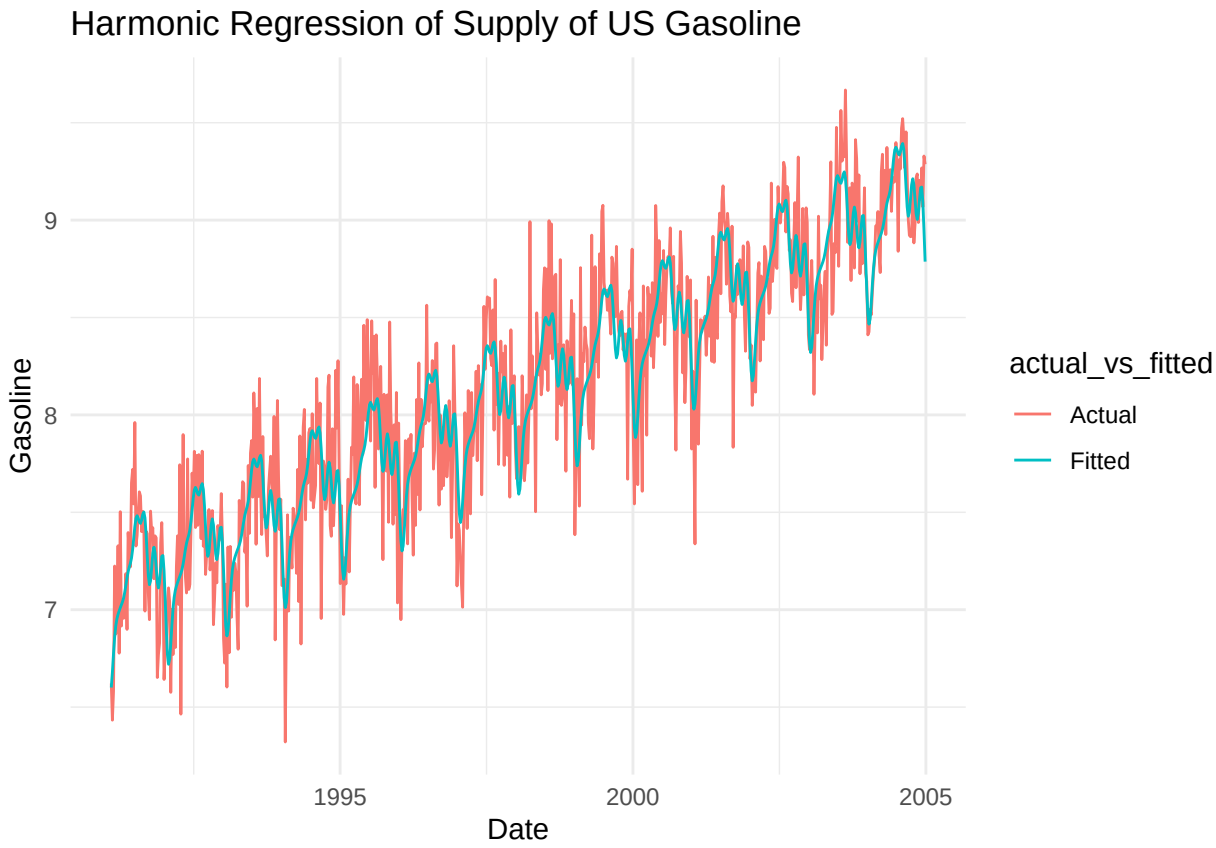
The best AIC occurs at f=7, so this will be our model:

```
fourier_gas <- gas.fit %>% model(TSLM(Barrels ~ trend() + fourier(K = 7)))
fourier_gas %>% glance()
```

```
## # A tibble: 1 x 15
##   .model r_squared adj_r_squared sigma2 statistic   p_value    df log_lik
##   <chr>      <dbl>         <dbl>  <dbl>     <dbl>     <dbl> <int>   <dbl>
## 1 TSLM(~     0.848         0.845 0.0724      264. 2.66e-278    16   -69.0
## # ... with 7 more variables: AIC <dbl>, AICc <dbl>, BIC <dbl>, CV <dbl>,
## #   deviance <dbl>, df.residual <int>, rank <int>
```

We now plot the observed and fitted gas values:

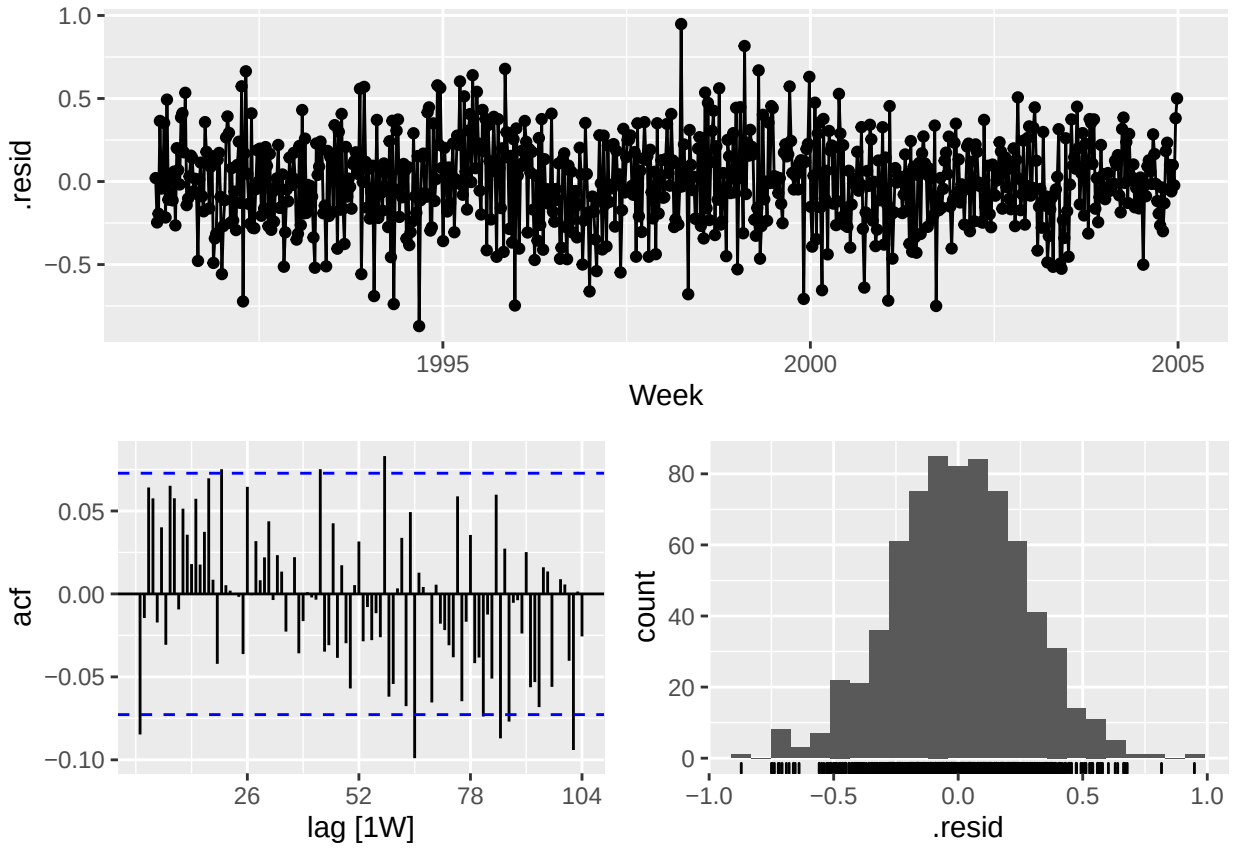
```
fitted.plot <- augment(fourier_gas)
colnames(fitted.plot) <- c(".model", "Week", "Actual", "Fitted",
    ".resid")
fitted.plot <- fitted.plot %>% gather(actual_vs_fitted, value,
    -.model, -Week, -.resid)

ggplot(data = fitted.plot, aes(x = Week, y = value, color = actual_vs_fitted)) +
    geom_line(alpha = 1) + labs(x = "Date", y = "Gasoline", title = "Harmonic Regression of Su
    theme_minimal()
```




We see that while the actual values fluctuates significantly more, the fitted values are perfectly periodic, even within the weekly fluctuations. However, the fitted series is able to capture the complex seasonality of the original trend, which is typically down toward the beginning of a year and climbs until summer, before coming down again. The trend is also neatly captured by the regression.

We now check the residuals of the final model, and use a Ljung-Box test to check for correlation in the residuals.

```
gg_tsresiduals(fourier_gas, lag_max = 104)
```




The residuals look stationary with constant variance with 0 mean, so the model has a good in-sample fit. There are also no extreme values, and the distribution looks roughly normal. There are few points on the far negative side and positive side (-1 and 1), but these are very few and overall there is nothing to worry about since they are a small fraction of the training data.

There are around 8 significant lags on the ACF plot. Since the CIs indicate 95% significant, we expect about 5% of them to be signficant. 8/104 is a bit larger than 5%.

We will test this with a Ljung_box test, where H0 is that the residual series is indistinguishable from white noise. We will also use lag = 2m as recommended by Hyndman. The m here is the seasonal period, corresponding to 52 weeks, so 2m = 104.

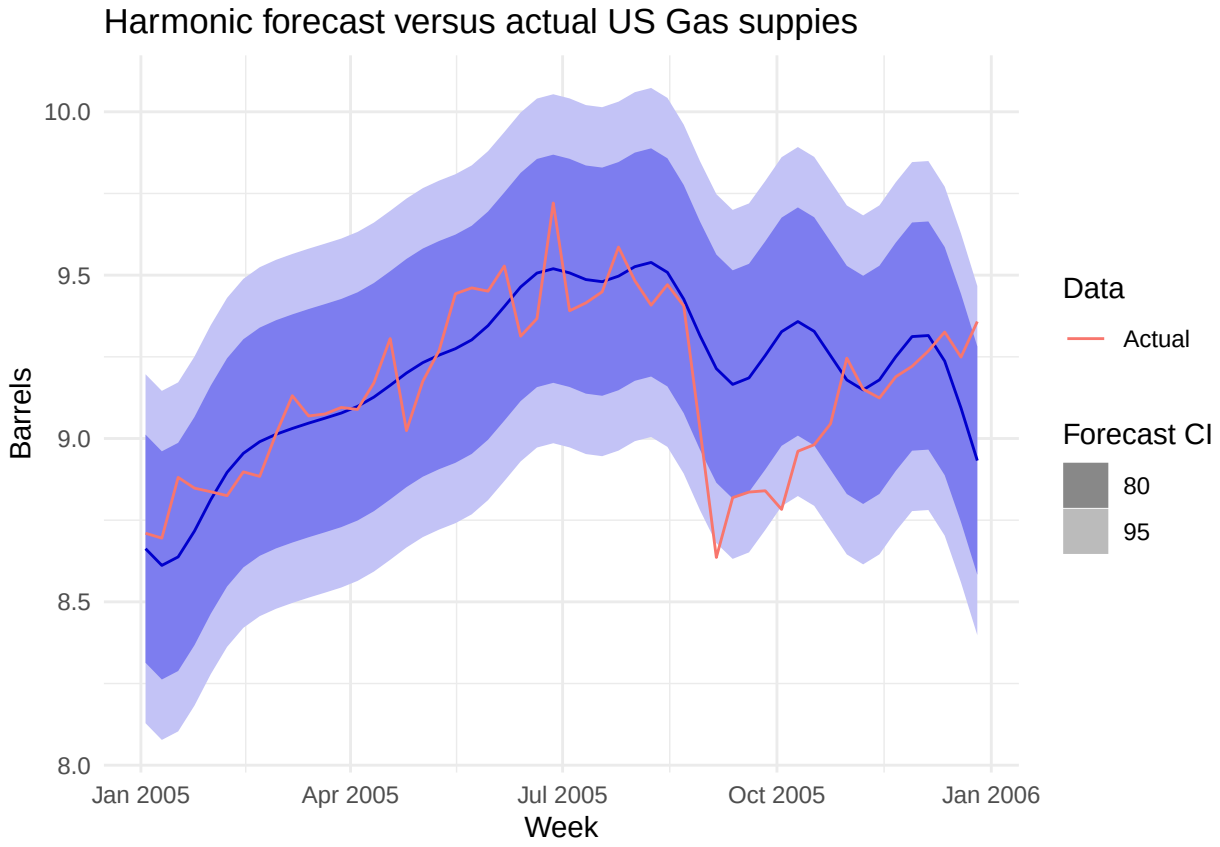
```
fourier_gas %>% augment() %>% features(.resid, ljung_box, lag = 104)
```

```
## # A tibble: 1 x 3
##   .model                                    lb_stat lb_pvalue
##   <chr>                                       <dbl>     <dbl>
## 1 TSLM(Barrels ~ trend() + fourier(K = 7))    144.   0.00561
```

Since the p-value is << 0.05, we reject H0 that the residuals are white noise and conclude that we do have serial correlation in the residual series. Even though the residuals fail the correlation tests, the results are probably not severe enough to make much difference to the forecasts and prediction intervals. (Note that the correlations are reltively small, even though they are significant.)

Forecasting into 2005:

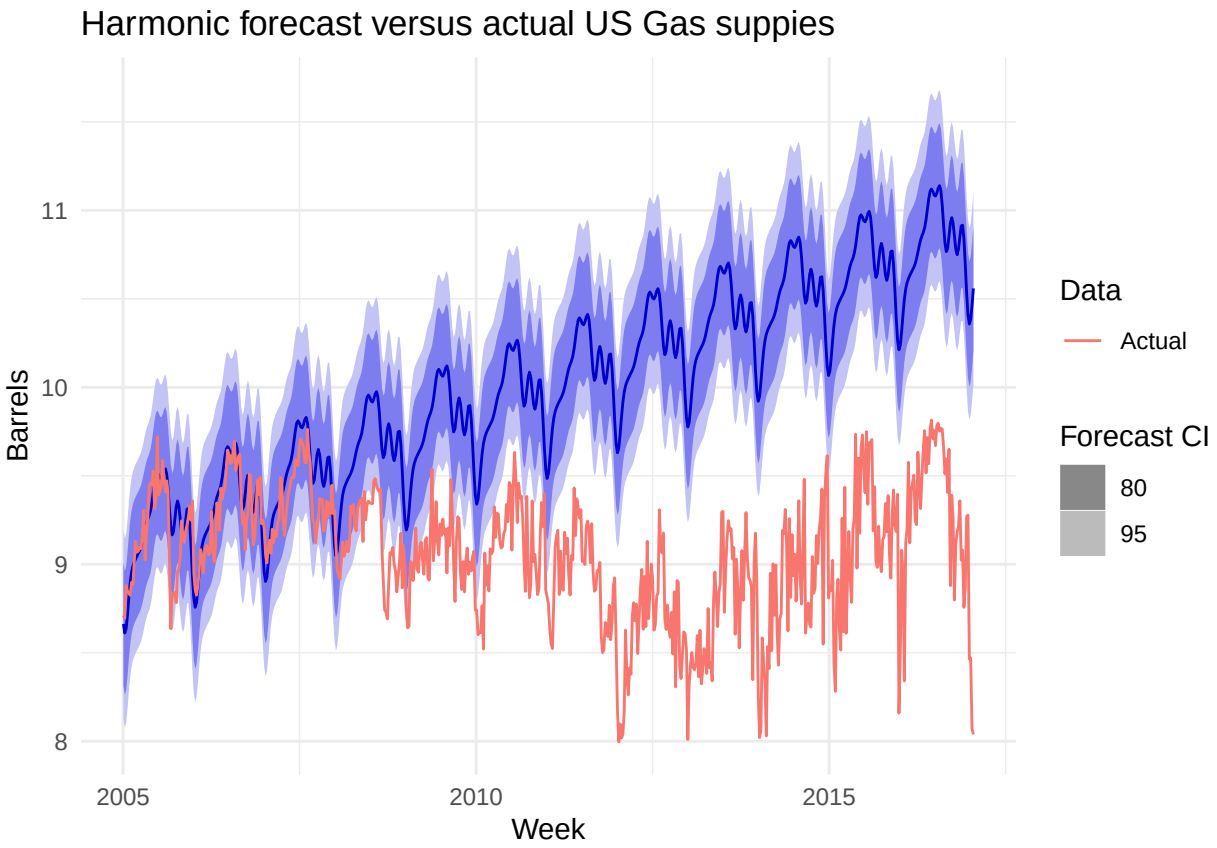
```
test.gas <- us_gasoline %>% dplyr::filter(year(Week) == 2005)
fc.gas <- fourier_gas %>% forecast(test.gas)

# Plot 2004 and 2005 fc.gas %>% autoplot(gas.fit %>%
# dplyr::filter(year(Week) >= 2004, year(Week) <= 2005)) +

# Plot only 2005
fc.gas %>% autoplot() + geom_line(data = test.gas, aes(x = Week,
    y = Barrels, color = "Actual")) + labs(title = "Harmonic forecast versus actual US Gas supp
    level = "Forecast CI", colour = "Data") + theme_minimal()
```




Zooming in closely, we see that most of the actual values are captured by the 95% CI. The dip around end of August is underestimated by the model (only week that falls outside of the 95% CI), but this could be a rare feature of this particular year that the drop was so drastic. The trend increase until July is captured very well, including some of the weekly fluctuations. The increases after Sept. as well as the small fluctuation in Nov. looks good as well.

Let's see for all dates in our test set:

```
test.gas <- us_gasoline %>% dplyr::filter(year(Week) > 2004)
fc.gas <- fourier_gas %>% forecast(test.gas)

fc.gas %>% autoplot() + geom_line(data = test.gas, aes(x = Week,
    y = Barrels, color = "Actual")) + labs(title = "Harmonic forecast versus actual US Gas supp
    level = "Forecast CI", colour = "Data") + theme_minimal()
```




The forecast is perfectly periodic series with a constant upward trend; however, in reality the product decreased starting in around 2008, and continuted to decrease until around 2013. Economic events that caused this cannot be predicted by the model as past data do not display such a fluctuating trend.

### Seasonal ARIMA model

The series of E-Commerce Retail Sales as a Percent of Total Sales is collected from FRED.

Citation: U.S. Census Bureau, E-Commerce Retail Sales as a Percent of Total Sales [ECOMPCTNSA], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/ECOMPCTNSA, April 20, 2020.

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

```
library(fredr)

ecom <- read.csv("data/ecom.csv")
ecom$date <- as.Date(ecom$date)
ecom <- as_tsibble(ecom, index = date)

# since we have quarterly data
ecom$date <- yearquarter(ecom$date)
head(ecom)
```

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

In order to choose an ARIMA model to fit to the ECOMPCTNSA series, we will take the following steps:

1. Perform EDA by plotting the data. Identify any unusual observations.
2. If necessary, transform the data (e.g. using a Box-Cox transformation) to stabilize the variance.
3. If the data are non-stationary: take first differences of the data until the data are stationary.
4. Examine the `ACF`/`PACF`: Is an `AR(p)` or `MA(q)` model appropriate?
5. Try your chosen model(s), and use appropriate metrics to choose a model.
6. Model evaluation Check the residuals from your chosen model by plotting the `ACF` of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
7. Once the residuals look like white noise, calculate forecasts.

*Step 1 - EDA*

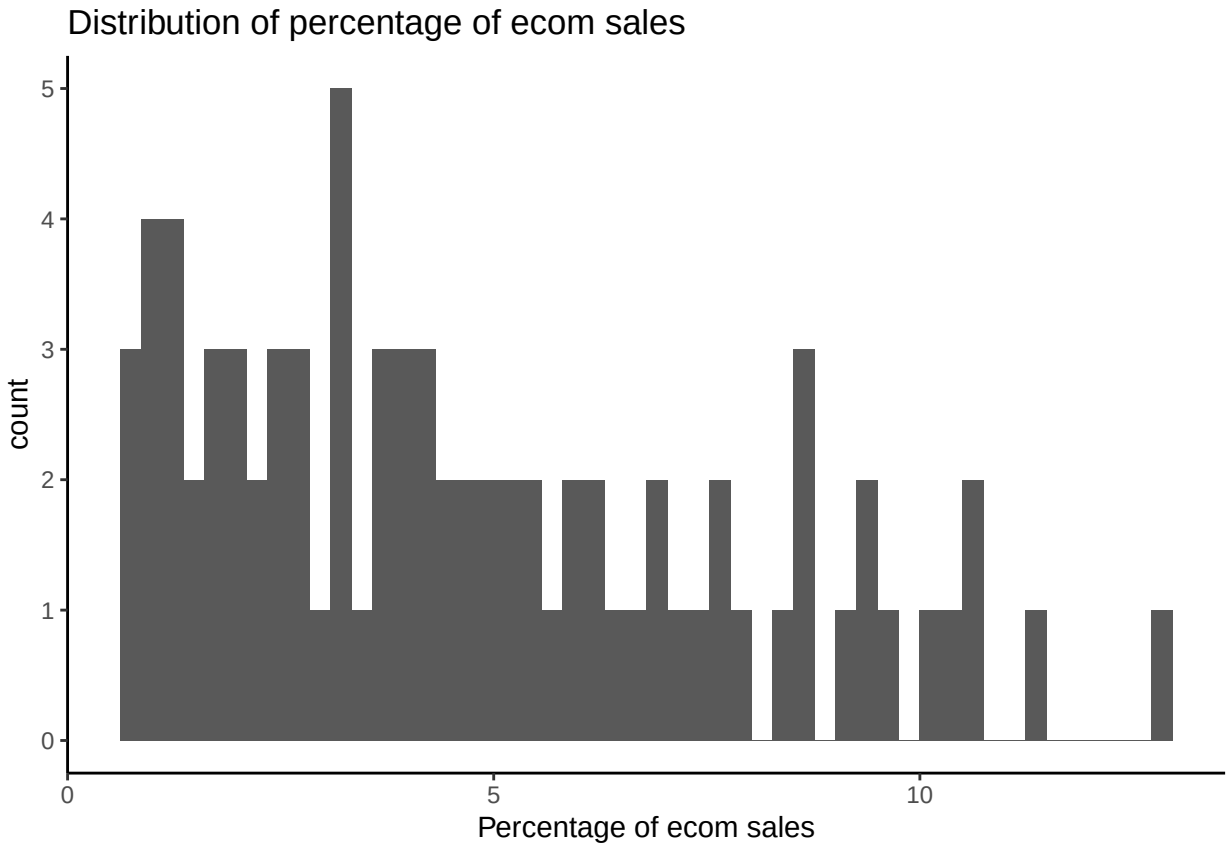
We will first look at aggregate statistics.

```
summary(ecom$value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.700   2.300   4.100   4.758   6.900  12.800
```

The range goes from 0.7 to 12.8, with a mean higher than the median indicating a likely right skew. We can see this from the histogram:

```
ggplot(data = data.frame(ecom), aes(x = value)) + geom_histogram(bins = 50) +
    labs(title = "Distribution of percentage of ecom sales",
        x = "Percentage of ecom sales") + theme_classic()
```

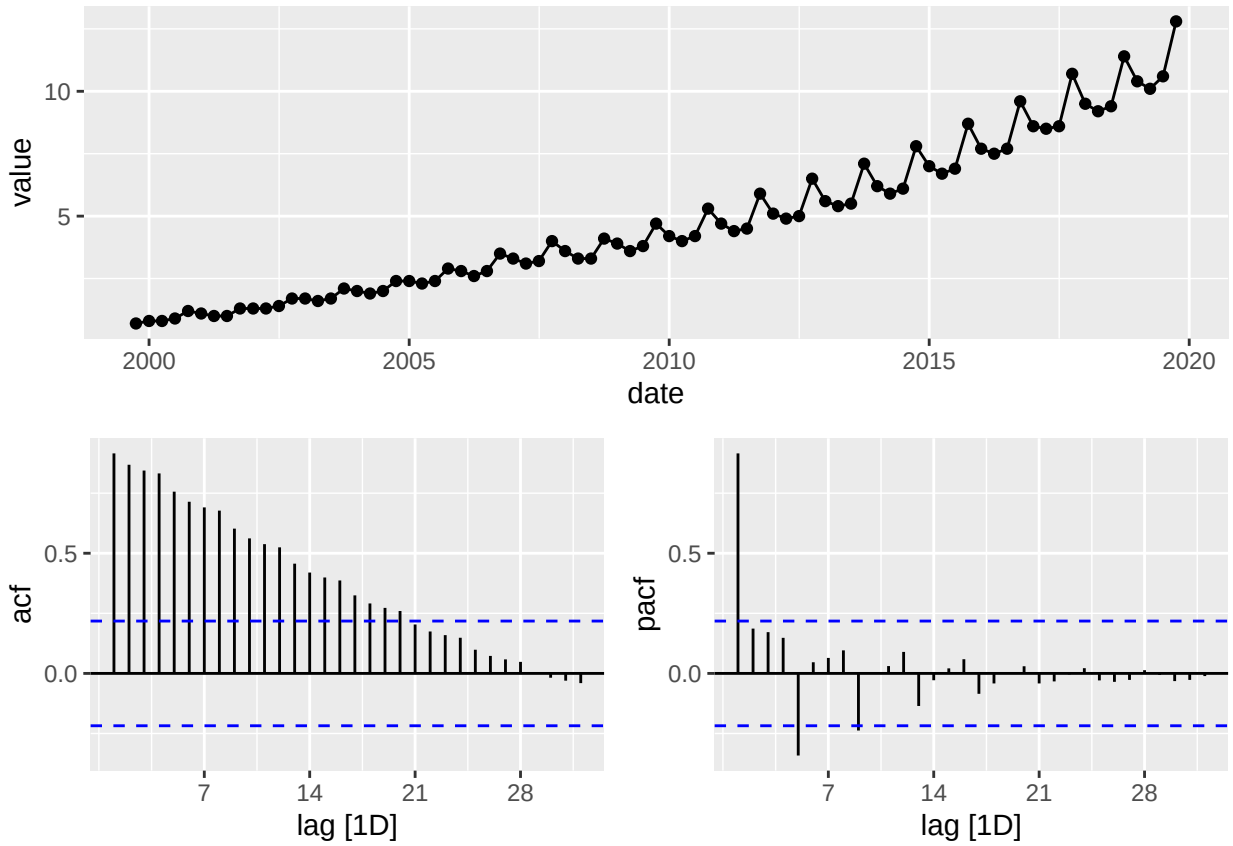



The histogram suggests that the distribution is not normal, and for this section of the time series, most values are under 5%, with a new outliers $> 10\%$.

Next we examine the time series, and ACF and PACF plots:

```
gg_tsdisplay(ecom, plot = "partial", lag_max = 32)
```

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



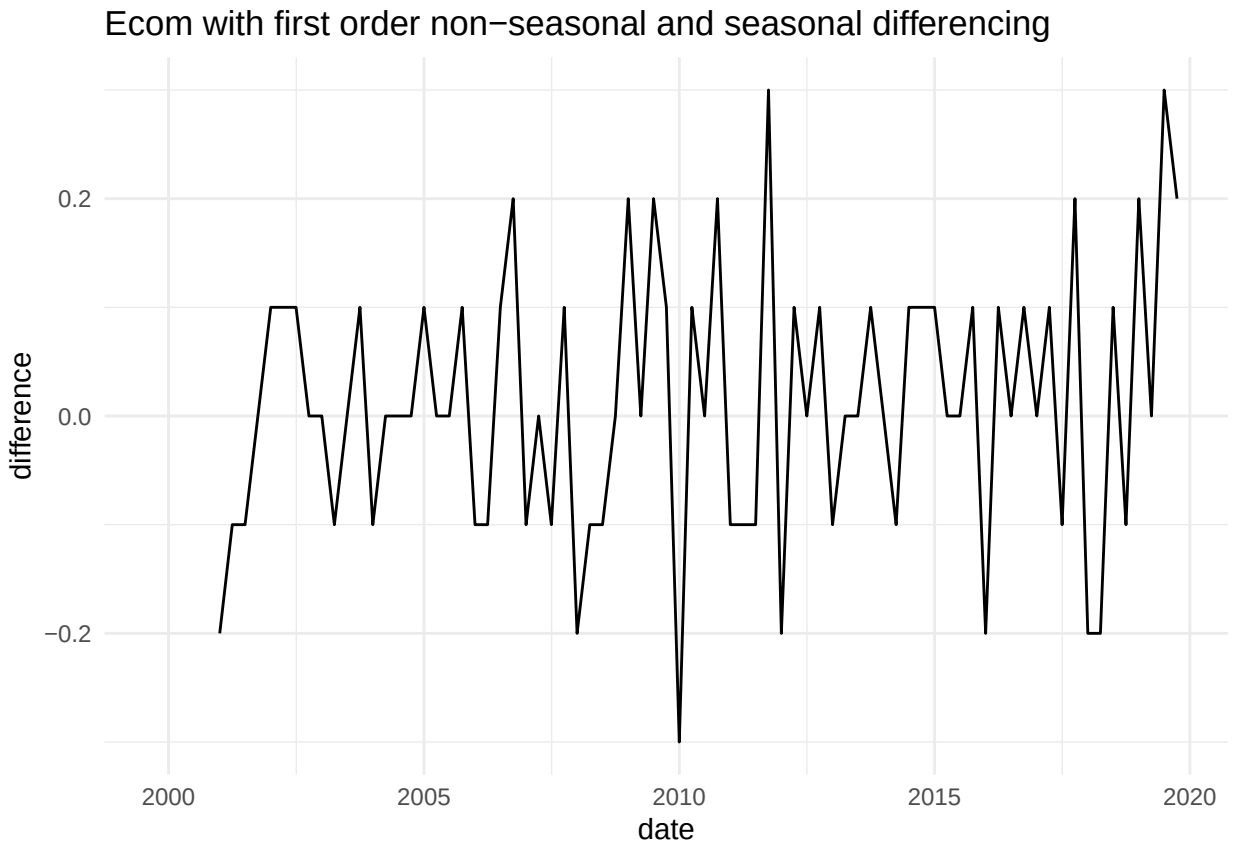

We see that there is a clear trend in the data that looks stronger than linear, and potentially quadratic or exponential. This indicates that the fraction of E-commerce is increasing. In addition, the variance within each year is increasing as year progresses, seeing less variation in the early part of the times series, and larger variation in the later part of the series. The plot also shows strong seasonality. It appears that in each year, Q4 has the highest percentage of ecom sales, and dipping at Q2.

The ACF shows a slow linear decay toward 0 with little seasonality, while the PACF cuts off after lag 1, but another significant value appears at lag 5. This is either due to chance, or suggests that the our model might require a mixture of MA and AR terms.

We next look at the trend in the series, and attempt to generate a stationary series. To do this, we difference the series to see whether the change in ecom is constant. Due to seasonality, we also difference at lag 4 in order to remove seasonal effects,

```
ecom.explore <- ecom
ecom.explore$difference <- ecom$value %>% difference() %>% difference(4)

ggplot(data = ecom.explore, aes(x = date, y = difference)) +
    geom_line() + ggtitle("Ecom with first order non-seasonal and seasonal differencing") +
    theme_minimal()
```

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




We can see that this series is now relatively stationary with mean near 0, meaning first order differencing in seasonal and non-seasonal terms removed the trend. We can verify this with the Augmented Dickey-Fuller Test:

```
library(tseries)
adf.test(ecom.explore$difference[-1:-5])
```

```
## Warning in adf.test(ecom.explore$difference[-1:-5]): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ecom.explore$difference[-1:-5]
## Dickey-Fuller = -4.3373, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Since the p-value $< 0.01$, we reject H0 that the series is not stationary, and conclude that the series is mean stationary, consistent with manual observation.

Next, we check for stability of variance as the ADF test only shows stability in the mean. The differenced series looks stable visually, but we can look for heteroskedastic behavior using an ARCH test from the FinTS package. The null hypothesis is that the series is homoskedastic (no ARCH effects). We will test at a value of 0.05:

```
ArchTest(ecom.explore$difference)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  ecom.explore$difference
## Chi-squared = 15.63, df = 12, p-value = 0.2088
```

Since the p-value $> 0.05$, we conclude that the series is stationary in the variance as well.

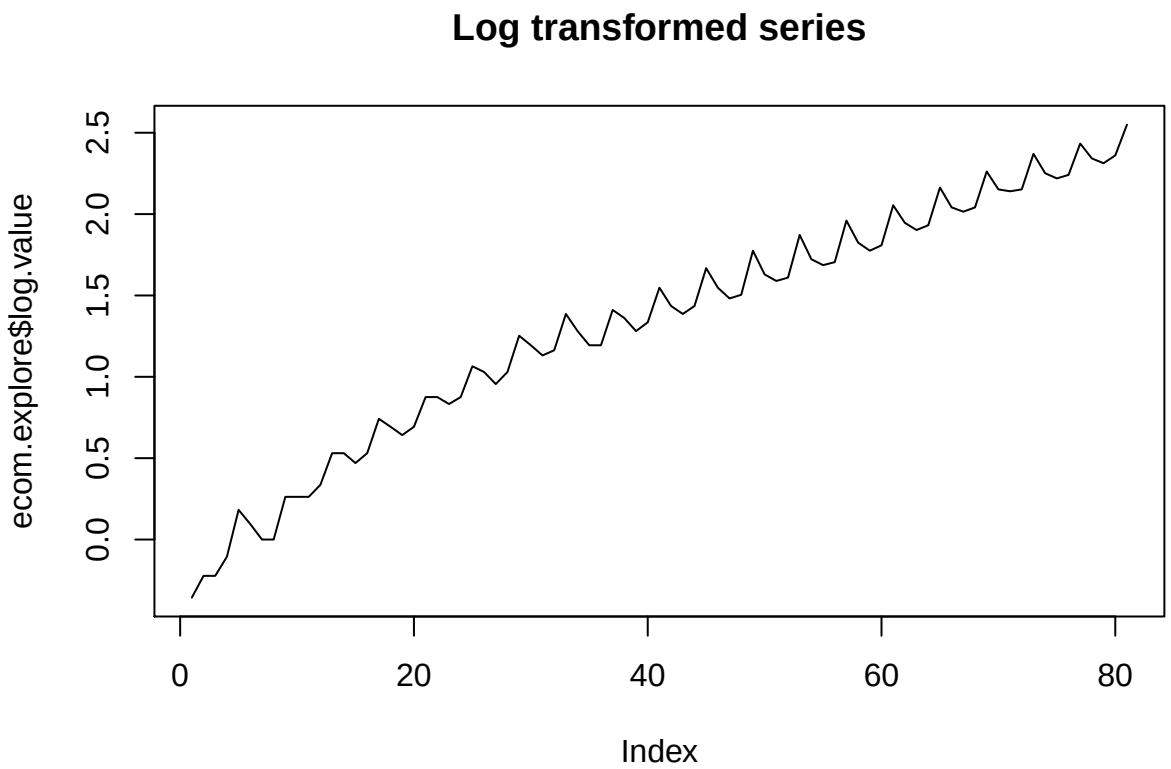
Next, we examine the variance in the original series. Since it is clearly increasing in the original series, we can stability this by performing a log or BoxCox transformation. To get the lambda for the BoxCox transformation, we use BoxCox.lambda, which returns approximately 0, so both BoxCox with this lambda and log transformations will provide very similar series.

```
BoxCox.lambda(ecom$value)
```

```
## [1] 0.01337716
```

We will opt for the log transform:

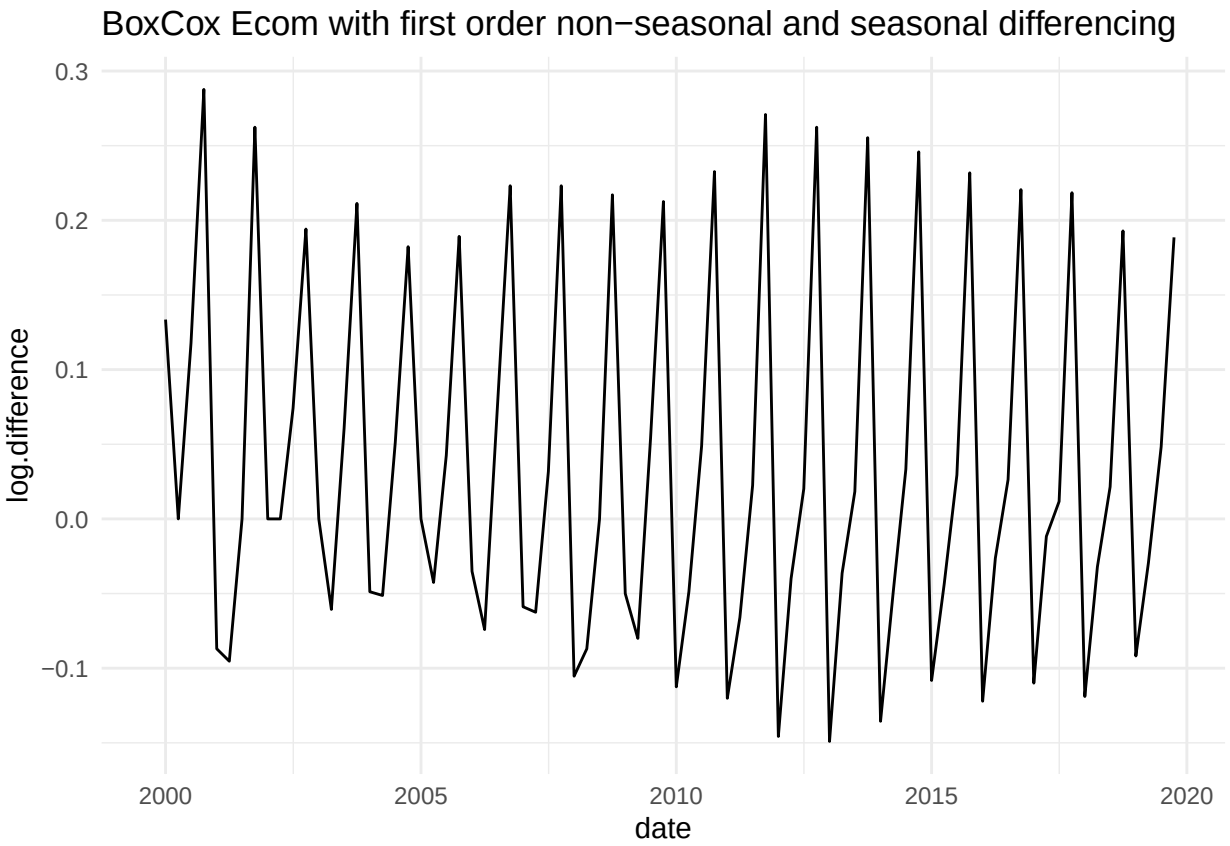
```
ecom.explore$log.value <- log(ecom$value)
plot(ecom.explore$log.value, type = "l", main = "Log transformed series")
```




We can see that compared to the original series, the variance has been stabilized. We now observe how differencing on this series in order to remove trend and seasonal affects looks:

```
ecom.explore$log.difference <- ecom.explore$log.value %>% difference()

ggplot(data = ecom.explore, aes(x = date, y = log.difference)) +
    geom_line() + ggtitle("BoxCox Ecom with first order non-seasonal and seasonal differencing
    theme_minimal()
```

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




This series has large amplitudes in around 2012, and the mean looks constant. The variance looks relatively constant as well. We will look at both the ADF and ARCH tests:

```
adf.test(ecom.explore$log.difference[-1])
```

```
## Warning in adf.test(ecom.explore$log.difference[-1]): p-value smaller than
## printed p-value
```

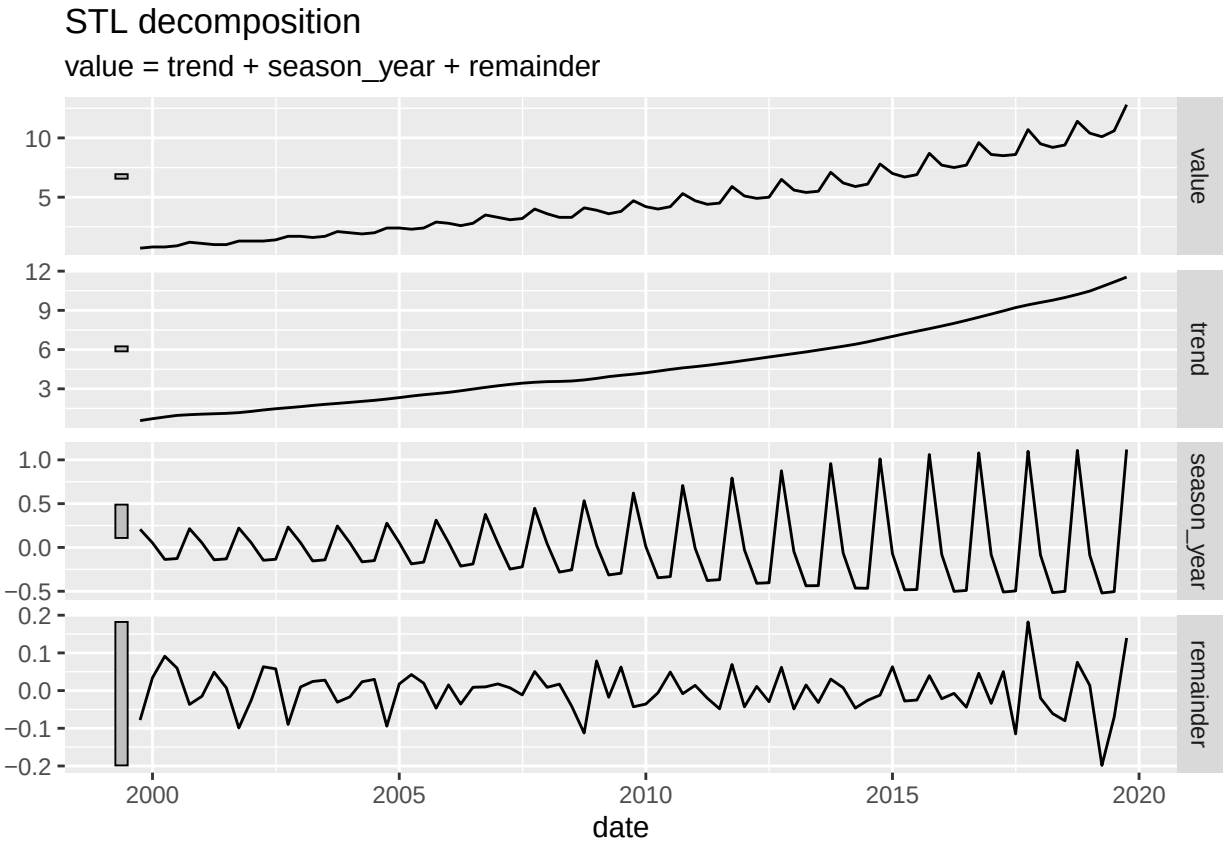
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ecom.explore$log.difference[-1]
## Dickey-Fuller = -4.6711, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Since the p-value $< 0.01$, we reject H0 that the series is not stationary, and conclude that the series is mean stationary, consistent with manual observation. Therefore we would want to use this series to fit an ARIMA model.

Finally, we perform a decomposition to see the trend and autocorrelation plots for both the original data, and BoxCox transformed data.

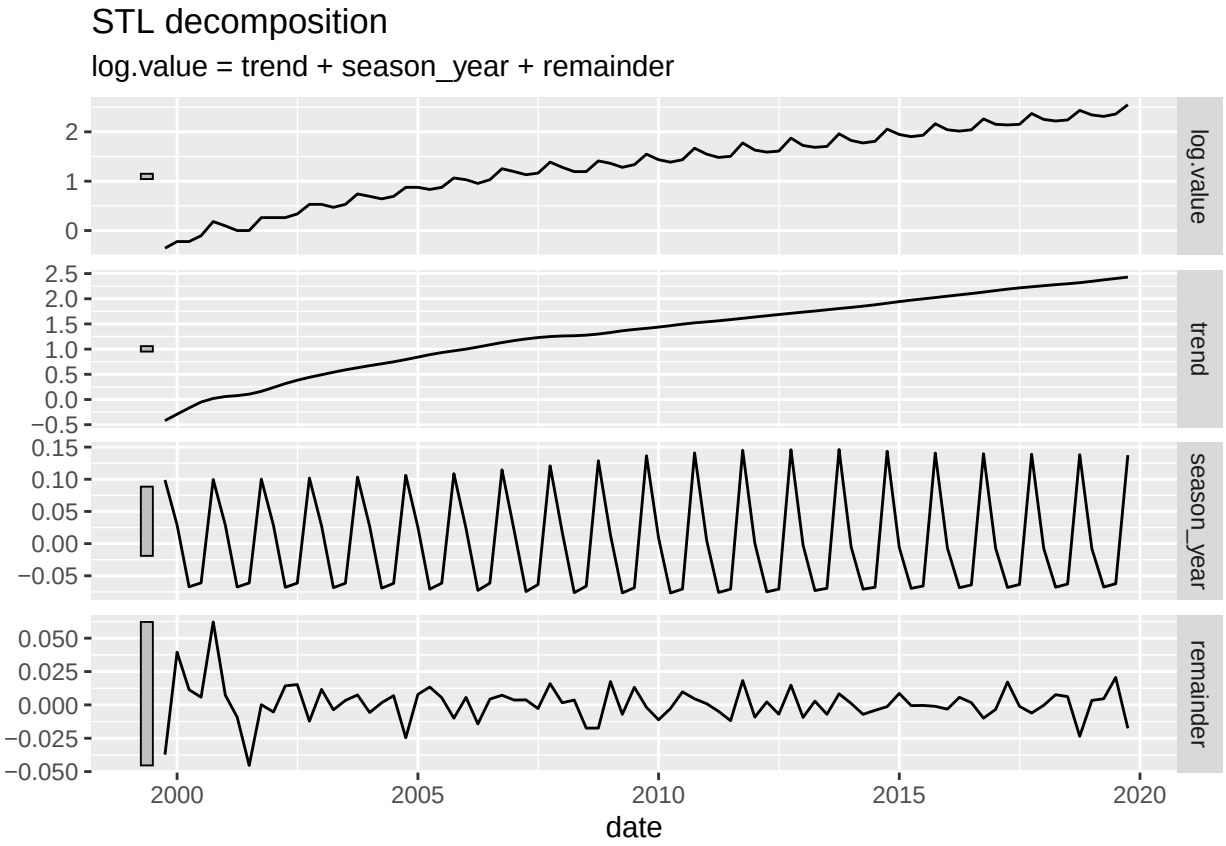
Original data:

```
stl <- ecom %>% model(STL(value)) %>% components()
autoplot(stl)
```




Log data:

```
stl.bc <- ecom.explore %>% model(STL(log.value)) %>% components()
autoplot(stl.bc)
```




The original data shows a somewhat quadratic trend, with seasonal variations that increase in amplitude. The log transformed data shows a flattened trend with decreasing slope, and seasonal variation that is somewhat constant in amplitude. Both remainder series look like random noise, with no clear autocorrelations.

*Step 2 - Stability variance*

From the EDA above, we saw that the Log transformation generated a series which is amendable to first order non-seasonal differencing in order to remove the trend. In addition, the original data after differencing (both seasonal and non-seasonal) produced a stationary series in both variance and mean. As a result, we choose to stabilize the variance of the original series with a log transformation, and use non-seasonal differencing to generate a stationary series for fitting an ARIMA model. We will also fit an ARIMA model with the seasonally and non-seasonally differenced original series, and choose the best model based on forecast performance.

*Step 3 - Differencing*

This step was explored thoroughly in the EDA.

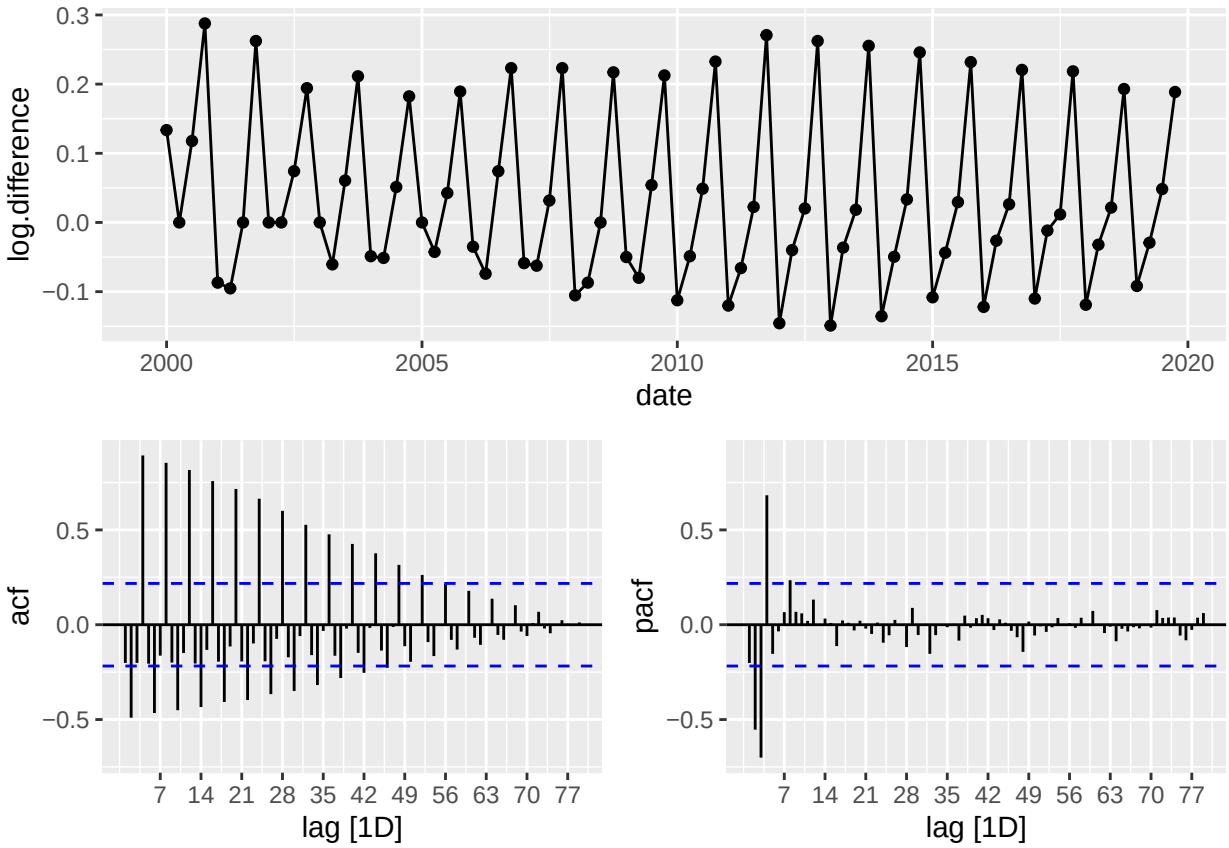
*Step 4 - Examine ACF/PACF*

For log transformed data: We will generate the ACF and PACF plots for the selected differenced series below:

```
gg_tsdisplay(ecom.explore, log.difference, plot_type = "partial",
    lag_max = 120)
```

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

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




The ACF oscillating and slowly decaying values, while the PACF cuts off at a lag of 4. This indicates that we might have a pure AR model with order 4.

First we will see what auto ARIMA gives us:

```
arima.auto.log <- ecom %>% model(fable::ARIMA(log(value)))
arima.auto.log %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,0)(2,1,0)[4]
## Transformation: log(.x)
##
## Coefficients:
##          sar1     sar2
##       -0.8522  -0.3223
## s.e.   0.1380   0.1563
##
## sigma^2 estimated as 0.001235:  log likelihood=146.13
## AIC=-286.25   AICc=-285.92   BIC=-279.26
```

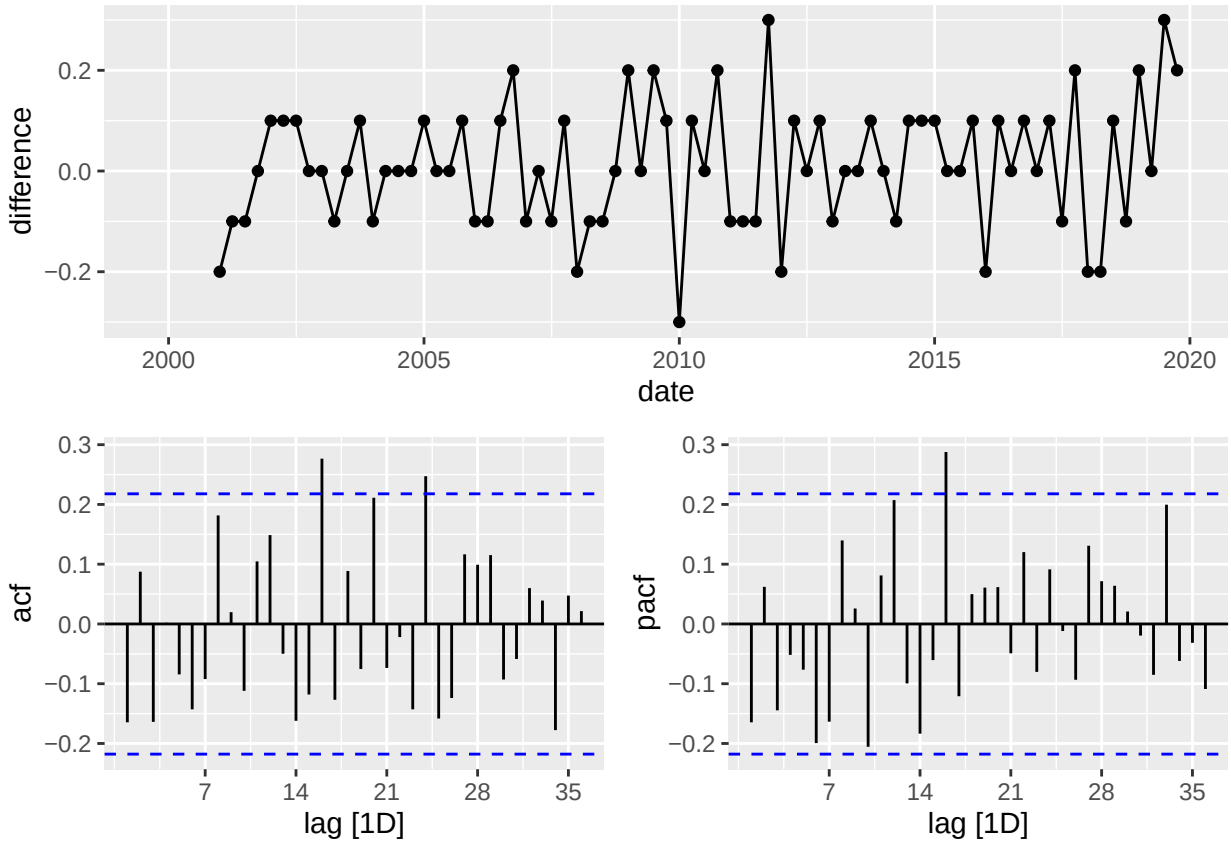
Auto arima finds that this model is AR with 2 seasonal terms, and differencing both seasonal and non-seasonal, with an AICc of -285.92.

For original series: We will generate the ACF and PACF plots for the selected differenced series below:

```
gg_tsdisplay(ecom.explore, difference, plot_type = "partial",
    lag_max = 36)
```

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

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




The ACF and PACF plots suggests that the series looks almost like white noise. There are significant ACF and PACF values above 14 lags, indicating that a mixture of MA and AR terms might be needed to capture this correlation. Since it is not clear whether to use pure MA or AR models, we will scan a range of models.

First we will see what auto ARIMA gives us:

```
arima.auto <- ecom %>% model(fable::ARIMA(value))
arima.auto %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,0)(0,1,0)[4]
##
## sigma^2 estimated as 0.01539:  log likelihood=50.76
## AIC=-99.52   AICc=-99.47   BIC=-97.19
```

Auto arima finds that this model is simply white noise, with an AICc of -99.47.

*Part 5 - Model selection*

We will scan a range of possible models using a seasonal and non-seasonal differencing of 1. To

select the best model, we will use AICc, which is a measure of in-sample fit, penalized by a function of the number of parameters and sample size. We could also use BIC, but this tends to penalize the number of parameters more so than AICc. We will start with AICc, and if the number of lag terms found is too large, which will try again with BIC.

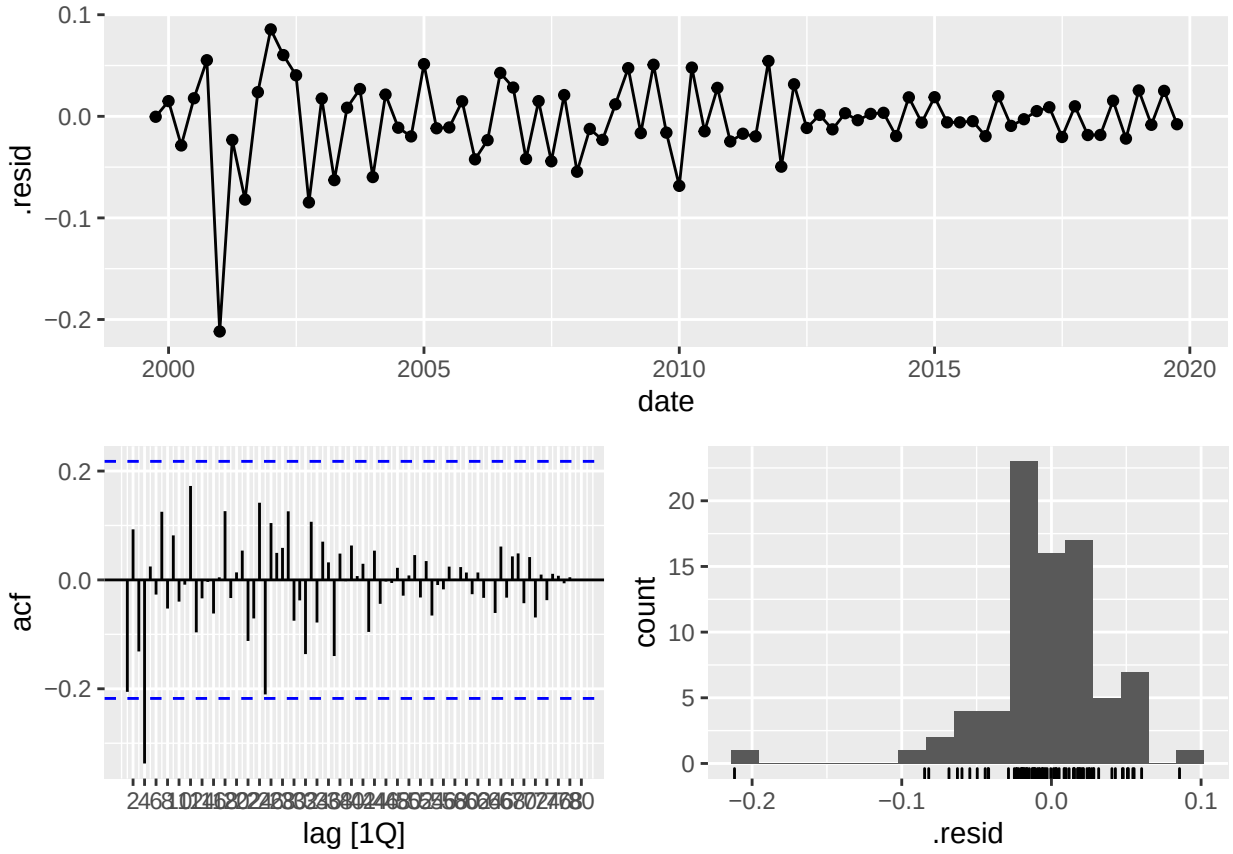
For the log series, since auto arima gave us 1 for seasonal differencing, we will scan with and without seasonal differencing:

```
results <- data.frame(p = integer(), q = integer(), P = integer(), Q = integer(), AICc = double

fit_aicc <- function(p,q,P,Q) {

  mod.fit <- ecom %>% model(fable::ARIMA(log(value) ~ pdq(p,1,q) + PDQ(P,1,Q, period = 4)))

  out <- tryCatch(
    {
      #If AICc cannot be found, then the model failed to converge
      AICc <- glance(mod.fit)$AICc
    },

    error = function(e) {
      return(NA)
    },

    warning = function(w) {
      return(NA)
    }
  )

  if (!is.na(out)){
      return(data.frame(cbind(p=p,q=q,P=P,Q=Q,AICc=AICc)))
  }
  else {
      return(NA)
  }
}

run = FALSE
if (run) {
  for (p in seq(0,4)) {
    for (q in seq(0,4)) {
      for (P in seq(0,4)) {
        for (Q in seq(0,4)) {
            answer = fit_aicc(p,q,P,Q)

            if (!is.na(answer)){
              if (dim(answer)[1] == 1) {
                  results <- rbind(results, answer)
```

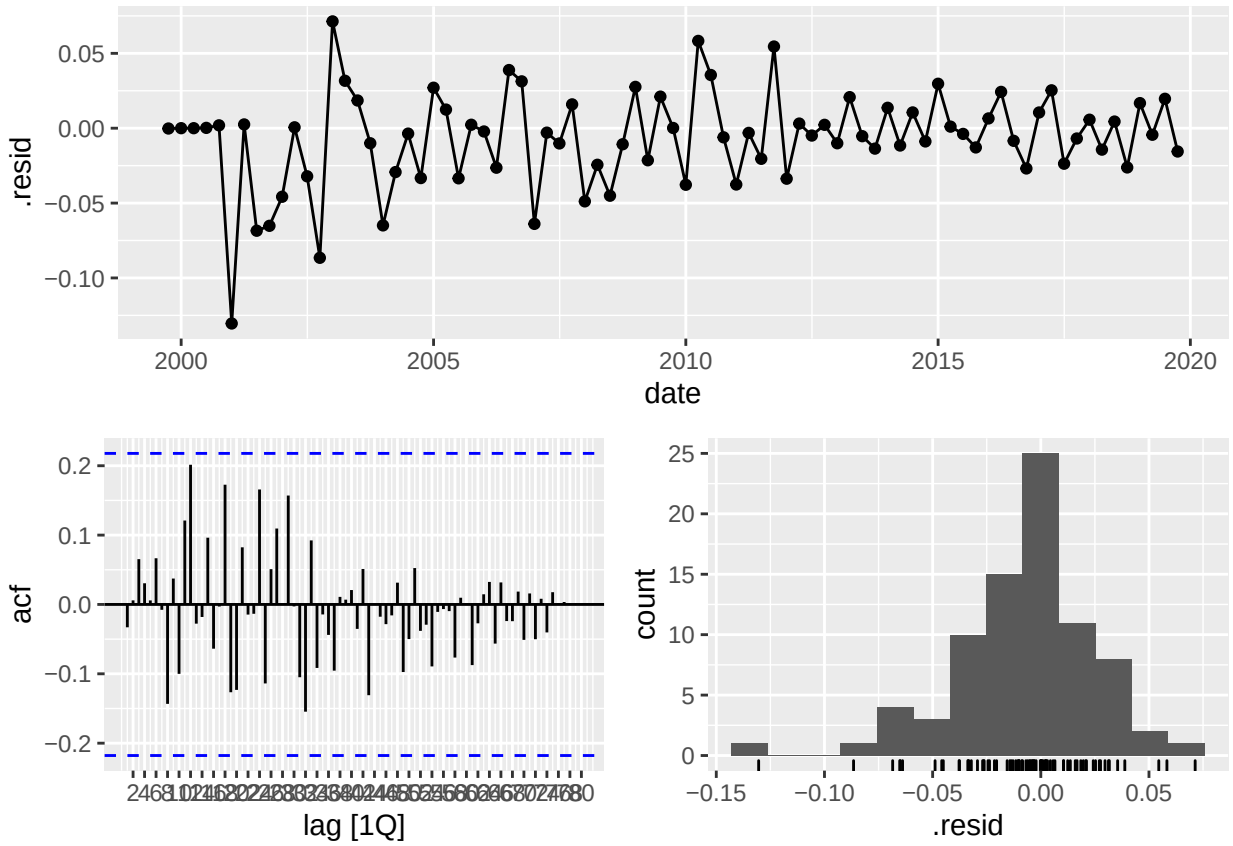
```
            }
          }
        }
      }
    }
  }
  results[which.min(results$AICc), ]
}

#RESULTS from the scan WITHOUT seasonal differencing
#Will not run each time due to time cost
#> results[order(results$AICc), ] %>% head()
#   p q P Q     AICc
#19 1 0 1 0 -268.3851
#8  0 1 1 0 -267.6495
#32 2 0 1 0 -266.8950
#23 1 1 1 0 -266.3759
#4  0 0 1 0 -264.6643
#42 4 1 0 0 -261.5758

#RESULTS from the scan WITH seasonal differencing
#Will not run each time due to time cost
#> results[order(results$AICc), ] %>% head()
#p q P Q AICc
#4 0 0 0 -286.9649
#4 0 1 0 -286.8374
#4 0 0 1 -286.7034
#0 0 0 2 -286.1582
#0 3 3 0 -286.1222
```

We will check the residuals for both models (for the conditions found to be best for both series):

```
mod.best.log.nonseasonal <- ecom %>% model(fable::ARIMA(log(value) ~
    pdq(1, 1, 0) + PDQ(1, 0, 0, period = 4)))

mod.best.log.seasonal <- ecom %>% model(fable::ARIMA(log(value) ~
    pdq(4, 1, 0) + PDQ(0, 1, 0, period = 4)))

gg_tsresiduals(mod.best.log.nonseasonal, lag_max = 360)
```




```
gg_tsresiduals(mod.best.log.seasonal, lag_max = 360)
```




The residual series for both models look like white noise, and we see that even up to lag of 360, there is at most a single significant autocorrelation. The distribution is skewed, but it is not of too much concern since there is no autocorrelation. To test whether the series is white noise, we use the Ljung-Box test:

```
mod.best.log.nonseasonal %>% augment() %>% features(.resid, ljung_box,
    lag = 52)
```

```
## # A tibble: 1 x 3
##   .model                                                lb_stat lb_pvalue
##   <chr>                                                   <dbl>     <dbl>
## 1 fable::ARIMA(log(value) ~ pdq(1, 1, 0) + PDQ(1, 0, 0, ~    53.1     0.431
```

```
mod.best.log.seasonal %>% augment() %>% features(.resid, ljung_box,
    lag = 52)
```

```
## # A tibble: 1 x 3
##   .model                                                lb_stat lb_pvalue
##   <chr>                                                   <dbl>     <dbl>
## 1 fable::ARIMA(log(value) ~ pdq(4, 1, 0) + PDQ(0, 1, 0, ~    44.5     0.759
```

Since the p-value >> 0.05 in both cases, we conclude that we do not reject H0 that the residual series is white noise.

To make a prediction, we will produce a pseudo out-of-sample fit using the last year as our test data. This is because ultimately, we're interested in the model's ability to produce 1-year ahead (4 quarter) forecasts:

```
dat.train <- ecom %>% filter(year(date) <= 2018)
dat.test <- ecom %>% filter(year(date) > 2018)

tail(dat.train)
```

```
## # A tsibble: 6 x 2 [1D]
##      date value
##     <qtr> <dbl>
## 1 2017 Q3   8.6
## 2 2017 Q4  10.7
## 3 2018 Q1   9.5
## 4 2018 Q2   9.2
## 5 2018 Q3   9.4
## 6 2018 Q4  11.4
```

```
head(dat.test)
```

```
## # A tsibble: 4 x 2 [1D]
##      date value
##     <qtr> <dbl>
## 1 2019 Q1  10.4
## 2 2019 Q2  10.1
## 3 2019 Q3  10.6
## 4 2019 Q4  12.8
```

```
# Fit the model
mod.train.test.log.nonseasonal <- dat.train %>% model(fable::ARIMA(log(value) ~
    pdq(1, 1, 0) + PDQ(1, 0, 0, period = 4)))

fabletools::accuracy(mod.train.test.log.nonseasonal %>% fabletools::forecast(h = 4),
    dat.test)
```

```
## # A tibble: 1 x 9
##   .model                    .type    ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>                     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 fable::ARIMA(log(value) ~ Test  0.399 0.426 0.399  3.56  3.56   NaN 0.281
```

```
mod.train.test.log.seasonal <- dat.train %>% model(fable::ARIMA(log(value) ~
    pdq(4, 1, 0) + PDQ(0, 1, 0)))
fabletools::accuracy(mod.train.test.log.seasonal %>% fabletools::forecast(h = 4),
    dat.test)
```

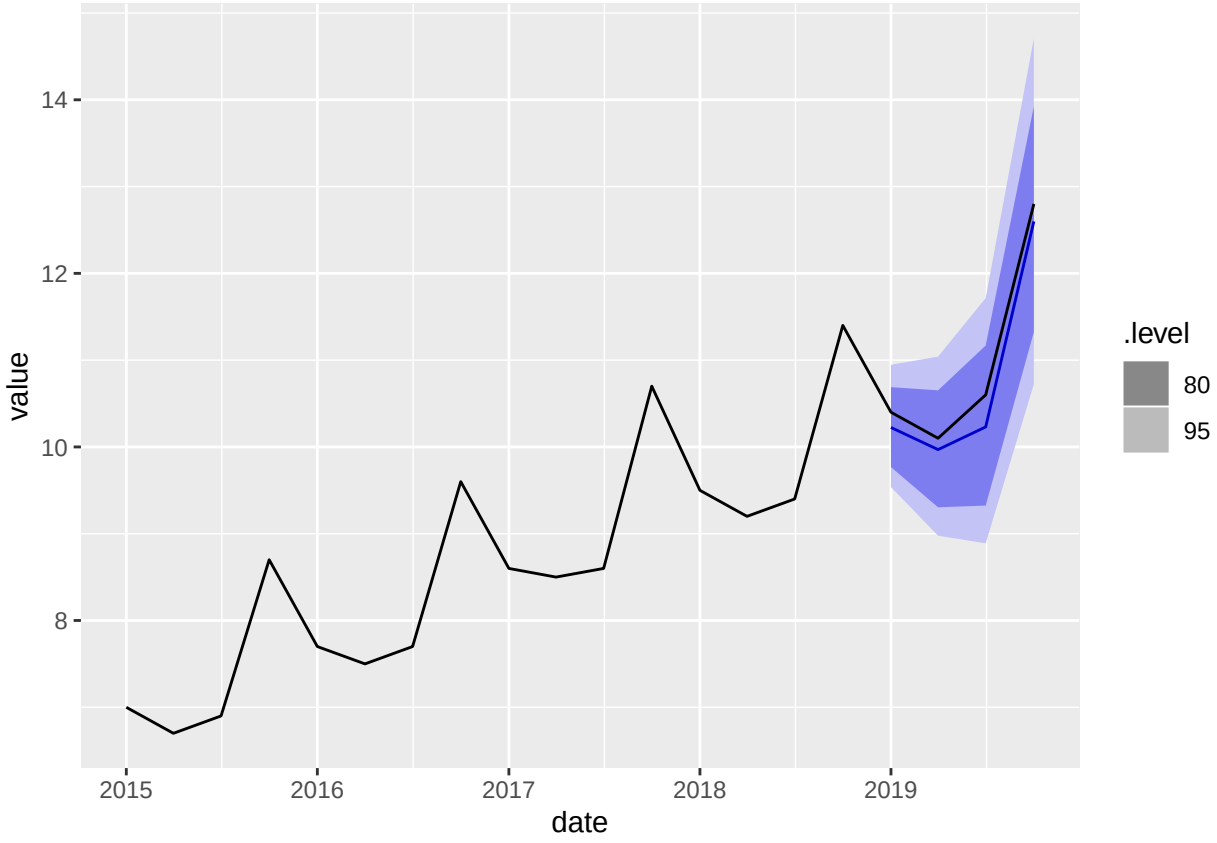
```
## # A tibble: 1 x 9
##   .model                   .type    ME  RMSE   MAE   MPE  MAPE  MASE   ACF1
##   <chr>                    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 fable::ARIMA(log(value)~ Test  0.219 0.237 0.219  2.01  2.01   NaN -0.372
```

To decide between models, we will use the MAE, which is a standard metric defined below:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}|$$

In this case, the errors are all much lower for the seasonal model, including MAE, which is not terrible given at the actual value at the beginning of 2018 was around 10. This is the model we will use going forward for the log transformed series. We generate a plot of the fit (starting from 2015 onward in order to zoom in on the fitted portion):

```
mod.train.test.log.seasonal %>% fabletools::forecast(h = 4) %>%
    autoplot(ecom %>% filter(year(date) >= 2015))
```




Based on the plot, the pseudo-out-of-sample fit looks reasonable. It captures the quarterly fluctuations at all time periods in terms of direction of change, although the model tends to slightly underestimate the actual value for all quarters of the following year. All of the 4 values however fall within the 95% CI of the forecast.

We now look at cross-validation in the same way as problem 3. However, rather than using one-step ahead forecasts, we will measure the overall MAE for 4 quarters ahead forecasts (same as pseudo-sample):

```
ecom_tr <- ecom %>%
  slice(1:(n()-4)) %>% #keeps all but last 4 rows (for 4 quarter ahead forecasts)
  stretch_tsibble(.init = 3, .step = 1) #.init says for the first window, start at 3. Then gen

fc.log <- ecom_tr %>%
```

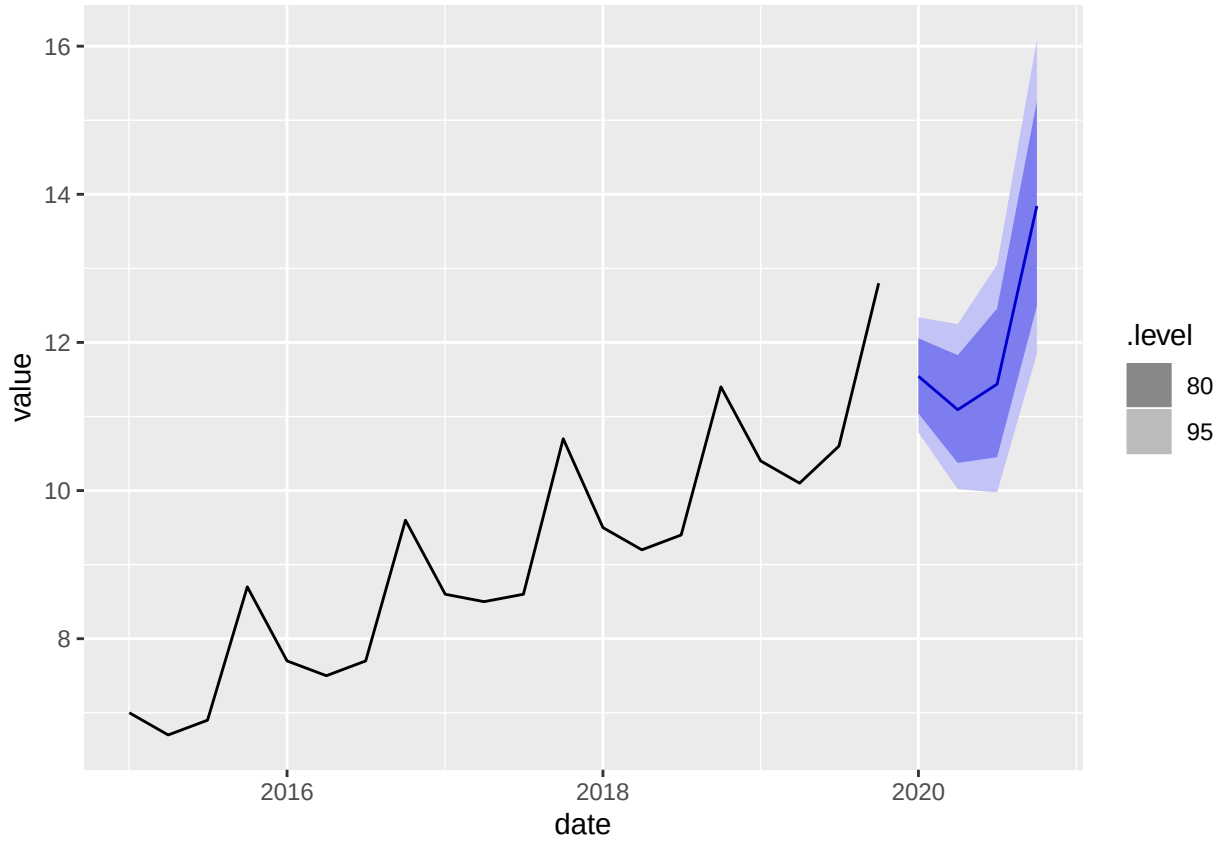
```
  model(fable::ARIMA(log(value) ~ pdq(4,1,0) + PDQ(0,1,0, period = 4))) %>%
  fabletools::forecast(h=4)

fc.log %>% fabletools::accuracy(ecom)
```

```
## # A tibble: 1 x 9
##   .model                    .type      ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>                     <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 fable::ARIMA(log(value~ Test  -0.0538 0.193 0.145 -1.66  3.58 0.292 0.627
```

The cross-validation MAE is smaller than the pseudo-out-of-sample error, which is a good sign that the model is working properly. This is because it is able to accurately forecast 4 quarters ahead for all slices of the data, starting from just 3 observations, up to n-4 observations.

We will now generate a forecast for the next year into the future. We will plot this forecast, along with the data from 2015 onward:

```
mod.best.log.seasonal %>% fabletools::forecast(h = 4) %>% autoplot(ecom %>%
    filter(year(date) >= 2015))
```




The forecast looks very reasonable. It captures the same seasonality as the previous few years, with an increasing trend that looks to be on the same scale as the previous years. The confidence intervals are relatively narrow, suggesting that the model is confident in its predictions.

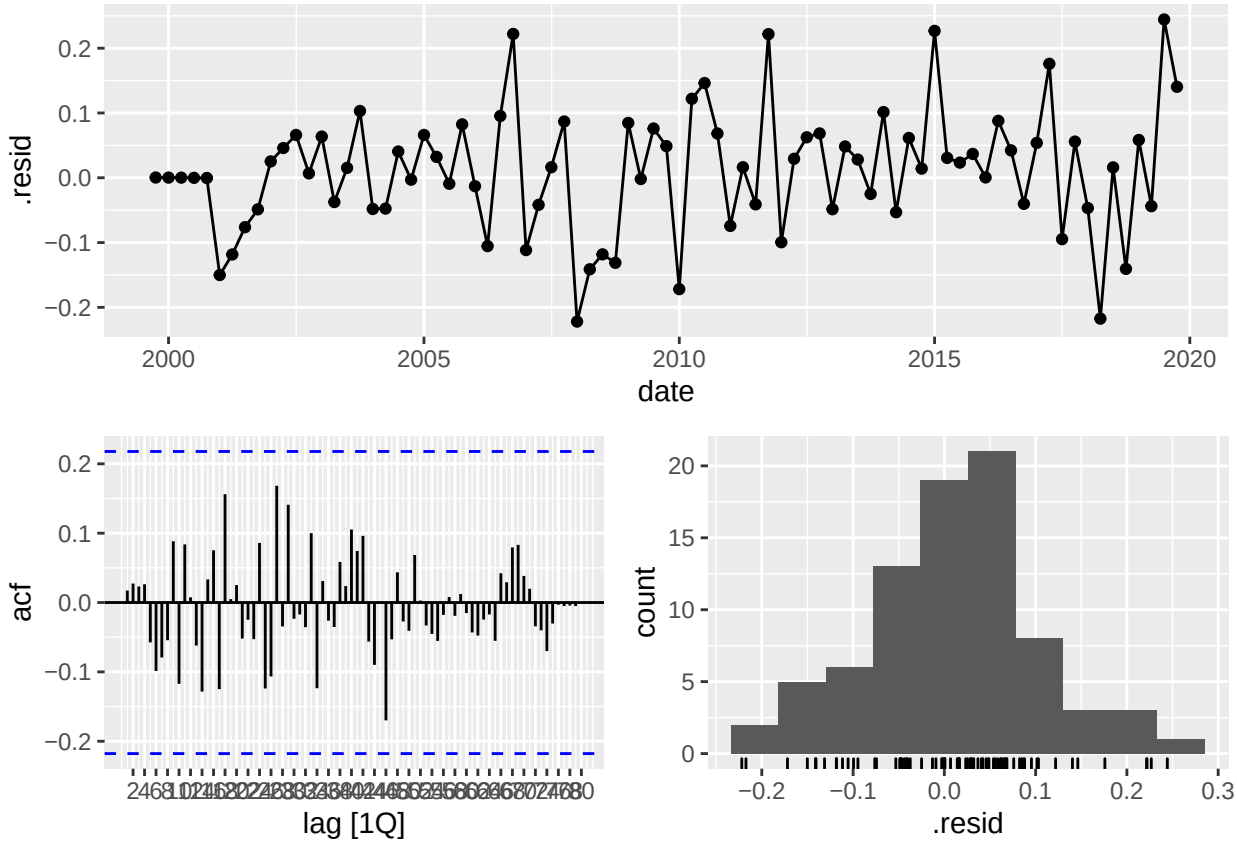
We now perform the same analysis for the seasonally and non-seasonally differenced original series, which we showed was stable in mean and variance:

```
results <- data.frame(p = integer(), q = integer(), P = integer(), Q = integer(), AICc = double

fit_aicc <- function(p,q,P,Q) {

  mod.fit <- ecom %>% model(fable::ARIMA(value ~ pdq(p,1,q) + PDQ(P,1,Q, period = 4)))

  out <- tryCatch(
    {
      #If AICc cannot be found, then the model failed to converge
      AICc <- glance(mod.fit)$AICc
    },

    error = function(e) {
      return(NA)
    },

    warning = function(w) {
      return(NA)
    }
  )

  if (!is.na(out)){
      return(data.frame(cbind(p=p,q=q,P=P,Q=Q,AICc=AICc)))
  }
  else {
      return(NA)
  }
}

run = FALSE
if (run) {
  for (p in seq(0,4)) {
    for (q in seq(0,4)) {
      for (P in seq(0,4)) {
        for (Q in seq(0,4)) {
            answer = fit_aicc(p,q,P,Q)

            if (!is.na(answer)){
              if (dim(answer)[1] == 1) {
                  results <- rbind(results, answer)
              }
            }
        }
      }
    }
  }
  results[which.min(results$AICc), ]
```

```
}

#RESULTS from the scan
#Will not run each time due to time cost
#> results[order(results$AICc), ] %>% head()
#    p q P Q     AICc
#56  0 3 1 2 -116.1103
#157 3 0 1 2 -114.4123
#73  1 0 1 2 -113.9098
#58  0 3 2 1 -113.7832
#126 2 0 1 2 -113.2885
#96  1 1 2 2 -112.7190
```

We see that by scanning the series, we get models with much better AICc than discovered by auto ARIMA. The best AICc we found is represented below, and the residuals plot shown:

```
mod.best <- ecom %>% model(fable::ARIMA(value ~ pdq(0, 1, 3) +
    PDQ(1, 1, 2, period = 4)))

gg_tsresiduals(mod.best, lag_max = 360)
```




The residual series looks like white noise, and we see that even up to lag of 360, there is not a single significant autocorrelation. The distribution is left skewed, but it is not of too much concern since there is no autocorrelation. To test whether the series is white noise, we use the Ljung-Box test:

```
mod.best %>% augment() %>% features(.resid, ljung_box, lag = 52)
```

```
## # A tibble: 1 x 3
##   .model                                                lb_stat lb_pvalue
##   <chr>                                                   <dbl>     <dbl>
## 1 fable::ARIMA(value ~ pdq(0, 1, 3) + PDQ(1, 1, 2, perio~    42.4     0.827
```

Since the p-value is >> 0.05, we conclude that we do not reject H0 that the residual series is white noise.

To make a prediction, we will produce a pseudo out-of-sample fit using the last year as our test data. This is because ultimately, we're interested in the model's ability to produce 1-year ahead (4 quarter) forecasts:

```
dat.train <- ecom %>% filter(year(date) <= 2018)
dat.test <- ecom %>% filter(year(date) > 2018)

tail(dat.train)
```

```
## # A tsibble: 6 x 2 [1D]
##      date value
##     <qtr> <dbl>
## 1 2017 Q3   8.6
## 2 2017 Q4  10.7
## 3 2018 Q1   9.5
## 4 2018 Q2   9.2
## 5 2018 Q3   9.4
## 6 2018 Q4  11.4
```

```
head(dat.test)
```

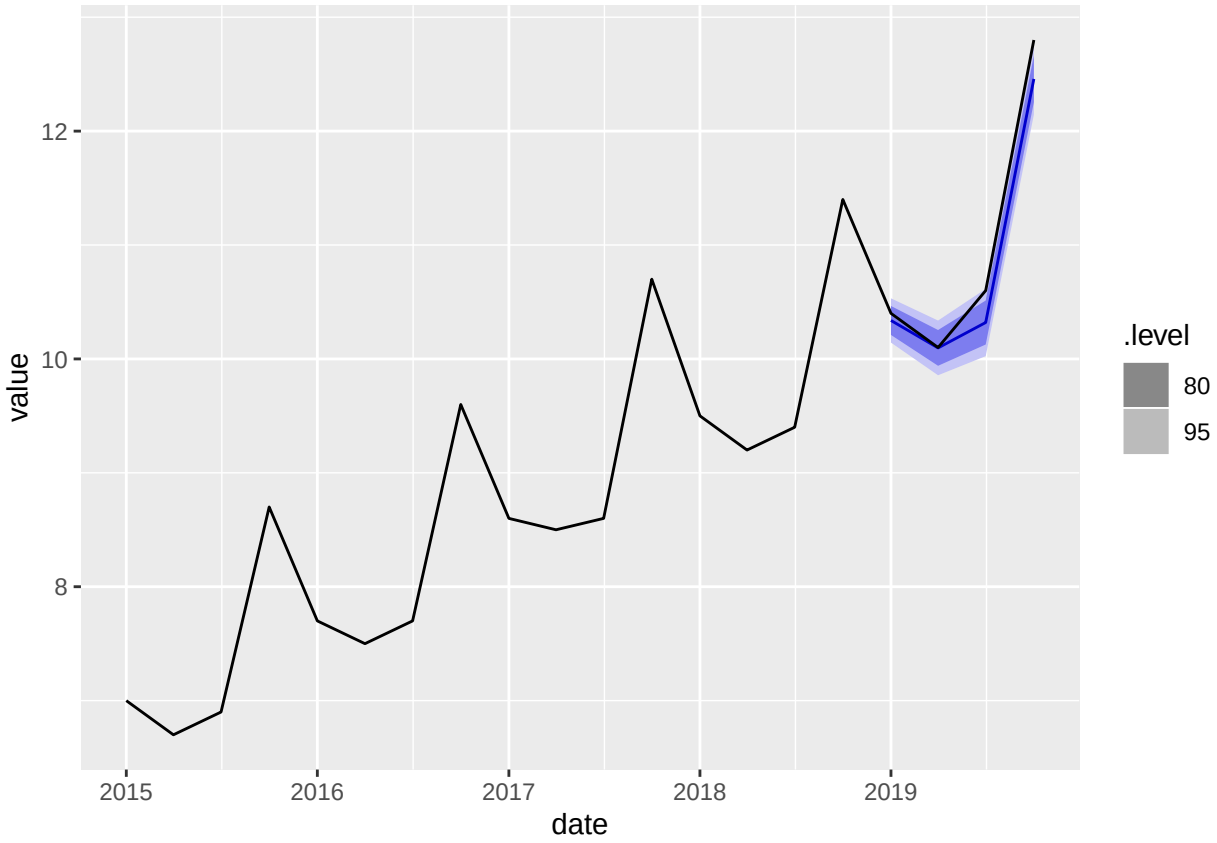
```
## # A tsibble: 4 x 2 [1D]
##      date value
##     <qtr> <dbl>
## 1 2019 Q1  10.4
## 2 2019 Q2  10.1
## 3 2019 Q3  10.6
## 4 2019 Q4  12.8
```

```
# Fit the model
mod.train.test <- dat.train %>% model(fable::ARIMA(value ~ pdq(0,
    1, 3) + PDQ(1, 1, 2, period = 4)))

fabletools::accuracy(mod.train.test %>% fabletools::forecast(h = 4),
    dat.test)
```

```
## # A tibble: 1 x 9
##   .model                    .type    ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>                     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 fable::ARIMA(value ~ pdq~ Test  0.172 0.223 0.172  1.49  1.49   NaN 0.230
```

The MAE is around 0.17, which is not terrible given at the actual value at the beginning of 2018 was around 10. We generate a plot of the fit (starting from 2015 onward in order to zoom in on the fitted portion):

```
mod.train.test %>% fabletools::forecast(h = 4) %>% autoplot(ecom %>%
    filter(year(date) >= 2015))
```




Based on the plot, the pseudo-out-of-sample fit looks very reasonable. It captures the quarterly fluctuations at all time periods in terms of direction of change, although the model tends to underestimates the size of the increase observed in the 3rd quarter of 2019. All of the 4 values however roughly fall within the 95% CI of the forecast.

We now look at cross-validation in the same way as problem 3. However, rather than using one-step ahead forecasts, we will measure the overall MAE for 4 quarters ahead forecasts (same as pseudo-sample):

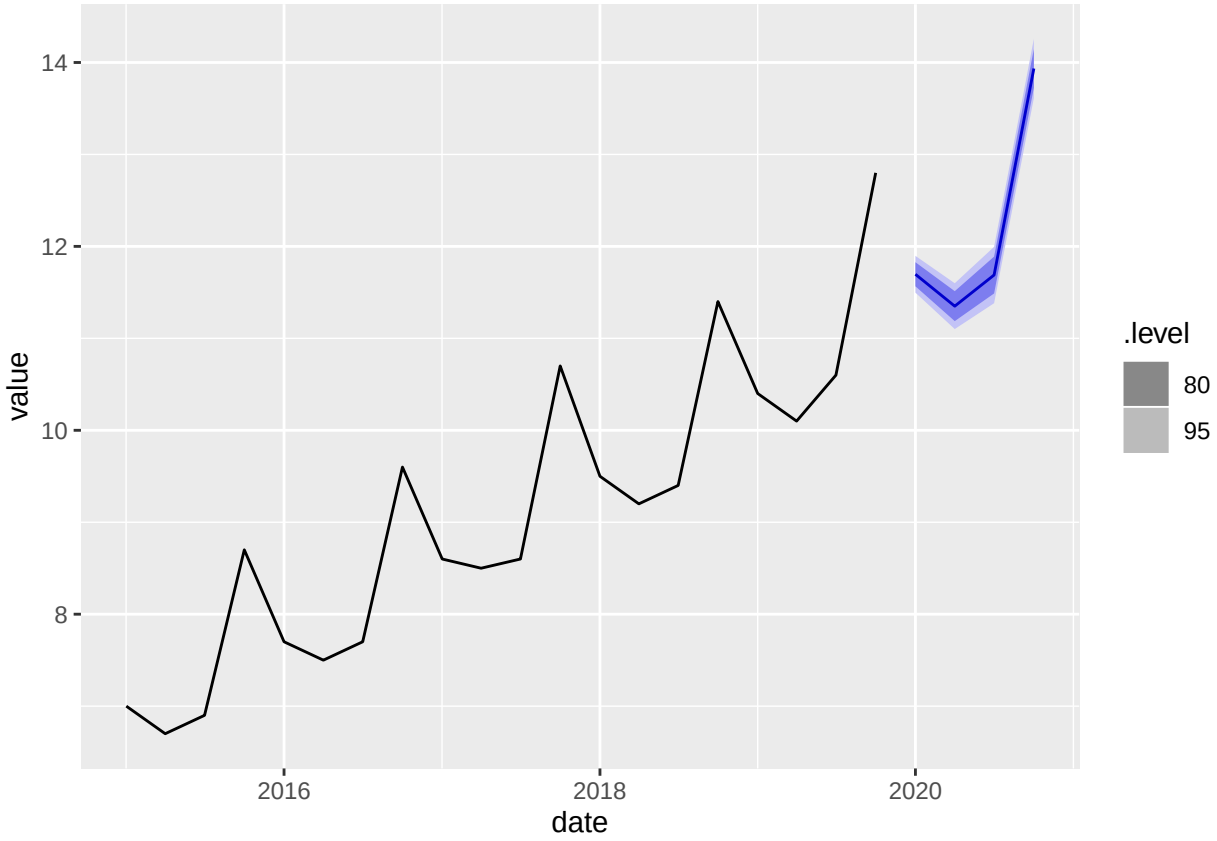
```
ecom_tr <- ecom %>%
  slice(1:(n()-4)) %>% #keeps all but last 4 rows (for 4 quarter ahead forecasts)
  stretch_tsibble(.init = 3, .step = 1) #.init says for the first window, start at 3. Then gen

fc <- ecom_tr %>%
  model(fable::ARIMA(value ~ pdq(0,1,3) + PDQ(1,1,2, period = 4))) %>%
  fabletools::forecast(h=4)

fc %>% fabletools::accuracy(ecom)
```

```
## # A tibble: 1 x 9
##   .model                    .type     ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>                     <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 fable::ARIMA(value ~ pd~ Test  0.0427 0.163 0.131 0.609  2.51 0.263 0.572
```

The cross-validation MAE is lower than the pseudo-out-of-sample error, which is a good sign that the model is working properly. This is because it is able to accurately forecast 4 quarters ahead for all slices of the data, starting from just 3 observations, up to n-4 observations.

We will now generate a forecast for the next year into the future. We will plot this forecast, along with the data from 2015 onward:

```
mod.best %>% fabletools::forecast(h = 4) %>% autoplot(ecom %>%
    filter(year(date) >= 2015))
```




The forecast looks very reasonable. The confidence intervals are even narrower than the log transformed model, suggesting that the model is confident in its predictions.

The summary for the errors of each model are below:

```
results <- data.frame(original_series = c(0.1719828, 0.1306777),
    log_transformed = c(0.2193847, 0.1452856))
rownames(results) <- c("MAE pseudo-out-of-sample-12mon", "MAE CV-12mon-ahead")
results
```

```
##                                original_series log_transformed
## MAE pseudo-out-of-sample-12mon       0.1719828       0.2193847
```

```
## MAE CV-12mon-ahead                  0.1306777       0.1452856
```

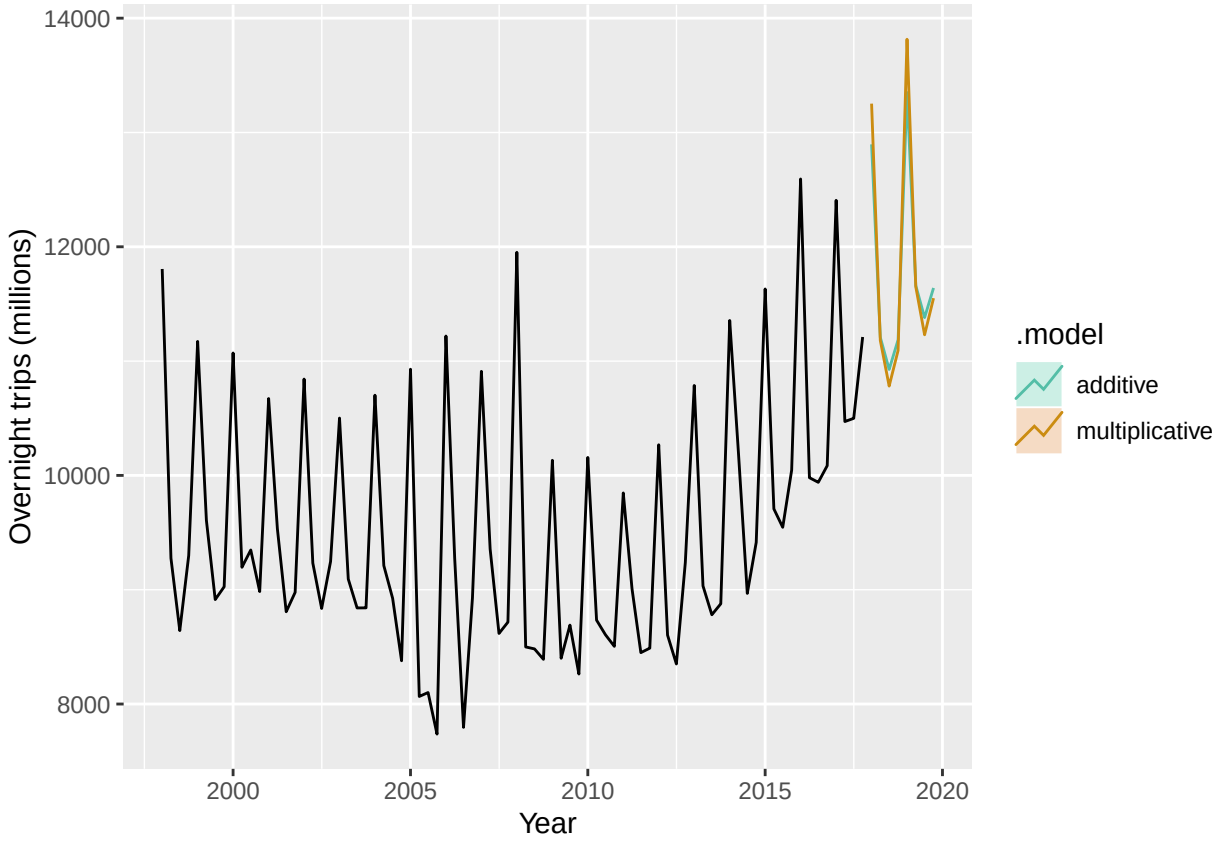
Overall, since the MAE tends to be lower for modeling on the original series with seasonal differencing, we will use this model for forecasting. The CIs are also narrower for this model, indicating that the model has higher confidence than the log transformed model.

### Holt-Winters and Model averaging

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

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

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

# using ETS() function from fable
fit <- aus_holidays %>% model(additive = ETS(Trips ~ error("A") +
    trend("A") + season("A")), multiplicative = ETS(Trips ~ error("M") +
    trend("A") + season("M")))
fc <- fit %>% forecast(h = 8)

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

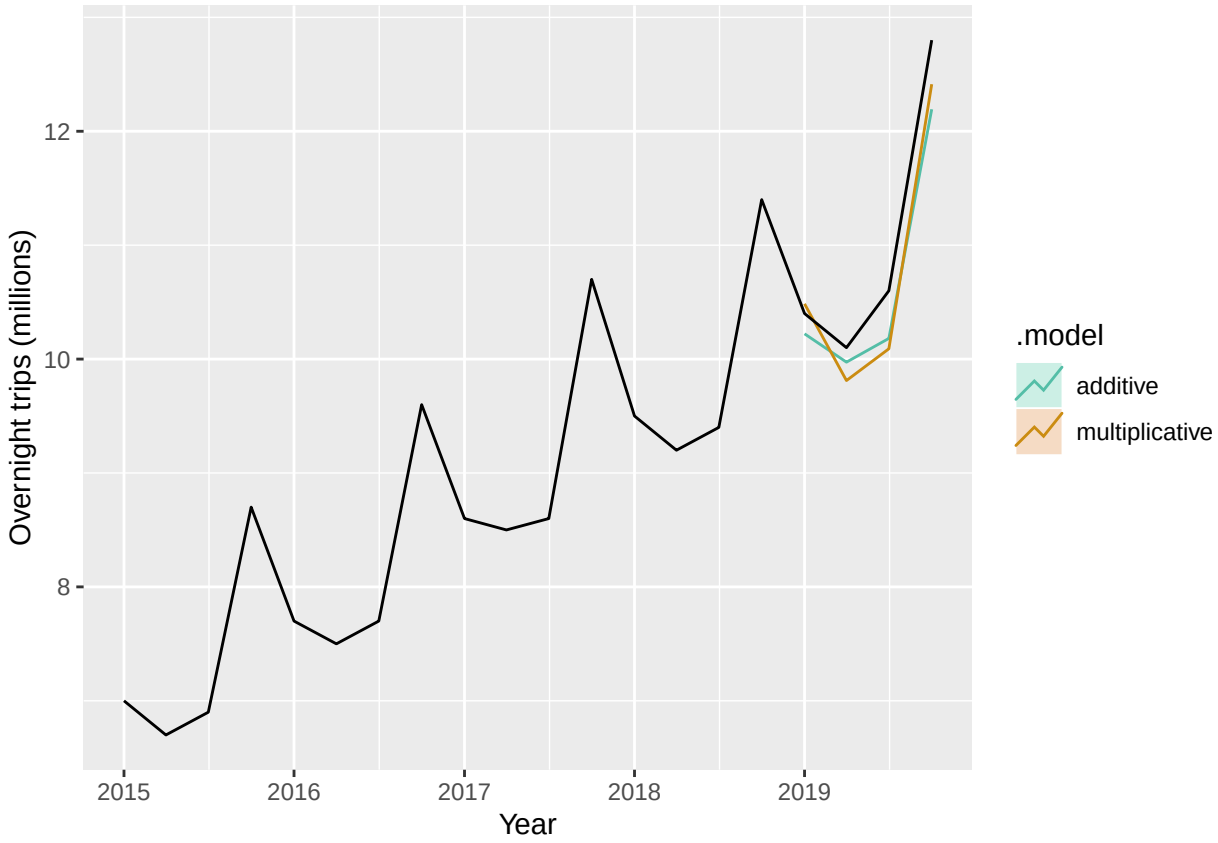



We apply a Holt-Winters model to the ecom time series from our Seasonal ARIMA model, and compare its forecasting performance to that model. Then we compare both to the performance of a

simple average of the ARIMA and Holt-Winters models.

We will compare forecasting performance for the following year (12month ahead).

```
# using ETS() function from fable dat.train is the previous
# set of data going up to 2018
fit <- dat.train %>% model(additive = ETS(value ~ error("A") +
    trend("A") + season("A")), multiplicative = ETS(value ~ error("M") +
    trend("A") + season("M")))
fc <- fit %>% fabletools::forecast(h = "1 years")

fc %>% autoplot(ecom %>% filter(year(date) >= 2015), level = NULL) +
    xlab("Year") + ylab("Overnight trips (millions)") + scale_color_brewer(type = "qual",
    palette = "Dark2")
```




Both the additive and multiplicative models appear worse than the ARIMA model for the next year forecast. We evaluate the accuracy of both models:

```
arima.forecast <- mod.train.test %>% fabletools::forecast(h = 4)
rbind(fabletools::accuracy(arima.forecast, ecom), fabletools::accuracy(fc,
    ecom))
```

```
## # A tibble: 3 x 9
##   .model                    .type    ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>                     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 fable::ARIMA(value ~ pdq~ Test  0.172 0.223 0.172  1.49  1.49 0.346 0.230
```

```
## 2 additive                  Test  0.333 0.385 0.333  2.92  2.92 0.669 0.253
## 3 multiplicative            Test  0.275 0.354 0.318  2.47  2.88 0.639 0.124
```

The error for all metrics including MAE for both Multiplicative and Additive models are higher than for the ARIMA model. ARIMA is clearly better at forecasting performance. In addition, the multiplicative model tends to be better than the additive for all metrics. We will do a simple average of both Multiplicative and Additive Holt Winters with ARIMA to see if we can improve the forecast accuracy of both models.

```
# Simple average of the two models
fc2 <- fc

# This will broadcast arima values and add to both additive
# and multiplicative models
avg.values <- (fc$value + arima.forecast$value)/2

fc2$value <- avg.values
fc2$.model <- c(replicate(4, "additive-arima-avg"), replicate(4,
    "multiplicative-arima-avg"))

rbind(fabletools::accuracy(arima.forecast, ecom), fabletools::accuracy(fc,
    ecom), fabletools::accuracy(fc2, ecom))
```

```
## # A tibble: 5 x 9
##   .model                    .type    ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>                     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 fable::ARIMA(value ~ pdq~ Test  0.172 0.223 0.172  1.49  1.49 0.346 0.230
## 2 additive                  Test  0.333 0.385 0.333  2.92  2.92 0.669 0.253
## 3 multiplicative            Test  0.275 0.354 0.318  2.47  2.88 0.639 0.124
## 4 additive-arima-avg        Test  0.252 0.302 0.252  2.20  2.20 0.507 0.253
## 5 multiplicative-arima-avg  Test  0.223 0.279 0.229  1.98  2.03 0.461 0.263
```

It looks like while averaging helped decrease the MAE on the Holt-Winters models, neither average were able to beat the ARIMA model for this particular time series. This is also true for all of the error metrics. Holt-Winter typically has less flexibility as ARIMA, which can take into account a specified and arbitrary number of autocorrelations. This model tends to underpredict our series. As a result, it appears that the ARIMA model, based on a one-year ahead forecast, is the best model to use.

*This work was done as part of the W271 - Statistical Methods for Discrete Response, Time Series, and Panel Data course under the U.C. Berkeley Master of Information and Data Science program.