W271 Lab 3

Tiffany Jaya, Joanna Huang, Robert Deng, Shan He

```r
# add packages
library(forecast)
library(knitr)
library(stats)
library(tseries)
library(xts)
# prevent source code from running off the page
opts_chunk$set(tidy.opts=list(width.cutoff=70), tidy=TRUE)
# remove all objects from current workspace
rm(list = ls())
# set seed number to reproduce results
set.seed(1)
# load data
raw.sales <- read.csv('./data/ECOMPCTNSA.csv', header=TRUE, sep=',')
```

Question 1: Forecasting using a SARIMA model

Since the emergence of the internet, more and more people are shopping at online retailers than brick-and-mortar stores. E-commerce is on the rise, and we would like to see what percentage of total retail sales e-commerce is accounted for in the fourth quarter of 2017. With data from the US Census Bureau ranging from 1999 to 2016, we were able to estimate it to be 10.20% using the seasonal autoregresive integrated moving average model (or SARIMA for short). While the number does not seem substantial compare to the perceived value of e-commerce, we have to remember that retail sales include motor vehicles, gas stations, and grocery stores where e-commerce has yet to play a major role in the field.

The SARIMA model that we use for the projected forecast is $\text{ARIMA}(0,1,0)(0,1,1)_4$.

Exploration Data Analysis

The first step once we obtained the dataset was to examine its structure.

```r
# convert raw data into a time-series object
sales <- ts(raw.sales$ECOMPCTNSA, start = c(1999, 4), frequency = 4)
# hold out test data for verification in forecast
sales.train <- ts(sales[time(sales) < 2015], start = c(1999, 4), frequency = 4)  # 1999-2014
sales.test <- ts(sales[time(sales) >= 2015], start = c(2015, 1), frequency = 4)  # 2015-2016
# examine the structure
kable(summary(raw.sales))
```

DATE	ECOMPCTNSA
1999-10-01: 1	Min. :0.700

DATE	ECOMPCTNSA
2000-01-01: 1	1st Qu.:2.000
2000-04-01: 1	Median :3.600
2000-07-01: 1	Mean :3.835
2000-10-01: 1	3rd Qu.:5.300
2001-01-01: 1	Max. :9.500
(Other) :63	NA

```
head(sales)
```

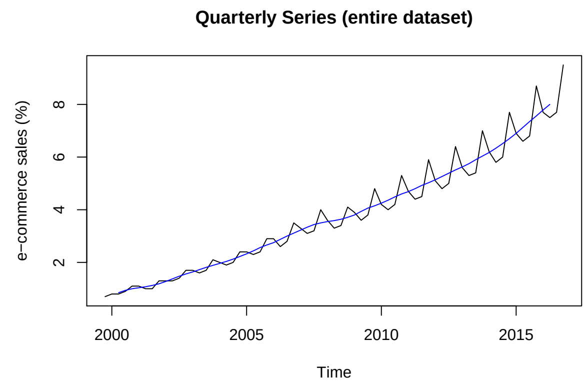
```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 1999                 0.7
## 2000  0.8  0.8  0.9  1.1
## 2001  1.1
```

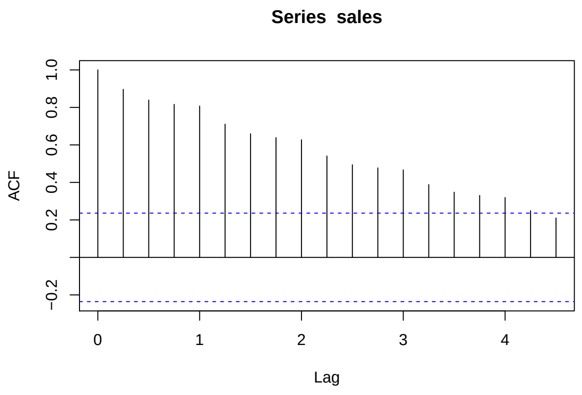
```
tail(sales)
```

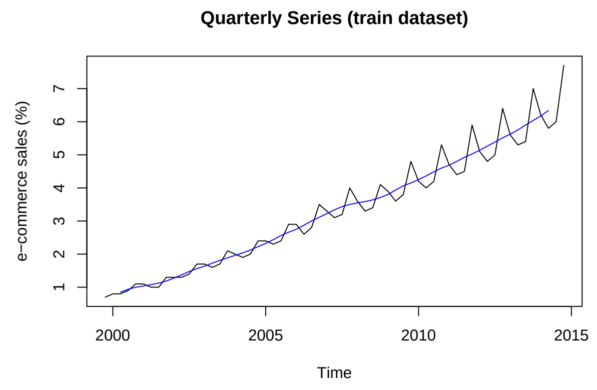
```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 2015            6.8  8.7
## 2016  7.7  7.5  7.7  9.5
```

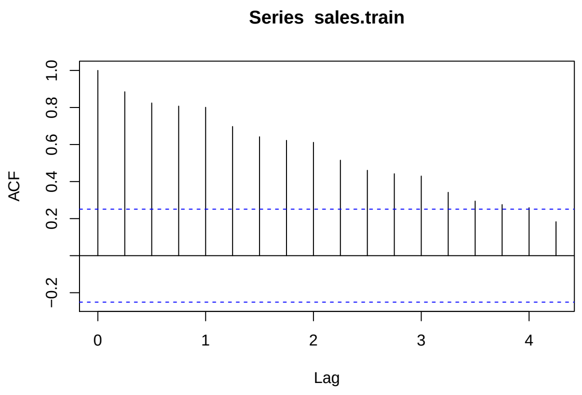
We were able to determine that there was no missing value among the 69 observations with sales appearing to increase overtime from 0.7% in the 4th quarter of 1999 to 9.5% in the 4th quarter of 2016. To confirm, we plot the time series as well as its associating MA(4) model. If the data expressed seasonality every quarter, the MA(4) model smooths out the variances and acts as an annual trend with the seasonal effects within each quarter removed.

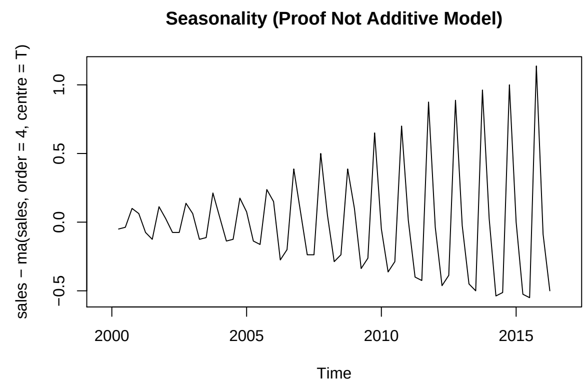
```
# using the entire dataset
plot(sales, ylab = "e-commerce sales (%)", main = "Quarterly Series (entire dataset)")
lines(ma(sales, order = 4, centre = T), col = "blue")
acf(sales)
# using the train dataset
plot(sales.train, ylab = "e-commerce sales (%)", main = "Quarterly Series (train dataset)")
lines(ma(sales.train, order = 4, centre = T), col = "blue")
acf(sales.train)
# remove trend to see seasonality
plot(sales - ma(sales, order = 4, centre = T), main = "Seasonality (Proof Not Additive Model)"
plot(sales/ma(sales, order = 4, centre = T), main = "Seasonality (Proof Is Multiplicative Mode
```

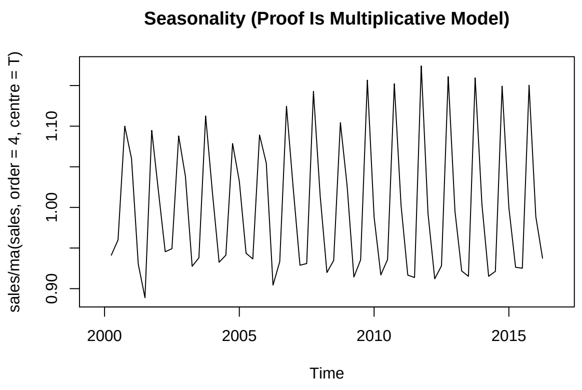












Given the upward trend and increasing variance, the series is a multiplicative model that is non-stationary with quarterly seasonality. The autocorrelation function further substantiates the series's non-stationary because of its slow decay and r_1s that are large and positive (r_1 indicates how successive values of y relate to each other). For this reason, we will perform two operations. One, we will difference the series to stabilize the mean. And two, we will apply a Box-Cox transformation (logarithm and power transformation) to stabilize the variance. We verify if differencing was necessary by running the unit root test.

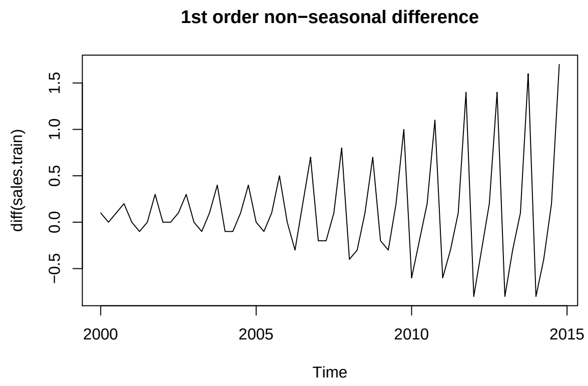
```r
# unit root test
cbind(adf.test(sales.train, alternative = "stationary")$p.value, kpss.test(sales.train)$p.valu
    ndiffs(sales.train), nsdiffs(sales.train))
```

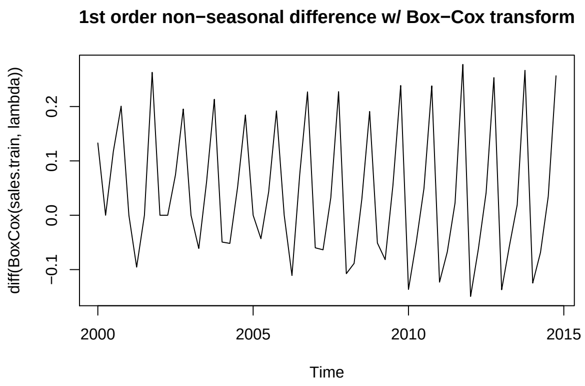
```
##      [,1] [,2] [,3] [,4]
## [1,] 0.99 0.01    1    1
```

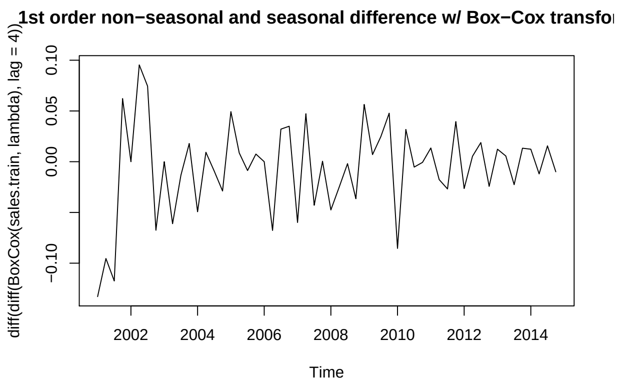
Large p-value in the Augmented Dickey-Fuller (ADF) test and small p-value in the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test confirm our intuition to difference the time series. As suggested by the non-seasonal (ndiffs) and seasonal (nsdiff) unit test, we will perform a non-seasonal difference to the data once and a seasonal difference once in order to make the series stationary in mean. The ADF and KPSS tests we run afterwards validate our differencing decision. We apply log and power transformation to the difference using the Box-Cox transformation by first finding the best lambda value that will give the optimal uniformity in the seasonal variation before administering the said transformation. With a lambda value of 0.01467236, the transformation is similar to a log transformation.

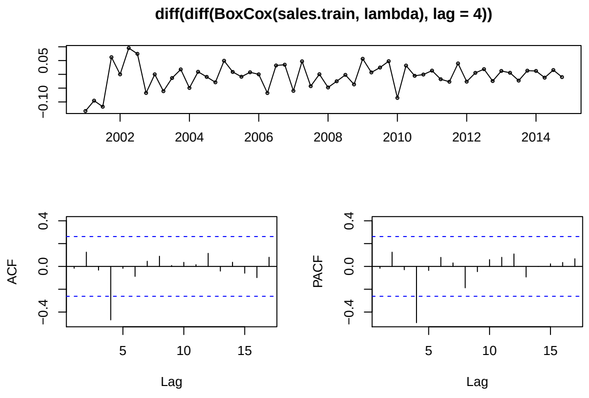
```
# find the best lambda for Box-Cox transformation lambda = 0.01467236,
# similar to a log-transformation
lambda <- BoxCox.lambda(sales.train)
# first-order non-seasonal differenced
plot(diff(sales.train), main = "1st order non-seasonal difference")
# ^ with Box-Cox transformed
plot(diff(BoxCox(sales.train, lambda)), main = "1st order non-seasonal difference w/ Box-Cox t
# ^ with first-order seasonal differenced
plot(diff(diff(BoxCox(sales.train, lambda), lag = 4)), main = "1st order non-seasonal and seas
# ^ and ACF and PACF
tsdisplay(diff(diff(BoxCox(sales.train, lambda), lag = 4)))
# unit root test on first-order differenced log-transformed series
cbind(adf.test(diff(diff(BoxCox(sales.train, lambda), lag = 4)), alternative = "stationary")$p
    kpss.test(diff(diff(BoxCox(sales.train, lambda), lag = 4)))$p.value)
```

```
##      [,1] [,2]
## [1,] 0.01  0.1
```









Modeling

Looking at the autocorrelation function (ACF) and partial autocorrelation function (PACF), we estimate the best-fitting model to be ARIMA$(0, 1, 0)(0, 1, 1)_4$. Our reasoning is as follows:

- Since we perform first order non-seasonal and seasonal difference, the non-seasonal difference d and seasonal difference D are equal to 1.
- With no significant spike in the non-seasonal lags of ACF and PACF plots, it suggests a possible non-seasonal AR(0) and MA(0) term.
- Since the seasonal correlograms in the ACF plot do not tail off to zero but those in the PACF plot does after lag 4, it signifies a potential moving average model. The only significant spike in ACF is at lag 4; all other autocorrelations are not significant. For this reason, it suggests a potential seasonal MA(1) term.

```
(base.m <- Arima(BoxCox(sales.train, lambda), order = c(0, 1, 0), seasonal = list(order = c(0,
    1, 1), 4)))
```

```
## Series: BoxCox(sales.train, lambda)
## ARIMA(0,1,0)(0,1,1)[4]
##
## Coefficients:
##          sma1
##       -0.5118
## s.e.   0.0973
##
## sigma^2 estimated as 0.00151:  log likelihood=102.31
## AIC=-200.62   AICc=-200.39   BIC=-196.57
```

By iterating through multiple parameters, we can confirm whether or not this model is the best-fitting model under the AICc criterion. We chose AICc instead of AIC since the number of observations, 69, is small and AICc can address AIC's potential problem of overfitting for small sample sizes.

```
best.manual.m <- base.m
for (p in 0:2) for (q in 0:2) for (d in 1:2) for (P in 0:2) for (Q in 0:2) for (D in 1:2) {
    m <- Arima(BoxCox(sales.train, lambda), order = c(p, d, q), seasonal = list(order = c(P,
        D, Q)))
    if (m$aicc < best.manual.m$aicc)
        best.manual.m <- m
}
best.manual.m
```

```
## Series: BoxCox(sales.train, lambda)
## ARIMA(0,1,0)(0,1,2)[4]
##
## Coefficients:
##          sma1    sma2
##       -0.7670  0.4052
## s.e.   0.1545  0.1342
##
## sigma^2 estimated as 0.001301:  log likelihood=106.17
## AIC=-206.35   AICc=-205.88   BIC=-200.27
```

What we have found is that ARIMA$(0,1,0)(0,1,2)_4$ has a lower AICc score than our estimated model we derived earlier ARIMA$(0,1,0)(0,1,1)_4$. In other words, ARIMA$(0,1,0)(0,1,2)_4$ may better explain the data but that fit might not be worth it at the cost of a loss in parsimony since we have to impose an additional seasonal MA lag into our estimated model. Similarly, ARIMA$(0,1,0)(0,1,1)_4$ may be more parsimonious, but it might not explain the data as well as ARIMA$(0,1,0)(0,1,2)_4$.

We compare our generated best-fitting model ARIMA$(0,1,0)(0,1,2)_4$ to the one generated by the auto.arima function and found it to be the same.

```
(best.auto.m <- auto.arima(BoxCox(sales.train, lambda), ic = "aicc", stepwise = FALSE,
    approximation = FALSE))
```

```
## Series: BoxCox(sales.train, lambda)
## ARIMA(0,1,0)(0,1,2)[4]
##
## Coefficients:
##          sma1    sma2
##       -0.7670  0.4052
## s.e.   0.1545  0.1342
##
## sigma^2 estimated as 0.001301:  log likelihood=106.17
## AIC=-206.35   AICc=-205.88   BIC=-200.27
```

For this reason, the two models we will compare moving forward are ARIMA$(0,1,0)(0,1,1)_4$, which is more parsimonous, and ARIMA$(0,1,0)(0,1,2)_4$, which has a better fit.

```
m1 <- base.m
m2 <- best.auto.m
```

Validating the models

Before we can forecast what percentage of total retail sales e-commerce sales will be in the future, we first need to validate that the residuals from the two models result in the following properties:

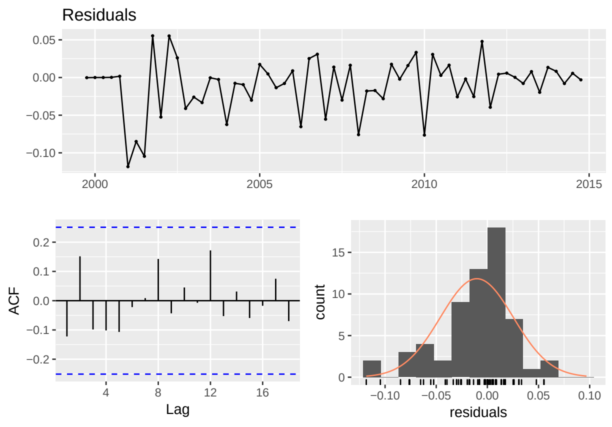
- uncorrelated, meaning there is no information left in the residuals that can be used in computing the forecast
- zero mean
- constant variance
- normally distributed

```
checkresiduals(m1$residuals)
h <- min(2 * 4, nrow(sales.train)/5)  # min(2m, T/5)
Box.test(m1$residuals, type = "Ljung-Box", lag = h)
```

```
##
##  Box-Ljung test
##
## data:  m1$residuals
## X-squared = 6.0888, df = 8, p-value = 0.6373
```

```
shapiro.test(m1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  m1$residuals
## W = 0.93758, p-value = 0.003854
```



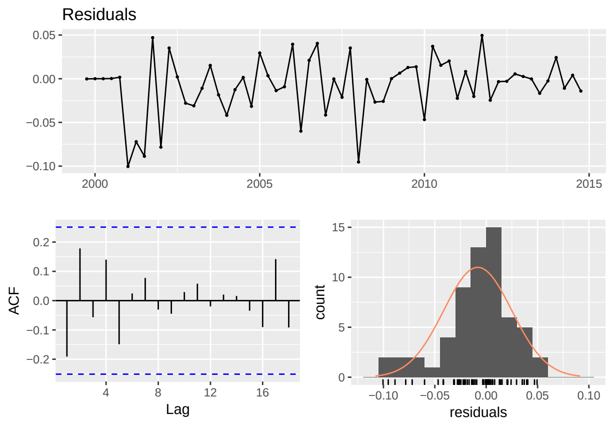
```
checkresiduals(m2$residuals)
h <- min(2 * 4, nrow(sales.train)/5)  # min(2m, T/5)
Box.test(m2$residuals, type = "Ljung-Box", lag = h)
```

```
##
##  Box-Ljung test
##
## data:  m2$residuals
## X-squared = 7.9999, df = 8, p-value = 0.4335
```

```
shapiro.test(m2$residuals)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  m2$residuals
## W = 0.93482, p-value = 0.002911
```



Looking at the ACF plots, all spikes of the two models are within the significant limits, meaning that the residuals are uncorrelated to one another. We then perform a test on a group of autocorrelations with the Box-Ljung test. With large p-values, the Box-Ljung test validates our assumption that the residuals are uncorrelated. The time plot of the residuals shows that the variation of the residuals stays approximately the same for both models, so we can treat the residual variance as constant. However, even with a mean close to zero, the histogram of all models suggests that it follows more of a negative skewed distribution than normal. The Shapiro-Wilk test confirms the non-normality of the distribution for the two models. What this signifies is that when we perform a prediction in the following section, its forecast will generally be quite good but prediction intervals computed assuming a normal distribution may be inaccurate.

Forecasting

Now that we have validated our models, it is time to extrapolate what percentage of e-sales commerce constitutes the total retail sales. First, we compare the forecasts of all the models to the hold out test data from 2015 till 2016 to see if their predictions are comparable to the actual. Then we plot out the forecasts. The blue represents the forecast and the orange represents the test data.

```
sales.test
```

```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 2015  6.9  6.6  6.8  8.7
## 2016  7.7  7.5  7.7  9.5
```

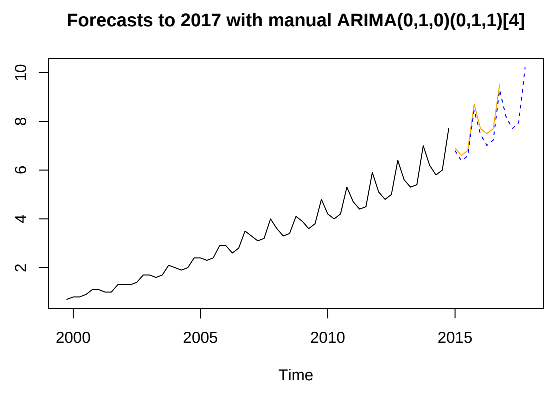
```
InvBoxCox(predict(m1, 3 * 4)$pred, lambda)
```

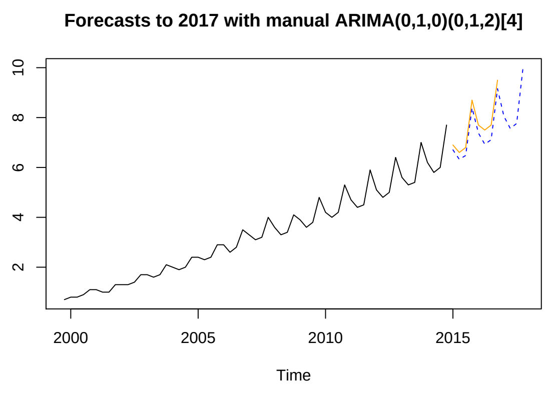
```
##           Qtr1      Qtr2      Qtr3      Qtr4
## 2015  6.793138  6.380801  6.582199  8.457164
## 2016  7.462415  7.010058  7.231008  9.287585
## 2017  8.196570  7.700373  7.942741 10.198236
```

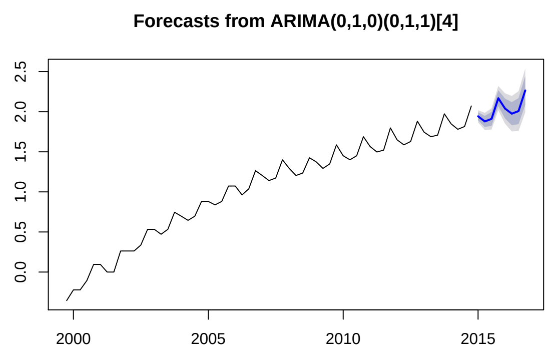
```
InvBoxCox(predict(m2, 3 * 4)$pred, lambda)
```

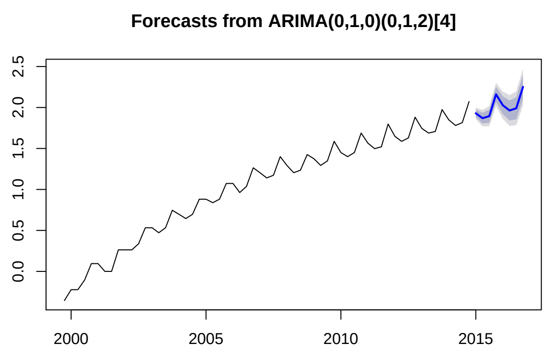
```
##          Qtr1      Qtr2      Qtr3      Qtr4
## 2015 6.705666 6.323977 6.479338 8.392189
## 2016 7.379740 6.930514 7.111839 9.157311
## 2017 8.053882 7.564228 7.761877 9.991076
```

```r
ts.plot(cbind(sales.train, sales.test, InvBoxCox(predict(m1, 4 * 3)$pred,
    lambda)), col = c("black", "orange", "blue"), lty = c(1, 1, 2), main = "Forecasts to 2017
ts.plot(cbind(sales.train, sales.test, InvBoxCox(predict(m2, 4 * 3)$pred,
    lambda)), col = c("black", "orange", "blue"), lty = c(1, 1, 2), main = "Forecasts to 2017
plot(forecast(m1))
plot(forecast(m2))
```









Both the forecasts as well as the graphs tell us that our parsimonous model ARIMA$(0,1,1)(0,1,1)_4$ predict much more closely to the test data than the autogenerated one ARIMA$(0,1,1)(0,1,2)_4$. We use this model to determine that e-commerce makes up approximately 10.20% of all retail sales by the fourth quarter of 2017.

Question 2: Learning how to use the xts library

If we could select one company to represent the e-commerce trend, Amazon is likely to be the first company that comes to mind. We will delve briefly into the Amazon stock as a way to better understand how to use the xts library.

1. Read AMAZ.csv and UMCSENT.csv into R as R DataFrames.

```r
raw.amaz <- read.csv("./data/AMAZ.csv", header = TRUE, sep = ",")
raw.sent <- read.csv("./data/UMCSENT.csv", header = TRUE, sep = ",")
```

2. Convert them to xts objects.

```r
# set local timezone
Sys.setenv(TZ = "America/Los_Angeles")
# assume stock data is collected in EST
amaz <- xts(raw.amaz[, -1], order.by = as.POSIXct(raw.amaz[, 1], tz = "EST"))
# assume sentiment data is collected in EST
sent <- xts(raw.sent[, -1], order.by = as.POSIXct(raw.sent[, 1], tz = "EST"))
```

3. Merge the two set of series together, preserving all of the observations in both set of series.

   a. Fill all of the missing values of the UMCSENT series with -9999.

```r
UMCSENT <- merge(amaz, sent, join = "outer", fill = -9999)
```

```
b. Then create a new series, named UMCSENT02, from the original UMCSENT series and replace all
```

```r
UMCSENT02 <- UMCSENT
UMCSENT02[UMCSENT02 == -9999] <- NA
```

```
c. Then create a new series, named UMCSENT03, and replace the NAs with the last observation.
```

```r
UMCSENT03 <- na.locf(UMCSENT02, fromLast = FALSE)
```

```
d. Then create a new series, named UMCSENT04, and replace the NAs using linear interpolation.
```

```r
UMCSENT04 <- na.approx(UMCSENT02, maxgap = 31)
```

```
e. Print out some observations to ensure that your merge as well as the missing value imputatio
```

```r
# list the top 3 index of NA's
index.nas <- which(is.na(UMCSENT))
# verify that they do contain NA
for (index in index.nas[1:3]) {
    print(UMCSENT[index])
}
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume  sent
## 2007-10-30        NA        NA       NA       10.4           0 -9999

## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume  sent
## 2007-10-31        NA        NA       NA       10.4           0 -9999

## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).
##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-11-01        NA        NA       NA       10.4           0 76.1
```

```
# verify 3b that NA -> -9999
for (index in index.nas[1:3]) {
    print(UMCSENT02[index])
}
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-10-30        NA        NA       NA       10.4           0   NA

## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-10-31        NA        NA       NA       10.4           0   NA

## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-11-01        NA        NA       NA       10.4           0 76.1
```

```
# verify 3c that NA -> last observation
print(end(UMCSENT02))
```

```
## [1] "2017-09-01 EST"
```

```
for (index in index.nas[1:3]) {
    print(UMCSENT03[index])
}
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-10-30      10.4      10.4     10.4       10.4           0 80.9

## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-10-31      10.4      10.4     10.4       10.4           0 80.9

## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).

##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-11-01      10.4      10.4     10.4       10.4           0 76.1
```

```
# 1st way: verify 3d that NA -> linear interpolation 1st index of NA is
# 505 2nd index of NA is 506 3rd index of NA is 507 we want to see
```

```r
# linear interpolation from index 505-507
print(UMCSENT04[index.nas[1]:index.nas[3]])
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).
```

```
##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume     sent
## 2007-10-30      10.6      11.1     10.6       10.4           0 76.40968
## 2007-10-31      10.8      11.8     10.8       10.4           0 76.25484
## 2007-11-01      11.0      12.5     11.0       10.4           0 76.10000
```

```r
# >> the interpolation seems reasonable 2nd way: verify 3d that NA ->
# linear interpolation (value at 1/3) == (value at 1/1) - (value at 1/1
# - value at 2/1) * (date difference b/w 1/1 and 1/3)/(date difference
# b/w 1/1 and 2/1)
coredata(UMCSENT04["2007-01-03", 6]) == coredata(sent["2007-01-01"]) -
    (coredata(sent["2007-01-01"]) - coredata(sent["2007-02-01"])) * 2/31
```

```
##      sent
## [1,] TRUE
```

```r
# verify merge is correct
print(amaz["2007-02-01"])
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).
```

```
##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume
## 2007-02-01        24        24       24         24         270
```

```r
print(sent["2007-02-01"])
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).
```

```
##            [,1]
## 2007-02-01 91.3
```

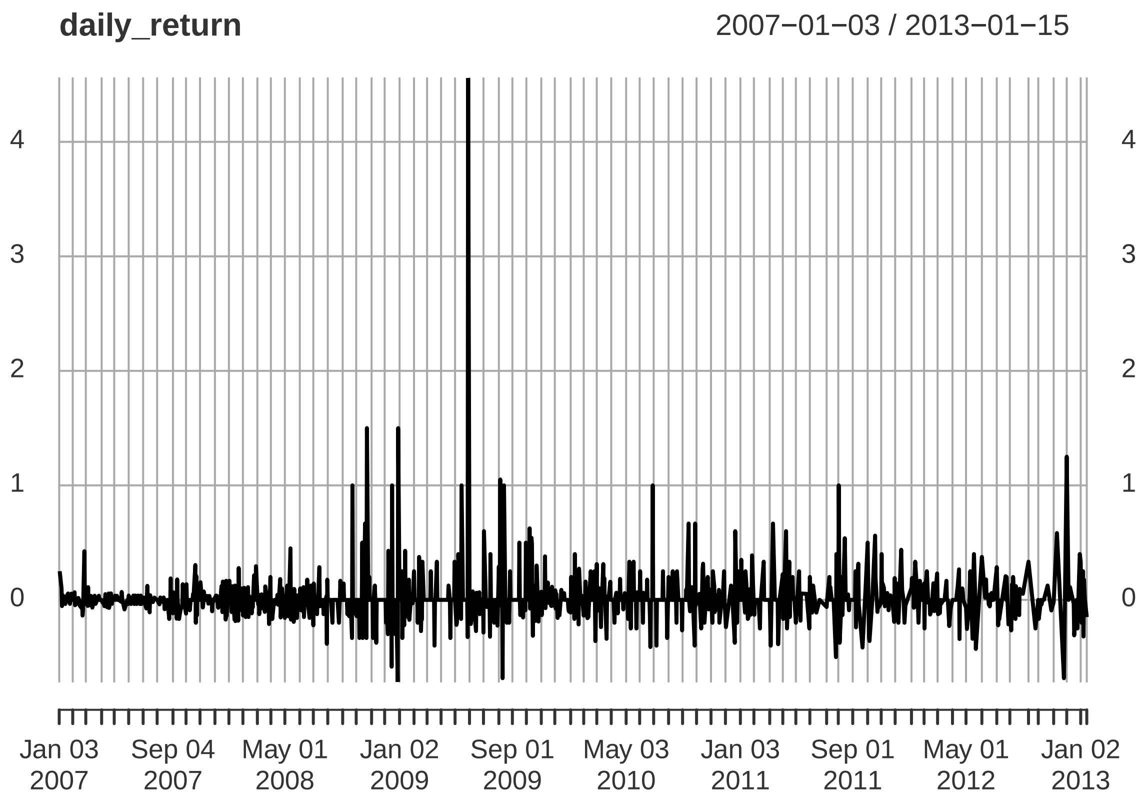
```r
print(UMCSENT["2007-02-01"])
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).
```

```
##            AMAZ.Open AMAZ.High AMAZ.Low AMAZ.Close AMAZ.Volume sent
## 2007-02-01        24        24       24         24         270 91.3
```

4. Calculate the daily return of the Amazon closing price (AMAZ.close), where daily return is defined as $(x(t) - x(t-1))/x(t-1)$. Plot the daily return series.

```r
daily_return <- (amaz[, 4] - lag(amaz[, 4], k = 1))/(lag(amaz[, 4], k = 1))
plot(daily_return)
```



Looking at Amazon's closing price between January 2007 to January 2013, we see much volatility. A few trends that may be gauged is that January to July generally sees lower daily returns and peaks of 1+ returns happens post-July (which may be related to product launch timelines). These trends hold with the exception of January-July 2009 which had unprecedented returns that may be due to the acquisition of Zappos during that time.

5. Create a 20-day and a 50-day rolling mean series from the AMAZ.close series.

```
head(cbind(amaz[, 4], rollapply(amaz[, 4], 20, FUN = mean, na.rm = TRUE)),
    30)
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).
```

```
##            AMAZ.Close AMAZ.Close.1
## 2007-01-03       16.0           NA
## 2007-01-04       20.0           NA
## 2007-01-08       22.0           NA
## 2007-01-09       20.8           NA
## 2007-01-10       20.8           NA
## 2007-01-11       21.6           NA
## 2007-01-12       22.0           NA
## 2007-01-16       21.2           NA
## 2007-01-17       21.6           NA
## 2007-01-22       22.8           NA
## 2007-01-23       22.8           NA
## 2007-01-26       22.0           NA
## 2007-01-29       23.2           NA
## 2007-01-31       24.0           NA
## 2007-02-01       24.0           NA
## 2007-02-02       24.0           NA
## 2007-02-05       25.6           NA
## 2007-02-06       24.4           NA
## 2007-02-09       23.6           NA
## 2007-02-12       23.2        22.28
## 2007-02-13       23.6        22.66
## 2007-02-14       23.6        22.84
## 2007-02-15       23.6        22.92
## 2007-02-16       22.4        23.00
## 2007-02-20       20.8        23.00
## 2007-02-21       20.4        22.94
## 2007-02-22       17.6        22.72
## 2007-02-23       16.0        22.46
## 2007-02-26       22.8        22.52
## 2007-02-27       22.0        22.48
```

```
head(cbind(amaz[, 4], rollapply(amaz[, 4], 50, FUN = mean, na.rm = TRUE)),
    60)
```

```
## Warning: timezone of object (EST) is different than current timezone
## (America/Los_Angeles).
```

```
##            AMAZ.Close AMAZ.Close.1
## 2007-01-03      16.00           NA
## 2007-01-04      20.00           NA
## 2007-01-08      22.00           NA
## 2007-01-09      20.80           NA
## 2007-01-10      20.80           NA
## 2007-01-11      21.60           NA
## 2007-01-12      22.00           NA
## 2007-01-16      21.20           NA
## 2007-01-17      21.60           NA
## 2007-01-22      22.80           NA
## 2007-01-23      22.80           NA
## 2007-01-26      22.00           NA
## 2007-01-29      23.20           NA
## 2007-01-31      24.00           NA
## 2007-02-01      24.00           NA
## 2007-02-02      24.00           NA
## 2007-02-05      25.60           NA
## 2007-02-06      24.40           NA
## 2007-02-09      23.60           NA
## 2007-02-12      23.20           NA
## 2007-02-13      23.60           NA
## 2007-02-14      23.60         NA
## 2007-02-15      23.60         NA
## 2007-02-16      22.40         NA
## 2007-02-20      20.80         NA
## 2007-02-21      20.40         NA
## 2007-02-22      17.60         NA
## 2007-02-23      16.00         NA
## 2007-02-26      22.80         NA
## 2007-02-27      22.00         NA
## 2007-02-28      22.00         NA
## 2007-03-01      22.00         NA
## 2007-03-02      22.80         NA
## 2007-03-05      21.60         NA
## 2007-03-06      24.00         NA
## 2007-03-07      22.80         NA
## 2007-03-09      22.80         NA
## 2007-03-12      23.60         NA
## 2007-03-15      22.00         NA
## 2007-03-19      22.00         NA
## 2007-03-20      22.80         NA
## 2007-03-22      22.00         NA
## 2007-03-29      22.80         NA
## 2007-03-30      22.80         NA
## 2007-04-04      22.80         NA
## 2007-04-05      22.40         NA
## 2007-04-09      21.60         NA
## 2007-04-10      20.40         NA
## 2007-04-11      20.00         NA
## 2007-04-12      21.00    22.0520
## 2007-04-13      21.60    22.1640
## 2007-04-16      20.20    22.1680
## 2007-04-17      21.04    22.1488
## 2007-04-18      21.60    22.1648
## 2007-04-19      22.80    22.2048
## 2007-04-20      21.20    22.1968
## 2007-04-24      22.40    22.2048
## 2007-04-26      21.60    22.2128
## 2007-04-27      21.60    22.2128
## 2007-04-30      21.60    22.1888
```