Amazon: Net Sales Forecasting

Siata Coulibaly

1/14/2020

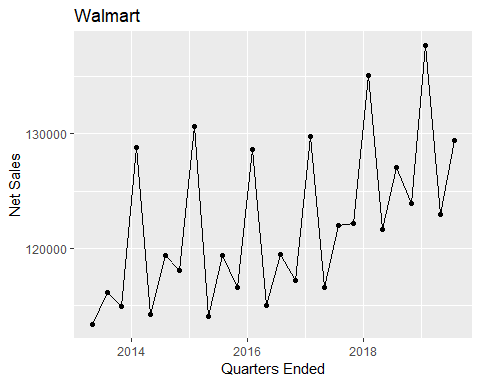
# Introduction

A time series is a series of data points indexed in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus, it is a sequence of discrete-time data. Time series analysis is a statistical technique that deals with time series data, or trend analysis. In this project, we will use the Amazon financial statement data with approximately all information about their income statements for the last five years in terms of quarters. The third quarter of this year was not included, but will probably be publicly available by the end of our analysis. In the following, we will explore the trend of the net sales data, model it for forecasting purposes, and add some other predictors to the modeling to sharpen our results.

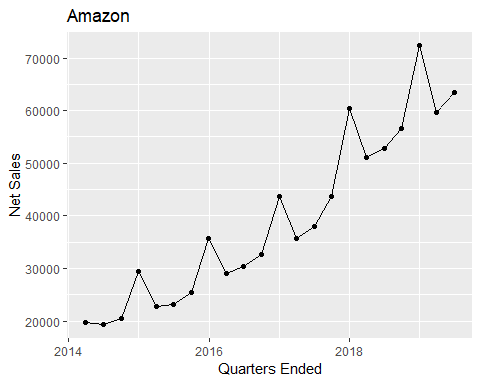
# Data Cleaning and Data Choice

Here is how we cleaned and refined the data so that it can easily be used for our analysis. After working on different subjects such as *Net Income* trend and *Net Sales* trend, we decided to choose the *Net Sales* data series for our analysis. You can see these different results in the following code section.

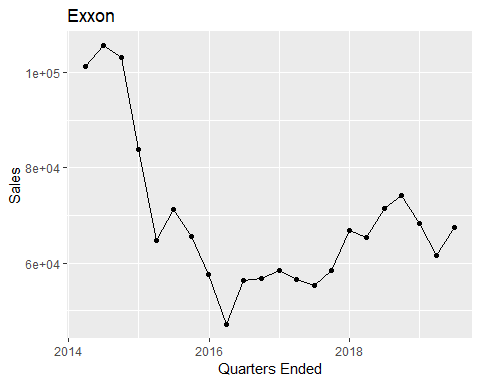
library(tidyverse)  
wal\_entire <- read.csv("Walmart Quarterly-Data.csv")  
amz\_entire <- read.csv("Amazon Quarterly-Data.csv")  
exx\_entire <- read.csv("Exxon Quarterly-Data.csv")  
  
wal\_cut <- wal\_entire[c(5,6), -1]  
amz\_cut <- amz\_entire[c(5,8), -1]  
exx\_cut <- exx\_entire[c(5,6), -1]  
  
wal <- t(wal\_cut) %>%   
 as.tibble(rownames = NULL)  
  
colnames(wal) = c("Quarters\_ended", "Net\_Sales")  
write.csv(wal, "Walmart.csv")  
  
amz <- t(amz\_cut) %>%   
 as.tibble(rownames = NULL)  
  
colnames(amz) = c("Quarters\_ended", "Net\_Sales")  
write.csv(amz, "amazon.csv")  
  
exx <- t(exx\_cut) %>%   
 as.tibble(rownames = NULL)  
  
colnames(exx) = c("Quarters\_ended", "Sales")  
write.csv(exx, "exxon.csv")  
  
  
wal <- read\_csv("walmart.csv",   
 col\_types = cols(Quarters\_ended = col\_character(),   
 X1 = col\_skip()))  
  
wal\_d <- as.character(wal$Quarters\_ended) %>%  
 parse\_date(format = "%b %d, %Y")   
  
wal <- mutate(wal, Quarters\_ended = wal\_d) %>%  
 arrange(Quarters\_ended)  
  
amz <- read\_csv("amazon.csv",   
 col\_types = cols(Quarters\_ended = col\_character(),   
 X1 = col\_skip()))  
amz\_d <- as.character(amz$Quarters\_ended) %>%  
 parse\_date(format = "%b %d, %Y")   
  
amz <- mutate(amz, Quarters\_ended = amz\_d) %>%  
 arrange(Quarters\_ended)  
  
exx <- read\_csv("exxon.csv",   
 col\_types = cols(Quarters\_ended = col\_character(),   
 X1 = col\_skip()))  
exx\_d <- as.character(exx$Quarters\_ended) %>%  
 parse\_date(format = "%b %d, %Y")   
  
exx <- mutate(exx, Quarters\_ended = exx\_d) %>%  
 arrange(Quarters\_ended)  
  
ggplot(wal, aes(Quarters\_ended, Net\_Sales)) + geom\_point() +  
 geom\_line() + xlab("Quarters Ended") + ylab("Net Sales") +   
 ggtitle("Walmart")



ggplot(amz, aes(Quarters\_ended, Net\_Sales)) + geom\_point() +  
 geom\_line() + xlab("Quarters Ended") + ylab("Net Sales") +  
 ggtitle("Amazon")



ggplot(exx, aes(Quarters\_ended, Sales)) + geom\_point() +  
 geom\_line() + xlab("Quarters Ended") + ylab("Sales") +  
 ggtitle("Exxon")



From the visualizations, we found Amazon more intuitive and more interesting to study. The next point was to look at the *net income* and choose which data would be more interesting to study and how possibly useful it could be for potential future investors.

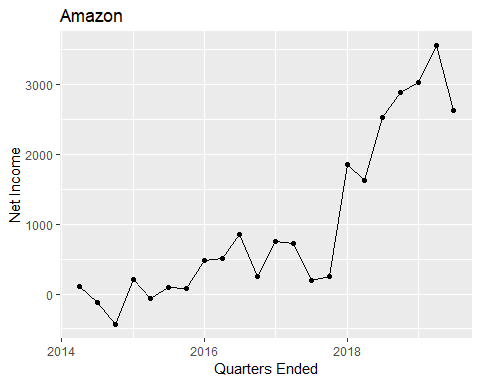
colname <- amz\_entire[5:24, 1]  
amazon\_cleaned <- amz\_entire[-c(1:4, 25), -1] %>%  
 t() %>%  
 as\_tibble(rownames = NULL)  
  
colnames(amazon\_cleaned) <- colname  
  
amazon\_cleaned$`3 months ended` <- as.character(amazon\_cleaned$`3 months ended`) %>%  
 parse\_date(format = "%b %d, %Y")  
  
amazon\_cleaned $quarters\_ended = amazon\_cleaned$`3 months ended`  
  
amazon\_cleaned <- amazon\_cleaned[c(1, 21, 2:20)]  
amazon\_cleaned <- amazon\_cleaned[-1]  
  
amazon\_used <- amazon\_cleaned[c(1, 20, 4, 12, 8, 9)]  
  
amazon\_used$`Net income (loss)`[c(18,20,21)] <- c("-57", "-437", "-126")  
  
amazon\_used$`Operating income (loss)`[c(20,21)] <- c("-544", "-15")  
  
  
colnames(amazon\_used) <- c("quarters\_ended", "net\_income", "net\_sales", "operating\_income",  
 "marketing\_exp", "technology\_exp")  
  
amazon\_used <- mutate(amazon\_used, marketing\_exp = str\_remove\_all(marketing\_exp, "[())]"))  
  
amazon\_used <- mutate(amazon\_used, technology\_exp = str\_remove\_all(technology\_exp, "[())]"))  
  
  
amazon\_used$net\_income <- parse\_number(amazon\_used$net\_income)  
amazon\_used$net\_sales <- parse\_number(amazon\_used$net\_sales)  
amazon\_used$operating\_income <- parse\_number(amazon\_used$operating\_income)  
amazon\_used$marketing\_exp <- parse\_number(amazon\_used$marketing\_exp)  
amazon\_used$technology\_exp <- parse\_number(amazon\_used$technology\_exp)  
summary(amazon\_used)

## quarters\_ended net\_income net\_sales operating\_income  
## Min. :2014-03-31 Min. :-437.0 Min. :19340 Min. :-544.0   
## 1st Qu.:2015-07-23 1st Qu.: 130.2 1st Qu.:26301 1st Qu.: 420.5   
## Median :2016-11-15 Median : 497.5 Median :35730 Median :1038.0   
## Mean :2016-11-14 Mean :1000.8 Mean :39356 Mean :1392.2   
## 3rd Qu.:2018-03-08 3rd Qu.:1799.2 3rd Qu.:52425 3rd Qu.:2076.2   
## Max. :2019-06-30 Max. :3561.0 Max. :72383 Max. :4420.0   
## marketing\_exp technology\_exp  
## Min. : 870 Min. :1991   
## 1st Qu.:1307 1st Qu.:3064   
## Median :1838 Median :4340   
## Mean :2212 Mean :4834   
## 3rd Qu.:2850 3rd Qu.:6648   
## Max. :4911 Max. :9065

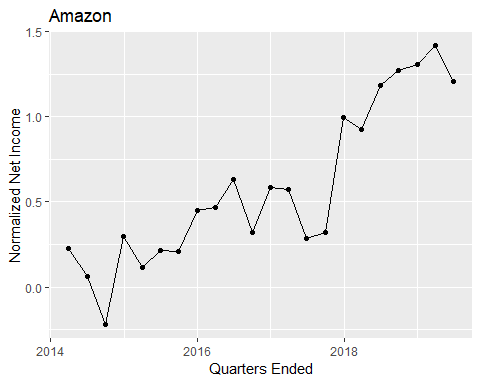
glimpse(amazon\_used)

## Observations: 22  
## Variables: 6  
## $ quarters\_ended <date> 2019-06-30, 2019-03-31, 2018-12-31, 2018-09-...  
## $ net\_income <dbl> 2625, 3561, 3027, 2883, 2534, 1629, 1856, 256...  
## $ net\_sales <dbl> 63404, 59700, 72383, 56576, 52886, 51042, 604...  
## $ operating\_income <dbl> 3084, 4420, 3787, 3724, 2983, 1927, 2126, 347...  
## $ marketing\_exp <dbl> 4291, 3664, 4911, 3303, 2901, 2699, 3441, 247...  
## $ technology\_exp <dbl> 9065, 7927, 7669, 7162, 7247, 6759, 6314, 594...

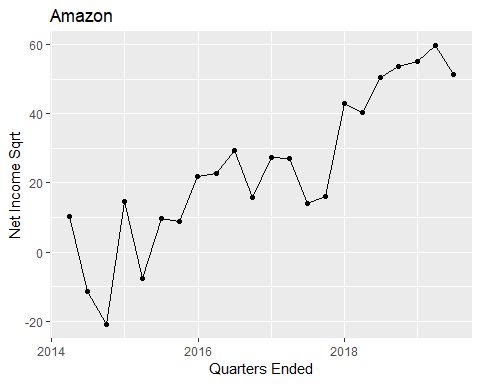
ggplot(amazon\_used, aes(quarters\_ended, net\_income)) + geom\_point() +  
 geom\_line() + xlab("Quarters Ended") + ylab("Net Income") +  
 ggtitle("Amazon")



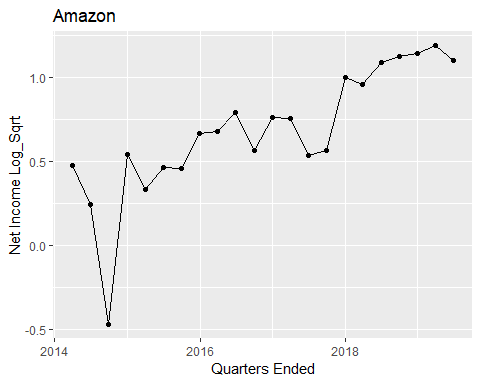
means <- c(mean(amazon\_used$net\_income), mean(amazon\_used$net\_sales),  
 mean(amazon\_used$operating\_income))  
  
sds <- c(sd(amazon\_used$net\_income), sd(amazon\_used$net\_sales), sd(amazon\_used$operating\_income))  
  
  
norm\_log\_amz\_used <- tibble(.rows = 22)  
  
for(i in 1:3)  
{  
 for(j in 1:22)  
 {  
 norm\_log\_amz\_used[j, i] <- log(2 + ((amazon\_used[j, i+1] - means[i]) / sds[i]))  
 }  
}  
  
norm\_log\_amz\_used$quarters\_ended <- amazon\_used$quarters\_ended  
norm\_log\_amz\_used <- norm\_log\_amz\_used[c(4, 1:3)]  
  
ggplot(norm\_log\_amz\_used, aes(quarters\_ended, net\_income)) + geom\_point() +  
 geom\_line() + xlab("Quarters Ended") + ylab("Normalized Net Income") +  
 ggtitle("Amazon")



norm\_sqrt\_amz\_used <- tibble(.rows = 22)  
  
for(i in 1:3)  
{  
 for(j in 1:22)  
 {  
 if(amazon\_used[j, i+1] < 0)  
 {  
 norm\_sqrt\_amz\_used[j, i] <- -sqrt(abs(amazon\_used[j, i+1]))  
 } else{  
 norm\_sqrt\_amz\_used[j, i] <- sqrt(amazon\_used[j, i+1])  
 }  
 }  
}  
  
norm\_sqrt\_amz\_used$quarters\_ended <- amazon\_used$quarters\_ended  
norm\_sqrt\_amz\_used <- norm\_sqrt\_amz\_used[c(4, 1:3)]  
  
ggplot(norm\_sqrt\_amz\_used, aes(quarters\_ended, net\_income)) + geom\_point() +  
 geom\_line() + xlab("Quarters Ended") + ylab("Net Income Sqrt") +  
 ggtitle("Amazon")



norm\_log\_sqrt\_amz\_used <- tibble(.rows = 22)  
  
for(i in 1:3)  
{  
 for(j in 1:22)  
 {  
 if(norm\_log\_amz\_used[j, i+1] < 0)  
 {  
 norm\_log\_sqrt\_amz\_used[j, i] <- -sqrt(abs(norm\_log\_amz\_used[j, i+1]))  
 } else{  
 norm\_log\_sqrt\_amz\_used[j, i] <- sqrt(norm\_log\_amz\_used[j, i+1])  
 }  
 }  
}  
  
norm\_log\_sqrt\_amz\_used$quarters\_ended <- amazon\_used$quarters\_ended  
norm\_log\_sqrt\_amz\_used <- norm\_log\_sqrt\_amz\_used[c(4, 1:3)]  
  
ggplot(norm\_log\_sqrt\_amz\_used, aes(quarters\_ended, net\_income)) + geom\_point() +  
 geom\_line() + xlab("Quarters Ended") + ylab("Net Income Log\_Sqrt") +  
 ggtitle("Amazon")

 The trend from the Amazon net income does not look as interesting as the net sales did, and it looked more complicated for our analysis purposes. The final data we will work on then is the Amazon *Net Sales*.

# Data Modeling

## Times Series Data Modeling

### Data Decomposition

As you have seen previously, Amazon’s net sales looks having an increasing trend and also a seasonal one. Given this characteristic, we can decompose our data into a time series object using the ts function.

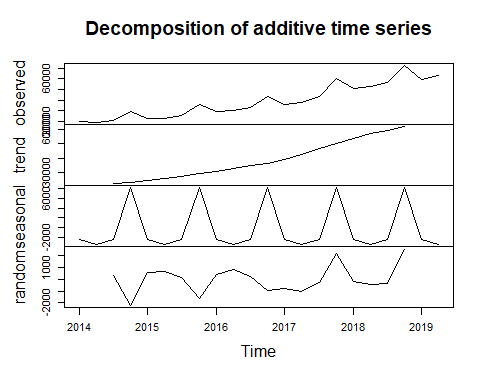
amz <- arrange(amazon\_used, quarters\_ended) [-c(1,2,4,5,6)]  
amz\_ts <- ts(amz, start = c(2014, 1), end = c(2019, 2), frequency = 4)  
amz\_ts

## Qtr1 Qtr2 Qtr3 Qtr4  
## 2014 19741 19340 20579 29328  
## 2015 22717 23185 25358 35746  
## 2016 29128 30404 32714 43741  
## 2017 35714 37955 43744 60453  
## 2018 51042 52886 56576 72383  
## 2019 59700 63404

dec\_amz\_ts <- stats::decompose(amz\_ts)  
dec\_amz\_ts

## $x  
## Qtr1 Qtr2 Qtr3 Qtr4  
## 2014 19741 19340 20579 29328  
## 2015 22717 23185 25358 35746  
## 2016 29128 30404 32714 43741  
## 2017 35714 37955 43744 60453  
## 2018 51042 52886 56576 72383  
## 2019 59700 63404   
##   
## $seasonal  
## Qtr1 Qtr2 Qtr3 Qtr4  
## 2014 -2381.353 -3394.478 -2329.797 8105.628  
## 2015 -2381.353 -3394.478 -2329.797 8105.628  
## 2016 -2381.353 -3394.478 -2329.797 8105.628  
## 2017 -2381.353 -3394.478 -2329.797 8105.628  
## 2018 -2381.353 -3394.478 -2329.797 8105.628  
## 2019 -2381.353 -3394.478   
##   
## $trend  
## Qtr1 Qtr2 Qtr3 Qtr4  
## 2014 NA NA 22619.00 23471.62  
## 2015 24549.62 25949.25 27552.88 29256.62  
## 2016 31078.50 32997.38 34820.00 36587.12  
## 2017 38909.75 42377.50 46382.50 50164.88  
## 2018 53635.25 56730.50 59304.00 61701.00  
## 2019 NA NA   
##   
## $random  
## Qtr1 Qtr2 Qtr3 Qtr4  
## 2014 NA NA 289.7969 -2249.2531  
## 2015 548.7281 630.2281 134.9219 -1616.2531  
## 2016 430.8531 801.1031 223.7969 -951.7531  
## 2017 -814.3969 -1028.0219 -308.7031 2182.4969  
## 2018 -211.8969 -450.0219 -398.2031 2576.3719  
## 2019 NA NA   
##   
## $figure  
## [1] -2381.353 -3394.478 -2329.797 8105.628  
##   
## $type  
## [1] "additive"  
##   
## attr(,"class")  
## [1] "decomposed.ts"

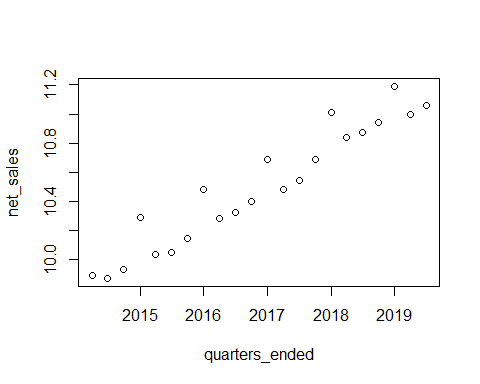
plot(dec\_amz\_ts)



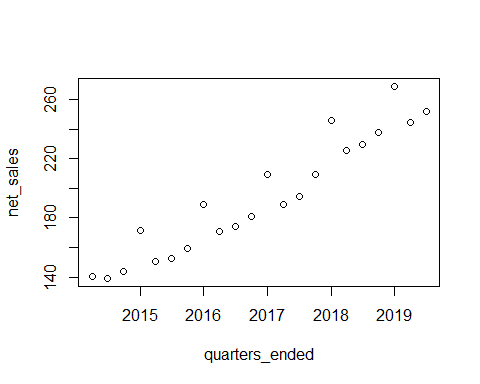
### Data Transformation

The blue line indicating the optimal lag value hits the correlation bar for most cases at lag = 4. This means that the best value that can be used to predict the net sales value of a quarter is the value of the same quarter from the previous year. That makes sense because of the seasonality pattern.

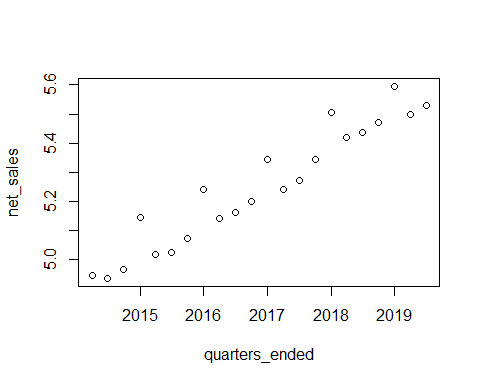
amz\_sub <- arrange(amazon\_used, quarters\_ended)[-c(2,4,5,6)]  
amz\_log <- mutate(amz\_sub, net\_sales = log(net\_sales))  
plot(amz\_log)



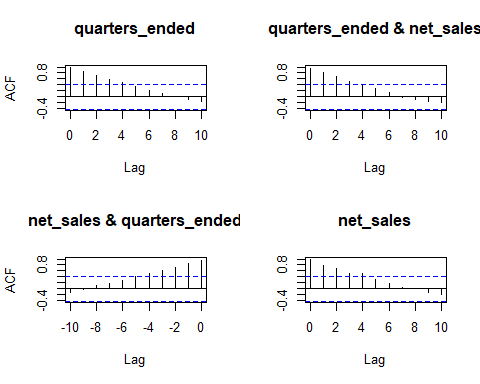
amz\_sqrt <- mutate(amz\_sub, net\_sales = sqrt(abs(net\_sales)))  
plot(amz\_sqrt)



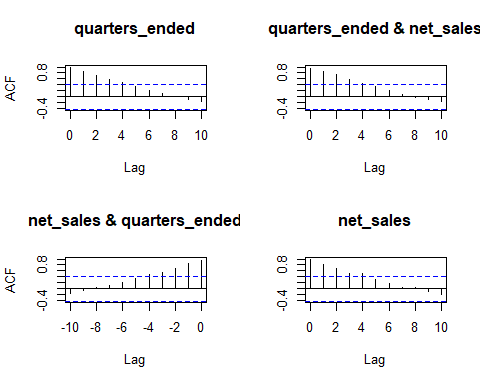
amz\_sqrt\_log <- mutate(amz\_sub, net\_sales = log(sqrt(abs(net\_sales))))  
plot(amz\_sqrt\_log)



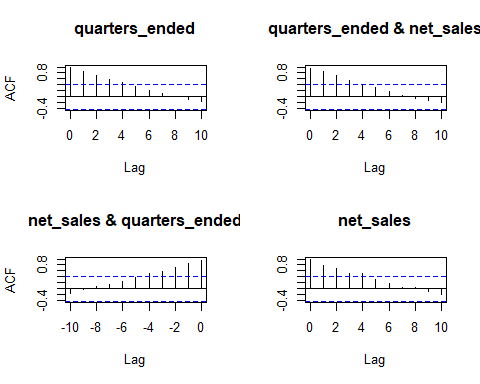
acf(amz\_sub)



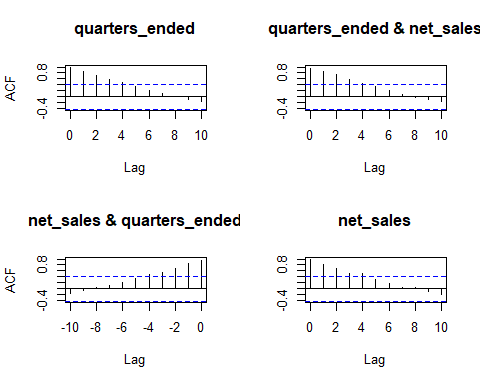
acf(amz\_log)



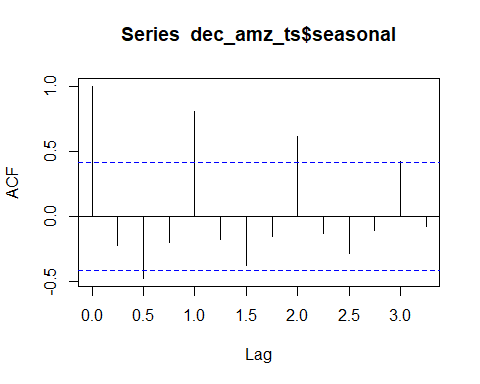
acf(amz\_sqrt)



acf(amz\_sqrt\_log)



acf(dec\_amz\_ts$seasonal)



## Regression Modeling

### Times Series Models

After visualizing how our data is behaving, we will try to fit a model that would help us be more accurate in forecasting the future quarters net sales. For that, we will try to fit multiple models and retain the one with the least error.

amztr <- amz[1:18,]  
amz\_ar1 <- arima(amztr, order = c(1, 0, 0), seasonal = list(order = c(1, 0, 0)))  
amz\_ar2 <- arima(amztr, order = c(2, 0, 0), seasonal = list(order = c(2, 0, 0)))  
amz\_ar3 <- arima(amztr, order = c(3, 0, 0), seasonal = list(order = c(3, 0, 0)))  
amz\_ar4 <- arima(amztr, order = c(4, 0, 0), seasonal = list(order = c(4, 0, 0)))  
  
AIC(amz\_ar1, amz\_ar2, amz\_ar3, amz\_ar4)

## df AIC  
## amz\_ar1 4 381.2604  
## amz\_ar2 6 383.2833  
## amz\_ar3 8 386.8106  
## amz\_ar4 10 373.2889

amz\_ma1 <- arima(amztr, order = c(0, 0, 1), seasonal = list(order = c(0, 0, 1)))  
amz\_ma2 <- arima(amztr, order = c(0, 0, 2), seasonal = list(order = c(0, 0, 2)))  
amz\_ma3 <- arima(amztr, order = c(0, 0, 3), seasonal = list(order = c(0, 0, 3)))  
amz\_ma4 <- arima(amztr, order = c(0, 0, 4), seasonal = list(order = c(0, 0, 4)))  
  
AIC(amz\_ma1, amz\_ma2, amz\_ma3, amz\_ma4)

## df AIC  
## amz\_ma1 4 386.8637  
## amz\_ma2 6 384.1205  
## amz\_ma3 8 385.0883  
## amz\_ma4 10 378.8451

amz\_arma104 <- arima(amztr, order = c(1, 0, 4), seasonal = list(order = c(1, 0, 4)))  
amz\_arma204 <- arima(amztr, order = c(2, 0, 4), seasonal = list(order = c(2, 0, 4)))  
amz\_arma304 <- arima(amztr, order = c(3, 0, 4), seasonal = list(order = c(3, 0, 4)))  
amz\_arma404 <- arima(amztr, order = c(4, 0, 4), seasonal = list(order = c(4, 0, 4)))  
  
AIC(amz\_arma104, amz\_arma204, amz\_arma304, amz\_arma404)

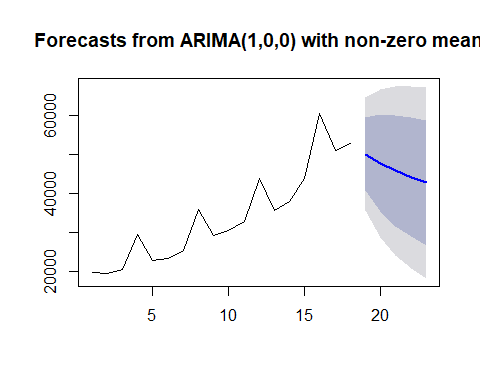
## df AIC  
## amz\_arma104 12 379.4757  
## amz\_arma204 14 376.8353  
## amz\_arma304 16 366.4078  
## amz\_arma404 18 399.1788

After our modeling phase, we will use the automatic version of arima function that is auto.arima. Using this function, we end up the auto-regressive model with order 1 (ARIMA(1,0,0)). This result is different than the step-by-step modeling process result, so which one will be optimal for our case? Apparently, the default model gives a further prediction from the actual value.

library(forecast)  
  
amz\_model <- auto.arima(amztr, d = 0, D = 0, max.P = 4, max.Q = 4, start.P = 0, start.Q = 0,  
 max.p = 4, max.q = 4, start.p = 0, start.q = 0, max.order = 12, stepwise = F)  
  
forecast(amz\_model, h = 5)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 19 50109.68 40614.91 59604.45 35588.68 64630.68  
## 20 47762.77 35330.09 60195.44 28748.63 66776.90  
## 21 45778.84 31615.30 59942.38 24117.58 67440.10  
## 22 44101.76 28821.00 59382.51 20731.86 67471.66  
## 23 42684.06 26652.58 58715.54 18166.02 67202.10

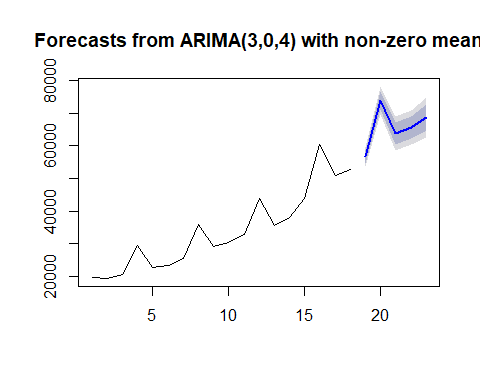
plot(forecast(amz\_model, h = 5))



forecast(amz\_arma304, h = 5)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 19 56581.65 54670.23 58493.08 53658.38 59504.93  
## 20 73852.31 70992.74 76711.89 69478.97 78225.66  
## 21 63769.56 60423.72 67115.40 58652.55 68886.58  
## 22 65580.17 62207.21 68953.13 60421.67 70738.67  
## 23 68636.95 64632.62 72641.28 62512.86 74761.04

plot(forecast(amz\_arma304, h = 5))



### Dynamic Regression Models

Using the dynamic regression models, we are integrating the marketing expenses and the technology expenses to observe how each of them could improve our predictions.

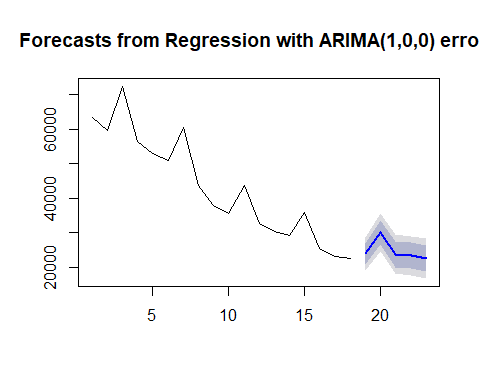
amz\_used <- amazon\_used[1:18,]  
  
dyn\_mod <- auto.arima(amz\_used$net\_sales, xreg = stats::lag(amz\_used$marketing\_exp, k = 1))  
summary(dyn\_mod)

## Series: amz\_used$net\_sales   
## Regression with ARIMA(1,0,0) errors   
##   
## Coefficients:  
## ar1 intercept xreg  
## 0.5628 11722.635 12.6169  
## s.e. 0.2486 2317.569 0.8869  
##   
## sigma^2 estimated as 5862940: log likelihood=-164.35  
## AIC=336.7 AICc=339.77 BIC=340.26  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 100.4455 2210.381 1943.737 -0.6417654 4.667694 0.2894366  
## ACF1  
## Training set -0.007154829

forecast(dyn\_mod, xreg = stats::lag(amazon\_used$marketing\_exp[18:22], k = 1))

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 19 23884.11 20781.03 26987.20 19138.35 28629.87  
## 20 30130.30 26569.46 33691.13 24684.47 35576.12  
## 21 23775.21 20081.18 27469.24 18125.68 29424.74  
## 22 23352.46 19617.23 27087.70 17639.91 29065.01  
## 23 22548.55 18800.36 26296.74 16816.18 28280.92

plot(forecast(dyn\_mod, xreg = amazon\_used$marketing\_exp[18:22]))



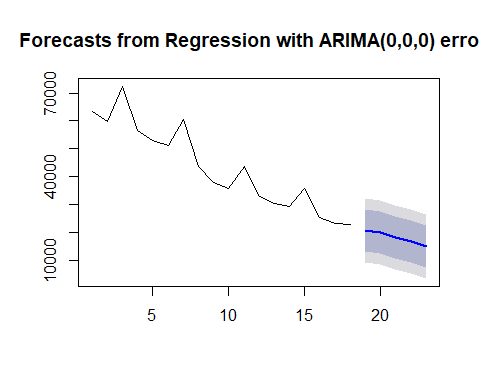
dyn\_mod1 <- auto.arima(amz\_used$net\_sales, xreg = lag(amz\_used$technology\_exp, k = 1))  
summary(dyn\_mod1)

## Series: amz\_used$net\_sales   
## Regression with ARIMA(0,0,0) errors   
##   
## Coefficients:  
## xreg  
## 7.5256  
## s.e. 0.2436  
##   
## sigma^2 estimated as 34404056: log likelihood=-171.63  
## AIC=347.26 AICc=348.11 BIC=348.92  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 213.3886 5865.497 4380.156 -0.524716 9.513096 0.6522372  
## ACF1  
## Training set -0.3298756

forecast(dyn\_mod1, xreg = amazon\_used$technology\_exp[18:22])

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## 19 20725.61 13208.676 28242.55 9229.450 32221.78  
## 20 19830.06 12313.125 27347.00 8333.899 31326.23  
## 21 18234.63 10717.690 25751.56 6738.463 29730.79  
## 22 16752.08 9235.138 24269.01 5255.912 28248.24  
## 23 14983.55 7466.613 22500.49 3487.387 26479.71

plot(forecast(dyn\_mod1, xreg = amazon\_used$technology\_exp[18:22]))



dyn\_mod2 <- arima(amz\_used$net\_sales, order = c(3, 0, 4), seasonal = list(order = c(3, 0, 4)),  
 xreg = stats::lag(amz\_used$marketing\_exp, k = 1))  
summary(dyn\_mod2)

##   
## Call:  
## arima(x = amz\_used$net\_sales, order = c(3, 0, 4), seasonal = list(order = c(3,   
## 0, 4)), xreg = stats::lag(amz\_used$marketing\_exp, k = 1))  
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 ma3 ma4 sar1  
## 0.6730 -0.0937 -0.3806 -1.1475 0.5191 -0.7109 0.7587 0.6730  
## s.e. 0.3635 0.6105 0.3612 0.3657 0.8403 0.7851 0.2531 0.3635  
## sar2 sar3 sma1 sma2 sma3 sma4 intercept  
## -0.0937 -0.3806 -1.2749 0.4130 -0.8871 0.8015 7706.7089  
## s.e. 0.6105 0.3612 0.4274 0.8659 0.8789 0.3913 361.3519  
## stats::lag(amz\_used$marketing\_exp, k = 1)  
## 14.6710  
## s.e. 0.1614  
##   
## sigma^2 estimated as 675583: log likelihood = -157.93, aic = 349.86  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -359.3286 821.9386 679.2397 -0.626844 1.621318 0.1011437  
## ACF1  
## Training set 0.2125014

predict(dyn\_mod2, newxreg = amazon\_used$marketing\_exp[18:22])

## $pred  
## Time Series:  
## Start = 19   
## End = 23   
## Frequency = 1   
## [1] 24778.22 31277.47 23457.83 22724.28 21653.29  
##   
## $se  
## Time Series:  
## Start = 19   
## End = 19   
## Frequency = 1   
## [1] 909.6016

dyn\_mod3 <- arima(amz\_used$net\_sales, order = c(3, 0, 4), seasonal = list(order = c(3, 0, 4)),  
 xreg = stats::lag(amz\_used$technology\_exp, k = 1))  
summary(dyn\_mod3)

##   
## Call:  
## arima(x = amz\_used$net\_sales, order = c(3, 0, 4), seasonal = list(order = c(3,   
## 0, 4)), xreg = stats::lag(amz\_used$technology\_exp, k = 1))  
##   
## Coefficients:  
## ar1 ar2 ar3 ma1 ma2 ma3 ma4 sar1  
## -1.0009 -0.9819 -0.9594 -0.6962 -0.9977 0.8418 0.1596 0.6122  
## s.e. 0.0726 0.0787 0.0431 0.4271 0.1024 0.4338 0.4467 0.0247  
## sar2 sar3 sma1 sma2 sma3 sma4 intercept  
## 0.6138 -0.9982 -0.6261 -0.9720 0.6232 0.1099 2674.3339  
## s.e. 0.0245 0.0031 0.4975 0.2001 0.5346 0.4054 133.7737  
## stats::lag(amz\_used$technology\_exp, k = 1)  
## 7.5282  
## s.e. 0.0266  
##   
## sigma^2 estimated as 372539: log likelihood = -157.03, aic = 348.05  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 39.03338 610.3599 488.3604 0.1067009 1.289355 0.07272043  
## ACF1  
## Training set -0.2184374

predict(dyn\_mod3, newxreg = amazon\_used$technology\_exp[18:22])

## $pred  
## Time Series:  
## Start = 19   
## End = 23   
## Frequency = 1   
## [1] 25834.83 24938.98 23343.00 21859.95 20090.83  
##   
## $se  
## Time Series:  
## Start = 19   
## End = 19   
## Frequency = 1   
## [1] 693.0824

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Date | Actual data | auto arima | arima304 | auto arima with marketing exp | auto arima with technology exp | arima304 with marketing exp | arima304 with technology exp |
| 09/30/18 | 56576 | 50109.68 | 56581.65 | 23884.11 | 20725.61 | 24778.22 | 25834.83 |
| 12/31/18 | 72383 | 47762.77 | 73852.31 | 30130.30 | 19830.06 | 31277.47 | 24938.98 |
| 03/31/19 | 59700 | 45778.84 | 63769.56 | 23775.21 | 18234.63 | 23457.83 | 23343.00 |
| 06/30/19 | 63404 | 44101.76 | 65580.17 | 23352.46 | 16752.08 | 22724.28 | 21859.95 |
| 09/30/19 | 69981 | 42684.06 | 68636.95 | 22548.55 | 14983.55 | 21653.29 | 20090.83 |

# Conclusion

In summary, the increasing and seasonal trends of Amazon’s net sales over time fit best to the auto-regressive integrated moving average model with order (3, 0, 4). this model’s results were in average much closer to the predicted values than the other models. Using the marketing expenses or the technology expenses was making the predictions even worse: **the simpler the model, the better the predictions** in our case. The last point to remember in this analysis is not to ignore the manual or step-wise method of choosing a fitting model, it is very useful for model choice justification.