VAR_Model_Cupcakes_v1

Steps for VAR Modeling -

1. Seasonally adjust the data
2. Individual ARIMA models, both time series needs to be I(1) process. Interpret ACF, PACF, Residuals - White noise plots?
3. ADF and KPSS test -> Not stationary.
4. Johanessen test for cointegration.
5. VAR Model - Train / test prediction error

Read seasonally adjusted data for modeling

```
library(tseries)
library(xlsx)
```

```
## Loading required package: rJava
## Loading required package: xlsxjars
```

```
library(forecast)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: timeDate
## This is forecast 5.9
```

```
setwd("/Users/kevalshah/Keval_Backup/University/UChicago/Capstone/Data/Data Clean up/Clean data to be used f
or Analysis/Cupcakes")

options(java.parameters = "-Xmx1000m")

cupcakes_data_social <- read.xlsx("Cupcakes_seasonally_adjusted.xlsx", sheetName = "SocialAdj")


# subset the data to have include observations that have both sales and social media data

cupcakes_data_social_ts <- cupcakes_data_social[1:260,]
head(cupcakes_data_social_ts)
```

```
##         Date Twitter...Seasonally.Adjusted Facebook...Seasonally.Adjusted
## 1 2010-01-10                      34975.69                     -717.89224
## 2 2010-01-17                      31292.11                    -2211.20474
## 3 2010-01-24                      13126.95                       94.73516
## 4 2010-01-31                     -21398.08                      -10.76724
## 5 2010-02-07                     -15742.75                     1628.79285
## 6 2010-02-14                    -106007.93                    -1230.11099
##   Blogs...Seasonally.Adjusted Comments...Seasonally.Adjusted
## 1                   6645.111                      -39.74903
## 2                   5952.515                       -4.33076
## 3                   6816.822                      -45.18172
## 4                   7375.750                      -82.09278
## 5                   8918.294                      -88.37403
## 6                   7917.022                     -126.57836
##   Forums...Seasonally.Adjusted Reviews...Seasonally.Adjusted
## 1                    1429.2462                     14.92211
## 2                    1859.8424                     30.24182
## 3                    1490.2270                     33.13124
## 4                    1962.3616                     75.73941
## 5                    2077.7342                     84.65047
## 6                     776.1525                     55.91009
##   Total.Social.Media
## 1          42307.325
## 2          36919.169
## 3          21516.688
## 4         -12077.084
## 5          -3121.649
## 6         -98615.536
```

```
# Read seasonally adjusted cupcakes sales volume data

cupcakes_data_sales <- read.xlsx("Cupcakes_seasonally_adjusted.xlsx", sheetName = "SalesAdj")


cupcakes_data_sales_ts <- cupcakes_data_sales
head(cupcakes_data_sales_ts)
```

```
##         Date Sales.Seasonally.Adjusted
## 1 2010-01-10                  92.90408
## 2 2010-01-17                  95.91610
## 3 2010-01-24                  74.10600
## 4 2010-01-31                  89.33437
## 5 2010-02-07                  56.99062
## 6 2010-02-14                 149.38485
```

ARIMA Model

```
# Function
  fun.illustrate.2=function(data,nperiod,p,d,q,P,D,Q) {

    error.holdout = rep(0,nperiod)
    r.sq.error.holdout = rep(0, nperiod)


    for(i in 1:nperiod) {

      # Keeping the first week as hold out for i[1] and then increment until 52nd value
      # 52nd value = 52 week = 1 year i.e last year as hold out.
      cutoff = length(data) - i
      #cutoff = cutoff - i

      #yvec.train=as.vector(data)[1:cutoff]
        if(cutoff >= nperiod) {
        yvec.train=as.vector(data)[1:cutoff]
        #break;
        yvec.hold=as.vector(data)[(cutoff+1):length(data)]
        #yvec.hold

        y=ts(yvec.train, start=2010, frequency=52)
        pred=predict(arima(y, order = c(p,d,q), seasonal = list(order = c(P,D,Q))),n.ahead=(length(data)-cut
off))
        # Predicted - Actual? or Actual - predicted.
        error.holdout[i]=mean((pred$pred-yvec.hold)^2)
        if(length(pred$pred) > 1) {r.sq.error.holdout[i] = (cor(pred$pred,yvec.hold))^2}
        #residuals.holdout[i] = yvec.hold - pred$pred
      }

    }
    # Ignore R Square of the i = 1 when holdout is last week.
    #return(list(error.holdout=error.holdout, Average = (error.holdout)^(1/length(error.holdout))))

    return(list(error.holdout=error.holdout, Average = mean(error.holdout), R.Squared = r.sq.error.holdout,
length(pred$pred), length(yvec.hold)))

    #predict(arima(y, order = c(p,d,q), seasonal = list(order = c(P,D,Q))),n.ahead=12)$pred
    #predict(arima(y, order = c(p,d,q), seasonal = list(order = c(P,D,Q))),n.ahead=12)$pred)^2

  }
```

ARIMA for cupcakes Social Media Mentions

```
# cupcakess Social Media Mentions

f1<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,0,0,0,0,0)
f2<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,0,1,0,0,0)
f3<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,0,2,0,0,0)
f4<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,1,1,0,0,0)
f5<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,1,2,0,0,0)
f6<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,1,0,0,0,0)
f7<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,2,0,0,0,0)
f8<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,2,1,0,0,0)
f9<- fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 2,2,2,0,0,0)

f10<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,0,0,0,0,0)
f11<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,0,1,0,0,0)
f12<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,0,2,0,0,0)
f13<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,1,1,0,0,0)
f14<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,1,2,0,0,0)
f15<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,1,0,0,0,0)
f16<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,2,0,0,0,0)
f17<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,2,1,0,0,0)
f18<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 1,2,2,0,0,0)

f19<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,0,0,0,0,0)
f20<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,0,1,0,0,0)
f21<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,0,2,0,0,0)
f22<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,1,0,0,0,0)
f23<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,1,1,0,0,0)
f24<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,1,2,0,0,0)
f25<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,2,0,0,0,0)
f26<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,2,1,0,0,0)
f27<-fun.illustrate.2(cupcakes_data_social$Total.Social.Media,52, 0,2,2,0,0,0)

# Concatenate
total.social_media <- c(f1$Average, f2$Average, f3$Average, f4$Average, f5$Average, f6$Average, f7$Average,
f8$Average,f9$Average, f10$Average, f11$Average, f12$Average, f13$Average, f14$Average, f15$Average, f16$Ave
rage, f17$Average, f18$Average, f19$Average, f20$Average, f21$Average, f22$Average, f23$Average,
f24$Average, f25$Average, f26$Average, f27$Average)

# Minimum
summary(total.social_media)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 9.020e+08 9.177e+08 1.033e+09 1.035e+10 1.930e+09 1.749e+11
```

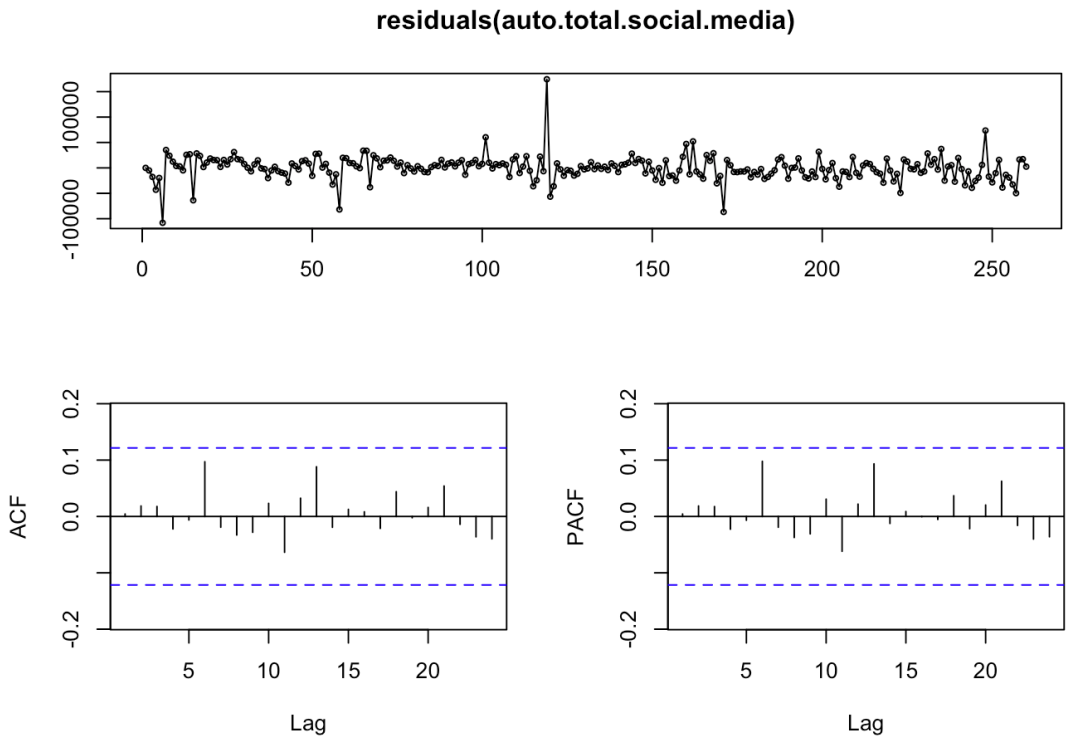
```
which.min(total.social_media)
```

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

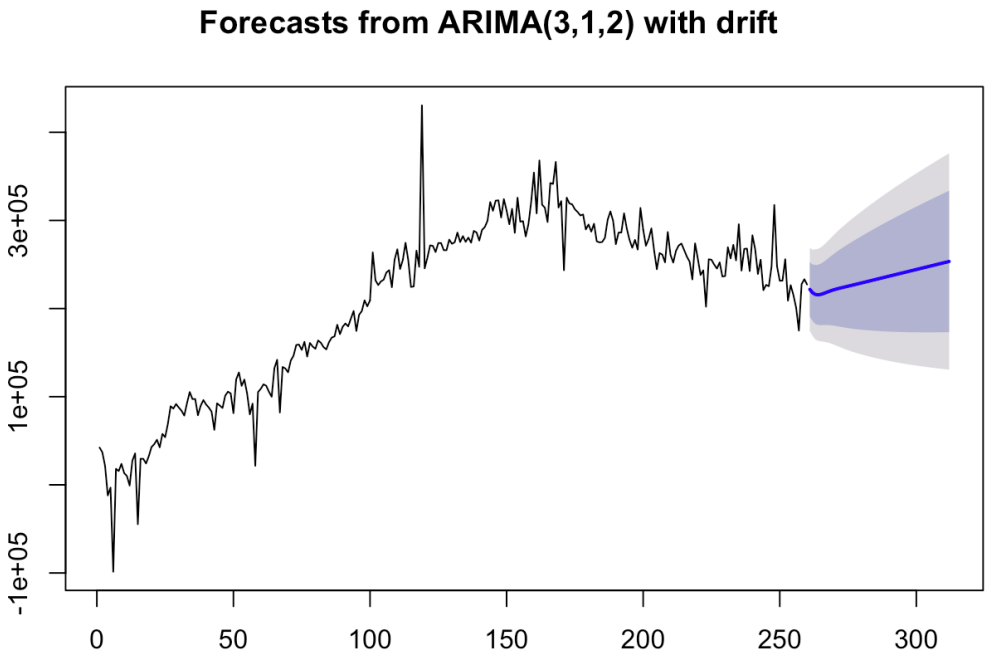
```
# Auto ARIMA (3,1,2)
auto.total.social.media <- auto.arima(cupcakes_data_social$Total.Social.Media)
auto.total.social.media
```

```
## Series: cupcakes_data_social$Total.Social.Media
## ARIMA(3,1,2) with drift
##
## Coefficients:
##          ar1      ar2      ar3      ma1     ma2     drift
##       0.9056  -0.0375  -0.1653  -1.6262  0.7288  759.2859
## s.e.  0.2370   0.0887   0.0905   0.2339  0.1580  517.7767
##
## sigma^2 estimated as 576172751:  log likelihood=-2968.19
## AIC=5950.38   AICc=5950.83   BIC=5975.28
```

```
tsdisplay(residuals(auto.total.social.media))
```



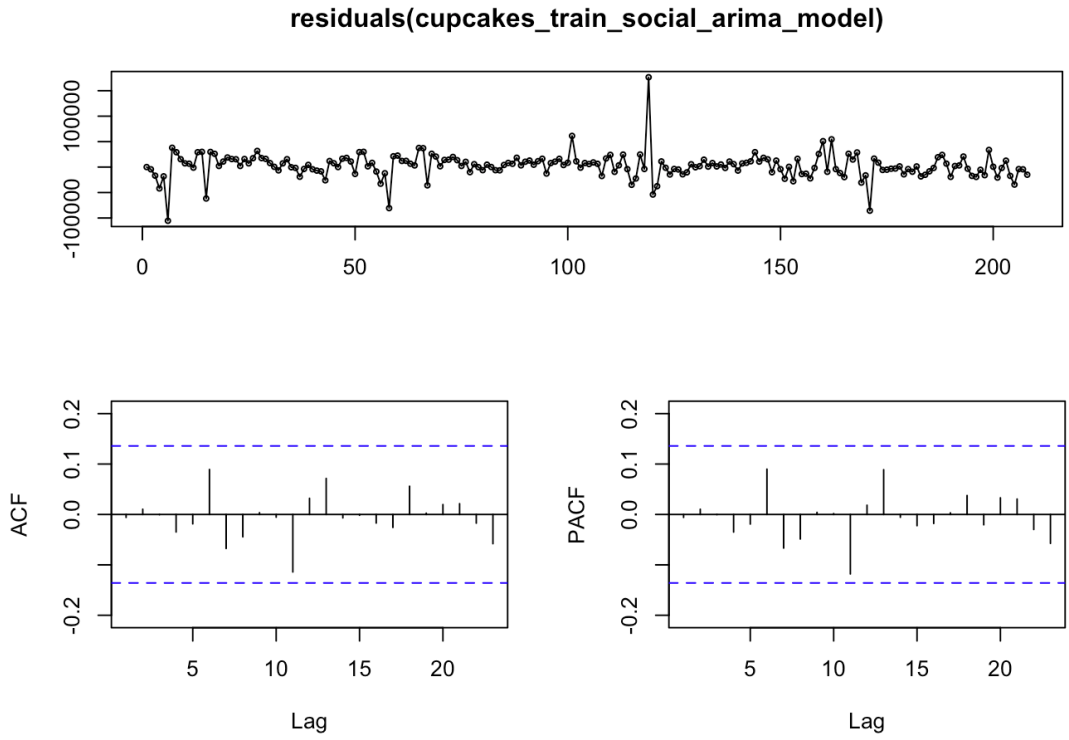

```
# Forecast Auto ARIMA
auto.total.social.media.forecast <- forecast(auto.total.social.media, h=52)
plot(auto.total.social.media.forecast)
```




```
# Train and Test Split
cupcakes_train_social_data_arima <- cupcakes_data_social_ts[1:208,8]
head(cupcakes_train_social_data_arima)
```

```
## [1]  42307.325  36919.169  21516.688 -12077.084  -3121.649 -98615.536
```

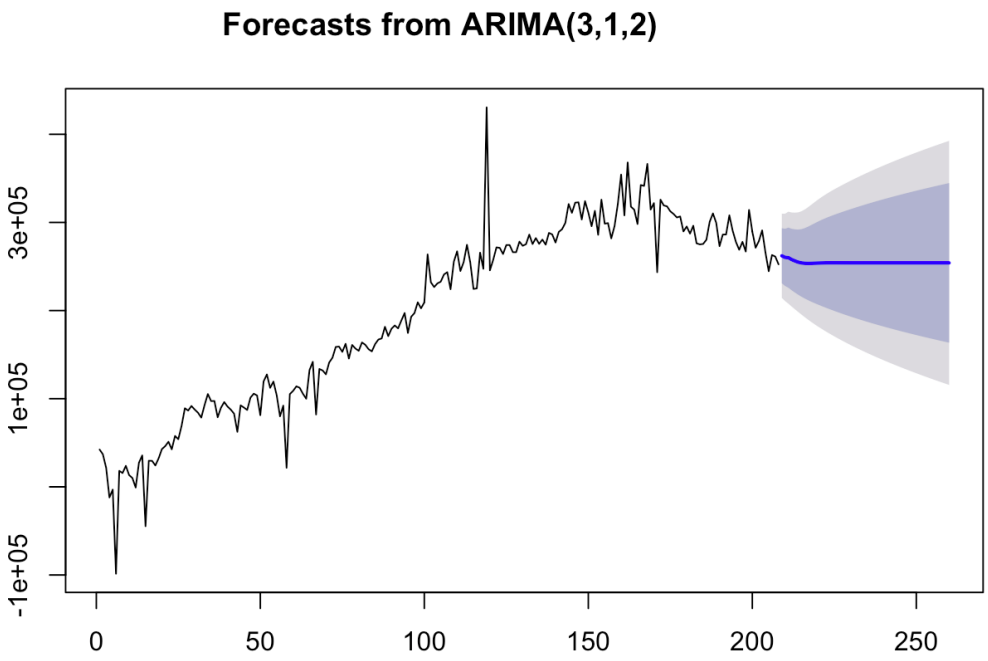
```
# Choosing Auto.arima - ARIMA(3,1,2,0,0,0)
cupcakes_train_social_arima_model <- Arima(cupcakes_train_social_data_arima, order=c(3,1,2))
tsdisplay(residuals(cupcakes_train_social_arima_model))
```




```
# Forecasting
forecast.cupcakes.social.arima <- forecast(cupcakes_train_social_arima_model, h=52)
forecast.cupcakes.social.arima$mean
```

```
## Time Series:
## Start = 209
## End = 260
## Frequency = 1
##  [1] 262131.1 260320.5 259988.0 257882.7 256263.6 254876.9 254030.9
##  [8] 253591.8 253472.5 253536.6 253684.6 253844.1 253976.8 254067.7
## [15] 254118.2 254137.3 254136.2 254124.9 254110.6 254097.8 254088.5
## [22] 254082.8 254080.3 254079.8 254080.6 254081.8 254083.0 254084.0
## [29] 254084.6 254084.9 254085.0 254085.0 254084.9 254084.8 254084.7
## [36] 254084.6 254084.6 254084.5 254084.5 254084.6 254084.6 254084.6
## [43] 254084.6 254084.6 254084.6 254084.6 254084.6 254084.6 254084.6
## [50] 254084.6 254084.6 254084.6
```

```
# Plot
plot(forecast(object=forecast.cupcakes.social.arima,h="52"))
```



Residuals look like white noise. The total social media mentions are of I(1), therefore the series is integrated.

ARIMA for cupcakes Sales Volume

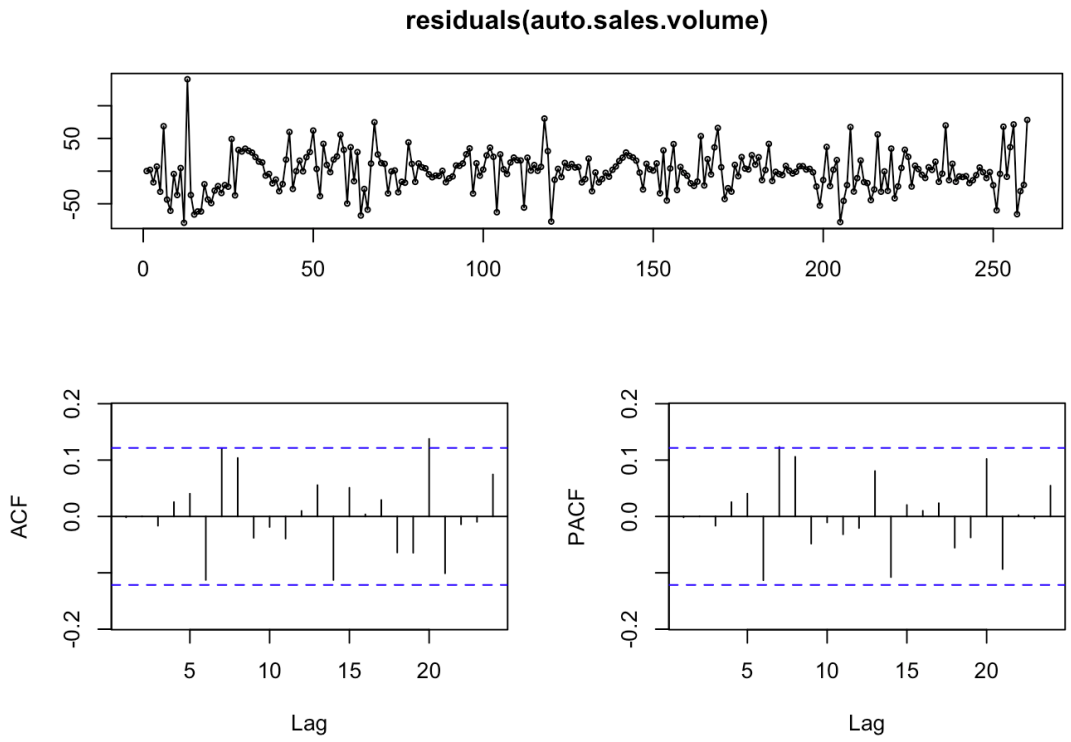
```
# cupcakess Sales Volume

f1<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,0,0,0,0,0)
f2<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,0,1,0,0,0)
f3<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,0,2,0,0,0)
f4<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,1,1,0,0,0)
f5<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,1,2,0,0,0)
f6<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,1,0,0,0,0)
f7<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,2,0,0,0,0)
f8<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,2,1,0,0,0)
f9<- fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 2,2,2,0,0,0)

f10<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,0,0,0,0,0)
f11<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,0,1,0,0,0)
f12<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,0,2,0,0,0)
f13<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,1,1,0,0,0)
f14<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,1,2,0,0,0)
f15<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,1,0,0,0,0)
f16<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,2,0,0,0,0)
f17<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,2,1,0,0,0)
f18<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 1,2,2,0,0,0)

f19<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,0,0,0,0,0)
f20<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,0,1,0,0,0)
f21<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,0,2,0,0,0)
f22<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,1,0,0,0,0)
f23<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,1,1,0,0,0)
f24<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,1,2,0,0,0)
f25<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,2,0,0,0,0)
f26<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,2,1,0,0,0)
f27<-fun.illustrate.2(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted,52, 0,2,2,0,0,0)

# Concatenate
total.sales.volume <- c(f1$Average, f2$Average, f3$Average, f4$Average, f5$Average, f6$Average, f7$Average,
f8$Average,f9$Average, f10$Average, f11$Average, f12$Average, f13$Average, f14$Average, f15$Average, f16$Ave
rage, f17$Average, f18$Average, f19$Average, f20$Average, f21$Average, f22$Average, f23$Average,
f24$Average, f25$Average, f26$Average, f27$Average)

# Minimum
summary(total.sales.volume)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1933    2070    2565   46780    4498  703400
```

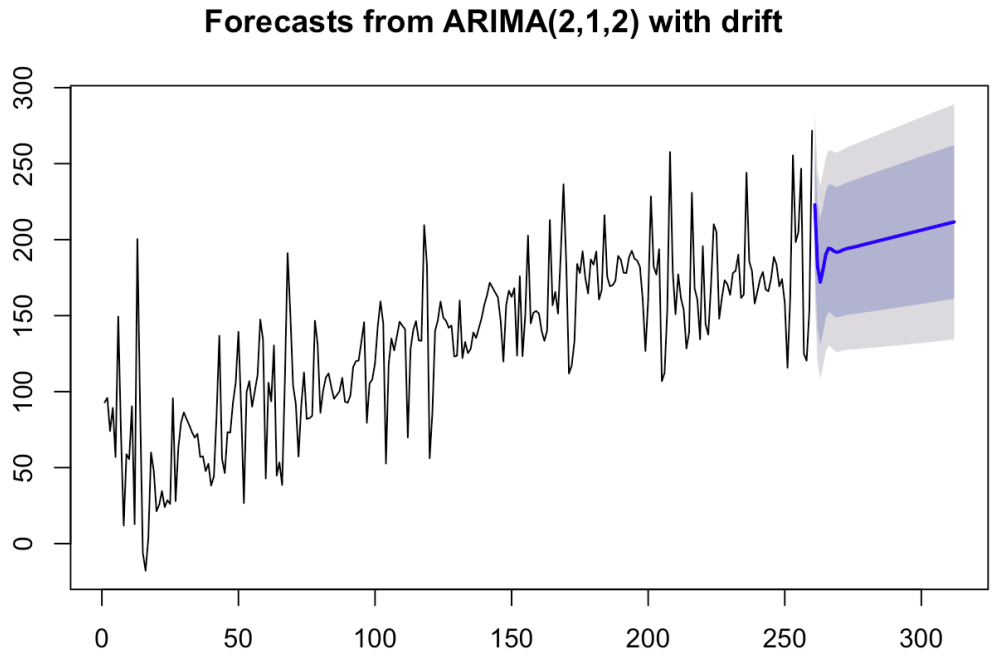
```
which.min(total.sales.volume)
```

```
## [1] 5
```

```
# Auto ARIMA (2,1,2)
auto.sales.volume <- auto.arima(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted)
tsdisplay(residuals(auto.sales.volume))
```



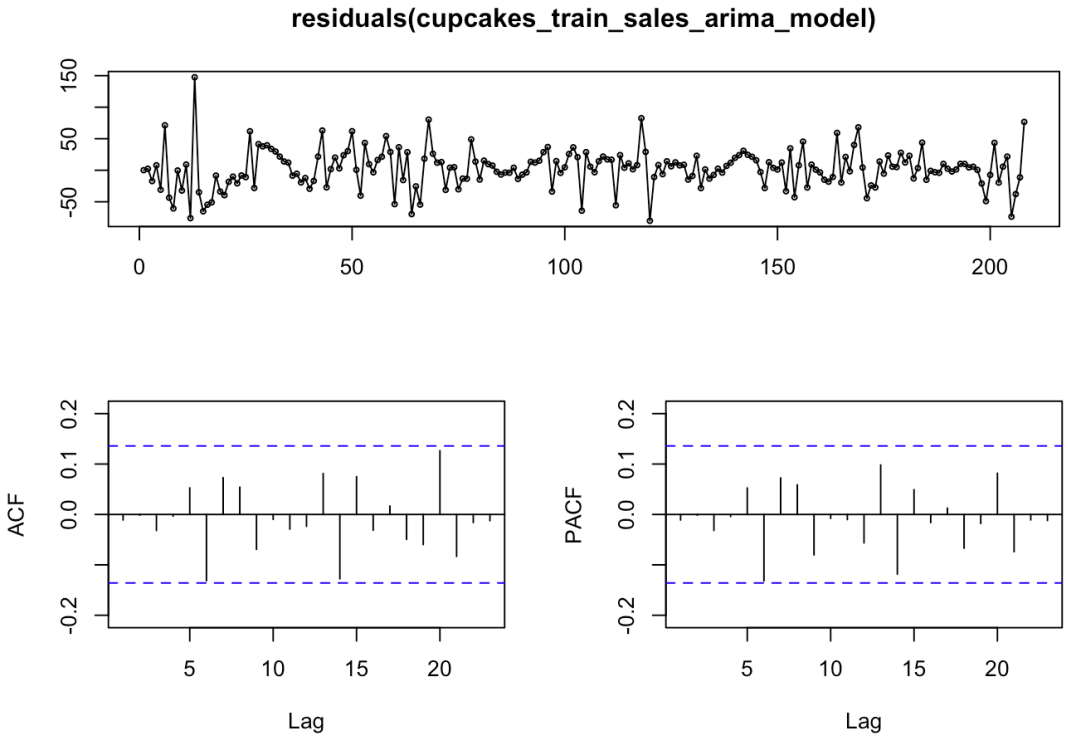

```
# Forecast Auto ARIMA
auto.sales.volume.forecast <- forecast(auto.sales.volume, h=52)
plot(auto.sales.volume.forecast)
```




```
# Train and Test Split
cupcakes_train_sales_data_arima <- cupcakes_data_sales_ts[1:208,2]
head(cupcakes_train_sales_data_arima)
```

```
## [1]  92.90408  95.91610  74.10600  89.33437  56.99062 149.38485
```

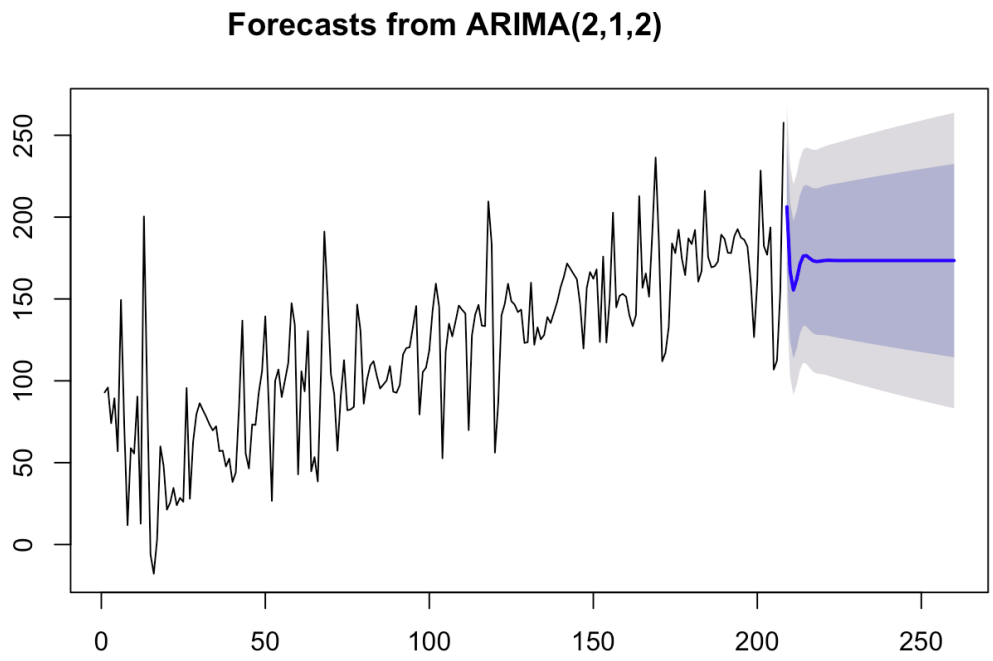
```
# Choosing Model 17 - ARIMA(2,1,2,0,0,0)
cupcakes_train_sales_arima_model <- Arima(cupcakes_train_sales_data_arima, order=c(2,1,2))
tsdisplay(residuals(cupcakes_train_sales_arima_model))
```




```
# Forecasting
forecast.cupcakes.sales.arima <- forecast(cupcakes_train_sales_arima_model, h=52)
forecast.cupcakes.sales.arima$mean
```

```
## Time Series:
## Start = 209
## End = 260
## Frequency = 1
##  [1] 206.2756 166.8020 155.5121 161.8652 171.2481 176.1993 176.4797
##  [8] 174.7860 173.3403 172.8539 173.0290 173.3554 173.5453 173.5691
## [15] 173.5144 173.4620 173.4418 173.4461 173.4573 173.4645 173.4659
## [22] 173.4641 173.4623 173.4614 173.4615 173.4619 173.4622 173.4622
## [29] 173.4622 173.4621 173.4621 173.4621 173.4621 173.4621 173.4621
## [36] 173.4621 173.4621 173.4621 173.4621 173.4621 173.4621 173.4621
## [43] 173.4621 173.4621 173.4621 173.4621 173.4621 173.4621 173.4621
## [50] 173.4621 173.4621 173.4621
```

```
# Plot
plot(forecast(object=forecast.cupcakes.sales.arima,h="52"))
```



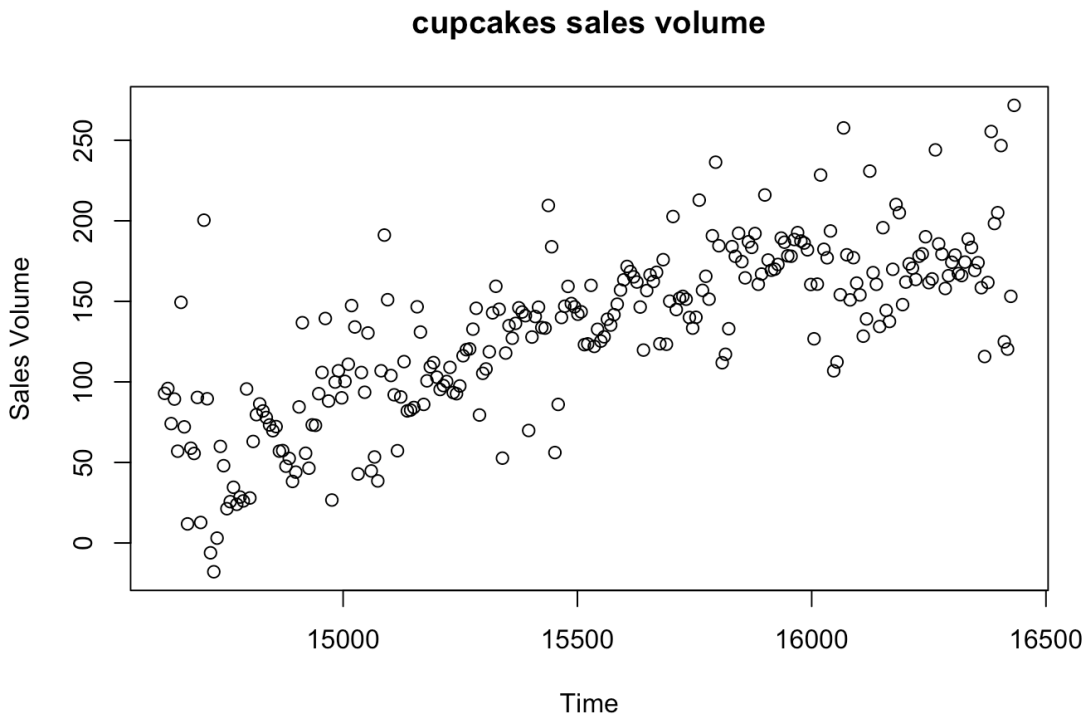
Both time series, cupcakes sales volume and social media are of I(1) OR > 1 process, therefore the series is not stationary and has a trend and drift, and is not showing a tendency to return back to mean.

Step 1: Check if the series is stationary.

```
sales.volume <- cupcakes_data_sales_ts$Sales.Seasonally.Adjusted
#d.sales.volume <- diff(sales.volume)
week <- cupcakes_data_sales_ts$Date


# Descriptive statistics and plotting the data
summary(sales.volume)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -17.86   94.85  140.00  131.50  168.10  271.70
```

```
#summary(d.sales.volume)

plot.ts(week, sales.volume, main = "cupcakes sales volume", xlab = "Time", ylab = "Sales Volume")
```




```
#Based on the cupcakes sales volume time series, there appears to be a linear trend.

#Therefore, for stationarity test we include a trend element in augmented dickey fuller test.

#Augmented Dickey Fuller test for stationarity



# Sales Volume
# Null hypothesis H0: Non - Stationary

library(urca)
summary(ur.df(y=sales.volume, lags = 52, type = "trend"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -68.552 -14.215  -0.161  12.796  60.018
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.38449   14.72259   2.132  0.03464 *
## z.lag.1      -0.24265    0.24622  -0.985  0.32595
## tt            0.06956    0.16009   0.435  0.66453
## z.diff.lag1  -0.43699    0.26087  -1.675  0.09597 .
## z.diff.lag2  -0.83428    0.26716  -3.123  0.00215 **
## z.diff.lag3  -0.91247    0.28246  -3.230  0.00151 **
## z.diff.lag4  -0.96346    0.29570  -3.258  0.00138 **
## z.diff.lag5  -0.94724    0.31591  -2.998  0.00317 **
## z.diff.lag6  -0.98136    0.33369  -2.941  0.00378 **
## z.diff.lag7  -0.82214    0.35359  -2.325  0.02139 *
## z.diff.lag8  -0.63406    0.37243  -1.703  0.09070 .
## z.diff.lag9  -0.54901    0.38486  -1.427  0.15577
## z.diff.lag10 -0.33252    0.39017  -0.852  0.39541
## z.diff.lag11 -0.23443    0.39178  -0.598  0.55049
## z.diff.lag12 -0.14615    0.38875  -0.376  0.70747
## z.diff.lag13 -0.01856    0.38028  -0.049  0.96113
## z.diff.lag14  0.03681    0.36800   0.100  0.92044
## z.diff.lag15  0.01055    0.35179   0.030  0.97612
## z.diff.lag16  0.09781    0.33602   0.291  0.77139
## z.diff.lag17  0.02416    0.31695   0.076  0.93933
## z.diff.lag18 -0.07447    0.30238  -0.246  0.80579
## z.diff.lag19 -0.14448    0.29081  -0.497  0.62003
## z.diff.lag20 -0.09697    0.28288  -0.343  0.73222
## z.diff.lag21 -0.31449    0.27669  -1.137  0.25749
## z.diff.lag22 -0.28426    0.27540  -1.032  0.30365
## z.diff.lag23 -0.31923    0.27594  -1.157  0.24913
## z.diff.lag24 -0.31101    0.27801  -1.119  0.26503
## z.diff.lag25 -0.20606    0.28309  -0.728  0.46780
## z.diff.lag26 -0.15298    0.28706  -0.533  0.59486
## z.diff.lag27 -0.11203    0.29181  -0.384  0.70158
## z.diff.lag28  0.03119    0.29616   0.105  0.91626
## z.diff.lag29  0.02998    0.29890   0.100  0.92025
## z.diff.lag30 -0.04420    0.29962  -0.148  0.88292
## z.diff.lag31  0.01092    0.29911   0.037  0.97092
## z.diff.lag32 -0.05690    0.29659  -0.192  0.84812
## z.diff.lag33 -0.11391    0.29130  -0.391  0.69631
## z.diff.lag34 -0.14877    0.28605  -0.520  0.60375
## z.diff.lag35 -0.16517    0.27815  -0.594  0.55351
## z.diff.lag36 -0.33534    0.27248  -1.231  0.22035
## z.diff.lag37 -0.26440    0.26857  -0.984  0.32644
## z.diff.lag38 -0.30884    0.26401  -1.170  0.24391
## z.diff.lag39 -0.28751    0.25845  -1.112  0.26771
## z.diff.lag40 -0.15919    0.24993  -0.637  0.52512
## z.diff.lag41 -0.24613    0.25042  -0.983  0.32724
## z.diff.lag42 -0.12422    0.24471  -0.508  0.61245
## z.diff.lag43 -0.14803    0.24006  -0.617  0.53839
## z.diff.lag44 -0.07267    0.23217  -0.313  0.75470
## z.diff.lag45  0.02961    0.22373   0.132  0.89490
```

```
## z.diff.lag46  0.10339    0.20836   0.496  0.62045
## z.diff.lag47  0.03022    0.18713   0.161  0.87193
## z.diff.lag48  0.18743    0.16803   1.115  0.26642
## z.diff.lag49  0.06902    0.14213   0.486  0.62793
## z.diff.lag50  0.18559    0.11629   1.596  0.11259
## z.diff.lag51  0.05942    0.08841   0.672  0.50254
## z.diff.lag52  0.22410    0.07198   3.113  0.00221 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.31 on 152 degrees of freedom
## Multiple R-squared:  0.6259, Adjusted R-squared:  0.493
## F-statistic:  4.71 on 54 and 152 DF,  p-value: 2.834e-14
##
##
## Value of test-statistic is: -0.9855 1.9627 1.6686
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.98 -3.42 -3.13
## phi2  6.15  4.71  4.05
## phi3  8.34  6.30  5.36
```

```
#adf.test(sales.volume, alternative = "stationary")

# KPSS test
# Null hypothesis: Series is stationary around a constant mean
# Alternative: Series is non stationary

kpss.test(sales.volume, null = "Level")
```

```
## Warning in kpss.test(sales.volume, null = "Level"): p-value smaller than
## printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  sales.volume
## KPSS Level = 5.5938, Truncation lag parameter = 3, p-value = 0.01
```

ADF Test: The test statistics values exceed the critical values. Therefore, the series is not stationary.

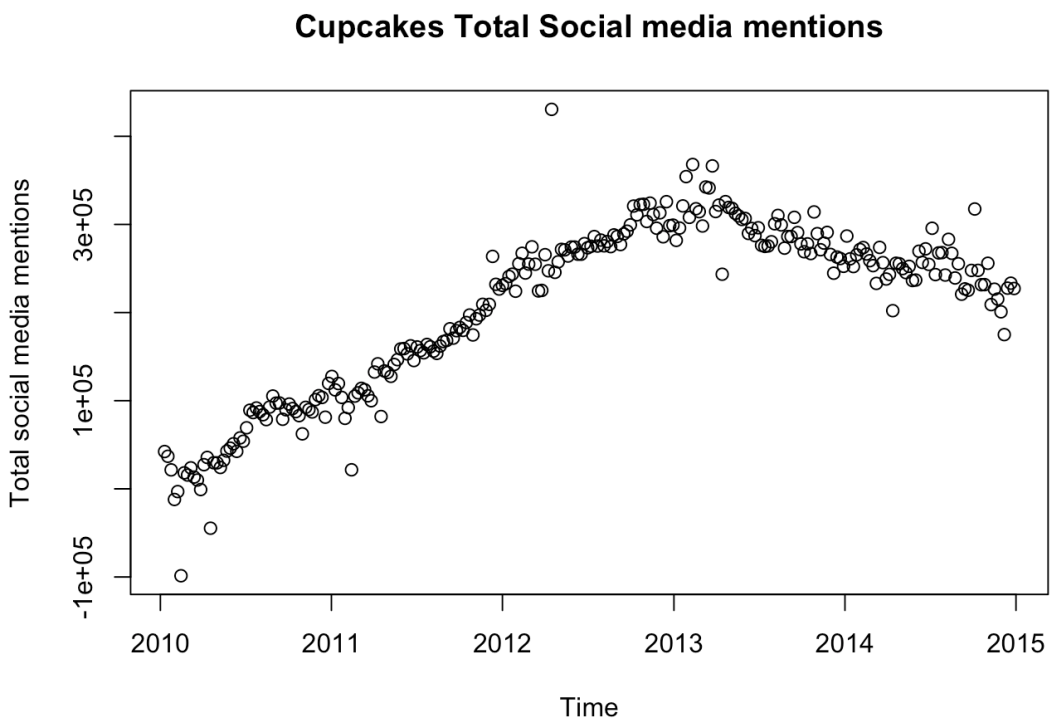
KPSS Test: At significance level of 5% or p-value > 0.05, we reject the 'Level' null hypothesis that the series is stationary. In other words, we have no evidence that the series is stationary.

:::::::::::::::::::: SOCIAL MEDIA MENTIONS ::::::::::::::::::::::::::

```
cupcakes_total_social_media <- cupcakes_data_social_ts$Total.Social.Media

#d.cupcakes_blogs <- diff(cupcakes_blogs)
week <- cupcakes_data_social_ts$Date


# Descriptive statistics and plotting the data
summary(cupcakes_total_social_media)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -98620  127600  243400  207200  277900  430500
```

```
plot(week, cupcakes_total_social_media, main = "Cupcakes Total Social media mentions", xlab = "Time", ylab =
 "Total social media mentions")
```




```
# Augmented Dickey Fuller test for stationarity
adf.test(cupcakes_total_social_media, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cupcakes_total_social_media
## Dickey-Fuller = -0.454, Lag order = 6, p-value = 0.9835
## alternative hypothesis: stationary
```

```
# KPSS test
# Null hypothesis: Series is stationary.
# Alternative: Series is non stationary

kpss.test(cupcakes_total_social_media, null = "Level")
```

```
## Warning in kpss.test(cupcakes_total_social_media, null = "Level"): p-value
## smaller than printed p-value
```

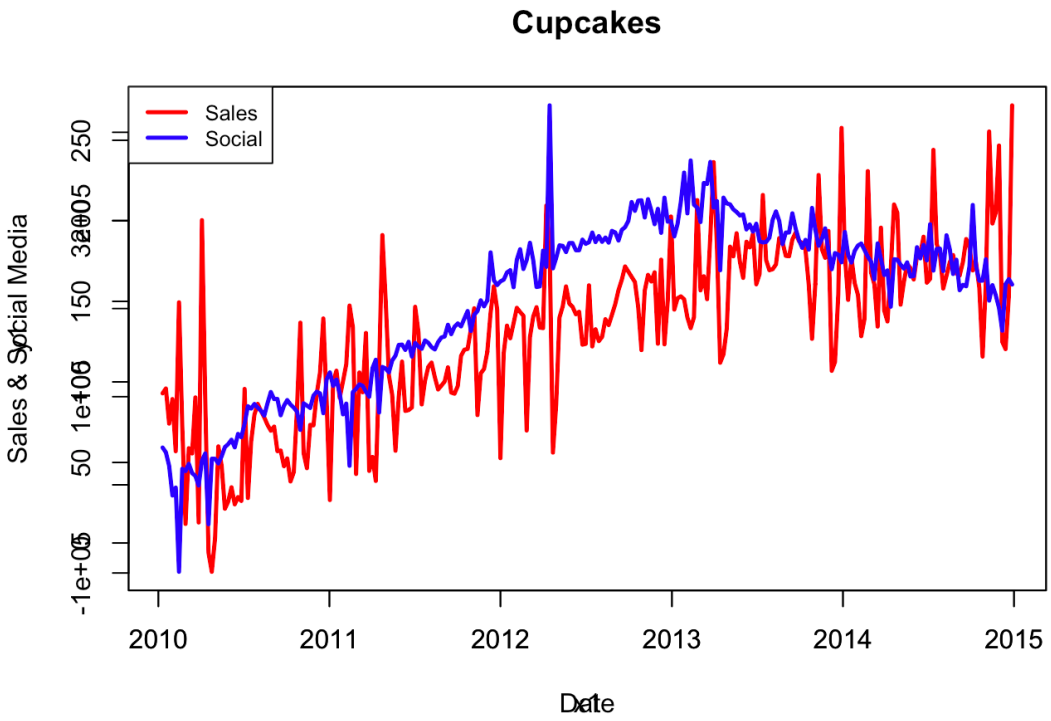
```
##
##  KPSS Test for Level Stationarity
##
## data:  cupcakes_total_social_media
## KPSS Level = 4.8255, Truncation lag parameter = 3, p-value = 0.01
```

```
# Low p-value suggests that the series is Non - Stationary. We reject the null hypothesis of stationarity.
```

Small p-value of 0.01 for KPSS test, less than significance level of 0.05 suggests that we reject the null hypothesis that the series is stationary.

Now that we have established both series are not stationary and has a trend or drift component, and are of I(1) or > process, we perform Johansen test for cointegration.

```
# Plot Sales and Sum Social Media Mentions

x1 <- cupcakes_data_sales_ts$Date
y1 <- sales.volume
y2 <- cupcakes_total_social_media

plot( x1, y1, type="l", col="red", main = "Cupcakes", xlab = "Date", lwd = "2.5")
par(new=TRUE)
plot( x1, y2, type="l", col="blue", ylab = "Sales & Social Media", lwd = "2.5")
legend("topleft", legend=c("Sales", "Social"), col=c("red", "blue"), lwd = 2.5, cex=0.8)
```




```
library("urca")

co.test.matrix <- cbind(sales.volume, cupcakes_total_social_media)

CoIntegrationTest =ca.jo(co.test.matrix,type="trace",K=6,ecdet="none", spec="longrun")
summary(CoIntegrationTest)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.09581302 0.04042481
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 | 10.48  6.50  8.18 11.65
## r = 0  | 36.06 15.66 17.95 23.52
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                                 sales.volume.l6
## sales.volume.l6                    1.0000000000
## cupcakes_total_social_media.l6    -0.0004089861
##                                 cupcakes_total_social_media.l6
## sales.volume.l6                                     1.000000000
## cupcakes_total_social_media.l6                      0.005773553
##
## Weights W:
## (This is the loading matrix)
##
##                                sales.volume.l6
## sales.volume.d                      -0.4832152
## cupcakes_total_social_media.d      -35.1484003
##                                cupcakes_total_social_media.l6
## sales.volume.d                                     0.001646439
## cupcakes_total_social_media.d                     -7.852846410
```

With lag of k=6, we see that our test statistic (r <= 1) of 10.48 is higher than at least one of # the critical values at 10% confidence level 6.50, we can assume there is cointegration of r time series.

http://denizstij.blogspot.com/2013/11/cointegration-tests-adf-and-johansen.html (http://denizstij.blogspot.com/2013/11/cointegration-tests-adf-and-johansen.html)

Running VAR model for Multivariate time series.

Multivariate time series analysis is used when one wants to model and explain the interactions and comovements among a group of time series variables

- http://faculty.washington.edu/ezivot/econ584/notes/multivariatetimeseries.pdf (http://faculty.washington.edu/ezivot/econ584/notes/multivariatetimeseries.pdf)

Granger Causality One of the main uses of VAR models is forecasting.

The following intuitive notion of a variable's forecasting ability is due to Granger (1969).

• If a variable, or group of variables, y1 (social media mentions) is found to be helpful for predicting another variable (sales volume), or group of variables, y2 then y1 is said to Granger-cause y2; otherwise it is said to fail to Granger-cause y2.

VAR Model Building and Evaluation steps:

1. Split Raw cupcakes sales and social data into train and validation (1 year).

2. Based on the # of observation split the ARIMA values into train and validation (1 year).
3. Run the VAR model on training set and forecast sales, social and measure the prediction accuracy by comparing the validation set.
4. Make plots of sales, social and arima (benchmark)
5. Run the model on validation set.
6. Make plots of sales, social and arima (benchmark)
7. Calculate the difference / lift / between sales arima forecasts and sales forecasts from var model.

```
library(vars)
```

```
## Loading required package: MASS
## Loading required package: strucchange
## Loading required package: sandwich
## Loading required package: lmtest
```

```
library(astsa)
```

```
##
## Attaching package: 'astsa'
##
## The following object is masked from 'package:forecast':
##
##     gas
```

```
# Read cupcakes Google Trends data in for exogenous variable in VAR model
#setwd("/Users/kevalshah/Keval_Backup/University/UChicago/Capstone/Data/Data Clean #up/Clean data to be used
 for Analysis")
#cupcakes_google_trends <- read.csv("Cupcakes_Google_Trends_Searches.csv")

# Plot sales, social media and google trends

x1 <- cupcakes_data_sales_ts$Date
y1 <- cupcakes_data_sales_ts$Sales.Seasonally.Adjusted
y2 <- cupcakes_data_social_ts$Total.Social.Media
#y3 <- cupcakes_google_trends$cupcakes.Searches

#plot( x1, y1, type="l", col="red", main = "cupcakess Sales, Social #Media and Google Trends", xlab = "Dat
e", lwd = "2.5")
#par(new=TRUE)
#plot( x1, y2, type="l", col="blue", ylab = "Social", lwd = "2.5")
#par(new=TRUE)
#plot( x1, y3, type="l", col="orange", ylab = "Social & GT", lwd = "2.5")
#legend("top", legend=c("Sales", "Social"), col=c("red", "blue"), lwd = #2.5, cex=0.8)


# Run VAR Model on Training set

length(cupcakes_data_social_ts$Total.Social.Media)
```

```
## [1] 260
```

```
length(cupcakes_data_sales_ts$Sales.Seasonally.Adjusted)
```

```
## [1] 260
```

```
length(cupcakes_data_sales_ts$Date)
```

```
## [1] 260
```

```
#length(cupcakes_google_trends$cupcakes.Searches)

Train_cupcakes_sales <- cupcakes_data_sales_ts[1:208,2]
Train_cupcakes_week <- cupcakes_data_sales_ts[1:208,1]
Train_cupcakes_social <- cupcakes_data_social_ts[1:208,8]
#Train_cupcakes_google_trends <- cupcakes_google_trends[1:208,3]

# Endogenous variables
Train_VAR_cupcakes <- cbind(Train_cupcakes_sales, Train_cupcakes_social)

#VAR Select
#VARselect(Train_VAR_cupcakes, lag.max = 10, type = "both", exogen = cbind(x3 #=Train_cupcakes_google_trend
s))

VARselect(Train_VAR_cupcakes, lag.max = 10, type = "both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      7      4      2      7
##
## $criteria
##                   1            2            3            4            5
## AIC(n) 2.730897e+01 2.703831e+01 2.697300e+01 2.694539e+01 2.696205e+01
## HQ(n)  2.736275e+01 2.711898e+01 2.708055e+01 2.707983e+01 2.712338e+01
## SC(n)  2.744183e+01 2.723760e+01 2.723872e+01 2.727754e+01 2.736063e+01
## FPE(n) 7.246710e+11 5.528468e+11 5.179199e+11 5.038590e+11 5.123864e+11
##                   6            7            8            9           10
## AIC(n) 2.692218e+01 2.690933e+01 2.692329e+01 2.693598e+01 2.695998e+01
## HQ(n)  2.711040e+01 2.712444e+01 2.716529e+01 2.720486e+01 2.725575e+01
## SC(n)  2.738718e+01 2.744077e+01 2.752116e+01 2.760027e+01 2.769071e+01
## FPE(n) 4.924454e+11 4.862732e+11 4.932587e+11 4.997433e+11 5.121196e+11
```

```
Train_VAR_model_cupcakes <- VAR(Train_VAR_cupcakes, p=7, type="both")
summary(Train_VAR_model_cupcakes)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: Train_cupcakes_sales, Train_cupcakes_social
## Deterministic variables: both
## Sample size: 201
## Log Likelihood: -3240.266
## Roots of the characteristic polynomial:
## 0.9997 0.8379 0.8379 0.8149 0.8149 0.8033 0.8033 0.7977 0.7727 0.7727 0.7495 0.7495 0.723 0.003
## Call:
## VAR(y = Train_VAR_cupcakes, p = 7, type = "both")
##
##
## Estimation results for equation Train_cupcakes_sales:
## =====================================================
## Train_cupcakes_sales = Train_cupcakes_sales.l1 + Train_cupcakes_social.l1 + Train_cupcakes_sales.l2 + Tra
in_cupcakes_social.l2 + Train_cupcakes_sales.l3 + Train_cupcakes_social.l3 + Train_cupcakes_sales.l4 + Train
_cupcakes_social.l4 + Train_cupcakes_sales.l5 + Train_cupcakes_social.l5 + Train_cupcakes_sales.l6 + Train_c
upcakes_social.l6 + Train_cupcakes_sales.l7 + Train_cupcakes_social.l7 + const + trend
##
##                             Estimate Std. Error t value Pr(>|t|)
## Train_cupcakes_sales.l1   3.231e-01  7.251e-02   4.456 1.44e-05 ***
## Train_cupcakes_social.l1 -8.352e-05  9.269e-05  -0.901 0.368749
## Train_cupcakes_sales.l2  -2.559e-01  7.362e-02  -3.475 0.000635 ***
## Train_cupcakes_social.l2  2.406e-04  8.789e-05   2.738 0.006785 **
## Train_cupcakes_sales.l3  -1.076e-01  7.722e-02  -1.393 0.165183
## Train_cupcakes_social.l3 -2.220e-04  9.112e-05  -2.436 0.015787 *
## Train_cupcakes_sales.l4   1.674e-02  7.836e-02   0.214 0.831087
## Train_cupcakes_social.l4  8.206e-05  9.263e-05   0.886 0.376824
## Train_cupcakes_sales.l5   3.801e-02  7.686e-02   0.495 0.621534
## Train_cupcakes_social.l5  7.810e-05  9.256e-05   0.844 0.399897
## Train_cupcakes_sales.l6  -1.121e-01  7.392e-02  -1.517 0.131088
## Train_cupcakes_social.l6  1.232e-04  8.968e-05   1.373 0.171279
## Train_cupcakes_sales.l7   1.905e-01  7.045e-02   2.704 0.007489 **
## Train_cupcakes_social.l7 -1.190e-04  8.937e-05  -1.332 0.184564
## const                     3.973e+01  9.108e+00   4.362 2.14e-05 ***
## trend                     4.762e-01  1.513e-01   3.146 0.001928 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 28.41 on 185 degrees of freedom
## Multiple R-Squared: 0.7215,  Adjusted R-squared: 0.6989
## F-statistic: 31.95 on 15 and 185 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Train_cupcakes_social:
## ======================================================
## Train_cupcakes_social = Train_cupcakes_sales.l1 + Train_cupcakes_social.l1 + Train_cupcakes_sales.l2 + Tr
ain_cupcakes_social.l2 + Train_cupcakes_sales.l3 + Train_cupcakes_social.l3 + Train_cupcakes_sales.l4 + Trai
n_cupcakes_social.l4 + Train_cupcakes_sales.l5 + Train_cupcakes_social.l5 + Train_cupcakes_sales.l6 + Train_
cupcakes_social.l6 + Train_cupcakes_sales.l7 + Train_cupcakes_social.l7 + const + trend
##
##                             Estimate Std. Error t value Pr(>|t|)
## Train_cupcakes_sales.l1   8.165e+01  5.730e+01   1.425 0.155870
## Train_cupcakes_social.l1  2.680e-01  7.325e-02   3.659 0.000330 ***
## Train_cupcakes_sales.l2  -1.561e+02  5.818e+01  -2.684 0.007946 **
## Train_cupcakes_social.l2  2.177e-01  6.946e-02   3.134 0.002004 **
## Train_cupcakes_sales.l3   1.556e+02  6.103e+01   2.550 0.011594 *
## Train_cupcakes_social.l3  3.211e-02  7.201e-02   0.446 0.656195
## Train_cupcakes_sales.l4   3.420e+00  6.193e+01   0.055 0.956013
## Train_cupcakes_social.l4  1.480e-01  7.321e-02   2.021 0.044697 *
## Train_cupcakes_sales.l5  -4.866e+01  6.074e+01  -0.801 0.424127
## Train_cupcakes_social.l5  3.878e-02  7.315e-02   0.530 0.596618
## Train_cupcakes_sales.l6   5.599e+01  5.842e+01   0.959 0.339060
```

```
## Train_cupcakes_social.l6  2.525e-01  7.087e-02   3.563 0.000466 ***
## Train_cupcakes_sales.l7  -6.365e+01  5.567e+01  -1.143 0.254350
## Train_cupcakes_social.l7  3.886e-02  7.063e-02   0.550 0.582858
## const                     1.189e+04  7.198e+03   1.652 0.100204
## trend                    -9.219e+01  1.196e+02  -0.771 0.441826
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 22450 on 185 degrees of freedom
## Multiple R-Squared: 0.9532,  Adjusted R-squared: 0.9494
## F-statistic: 251.4 on 15 and 185 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                       Train_cupcakes_sales Train_cupcakes_social
## Train_cupcakes_sales                 807.2                 12152
## Train_cupcakes_social              12152.3             504137565
##
## Correlation matrix of residuals:
##                       Train_cupcakes_sales Train_cupcakes_social
## Train_cupcakes_sales               1.00000               0.01905
## Train_cupcakes_social              0.01905               1.00000
```

The adjusted R-Squared of 69% for equation predicting sales as dependent variable and endogenous variables of social media and sales with lag order of 7 and with constant and trend deterministic variable indicates a good fit.

On the other hand, the inverse, of predicting social media with sales as predictors has Adj. R-Squared of 94% which indicates that sales is a good lagging indicator of social media.

Now, we fit our training model on our validation set and check the prediction error / accuracy.

```
# Run VAR Model on Validation / Test Set

Test_cupcakes_sales <- cupcakes_data_sales_ts[209:260,2]
Test_cupcakes_week <- cupcakes_data_sales_ts[209:260,1]
Test_cupcakes_social <- cupcakes_data_social_ts[209:260,8]
#Test_cupcakes_google_trends <- cupcakes_google_trends[209:260,3]

length(Test_cupcakes_sales)
```

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

```
length(Test_cupcakes_week)
```

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

```
length(Test_cupcakes_social)
```

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

```
#length(Test_cupcakes_google_trends)

# Endogenous variables
Test_VAR_cupcakes <- cbind(Test_cupcakes_sales, Test_cupcakes_social)
```

```
# cupcakes prediction

#var_train_forecasts <- predict(Train_VAR_model_cupcakes, n.ahead = 52, ci = 0.95, dumvar = #cbind(x3 =Test_
cupcakes_google_trends))

var_train_forecasts <- predict(Train_VAR_model_cupcakes, n.ahead = 52, ci = 0.95)



summary(var_train_forecasts)
```

```
##          Length Class  Mode
## fcst       2    -none- list
## endog    416    -none- numeric
## model     10    varest list
## exo.fcst   0    -none- NULL
```

```
head(var_train_forecasts)
```

```
## $fcst
## $fcst$Train_cupcakes_sales
##           fcst    lower    upper       CI
##  [1,] 221.9178 166.2320 277.6036 55.68581
##  [2,] 166.6991 108.0846 225.3136 58.61449
##  [3,] 167.5273 107.6828 227.3718 59.84452
##  [4,] 171.6834 110.5094 232.8575 61.17405
##  [5,] 179.0979 117.6853 240.5106 61.41264
##  [6,] 175.5768 113.6656 237.4880 61.91121
##  [7,] 191.8087 129.3309 254.2865 62.47777
##  [8,] 198.3002 135.5359 261.0646 62.76432
##  [9,] 186.8872 123.8706 249.9037 63.01652
## [10,] 181.5634 118.5449 244.5818 63.01844
## [11,] 179.9631 116.8613 243.0650 63.10182
## [12,] 186.1872 122.9269 249.4474 63.26021
## [13,] 186.0893 122.7622 249.4164 63.32709
## [14,] 187.1124 123.7509 250.4739 63.36146
## [15,] 190.0563 126.6684 253.4442 63.38789
## [16,] 189.6873 126.2901 253.0845 63.39719
## [17,] 188.4062 125.0017 251.8107 63.40452
## [18,] 186.8019 123.3820 250.2217 63.41989
## [19,] 188.0851 124.6199 251.5502 63.46515
## [20,] 189.4567 125.9704 252.9430 63.48630
## [21,] 189.9561 126.4523 253.4599 63.50378
## [22,] 190.7346 127.2146 254.2546 63.51998
## [23,] 190.9615 127.4280 254.4950 63.53349
## [24,] 191.1920 127.6459 254.7381 63.54611
## [25,] 190.9367 127.3751 254.4983 63.56162
## [26,] 191.1923 127.6123 254.7723 63.58002
## [27,] 191.9002 128.3014 255.4990 63.59878
## [28,] 192.4607 128.8460 256.0753 63.61461
## [29,] 192.9208 129.2904 256.5512 63.63037
## [30,] 193.1755 129.5300 256.8209 63.64549
## [31,] 193.4836 129.8227 257.1445 63.66090
## [32,] 193.7325 130.0562 257.4087 63.67623
## [33,] 193.9997 130.3074 257.6919 63.69226
## [34,] 194.4038 130.6951 258.1125 63.70868
## [35,] 194.8136 131.0888 258.5384 63.72478
## [36,] 195.2109 131.4704 258.9514 63.74049
## [37,] 195.5210 131.7650 259.2771 63.75603
## [38,] 195.8244 132.0529 259.5960 63.77156
## [39,] 196.1365 132.3493 259.9237 63.78720
## [40,] 196.4473 132.6444 260.2502 63.80289
## [41,] 196.7819 132.9631 260.6006 63.81874
## [42,] 197.1200 133.2855 260.9546 63.83453
## [43,] 197.4611 133.6109 261.3114 63.85025
## [44,] 197.7809 133.9151 261.6467 63.86583
## [45,] 198.0862 134.2048 261.9676 63.88140
## [46,] 198.3929 134.4959 262.2899 63.89700
## [47,] 198.6996 134.7870 262.6122 63.91262
## [48,] 199.0111 135.0829 262.9394 63.92825
## [49,] 199.3208 135.3769 263.2647 63.94388
## [50,] 199.6297 135.6702 263.5892 63.95947
## [51,] 199.9336 135.9585 263.9086 63.97503
## [52,] 200.2305 136.2399 264.2211 63.99056
##
## $fcst$Train_cupcakes_social
##           fcst     lower    upper        CI
##  [1,] 260123.6 216116.51 304130.7  44007.09
##  [2,] 249081.4 203272.47 294890.2  45808.88
##  [3,] 252669.8 204852.77 300486.8  47817.03
##  [4,] 261088.3 212283.73 309892.9  48804.61
##  [5,] 253203.6 203320.23 303086.9  49883.34
##  [6,] 254092.0 203159.31 305024.6  50932.67
##  [7,] 247791.5 193619.45 301963.6  54172.08
```

```
##  [8,] 247032.9 191179.09 302886.6  55853.77
##  [9,] 245863.4 188442.97 303283.8  57420.43
## [10,] 247568.7 188880.76 306256.5  58687.89
## [11,] 247827.7 187732.25 307923.1  60095.41
## [12,] 245187.8 183924.00 306451.6  61263.81
## [13,] 243448.1 180581.52 306314.8  62866.62
## [14,] 240732.5 176570.00 304895.1  64162.54
## [15,] 239497.3 173990.82 305003.8  65506.48
## [16,] 239024.5 172266.10 305783.0  66758.43
## [17,] 237836.9 169812.00 305861.9  68024.93
## [18,] 237027.8 167799.12 306256.4  69228.64
## [19,] 235185.2 164725.49 305645.0  70459.75
## [20,] 233588.7 161959.85 305217.5  71628.84
## [21,] 231805.3 159012.80 304597.8  72792.51
## [22,] 230370.3 156439.04 304301.6  73931.30
## [23,] 229065.0 153991.86 304138.2  75073.17
## [24,] 227603.8 151423.73 303783.9  76180.09
## [25,] 226180.7 148895.72 303465.7  77285.01
## [26,] 224542.8 146186.18 302899.5  78356.65
## [27,] 222893.9 143476.86 302311.0  79417.07
## [28,] 221238.8 140777.82 301699.7  80460.96
## [29,] 219564.9 138067.73 301062.1  81497.17
## [30,] 217965.5 135446.08 300485.0  82519.47
## [31,] 216300.7 132770.62 299830.8  83530.09
## [32,] 214637.7 130112.63 299162.7  84525.05
## [33,] 212903.6 127396.32 298410.9  85507.29
## [34,] 211143.3 124666.58 297620.0  86476.71
## [35,] 209361.8 121925.22 296798.4  87436.57
## [36,] 207556.5 119171.21 295941.8  88385.27
## [37,] 205753.6 116428.90 295078.4  89324.75
## [38,] 203925.5 113672.42 294178.5  90253.04
## [39,] 202077.5 110906.19 293248.8  91171.33
## [40,] 200199.5 108120.11 292278.8  92079.36
## [41,] 198290.7 105312.56 291268.9  92978.16
## [42,] 196362.7 102494.83 290230.6  93867.90
## [43,] 194410.3  99661.31 289159.4  94749.04
## [44,] 192442.9  96821.26 288064.5  95621.61
## [45,] 190453.2  93967.34 286939.0  96485.82
## [46,] 188441.8  91100.09 285783.6  97341.73
## [47,] 186406.4  88216.74 284596.1  98189.69
## [48,] 184345.9  85316.08 283375.7  99029.81
## [49,] 182263.5  82400.99 282126.0  99862.48
## [50,] 180158.5  79470.73 280846.3 100687.80
## [51,] 178032.8  76526.83 279538.8 101506.00
## [52,] 175885.4  73568.26 278202.6 102317.18
##
##
## $endog
##        Train_cupcakes_sales Train_cupcakes_social
##   [1,]            92.904077            42307.3248
##   [2,]            95.916097            36919.1686
##   [3,]            74.106000            21516.6878
##   [4,]            89.334366           -12077.0838
##   [5,]            56.990616            -3121.6487
##   [6,]           149.384847           -98615.5357
##   [7,]            72.089173            18151.3345
##   [8,]            11.843981            15646.2071
##   [9,]            58.834366            23804.0027
##  [10,]            55.574750            13308.6950
##  [11,]            90.368020            10003.6518
##  [12,]            12.738212             -812.7617
##  [13,]           200.392058            27387.9715
##  [14,]            89.521866            35647.1758
##  [15,]            -6.100730           -44679.3362
##  [16,]           -17.860346            29584.4859
```

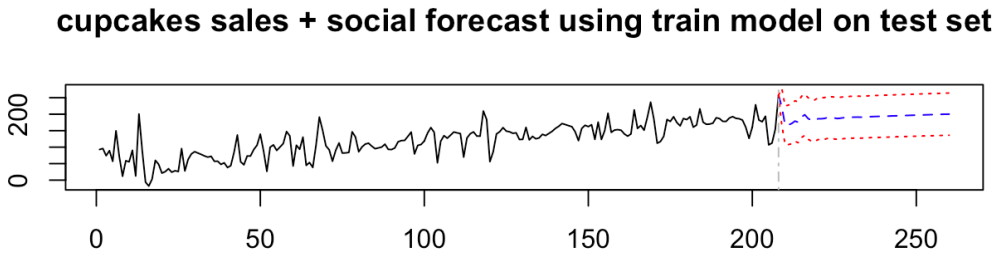
```
##  [17,]            3.017058           29481.7647
##  [18,]           59.971385           24234.6806
##  [19,]           47.978597           32375.8056
##  [20,]           21.281481           42823.1422
##  [21,]           25.649270           46157.7647
##  [22,]           34.586770           51145.2479
##  [23,]           24.002635           42530.6205
##  [24,]           28.502635           57755.4691
##  [25,]           26.113212           54014.7936
##  [26,]           95.608404           69257.2287
##  [27,]           27.937731           89182.0123
##  [28,]           63.026673           86524.5292
##  [29,]           79.656481           91718.7215
##  [30,]           86.327154           87666.0436
##  [31,]           81.983404           84077.6013
##  [32,]           77.916097           78540.6470
##  [33,]           73.276673           92727.0316
##  [34,]           69.714173          105294.9330
##  [35,]           72.252635           97255.5580
##  [36,]           57.017058           97419.8176
##  [37,]           57.375231           78925.7455
##  [38,]           47.670904           89701.4907
##  [39,]           52.476193           96170.9619
##  [40,]           38.206962           91087.0172
##  [41,]           43.995423           87575.4595
##  [42,]           84.466577           83019.6037
##  [43,]          136.750231           62371.4426
##  [44,]           55.697347           92462.5340
##  [45,]           46.432923           89954.3633
##  [46,]           73.322347           87205.8489
##  [47,]           73.036289          101230.1157
##  [48,]           92.694943          105698.2335
##  [49,]          105.875231          103690.8849
##  [50,]          139.358404           81267.6422
##  [51,]           88.115616          119589.9715
##  [52,]           26.668500          127516.9330
##  [53,]           99.904077          112287.3248
##  [54,]          106.916097          119447.1686
##  [55,]           90.106000          103728.6878
##  [56,]          100.334366           79977.9162
##  [57,]          110.990616           92141.3513
##  [58,]          147.384847           21612.4643
##  [59,]          134.089173          105200.3345
##  [60,]           42.843981          108917.2071
##  [61,]          105.834366          114018.0027
##  [62,]           93.574750          112391.6950
##  [63,]          130.368020          105543.6518
##  [64,]           44.738212           99944.2383
##  [65,]           53.392058          132509.9715
##  [66,]           38.521866          141855.1758
##  [67,]          106.899270           82041.6638
##  [68,]          191.139654          133746.4859
##  [69,]          151.017058          132025.7647
##  [70,]          103.971385          127630.6806
##  [71,]           91.978597          141082.8056
##  [72,]           57.281481          146524.1422
##  [73,]           90.649270          158731.7647
##  [74,]          112.586770          159119.2479
##  [75,]           82.002635          153115.6205
##  [76,]           82.502635          162259.4691
##  [77,]           84.113212          145530.7936
##  [78,]          146.608404          160985.2287
##  [79,]          130.937731          156899.0123
##  [80,]           86.026673          154335.5292
##  [81,]          100.656481          163799.7215
```

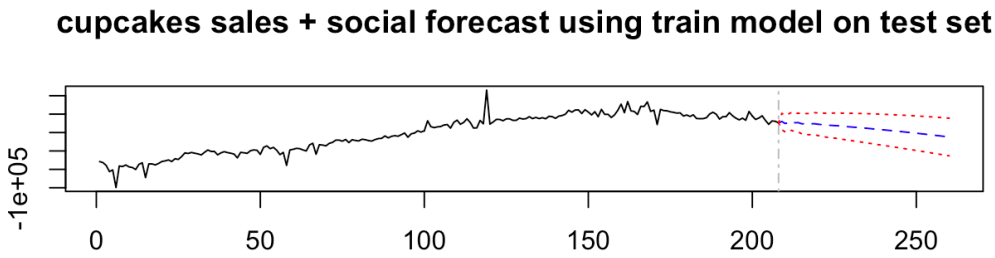
```
##  [82,]          109.327154          161187.0436
##  [83,]          111.983404          156195.6013
##  [84,]          102.916097          153650.6470
##  [85,]           95.276673          161961.0316
##  [86,]           97.714173          167125.9330
##  [87,]          100.252635          168343.5580
##  [88,]          109.017058          181598.8176
##  [89,]           93.375231          170962.7455
##  [90,]           92.670904          179385.4907
##  [91,]           97.476193          183097.9619
##  [92,]          116.206962          179775.0172
##  [93,]          119.995423          188675.4595
##  [94,]          120.466577          197199.6037
##  [95,]          132.750231          174569.4426
##  [96,]          145.697347          192953.5340
##  [97,]           79.432923          197139.3633
##  [98,]          105.322347          209331.8489
##  [99,]          108.036289          202511.1157
## [100,]          118.694943          209247.2335
## [101,]          142.875231          263695.8849
## [102,]          159.358404          232052.6422
## [103,]          145.115616          226656.9715
## [104,]           52.668500          230807.9330
## [105,]          117.904077          232761.3248
## [106,]          134.916097          240959.1686
## [107,]          127.106000          243538.6878
## [108,]          136.334366          224187.9162
## [109,]          145.990616          255457.3513
## [110,]          143.384847          267227.4643
## [111,]          141.089173          244847.3345
## [112,]           69.843981          254981.2071
## [113,]          127.834366          274585.0027
## [114,]          140.574750          254820.6950
## [115,]          146.368020          224481.6518
## [116,]          133.738212          225119.2383
## [117,]          133.392058          265657.9715
## [118,]          209.521866          247377.1758
## [119,]          183.899270          430520.6638
## [120,]           56.139654          245613.4859
## [121,]           86.017058          257548.7647
## [122,]          139.971385          271694.6806
## [123,]          146.978597          271101.8056
## [124,]          159.281481          264112.1422
## [125,]          148.649270          274307.7647
## [126,]          146.586770          274331.2479
## [127,]          142.002635          266198.6205
## [128,]          143.502635          266189.4691
## [129,]          123.113212          278178.7936
## [130,]          123.608404          273463.2287
## [131,]          159.937731          275247.0123
## [132,]          122.026673          286203.5292
## [133,]          132.656481          275436.7215
## [134,]          125.327154          282210.0436
## [135,]          127.983404          275691.6013
## [136,]          138.916097          280514.6470
## [137,]          135.276673          274768.0316
## [138,]          141.714173          288098.9330
## [139,]          148.252635          286337.5580
## [140,]          157.017058          277160.8176
## [141,]          163.375231          289208.7455
## [142,]          171.670904          292311.4907
## [143,]          168.476193          299485.9619
## [144,]          165.206962          320827.0172
## [145,]          161.995423          310860.4595
## [146,]          146.466577          322406.6037
```

```
## [147,]          119.750231          322820.4426
## [148,]          156.697347          303322.5340
## [149,]          166.432923          324124.3633
## [150,]          162.322347          311170.8489
## [151,]          168.036289          295697.1157
## [152,]          123.694943          313107.2335
## [153,]          175.875231          285933.8849
## [154,]          123.358404          325787.6422
## [155,]          150.115616          298572.9715
## [156,]          202.668500          299202.9330
## [157,]          144.904077          281859.3248
## [158,]          151.916097          295877.1686
## [159,]          153.106000          320981.6878
## [160,]          151.334366          354226.9162
## [161,]          139.990616          307986.3513
## [162,]          133.384847          367996.4643
## [163,]          140.089173          317778.3345
## [164,]          212.843981          314447.2071
## [165,]          156.834366          298179.0027
## [166,]          165.574750          342433.6950
## [167,]          151.368020          341378.6518
## [168,]          190.738212          366363.2383
## [169,]          236.392058          314597.9715
## [170,]          184.521866          321912.1758
## [171,]          111.899270          243478.6638
## [172,]          117.139654          325834.4859
## [173,]          133.017058          319257.7647
## [174,]          183.971385          318201.6806
## [175,]          177.978597          312570.8056
## [176,]          192.281481          309558.1422
## [177,]          174.649270          305617.7647
## [178,]          164.586770          306748.2479
## [179,]          187.002635          289752.6205
## [180,]          183.502635          295373.4691
## [181,]          192.113212          287277.7936
## [182,]          160.608404          296203.2287
## [183,]          166.937731          276286.0123
## [184,]          216.026673          275055.5292
## [185,]          175.656481          275535.7215
## [186,]          169.327154          280314.0436
## [187,]          169.983404          300721.6013
## [188,]          172.916097          310201.6470
## [189,]          189.276673          299419.0316
## [190,]          186.714173          273080.9330
## [191,]          178.252635          286208.5580
## [192,]          178.017058          286284.8176
## [193,]          188.375231          308017.7455
## [194,]          192.670904          290729.4907
## [195,]          187.476193          277873.9619
## [196,]          186.206962          268962.0172
## [197,]          181.995423          277909.4595
## [198,]          160.466577          266944.6037
## [199,]          126.750231          314157.4426
## [200,]          160.697347          289521.5340
## [201,]          228.432923          271259.3633
## [202,]          182.322347          278831.8489
## [203,]          177.036289          290946.1157
## [204,]          193.694943          265946.2335
## [205,]          106.875231          244646.8849
## [206,]          112.358404          262957.6422
## [207,]          154.115616          261258.9715
## [208,]          257.668500          252431.9330
##
## $model
##
```

```
## VAR Estimation Results:
## ========================
##
## Estimated coefficients for equation Train_cupcakes_sales:
## =========================================================
## Call:
## Train_cupcakes_sales = Train_cupcakes_sales.l1 + Train_cupcakes_social.l1 + Train_cupcakes_sales.l2 + Tra
in_cupcakes_social.l2 + Train_cupcakes_sales.l3 + Train_cupcakes_social.l3 + Train_cupcakes_sales.l4 + Train
_cupcakes_social.l4 + Train_cupcakes_sales.l5 + Train_cupcakes_social.l5 + Train_cupcakes_sales.l6 + Train_c
upcakes_social.l6 + Train_cupcakes_sales.l7 + Train_cupcakes_social.l7 + const + trend
##
##  Train_cupcakes_sales.l1 Train_cupcakes_social.l1  Train_cupcakes_sales.l2
##             3.231226e-01            -8.351825e-05            -2.558688e-01
## Train_cupcakes_social.l2  Train_cupcakes_sales.l3 Train_cupcakes_social.l3
##             2.406405e-04            -1.076005e-01            -2.220006e-04
##  Train_cupcakes_sales.l4 Train_cupcakes_social.l4  Train_cupcakes_sales.l5
##             1.673850e-02             8.206413e-05             3.800949e-02
## Train_cupcakes_social.l5  Train_cupcakes_sales.l6 Train_cupcakes_social.l6
##             7.809908e-05            -1.121056e-01             1.231669e-04
##  Train_cupcakes_sales.l7 Train_cupcakes_social.l7                    const
##             1.904861e-01            -1.190271e-04             3.972651e+01
##                    trend
##             4.761644e-01
##
##
## Estimated coefficients for equation Train_cupcakes_social:
## ==========================================================
## Call:
## Train_cupcakes_social = Train_cupcakes_sales.l1 + Train_cupcakes_social.l1 + Train_cupcakes_sales.l2 + Tr
ain_cupcakes_social.l2 + Train_cupcakes_sales.l3 + Train_cupcakes_social.l3 + Train_cupcakes_sales.l4 + Trai
n_cupcakes_social.l4 + Train_cupcakes_sales.l5 + Train_cupcakes_social.l5 + Train_cupcakes_sales.l6 + Train_
cupcakes_social.l6 + Train_cupcakes_sales.l7 + Train_cupcakes_social.l7 + const + trend
##
##  Train_cupcakes_sales.l1 Train_cupcakes_social.l1  Train_cupcakes_sales.l2
##             8.165144e+01             2.680163e-01            -1.561288e+02
## Train_cupcakes_social.l2  Train_cupcakes_sales.l3 Train_cupcakes_social.l3
##             2.176975e-01             1.556005e+02             3.211005e-02
##  Train_cupcakes_sales.l4 Train_cupcakes_social.l4  Train_cupcakes_sales.l5
##             3.420333e+00             1.479669e-01            -4.865859e+01
## Train_cupcakes_social.l5  Train_cupcakes_sales.l6 Train_cupcakes_social.l6
##             3.878271e-02             5.599425e+01             2.525117e-01
##  Train_cupcakes_sales.l7 Train_cupcakes_social.l7                    const
##            -6.365415e+01             3.885894e-02             1.189158e+04
##                    trend
##            -9.218784e+01
##
##
##
## $exo.fcst
## NULL
```

```
plot(var_train_forecasts, type = "l", main = "cupcakes sales + social forecast using train model on test se
t")
```






```
# Check accuracy of our forecasts using train model on test data

# cupcakes sales volume forecast
accuracy(var_train_forecasts$fcst$Train_cupcakes_sales[,1], Test_cupcakes_sales)
```

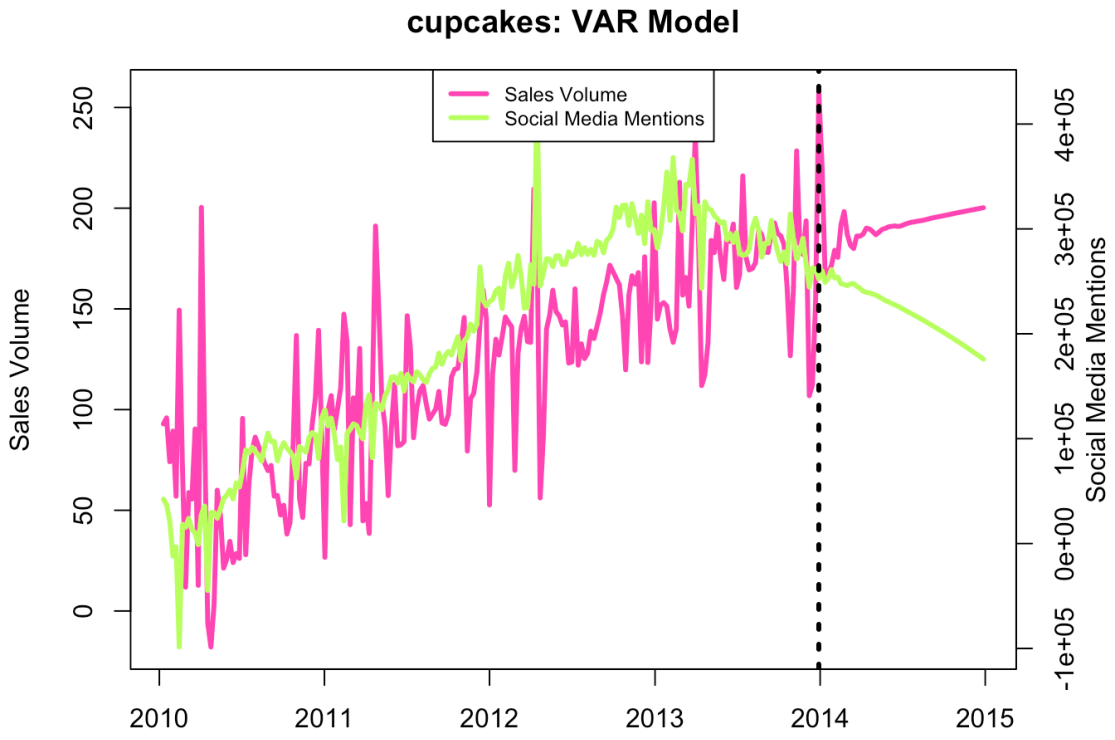
```
##                  ME     RMSE      MAE       MPE     MAPE
## Test set -17.37229 35.65068 29.79953 -13.31054 18.48432
```

```
# cupcakes social media forecast
accuracy(var_train_forecasts$fcst$Train_cupcakes_social[,1], Test_cupcakes_social)
```

```
##                ME    RMSE      MAE      MPE     MAPE
## Test set 26985.46 36730.7 29487.94 10.45574 11.66127
```

Plot Raw Sales and Social media with Forecasts

```
# Append raw + forecast
cupcakes.sales.VAR.All <- append(Train_cupcakes_sales, var_train_forecasts$fcst$Train_cupcakes_sales[,1])
cupcakes.social.VAR.All <- append(Train_cupcakes_social, var_train_forecasts$fcst$Train_cupcakes_social[,1])
cupcakes.week.All <- append(Train_cupcakes_week, Test_cupcakes_week)

cupcakes_df_total <- data.frame(cupcakes.week.All, cupcakes.sales.VAR.All, cupcakes.social.VAR.All)


mar.default <- c(3,3,3,3) + 0.1
par(mar = mar.default + c(0, 1, 0, 0))
plot(cupcakes_df_total[,1:2], type="l",
     ylab="Sales Volume", xlab="Time (Year)",
     lwd=3, main="cupcakes: VAR Model", col="hotpink")
par(new=TRUE)
plot(cupcakes_df_total[,3], type="l", col="darkolivegreen1", axes=FALSE,
     ylab="", xlab="", lwd=3)
axis(4)
mtext("Social Media Mentions", side=4, line=+2, adj=0.5)
abline(v=208, lty=3, lwd=3)
legend("top", legend=c("Sales Volume", "Social Media Mentions"),
       col=c("hotpink","darkolivegreen1"), lwd=3, cex=0.75)
```



Comparing sales prediction to ARIMA benchmark, make plots and calculate lift

```
# Calculate accuracy of arima sales forecast and VAR model sales forecast to actual sales values in
# test set
# Compare prediction error of each and calculate the lift obtained from prediction error.


# Difference in Predictions cupcakess

All_cupcakes_Forecasts <- cbind.data.frame(Test_week = as.Date(Test_cupcakes_week), AR=forecast.cupcakes.sal
es.arima$mean, VAR=var_train_forecasts$fcst$Train_cupcakes_sales[,1],
                   ACTUAL=Test_cupcakes_sales)
head(All_cupcakes_Forecasts)
```

```
##    Test_week       AR      VAR   ACTUAL
## 1 2014-01-05 206.2756 221.9178 178.9041
## 2 2014-01-12 166.8020 166.6991 150.9161
## 3 2014-01-19 155.5121 167.5273 177.1060
## 4 2014-01-26 161.8652 171.6834 161.3344
## 5 2014-02-02 171.2481 179.0979 153.9906
## 6 2014-02-09 176.1993 175.5768 128.3848
```

```
# Calculate the RMSE. Predicted - Actual values - ARIMA.
AR.error <- forecast.cupcakes.sales.arima$mean - Test_cupcakes_sales
ar.cupcakes.sales.rmse <- sqrt(mean(AR.error^2))
# Calculate MAE
ar.cupcakes.sales.mae <- mean(abs(AR.error))


# Calculate the RMSE. Predicted - Actual values - VAR.
VAR.error <- var_train_forecasts$fcst$Train_cupcakes_sales[,1] - Test_cupcakes_sales
var.cupcakes.sales.rmse <- sqrt(mean(VAR.error^2))
# Calculate MAE
var.cupcakes.sales.mae <- mean(abs(VAR.error))


# Calculate Lift in prediction accuracy
paste(round((((ar.cupcakes.sales.rmse - var.cupcakes.sales.rmse)/ar.cupcakes.sales.rmse))*100, digits = 2),
"%", sep = "")
```

```
## [1] "-9.33%"
```

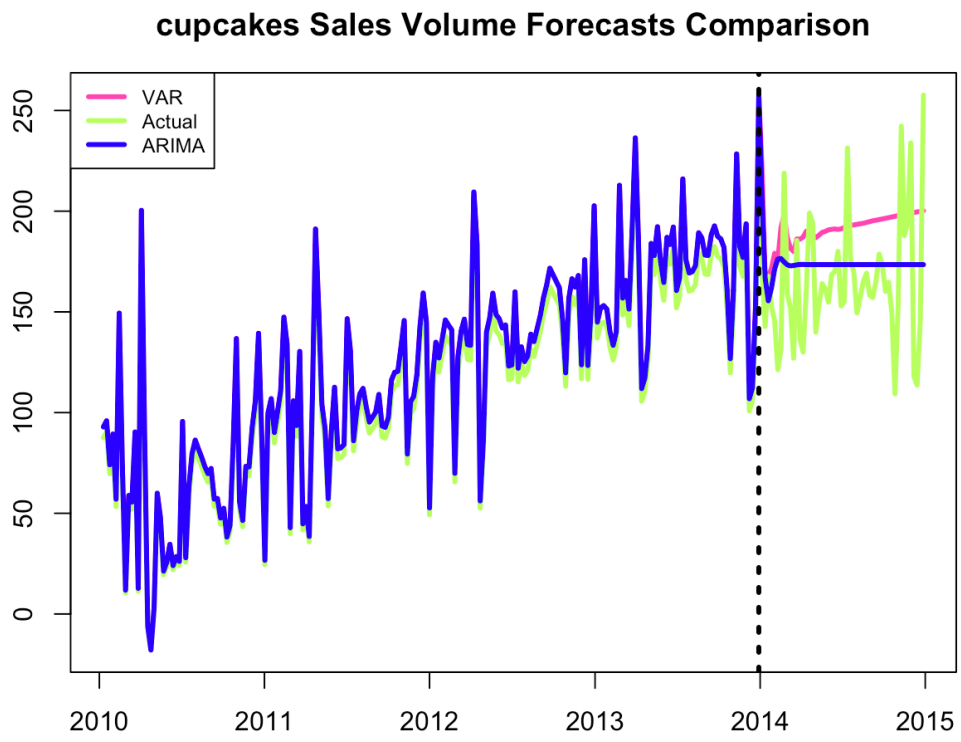
```
paste(round((((ar.cupcakes.sales.mae - var.cupcakes.sales.mae)/ar.cupcakes.sales.mae))*100, digits = 2),
"%", sep = "")
```

```
## [1] "-27.34%"
```

```
# Create a table to compare RMSE and MAE
accuracy_table <- matrix(c(32.60731,35.65068,"-9.33%",23.40172,29.79953,"-27.34%"),ncol=3,byrow=TRUE)
colnames(accuracy_table) <- c("ARIMA","VAR", "Lift in Prediction accuracy")
rownames(accuracy_table) <- c("RMSE","MAE")
accuracy_table
```

```
##      ARIMA      VAR        Lift in Prediction accuracy
## RMSE "32.60731" "35.65068" "-9.33%"
## MAE  "23.40172" "29.79953" "-27.34%"
```

```
# Append ARIMA sales data and forecasts
cupcakes.sales.arima.All <- append(cupcakes_train_sales_data_arima, forecast.cupcakes.sales.arima$mean)

cupcakes.sales.actual.All <- append(Train_cupcakes_sales, Test_cupcakes_sales)

# Create a dataframe w Raw sales data, VAR Forecasts, ARIMA Forecasts and
# Test data.

cupcakes_df_total_final <- cbind.data.frame(cupcakes.week.All, cupcakes.sales.VAR.All, cupcakes.sales.actua
l.All, cupcakes.sales.arima.All)


mar.default <- c(3,3,3,3) + 0.1
par(mar = mar.default + c(0, 1, 0, 0))
plot(cupcakes_df_total_final[,1:2], type="l",
     ylab="", xlab="Time (Year)",
     lwd=3, main="cupcakes Sales Volume Forecasts Comparison", col="hotpink")
par(new=TRUE)
# Actual data train + test
plot(cupcakes_df_total_final[,3], type="l", col="darkolivegreen1", axes=FALSE,
     ylab="", xlab="", lwd=3)
par(new=TRUE)
# ARIMA Forecast
plot(cupcakes_df_total_final[,4], type="l", col="blue", axes=FALSE,
     ylab="", xlab="", lwd=3)

abline(v=208, lty=3, lwd=3)
legend("topleft", legend=c("VAR", "Actual", "ARIMA"),
       col=c("hotpink","darkolivegreen1","blue"), lwd=3, cex=0.75)
```



Based on the above plot we can see that ARIMA model predicts better than VAR model. This is in contrast to clementine where VAR model indeed had better accuracy rates.

```
print(accuracy_table)
```

```
##      ARIMA      VAR        Lift in Prediction accuracy
## RMSE "32.60731" "35.65068" "-9.33%"
## MAE  "23.40172" "29.79953" "-27.34%"
```

Predictions for 2015

```
var_cupcakes_forecasts_2015 <- predict(Train_VAR_model_cupcakes, n.ahead = 104, ci = 0.95)

plot(var_cupcakes_forecasts_2015, type = "l", main = "2015 Cupcakes sales + social forecast using train mode
l on test set")
```