VAR_Model_Clem_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")

options(java.parameters = "-Xmx1000m")

clem_data_social <- read.xlsx("clem_social_seasonally_adjusted_v1_average_weekly.xlsx", sheetName = "Seasona
lly Adjusted data")


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

clem_data_social_ts <- clem_data_social[1:260,]
head(clem_data_social_ts)
```

```
##   Analysis_Date clemBlogSeasonallyAdjusted clemFacebookSeasonallyAdjusted
## 1    2010-01-03                  581.6909                     -74.357581
## 2    2010-01-10                  942.6203                    -140.438351
## 3    2010-01-17                  505.8102                    -158.056139
## 4    2010-01-24                  527.6058                     -45.693158
## 5    2010-01-31                  569.3438                     -14.529697
## 6    2010-02-07                  629.9015                       4.919822
##   clemTwitterSeasonallyAdjusted clemCommentsSeasonallyAdjusted
## 1                     -1464.1786                      -3.356490
## 2                     -2532.8262                      -2.658894
## 3                     -1907.9080                     -12.057933
## 4                     -1636.2469                     -21.456971
## 5                     -1177.8623                      -2.800721
## 6                      -524.3407                      -5.286298
##   clemReviewsSeasonallyAdjusted clemForumsSeasonallyAdjusted
## 1                     -22.712518                   -148.30772
## 2                      -5.244249                   -297.93560
## 3                       2.087482                   -196.14474
## 4                      -2.549538                   -106.06060
## 5                       5.231712                     96.99228
## 6                       4.123539                    256.67978
##   Total.social.media
## 1         -1131.2220
## 2         -2036.4831
## 3         -1766.2691
## 4         -1284.4013
## 5          -523.6249
## 6           365.9977
```

```
# Read seasonally adjusted Clementine sales volume data

clem_data_sales <- read.xlsx("clem_social_seasonally_adjusted_v1_average_weekly.xlsx", sheetName = "Clem Sal
es Volume")

myvars = c("Date", "salestszSeasonallyAdjusted")

clem_data_sales_ts <- clem_data_sales[myvars]
head(clem_data_sales_ts)
```

```
##         Date salestszSeasonallyAdjusted
## 1 2010-01-09                   2077991.2
## 2 2010-01-16                   1320663.6
## 3 2010-01-23                   1909651.8
## 4 2010-01-30                    689723.6
## 5 2010-02-06                   1220800.4
## 6 2010-02-13                    979856.2
```

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 Clementine Social Media Mentions

```
# Clementines Social Media Mentions

f1<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,0,0,0,0,0)
f2<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,0,1,0,0,0)
f3<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,0,2,0,0,0)
f4<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,1,1,0,0,0)
f5<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,1,2,0,0,0)
f6<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,1,0,0,0,0)
f7<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,2,0,0,0,0)
f8<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,2,1,0,0,0)
f9<- fun.illustrate.2(clem_data_social$Total.social.media,52, 2,2,2,0,0,0)

f10<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,0,0,0,0,0)
f11<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,0,1,0,0,0)
f12<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,0,2,0,0,0)
f13<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,1,1,0,0,0)
f14<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,1,2,0,0,0)
f15<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,1,0,0,0,0)
f16<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,2,0,0,0,0)
f17<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,2,1,0,0,0)
f18<-fun.illustrate.2(clem_data_social$Total.social.media,52, 1,2,2,0,0,0)

f19<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,0,0,0,0,0)
f20<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,0,1,0,0,0)
f21<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,0,2,0,0,0)
f22<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,1,0,0,0,0)
f23<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,1,1,0,0,0)
f24<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,1,2,0,0,0)
f25<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,2,0,0,0,0)
f26<-fun.illustrate.2(clem_data_social$Total.social.media,52, 0,2,1,0,0,0)
f27<-fun.illustrate.2(clem_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.
## 5.122e+06 5.631e+06 6.400e+06 7.654e+07 7.189e+06 1.134e+09
```

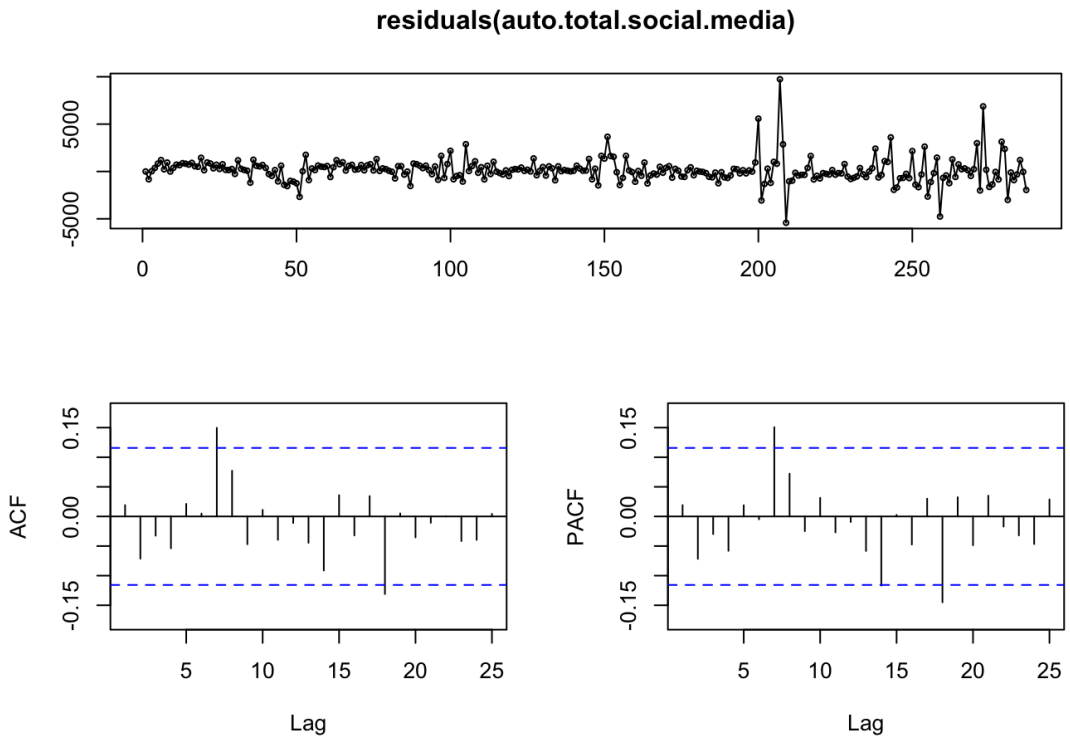
```
which.min(total.social_media)
```

```
## [1] 13
```

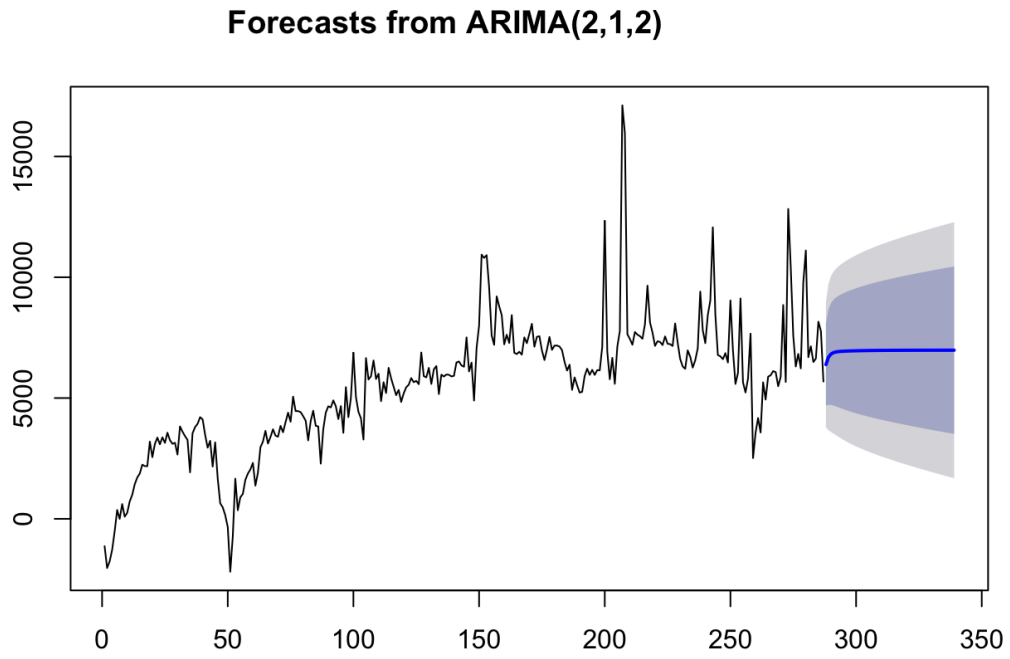
```
# Auto ARIMA (2,1,2)
auto.total.social.media <- auto.arima(clem_data_social$Total.social.media)
auto.total.social.media
```

```
## Series: clem_data_social$Total.social.media
## ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.3254  -0.3817  -1.7204  0.7322
## s.e.  0.1317   0.0966   0.1138  0.1051
##
## sigma^2 estimated as 1737494:  log likelihood=-2460.74
## AIC=4931.47   AICc=4931.69   BIC=4949.75
```

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



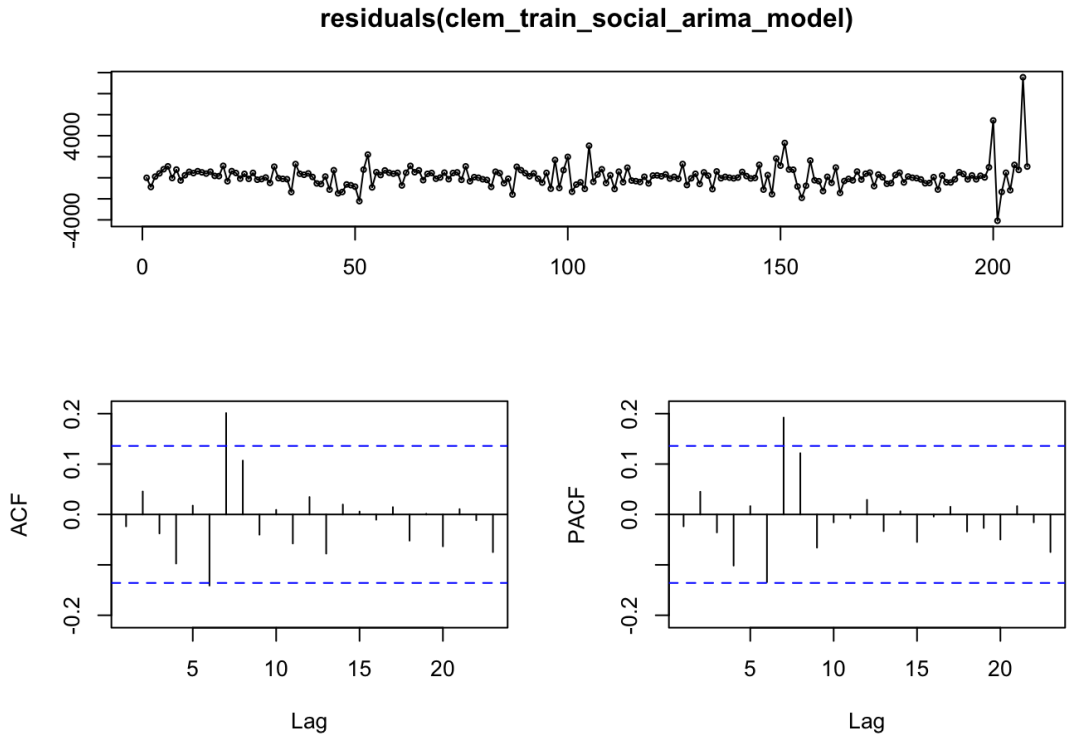

```
# Forecast Auto ARIMA
auto.total.social.media.forecast <- forecast(auto.total.social.media, h=52)
plot(auto.total.social.media.forecast)
```




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

```
## [1] -1131.2220 -2036.4831 -1766.2691 -1284.4013  -523.6249   365.9977
```

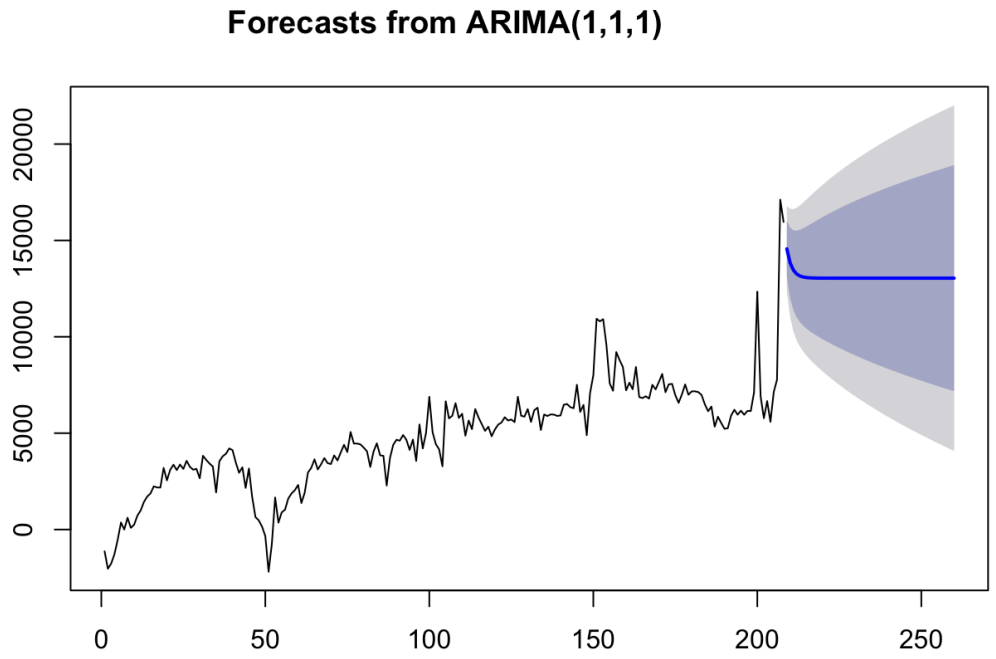
```
# Choosing Model 13 - ARIMA(1,1,1,0,0,0)
clem_train_social_arima_model <- Arima(clem_train_social_data_arima, order=c(1,1,1))
tsdisplay(residuals(clem_train_social_arima_model))
```




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

```
## Time Series:
## Start = 209
## End = 260
## Frequency = 1
##  [1] 14564.48 13837.09 13457.46 13259.34 13155.94 13101.97 13073.81
##  [8] 13059.11 13051.44 13047.43 13045.34 13044.25 13043.68 13043.39
## [15] 13043.23 13043.15 13043.11 13043.09 13043.07 13043.07 13043.07
## [22] 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06
## [29] 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06
## [36] 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06
## [43] 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06 13043.06
## [50] 13043.06 13043.06 13043.06
```

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



ARIMA for Clementine Sales Volume

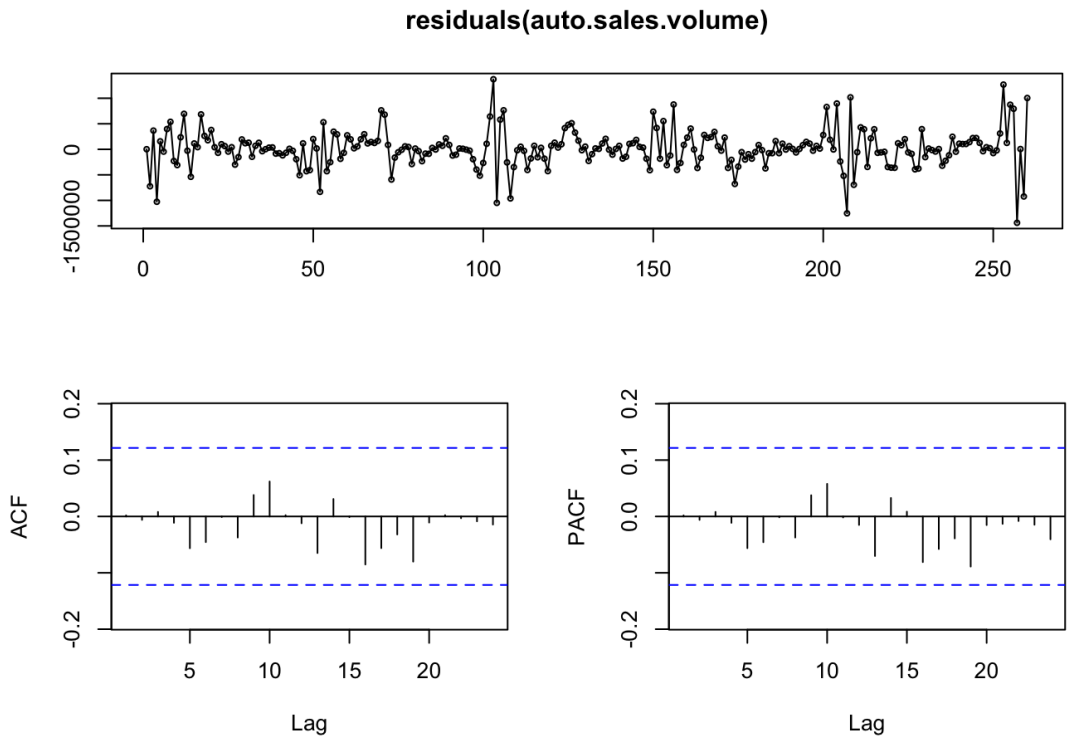
```
# Clementines Sales Volume

f1<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,0,0,0,0,0)
f2<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,0,1,0,0,0)
f3<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,0,2,0,0,0)
f4<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,1,1,0,0,0)
f5<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,1,2,0,0,0)
f6<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,1,0,0,0,0)
f7<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,2,0,0,0,0)
f8<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,2,1,0,0,0)
f9<- fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 2,2,2,0,0,0)

f10<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,0,0,0,0,0)
f11<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,0,1,0,0,0)
f12<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,0,2,0,0,0)
f13<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,1,1,0,0,0)
f14<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,1,2,0,0,0)
f15<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,1,0,0,0,0)
f16<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,2,0,0,0,0)
f17<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,2,1,0,0,0)
f18<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 1,2,2,0,0,0)

f19<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,0,0,0,0,0)
f20<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,0,1,0,0,0)
f21<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,0,2,0,0,0)
f22<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,1,0,0,0,0)
f23<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,1,1,0,0,0)
f24<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,1,2,0,0,0)
f25<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,2,0,0,0,0)
f26<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,52, 0,2,1,0,0,0)
f27<-fun.illustrate.2(clem_data_sales_ts$salestszSeasonallyAdjusted,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.
## 1.343e+12 1.404e+12 1.411e+12 5.285e+12 1.873e+12 7.704e+13
```

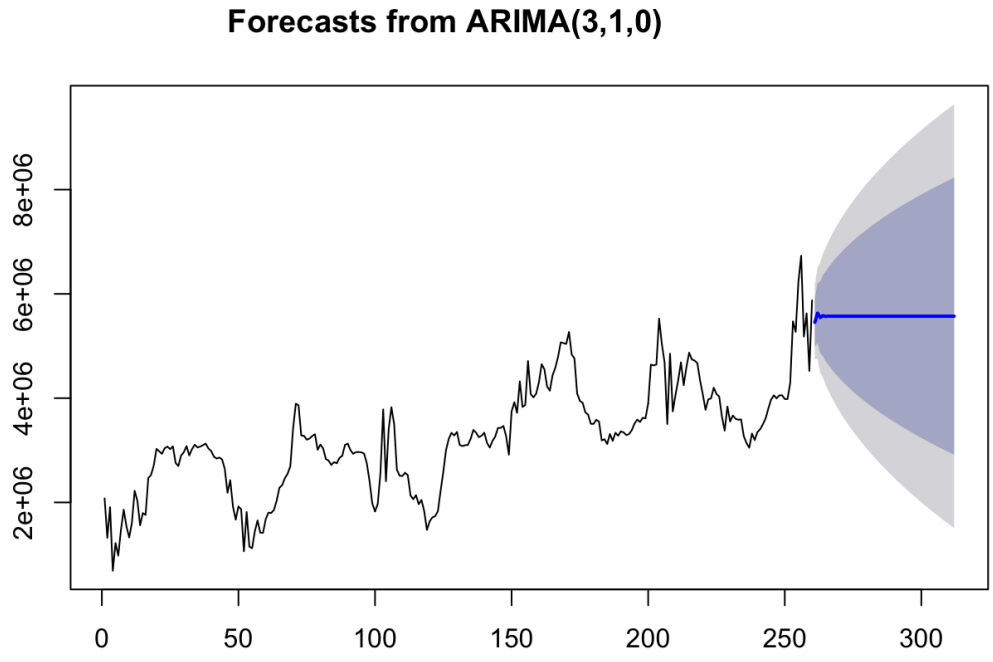
```
which.min(total.sales.volume)
```

```
## [1] 17
```

```
# Auto ARIMA (3,1,0)
auto.sales.volume <- auto.arima(clem_data_sales_ts$salestszSeasonallyAdjusted)
tsdisplay(residuals(auto.sales.volume))
```



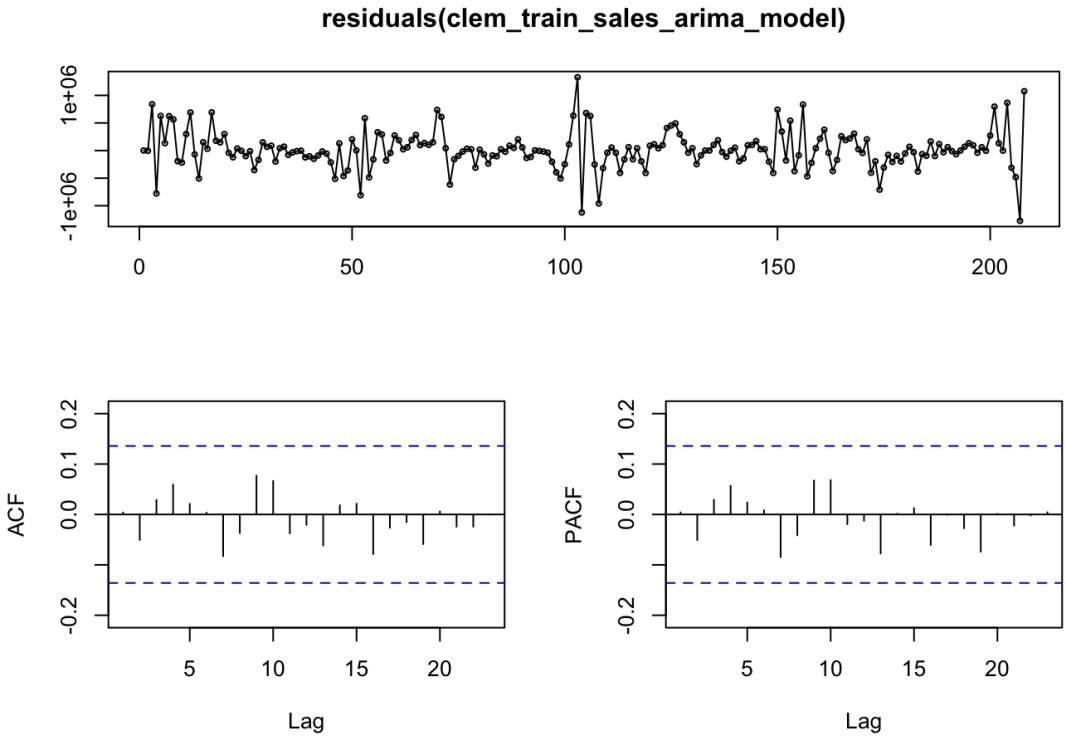

```
# Forecast Auto ARIMA
auto.sales.volume.forecast <- forecast(auto.sales.volume, h=52)
plot(auto.sales.volume.forecast)
```




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

```
## [1] 2077991.2 1320663.6 1909651.8  689723.6 1220800.4  979856.2
```

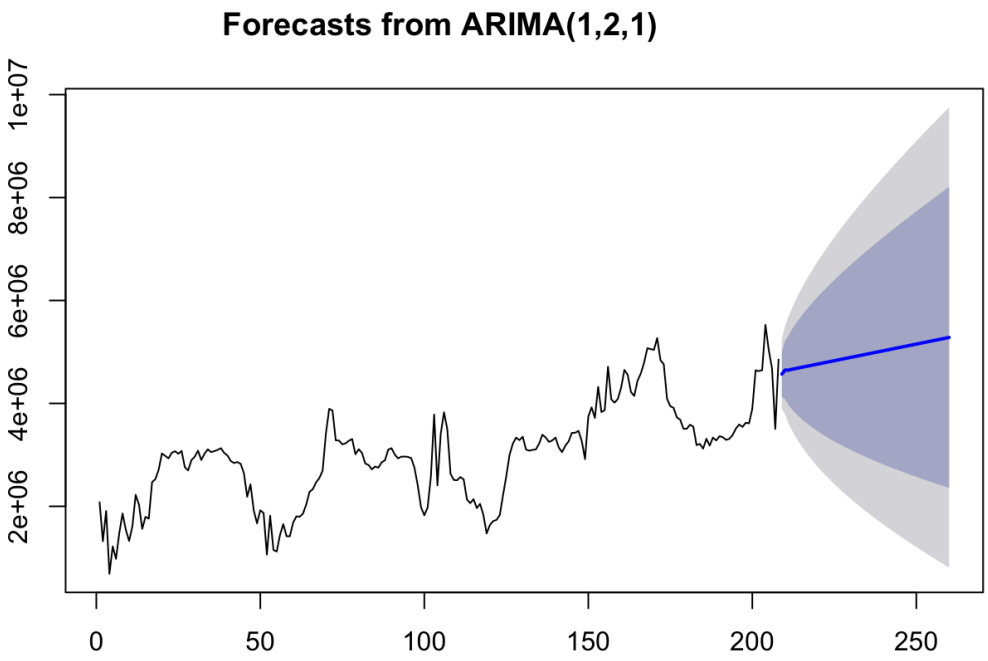
```
# Choosing Model 17 - ARIMA(1,2,1,0,0,0)
clem_train_sales_arima_model <- Arima(clem_train_sales_data_arima, order=c(1,2,1))
tsdisplay(residuals(clem_train_sales_arima_model))
```




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

```
## Time Series:
## Start = 209
## End = 260
## Frequency = 1
##  [1] 4569266 4647909 4646237 4662379 4674570 4687638 4700511 4713427
##  [9] 4726333 4739242 4752150 4765058 4777966 4790874 4803783 4816691
## [17] 4829599 4842507 4855415 4868324 4881232 4894140 4907048 4919956
## [25] 4932865 4945773 4958681 4971589 4984497 4997406 5010314 5023222
## [33] 5036130 5049038 5061947 5074855 5087763 5100671 5113579 5126488
## [41] 5139396 5152304 5165212 5178121 5191029 5203937 5216845 5229753
## [49] 5242662 5255570 5268478 5281386
```

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



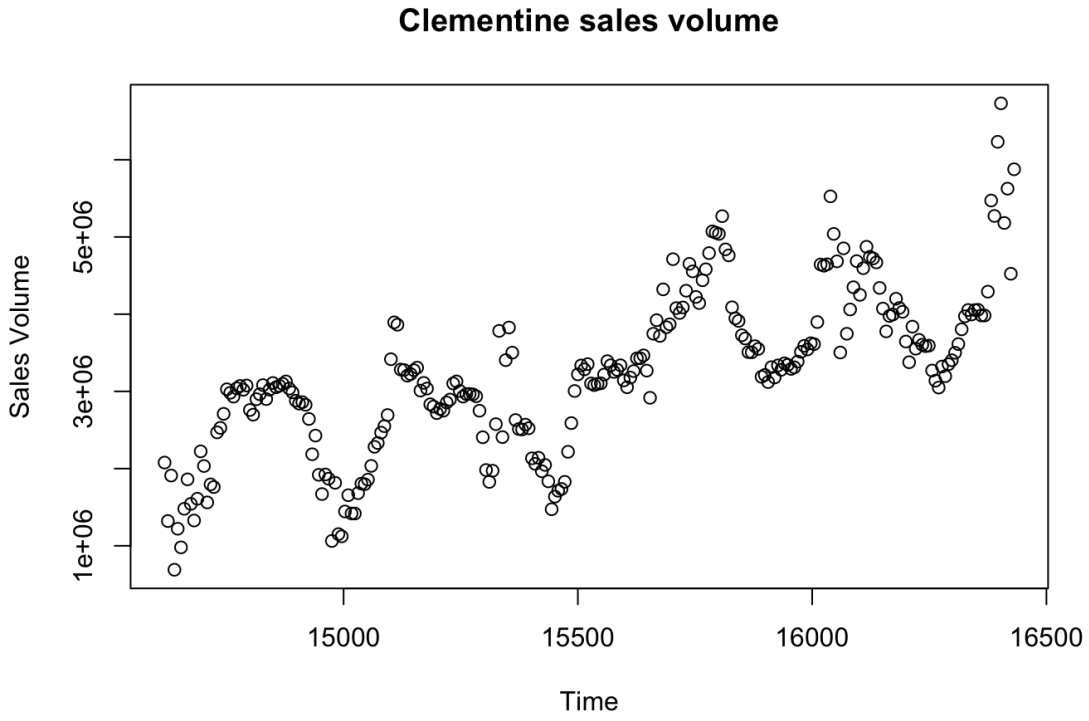
Both time series, clementine sales volume and social media are of I(1) OR > 1 process, therefore the series is not stationary and has a trend and drift, and is not showing a tendency to return back to mean.

Step 1: Check if the series is stationary.

```
sales.volume <- clem_data_sales_ts$salestszSeasonallyAdjusted
#d.sales.volume <- diff(sales.volume)
week <- clem_data_sales_ts$Date


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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  689700 2639000 3219000 3229000 3865000 6730000
```

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

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




```
#Based on the 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"))

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  sales.volume
## Dickey-Fuller = -3.2802, Lag order = 6, p-value = 0.07494
## alternative hypothesis: stationary
```

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

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

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

```
##
##  KPSS Test for Trend Stationarity
##
## data:  sales.volume
## KPSS Trend = 0.1167, Truncation lag parameter = 3, p-value = 0.1
```

ADF Test: With the p-value of 0.07, greater than significance level of 0.05 suggests that we fail reject the null hypothesis that the series is not stationary and unit root. In other words, we have no evidence that the series is stationary.

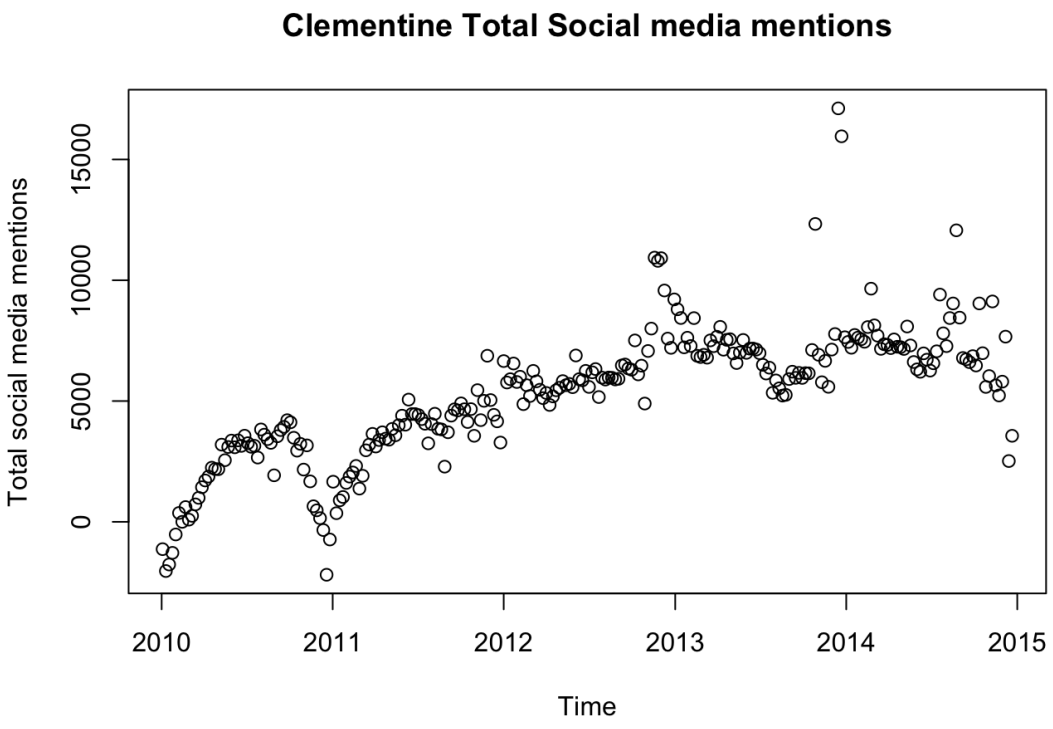
KPSS Test: At significance level of 5% or p-value > 0.05, we reject the 'Trend' null hypothesis that the series is stationary with a linear trend. In other words, we have no evidence that the series is stationary.

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

```
clem_total_social_media <- clem_data_social_ts$Total.social.media

#d.clem_blogs <- diff(clem_blogs)
week <- clem_data_social_ts$Analysis_Date


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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -2191    3556    5772    5276    7020   17110
```

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




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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  clem_total_social_media
## Dickey-Fuller = -2.3232, Lag order = 6, p-value = 0.4398
## alternative hypothesis: stationary
```

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

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

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

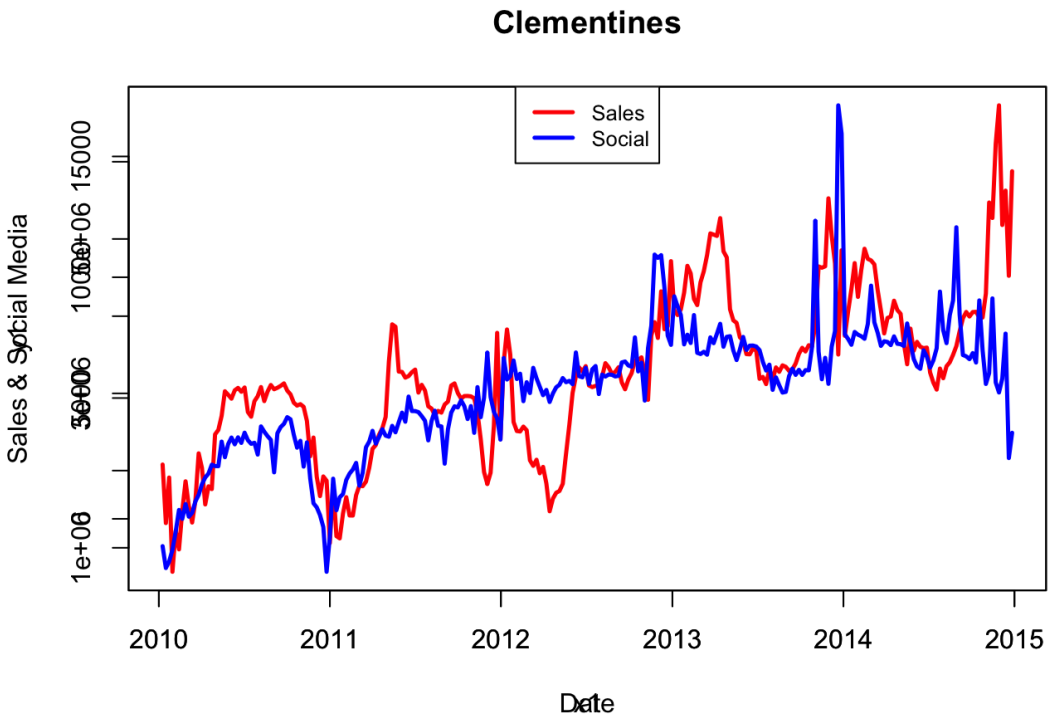
```
##
##  KPSS Test for Level Stationarity
##
## data:  clem_total_social_media
## KPSS Level = 4.7557, Truncation lag parameter = 3, p-value = 0.01
```

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

Small p-value of 0.01, 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 <- clem_data_sales_ts$Date
y1 <- sales.volume
y2 <- clem_total_social_media

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




```
library("urca")

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

CoIntegrationTest =ca.jo(co.test.matrix,type="trace",K=4,ecdet="none", spec="longrun")
summary(CoIntegrationTest)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.05653831 0.03226990
##
## Values of teststatistic and critical values of test:
##
##          test 10pct  5pct  1pct
## r <= 1 |  8.4  6.50  8.18 11.65
## r = 0  | 23.3 15.66 17.95 23.52
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                            sales.volume.l4 clem_total_social_media.l4
## sales.volume.l4                     1.0000                     1.0000
## clem_total_social_media.l4       -432.5455                   149.9026
##
## Weights W:
## (This is the loading matrix)
##
##                           sales.volume.l4 clem_total_social_media.l4
## sales.volume.d              -0.0965732245              -0.0292266540
## clem_total_social_media.d    0.0002522272              -0.0001353827
```

With lag of k=4, we see that our test statistic (r <= 1) of 8.4 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 Clem 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 Clementine 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 f
or Analysis")
clem_google_trends <- read.csv("Clem_Google_Trends_Searches.csv")

# Plot sales, social media and google trends

x1 <- clem_data_sales_ts$Date
y1 <- clem_data_sales_ts$salestszSeasonallyAdjusted
y2 <- clem_data_social_ts$Total.social.media
y3 <- clem_google_trends$clementine.Searches

#plot( x1, y1, type="l", col="red", main = "Clementines 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(clem_data_social_ts$Total.social.media)
```

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

```
length(clem_data_sales_ts$salestszSeasonallyAdjusted)
```

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

```
length(clem_data_sales_ts$Date)
```

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

```
length(clem_google_trends$clementine.Searches)
```

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

```
Train_clem_sales <- clem_data_sales_ts[1:208,2]
Train_clem_week <- clem_data_sales_ts[1:208,1]
Train_clem_social <- clem_data_social_ts[1:208,8]
Train_clem_google_trends <- clem_google_trends[1:208,3]

# Endogenous variables
Train_VAR_clem <- cbind(Train_clem_sales, Train_clem_social)

#VAR Select
#VARselect(Train_VAR_clem, lag.max = 10, type = "both", exogen = cbind(x3 #=Train_clem_google_trends))

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

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      1      1      9
##
## $criteria
##                   1             2             3             4             5
## AIC(n) 3.940829e+01 3.939856e+01 3.942795e+01 3.943706e+01 3.945760e+01
## HQ(n)  3.946207e+01 3.947923e+01 3.953550e+01 3.957150e+01 3.961893e+01
## SC(n)  3.954115e+01 3.959785e+01 3.969367e+01 3.976921e+01 3.985618e+01
## FPE(n) 1.302588e+17 1.290014e+17 1.328555e+17 1.340822e+17 1.368819e+17
##                   6             7             8             9            10
## AIC(n) 3.947894e+01 3.940146e+01 3.938349e+01 3.935874e+01 3.938717e+01
## HQ(n)  3.966716e+01 3.961657e+01 3.962548e+01 3.962762e+01 3.968294e+01
## SC(n)  3.994394e+01 3.993290e+01 3.998135e+01 4.002303e+01 4.011790e+01
## FPE(n) 1.398586e+17 1.294622e+17 1.271946e+17 1.241315e+17 1.277706e+17
```

```
Train_VAR_model_clem <- VAR(Train_VAR_clem, p=9, type="both")
summary(Train_VAR_model_clem)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: Train_clem_sales, Train_clem_social
## Deterministic variables: both
## Sample size: 199
## Log Likelihood: -4440.46
## Roots of the characteristic polynomial:
## 0.9426 0.9426 0.9406 0.9406 0.9212 0.9212 0.9078 0.9036 0.9036 0.8158 0.8158 0.7985 0.7985 0.7784 0.7784
0.7666 0.7666 0.3757
## Call:
## VAR(y = Train_VAR_clem, p = 9, type = "both")
##
##
## Estimation results for equation Train_clem_sales:
## ================================================
## Train_clem_sales = Train_clem_sales.l1 + Train_clem_social.l1 + Train_clem_sales.l2 + Train_clem_social.l
2 + Train_clem_sales.l3 + Train_clem_social.l3 + Train_clem_sales.l4 + Train_clem_social.l4 + Train_clem_sal
es.l5 + Train_clem_social.l5 + Train_clem_sales.l6 + Train_clem_social.l6 + Train_clem_sales.l7 + Train_clem
_social.l7 + Train_clem_sales.l8 + Train_clem_social.l8 + Train_clem_sales.l9 + Train_clem_social.l9 + const
+ trend
##
##                         Estimate Std. Error t value Pr(>|t|)
## Train_clem_sales.l1   7.824e-01  7.440e-02  10.517  < 2e-16 ***
## Train_clem_social.l1  6.192e+01  2.241e+01   2.763  0.00633 **
## Train_clem_sales.l2   6.707e-02  9.728e-02   0.690  0.49139
## Train_clem_social.l2 -1.522e+01  2.950e+01  -0.516  0.60658
## Train_clem_sales.l3   8.403e-02  9.861e-02   0.852  0.39528
## Train_clem_social.l3  1.508e+00  3.054e+01   0.049  0.96067
## Train_clem_sales.l4  -3.986e-04  9.857e-02  -0.004  0.99678
## Train_clem_social.l4  3.787e+01  3.056e+01   1.239  0.21684
## Train_clem_sales.l5  -3.275e-02  1.006e-01  -0.325  0.74521
## Train_clem_social.l5  2.038e+01  3.054e+01   0.667  0.50546
## Train_clem_sales.l6   1.248e-02  9.728e-02   0.128  0.89802
## Train_clem_social.l6 -1.369e+01  3.053e+01  -0.448  0.65448
## Train_clem_sales.l7  -1.539e-01  9.421e-02  -1.633  0.10419
## Train_clem_social.l7 -8.792e+01  3.121e+01  -2.817  0.00539 **
## Train_clem_sales.l8  -3.376e-03  9.252e-02  -0.036  0.97094
## Train_clem_social.l8  7.445e+01  3.311e+01   2.249  0.02573 *
## Train_clem_sales.l9   1.187e-01  7.514e-02   1.580  0.11584
## Train_clem_social.l9 -4.461e+01  2.965e+01  -1.504  0.13422
## const                 1.974e+05  9.550e+04   2.067  0.04015 *
## trend                 1.747e+02  8.329e+02   0.210  0.83406
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 311100 on 179 degrees of freedom
## Multiple R-Squared: 0.8926,  Adjusted R-squared: 0.8812
## F-statistic: 78.27 on 19 and 179 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation Train_clem_social:
## ==================================================
## Train_clem_social = Train_clem_sales.l1 + Train_clem_social.l1 + Train_clem_sales.l2 + Train_clem_social.
l2 + Train_clem_sales.l3 + Train_clem_social.l3 + Train_clem_sales.l4 + Train_clem_social.l4 + Train_clem_sa
les.l5 + Train_clem_social.l5 + Train_clem_sales.l6 + Train_clem_social.l6 + Train_clem_sales.l7 + Train_cle
m_social.l7 + Train_clem_sales.l8 + Train_clem_social.l8 + Train_clem_sales.l9 + Train_clem_social.l9 + cons
t + trend
##
##                         Estimate Std. Error t value Pr(>|t|)
## Train_clem_sales.l1  -4.532e-04  2.458e-04  -1.843 0.066928 .
## Train_clem_social.l1  6.491e-01  7.406e-02   8.765 1.42e-15 ***
## Train_clem_sales.l2   7.296e-04  3.214e-04   2.270 0.024408 *
## Train_clem_social.l2  8.506e-02  9.748e-02   0.873 0.384007
```

```
## Train_clem_sales.l3   3.243e-04  3.258e-04   0.995 0.320915
## Train_clem_social.l3 -5.985e-02  1.009e-01  -0.593 0.553914
## Train_clem_sales.l4  -3.998e-04  3.257e-04  -1.228 0.221237
## Train_clem_social.l4 -9.680e-02  1.010e-01  -0.959 0.339054
## Train_clem_sales.l5   6.902e-05  3.325e-04   0.208 0.835785
## Train_clem_social.l5  1.272e-01  1.009e-01   1.260 0.209162
## Train_clem_sales.l6   3.061e-04  3.214e-04   0.952 0.342239
## Train_clem_social.l6 -1.654e-01  1.009e-01  -1.640 0.102750
## Train_clem_sales.l7  -5.244e-05  3.113e-04  -0.168 0.866412
## Train_clem_social.l7  4.190e-01  1.031e-01   4.064 7.23e-05 ***
## Train_clem_sales.l8  -5.273e-04  3.057e-04  -1.725 0.086308 .
## Train_clem_social.l8  3.081e-02  1.094e-01   0.282 0.778525
## Train_clem_sales.l9   4.647e-05  2.483e-04   0.187 0.851742
## Train_clem_social.l9 -2.538e-01  9.797e-02  -2.591 0.010362 *
## const                 2.130e+02  3.156e+02   0.675 0.500491
## trend                 9.261e+00  2.752e+00   3.365 0.000937 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 1028 on 179 degrees of freedom
## Multiple R-Squared: 0.8544,  Adjusted R-squared: 0.8389
## F-statistic: 55.28 on 19 and 179 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                   Train_clem_sales Train_clem_social
## Train_clem_sales         9.677e+10         -15374636
## Train_clem_social       -1.537e+07           1056583
##
## Correlation matrix of residuals:
##                   Train_clem_sales Train_clem_social
## Train_clem_sales           1.00000          -0.04808
## Train_clem_social         -0.04808           1.00000
```

The adjusted R-Squared of 87% for equation predicting sales as dependent variable and endogenous variables of social media and sales with lag order of 2 and exogenous variable of google trends 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 67%.

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

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

Test_clem_sales <- clem_data_sales_ts[209:260,2]
Test_clem_week <- clem_data_sales_ts[209:260,1]
Test_clem_social <- clem_data_social_ts[209:260,8]
Test_clem_google_trends <- clem_google_trends[209:260,3]

length(Test_clem_sales)
```

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

```
length(Test_clem_week)
```

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

```
length(Test_clem_social)
```

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

```
length(Test_clem_google_trends)
```

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

```
# Endogenous variables
#Test_VAR_clem <- cbind(Test_clem_sales, Test_clem_social)

#VAR Select
#VARselect(Test_VAR_clem, lag.max = 10, type = "both", exogen = cbind(x3 =Test_clem_google_trends))

#Test_VAR_model_clem <- VAR(Test_VAR_clem, p=7, type="both", exogen = cbind(x3 =Test_clem_google_trends))
#summary(Test_VAR_model_clem)
```

```
# Clementine prediction

#var_train_forecasts <- predict(Train_VAR_model_clem, n.ahead = 52, ci = 0.95, dumvar = #cbind(x3 =Test_clem
_google_trends))

var_train_forecasts <- predict(Train_VAR_model_clem, 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_clem_sales
##          fcst   lower   upper        CI
##  [1,] 4758695 4148997 5368392  609697.7
##  [2,] 4668234 3887763 5448705  780471.0
##  [3,] 5157530 4272658 6042402  884872.0
##  [4,] 5435878 4453369 6418387  982509.0
##  [5,] 5536223 4449022 6623423 1087200.3
##  [6,] 4825124 3640920 6009328 1184204.0
##  [7,] 5064168 3791384 6336951 1272783.7
##  [8,] 5121567 3813841 6429292 1307725.5
##  [9,] 4779025 3428620 6129429 1350404.4
## [10,] 4813071 3430583 6195560 1382488.4
## [11,] 4976439 3570920 6381957 1405518.6
## [12,] 5048793 3620133 6477453 1428659.9
## [13,] 4683637 3237043 6130232 1446594.7
## [14,] 4424901 2964226 5885576 1460675.0
## [15,] 4655516 3189130 6121902 1466386.0
## [16,] 4519282 3046862 5991702 1472420.3
## [17,] 4382303 2905829 5858776 1476473.3
## [18,] 4480120 3002349 5957891 1477771.0
## [19,] 4603556 3124665 6082447 1478890.9
## [20,] 4535910 3056196 6015623 1479713.5
## [21,] 4288091 2808083 5768098 1480007.4
## [22,] 4357739 2877697 5837780 1480041.4
## [23,] 4429597 2948440 5910753 1481156.7
## [24,] 4325035 2843328 5806743 1481707.4
## [25,] 4350792 2868500 5833083 1482291.5
## [26,] 4477562 2994153 5960970 1483408.8
## [27,] 4548633 3064299 6032967 1484334.3
## [28,] 4438351 2953610 5923093 1484741.6
## [29,] 4399742 2914756 5884728 1484985.8
## [30,] 4511901 3026195 5997608 1485706.5
## [31,] 4506637 3020242 5993031 1486394.5
## [32,] 4483239 2996565 5969914 1486674.6
## [33,] 4560228 3073167 6047289 1487061.1
## [34,] 4657912 3170496 6145328 1487416.0
## [35,] 4651337 3163784 6138891 1487553.4
## [36,] 4589579 3102021 6077138 1487558.5
## [37,] 4639303 3151727 6126880 1487576.8
## [38,] 4687881 3200223 6175538 1487657.5
## [39,] 4665070 3177391 6152750 1487679.2
## [40,] 4683292 3195609 6170975 1487682.9
## [41,] 4753772 3266074 6241469 1487697.3
## [42,] 4790513 3302785 6278241 1487727.8
## [43,] 4753097 3265316 6240878 1487780.8
## [44,] 4746981 3259128 6234834 1487853.1
## [45,] 4791864 3303990 6279738 1487874.2
## [46,] 4793591 3305678 6281504 1487913.0
## [47,] 4782654 3294687 6270620 1487966.5
## [48,] 4815235 3327241 6303230 1487994.5
## [49,] 4858170 3370148 6346192 1488021.9
## [50,] 4853827 3365761 6341894 1488066.6
## [51,] 4831871 3343742 6319999 1488128.5
## [52,] 4852072 3363914 6340231 1488158.7
##
## $fcst$Train_clem_social
##           fcst     lower    upper       CI
##  [1,] 11075.123  9060.472 13089.77 2014.651
##  [2,] 10343.715  7918.848 12768.58 2424.867
##  [3,]  8972.064  6363.140 11580.99 2608.924
##  [4,]  8416.950  5708.924 11124.98 2708.026
##  [5,]  7650.013  4912.609 10387.42 2737.403
##  [6,] 11365.900  8591.628 14140.17 2774.272
##  [7,] 15032.955 12213.984 17851.93 2818.971
```

```
##  [8,] 11870.905  8884.617 14857.19 2986.288
##  [9,] 10147.967  6907.402 13388.53 3240.565
## [10,]  9495.954  6182.855 12809.05 3313.099
## [11,]  8642.518  5291.910 11993.13 3350.608
## [12,]  7799.211  4441.524 11156.90 3357.687
## [13,]  8368.604  5009.757 11727.45 3358.847
## [14,] 11558.231  8192.342 14924.12 3365.889
## [15,] 11410.723  8039.669 14781.78 3371.054
## [16,]  9472.141  6068.347 12875.93 3403.794
## [17,]  9003.340  5580.089 12426.59 3423.251
## [18,]  8610.467  5184.152 12036.78 3426.315
## [19,]  8053.811  4626.161 11481.46 3427.650
## [20,]  7717.610  4283.466 11151.75 3434.144
## [21,]  9353.986  5906.328 12801.64 3447.658
## [22,] 10741.258  7272.006 14210.51 3469.252
## [23,]  9820.013  6350.119 13289.91 3469.894
## [24,]  9152.505  5676.054 12628.95 3476.450
## [25,]  9086.390  5604.332 12568.45 3482.058
## [26,]  8859.456  5373.624 12345.29 3485.832
## [27,]  8462.999  4972.059 11953.94 3490.941
## [28,]  8925.241  5427.132 12423.35 3498.109
## [29,] 10276.646  6766.668 13786.62 3509.978
## [30,] 10418.668  6903.894 13933.44 3514.775
## [31,]  9811.939  6296.226 13327.65 3515.714
## [32,]  9703.137  6185.109 13221.16 3518.028
## [33,]  9695.163  6176.181 13214.14 3518.982
## [34,]  9447.327  5927.853 12966.80 3519.474
## [35,]  9370.394  5850.209 12890.58 3520.186
## [36,] 10106.134  6584.270 13628.00 3521.864
## [37,] 10702.590  7178.590 14226.59 3524.000
## [38,] 10443.014  6918.911 13967.12 3524.103
## [39,] 10203.791  6678.121 13729.46 3525.670
## [40,] 10231.403  6705.107 13757.70 3526.297
## [41,] 10143.759  6617.371 13670.15 3526.389
## [42,]  9945.699  6419.265 13472.13 3526.434
## [43,] 10157.203  6630.763 13683.64 3526.440
## [44,] 10696.393  7169.543 14223.24 3526.850
## [45,] 10781.768  7254.677 14308.86 3527.091
## [46,] 10560.187  7032.303 14088.07 3527.884
## [47,] 10521.916  6993.238 14050.59 3528.678
## [48,] 10532.216  7003.326 14061.11 3528.890
## [49,] 10400.545  6871.543 13929.55 3529.003
## [50,] 10362.547  6833.507 13891.59 3529.041
## [51,] 10663.292  7134.203 14192.38 3529.089
## [52,] 10908.492  7379.206 14437.78 3529.287
##
##
## $endog
##       Train_clem_sales Train_clem_social
##   [1,]        2077991.2     -1131.2220137
##   [2,]        1320663.6     -2036.4830714
##   [3,]        1909651.8     -1766.2691291
##   [4,]         689723.6     -1284.4013406
##   [5,]        1220800.4      -523.6248983
##   [6,]         979856.2       365.9976979
##   [7,]        1479185.4         0.3558709
##   [8,]        1860749.0       611.2741402
##   [9,]        1542671.6        90.0217363
##  [10,]        1327480.1       248.4544286
##  [11,]        1609361.3       721.6827940
##  [12,]        2222929.9       992.8246209
##  [13,]        2033281.7      1432.3174094
##  [14,]        1562119.6      1711.1755825
##  [15,]        1795808.3      1873.9496209
##  [16,]        1760854.6      2241.0385632
```

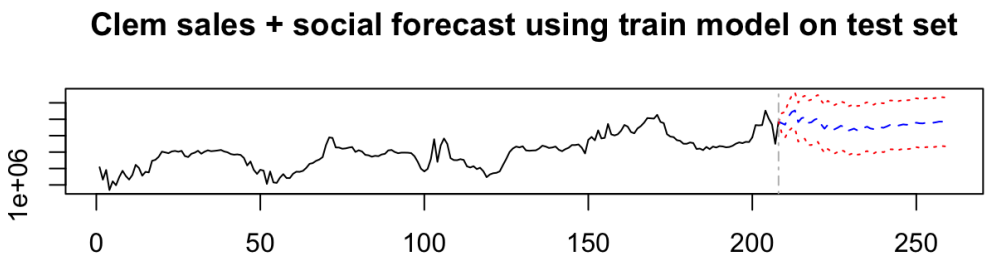
```
##  [17,]        2469287.5      2183.4568325
##  [18,]        2527210.0      2178.9496209
##  [19,]        2709033.8      3195.3198132
##  [20,]        3027068.1      2551.1202940
##  [21,]        2981474.2      3097.3005825
##  [22,]        2931852.3      3363.2573132
##  [23,]        3039766.0      3085.2693325
##  [24,]        3070881.7      3372.0938517
##  [25,]        3021445.2      3143.9015440
##  [26,]        3076130.2      3565.0361594
##  [27,]        2758538.9      3255.1856786
##  [28,]        2698152.9      3105.3760632
##  [29,]        2898349.3      3147.4260632
##  [30,]        2963534.5      2660.4491402
##  [31,]        3081420.0      3824.7356786
##  [32,]        2900307.0      3607.7818325
##  [33,]        3022943.2      3422.6626017
##  [34,]        3108343.1      3268.9222171
##  [35,]        3053562.0      1925.0106786
##  [36,]        3072462.0      3541.6299094
##  [37,]        3097452.3      3797.8914479
##  [38,]        3129129.2      3931.5087556
##  [39,]        3036085.4      4207.6068325
##  [40,]        2987174.6      4119.8491402
##  [41,]        2879341.9      3479.5395248
##  [42,]        2841920.4      2945.5145248
##  [43,]        2860287.6      3228.0587555
##  [44,]        2826199.7      2162.7683709
##  [45,]        2642833.1      3164.6414479
##  [46,]        2185830.1      1674.1837555
##  [47,]        2427464.4       644.1472171
##  [48,]        1919336.4       475.3202940
##  [49,]        1670006.9       152.5318325
##  [50,]        1923753.6      -340.0412444
##  [51,]        1868952.9     -2190.7181675
##  [52,]        1062789.1      -731.7797060
##  [53,]        1816196.4      1660.7779863
##  [54,]        1151283.9       357.5169286
##  [55,]        1122230.4       889.7308709
##  [56,]        1444634.3      1028.5986594
##  [57,]        1655096.4      1610.3751017
##  [58,]        1418642.3      1871.9976979
##  [59,]        1417775.8      2042.3558709
##  [60,]        1681882.6      2314.2741402
##  [61,]        1805371.3      1375.0217363
##  [62,]        1797987.2      1905.4544286
##  [63,]        1857040.8      2959.6827940
##  [64,]        2034920.6      3196.8246209
##  [65,]        2282016.0      3641.3174094
##  [66,]        2331639.7      3116.1755825
##  [67,]        2465413.2      3385.9496209
##  [68,]        2549572.6      3707.0385632
##  [69,]        2693755.2      3443.4568325
##  [70,]        3416675.3      3398.9496209
##  [71,]        3893915.3      3847.3198132
##  [72,]        3863239.9      3583.1202940
##  [73,]        3282693.9      3999.3005825
##  [74,]        3277275.7      4394.2573132
##  [75,]        3204178.0      4020.2693325
##  [76,]        3224071.2      5059.0938517
##  [77,]        3274183.2      4461.9015440
##  [78,]        3307544.7      4460.0361594
##  [79,]        3010483.5      4417.1856786
##  [80,]        3110268.3      4250.3760632
##  [81,]        3036715.1      4060.4260632
```

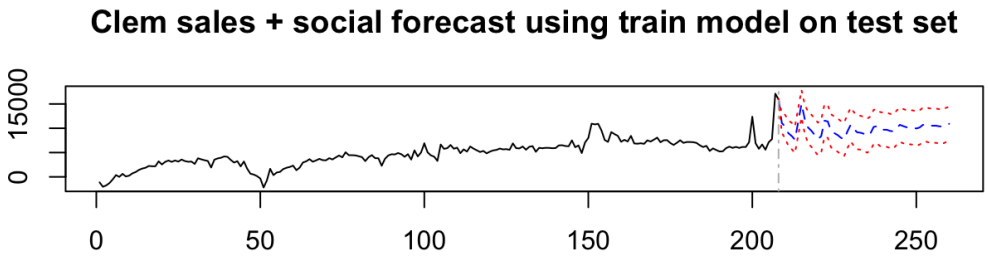
```
##  [82,]        2832657.0      3248.4491402
##  [83,]        2802963.4      4046.7356786
##  [84,]        2718567.4      4473.7818325
##  [85,]        2772830.2      3851.6626017
##  [86,]        2750812.7      3824.9222171
##  [87,]        2855598.4      2284.0106786
##  [88,]        2893680.6      3707.6299094
##  [89,]        3102093.2      4395.8914479
##  [90,]        3128923.4      4657.5087556
##  [91,]        3005128.3      4610.6068325
##  [92,]        2932257.0      4902.8491402
##  [93,]        2965397.8      4659.5395248
##  [94,]        2966578.1      4131.5145248
##  [95,]        2961161.5      4665.0587555
##  [96,]        2936383.4      3560.7683709
##  [97,]        2749571.7      5450.6414479
##  [98,]        2405608.3      4209.1837555
##  [99,]        1981605.4      5008.1472171
## [100,]        1827741.7      6874.3202940
## [101,]        1972377.4      5040.5318325
## [102,]        2575755.0      4428.9587556
## [103,]        3783529.3      4158.2818325
## [104,]        2407656.1      3281.2202940
## [105,]        3404523.8      6653.7779863
## [106,]        3827436.2      5766.5169286
## [107,]        3501785.9      5904.7308709
## [108,]        2629319.0      6553.5986594
## [109,]        2512624.7      5788.3751017
## [110,]        2506626.4      6005.9976979
## [111,]        2569186.5      4871.3558709
## [112,]        2522442.1      5654.2741402
## [113,]        2132970.4      5208.0217363
## [114,]        2061745.6      6248.4544286
## [115,]        2139889.8      5815.6827940
## [116,]        1965566.4      5466.8246209
## [117,]        2047397.1      5119.3174094
## [118,]        1834710.0      5334.1755825
## [119,]        1473312.3      4837.9496209
## [120,]        1637293.6      5191.0385632
## [121,]        1712948.7      5445.4568325
## [122,]        1736100.5      5549.9496209
## [123,]        1828442.3      5825.3198132
## [124,]        2217328.1      5656.1202940
## [125,]        2589052.4      5709.3005825
## [126,]        3005289.8      5573.2573132
## [127,]        3220444.8      6885.2693325
## [128,]        3335409.4      5905.0938517
## [129,]        3285071.4      5856.9015440
## [130,]        3352387.0      6249.0361594
## [131,]        3102707.3      5584.1856786
## [132,]        3081498.6      6188.3760632
## [133,]        3097255.0      6323.4260632
## [134,]        3104773.8      5165.4491402
## [135,]        3217716.8      5967.7356786
## [136,]        3391157.8      5889.7818325
## [137,]        3337973.0      5973.6626017
## [138,]        3251749.7      5961.9222171
## [139,]        3279336.1      5894.0106786
## [140,]        3337459.8      5919.6299094
## [141,]        3141980.4      6472.8914479
## [142,]        3053319.6      6511.5087555
## [143,]        3178707.0      6347.6068325
## [144,]        3262688.0      6295.8491402
## [145,]        3428266.8      7508.5395248
## [146,]        3428911.5      6103.5145248
```

```
## [147,]        3466194.8      6464.0587555
## [148,]        3268942.3      4895.7683709
## [149,]        2916938.9      7071.6414479
## [150,]        3746737.1      7998.1837555
## [151,]        3922153.6     10936.1472171
## [152,]        3718288.6     10801.3202940
## [153,]        4321226.9     10913.5318325
## [154,]        3830955.8      9575.9587556
## [155,]        3868903.8      7584.2818325
## [156,]        4712227.8      7202.2202940
## [157,]        4078340.8      9206.7779863
## [158,]        4014880.9      8796.5169286
## [159,]        4087165.5      8436.7308709
## [160,]        4303987.9      7222.5986594
## [161,]        4651768.6      7620.3751017
## [162,]        4554273.3      7278.9976979
## [163,]        4223135.7      8432.3558709
## [164,]        4143781.8      6872.2741402
## [165,]        4439329.0      6822.0217363
## [166,]        4581554.0      6914.4544286
## [167,]        4790849.7      6794.6827940
## [168,]        5071836.4      7509.8246209
## [169,]        5056321.8      7271.3174094
## [170,]        5039456.1      7660.1755825
## [171,]        5269521.9      8070.9496209
## [172,]        4839400.7      7125.0385632
## [173,]        4761428.7      7533.4568325
## [174,]        4089047.1      7558.9496209
## [175,]        3946073.5      6970.3198132
## [176,]        3912808.2      6573.1202940
## [177,]        3727163.8      7007.3005825
## [178,]        3684843.5      7528.2573132
## [179,]        3508703.5      6994.2693325
## [180,]        3505929.6      7168.0938517
## [181,]        3585424.6      7172.9015440
## [182,]        3549844.9      7131.0361594
## [183,]        3188808.8      6982.1856786
## [184,]        3213007.1      6497.3760632
## [185,]        3120410.0      6138.4260632
## [186,]        3313654.9      6380.4491402
## [187,]        3180083.5      5339.7356786
## [188,]        3336044.6      5852.7818325
## [189,]        3279727.8      5526.6626017
## [190,]        3362907.8      5225.9222171
## [191,]        3343593.6      5256.0106786
## [192,]        3291203.4      5909.6299094
## [193,]        3311678.7      6214.8914479
## [194,]        3385868.4      5959.5087556
## [195,]        3511870.9      6165.6068325
## [196,]        3589654.9      5956.8491402
## [197,]        3543135.9      6154.5395248
## [198,]        3623337.3      6146.5145248
## [199,]        3612042.2      7113.0587555
## [200,]        3898106.6     12330.7683709
## [201,]        4643754.5      6915.6414479
## [202,]        4627300.3      5778.1837555
## [203,]        4645862.9      6663.1472171
## [204,]        5525928.9      5588.3202940
## [205,]        5039683.8      7129.5318325
## [206,]        4683987.3      7772.9587555
## [207,]        3503620.6     17114.2818325
## [208,]        4852728.4     15958.2202940
##
## $model
##
```

```
## VAR Estimation Results:
## ========================
##
## Estimated coefficients for equation Train_clem_sales:
## ====================================================
## Call:
## Train_clem_sales = Train_clem_sales.l1 + Train_clem_social.l1 + Train_clem_sales.l2 + Train_clem_social.l
2 + Train_clem_sales.l3 + Train_clem_social.l3 + Train_clem_sales.l4 + Train_clem_social.l4 + Train_clem_sal
es.l5 + Train_clem_social.l5 + Train_clem_sales.l6 + Train_clem_social.l6 + Train_clem_sales.l7 + Train_clem
_social.l7 + Train_clem_sales.l8 + Train_clem_social.l8 + Train_clem_sales.l9 + Train_clem_social.l9 + const
+ trend
##
##  Train_clem_sales.l1 Train_clem_social.l1  Train_clem_sales.l2
##         7.824150e-01         6.192138e+01         6.707404e-02
## Train_clem_social.l2  Train_clem_sales.l3 Train_clem_social.l3
##        -1.521770e+01         8.403037e-02         1.508041e+00
##  Train_clem_sales.l4 Train_clem_social.l4  Train_clem_sales.l5
##        -3.986068e-04         3.787383e+01        -3.274929e-02
## Train_clem_social.l5  Train_clem_sales.l6 Train_clem_social.l6
##         2.037668e+01         1.248483e-02        -1.368525e+01
##  Train_clem_sales.l7 Train_clem_social.l7  Train_clem_sales.l8
##        -1.538542e-01        -8.791747e+01        -3.375587e-03
## Train_clem_social.l8  Train_clem_sales.l9 Train_clem_social.l9
##         7.445496e+01         1.187393e-01        -4.460519e+01
##                const                trend
##         1.974218e+05         1.747395e+02
##
##
## Estimated coefficients for equation Train_clem_social:
## ======================================================
## Call:
## Train_clem_social = Train_clem_sales.l1 + Train_clem_social.l1 + Train_clem_sales.l2 + Train_clem_social.
l2 + Train_clem_sales.l3 + Train_clem_social.l3 + Train_clem_sales.l4 + Train_clem_social.l4 + Train_clem_sa
les.l5 + Train_clem_social.l5 + Train_clem_sales.l6 + Train_clem_social.l6 + Train_clem_sales.l7 + Train_cle
m_social.l7 + Train_clem_sales.l8 + Train_clem_social.l8 + Train_clem_sales.l9 + Train_clem_social.l9 + cons
t + trend
##
##  Train_clem_sales.l1 Train_clem_social.l1  Train_clem_sales.l2
##        -4.531721e-04         6.490940e-01         7.296165e-04
## Train_clem_social.l2  Train_clem_sales.l3 Train_clem_social.l3
##         8.506463e-02         3.243271e-04        -5.984888e-02
##  Train_clem_sales.l4 Train_clem_social.l4  Train_clem_sales.l5
##        -3.998133e-04        -9.679771e-02         6.902133e-05
## Train_clem_social.l5  Train_clem_sales.l6 Train_clem_social.l6
##         1.271834e-01         3.061069e-04        -1.654337e-01
##  Train_clem_sales.l7 Train_clem_social.l7  Train_clem_sales.l8
##        -5.243927e-05         4.190137e-01        -5.272616e-04
## Train_clem_social.l8  Train_clem_sales.l9 Train_clem_social.l9
##         3.081235e-02         4.647388e-05        -2.538254e-01
##                const                trend
##         2.130318e+02         9.260544e+00
##
##
##
## $exo.fcst
## NULL
```

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






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

# Clem sales volume forecast
accuracy(var_train_forecasts$fcst$Train_clem_sales[,1], Test_clem_sales)
```

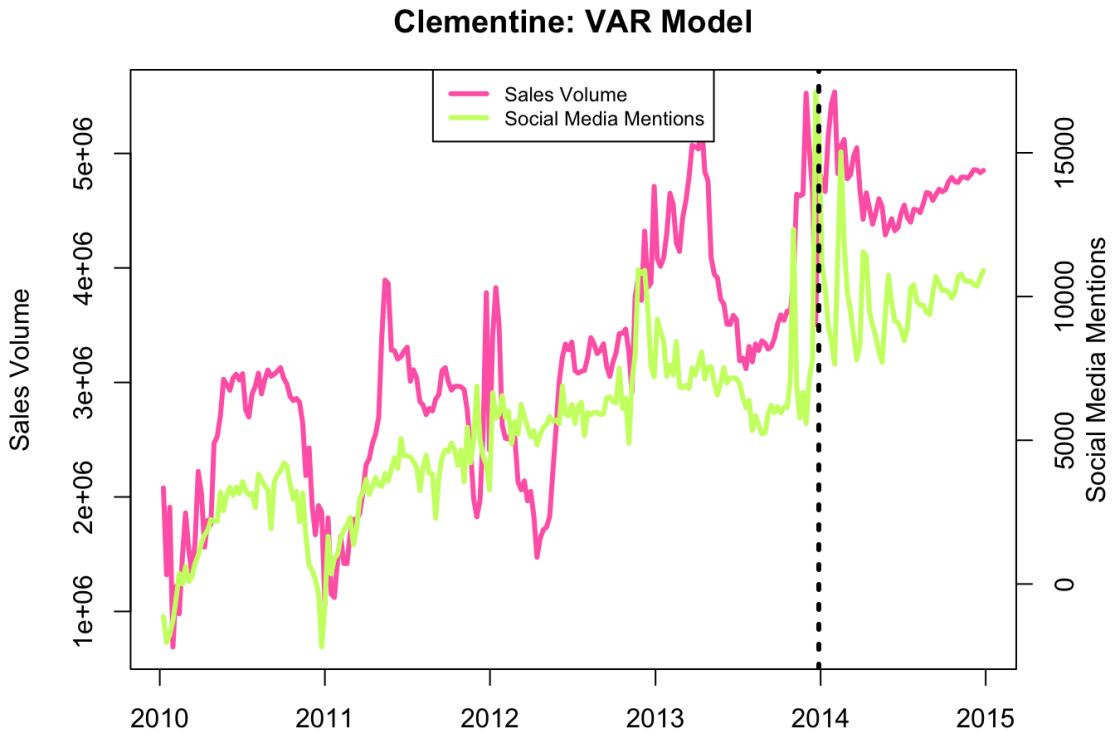
```
##                  ME      RMSE       MAE        MPE     MAPE
## Test set -526193.8 868077.2 781706.3 -15.50979 19.76432
```

```
# Clem social media forecast
accuracy(var_train_forecasts$fcst$Train_clem_social[,1], Test_clem_social)
```

```
##                  ME      RMSE       MAE        MPE     MAPE
## Test set -2690.063 3351.577 2807.876 -45.64167 46.67575
```

Plot Raw Sales and Social media with Forecasts

```
# Append raw + forecast
clem.sales.VAR.All <- append(Train_clem_sales, var_train_forecasts$fcst$Train_clem_sales[,1])
clem.social.VAR.All <- append(Train_clem_social, var_train_forecasts$fcst$Train_clem_social[,1])
clem.week.All <- append(Train_clem_week, Test_clem_week)

clem_df_total <- data.frame(clem.week.All, clem.sales.VAR.All, clem.social.VAR.All)


mar.default <- c(3,3,3,3) + 0.1
par(mar = mar.default + c(0, 1, 0, 0))
plot(clem_df_total[,1:2], type="l",
     ylab="Sales Volume", xlab="Time (Year)",
     lwd=3, main="Clementine: VAR Model", col="hotpink")
par(new=TRUE)
plot(clem_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 Clementines

All_Clem_Forecasts <- cbind.data.frame(Test_week = as.Date(Test_clem_week), AR=forecast.clem.sales.arima$mea
n, VAR=var_train_forecasts$fcst$Train_clem_sales[,1],
                   ACTUAL=Test_clem_sales)
head(All_Clem_Forecasts)
```

```
##    Test_week      AR     VAR  ACTUAL
## 1 2014-01-04 4569266 4758695 3746226
## 2 2014-01-11 4647909 4668234 4060837
## 3 2014-01-18 4646237 5157530 4348938
## 4 2014-01-25 4662379 5435878 4687070
## 5 2014-02-01 4674570 5536223 4250070
## 6 2014-02-08 4687638 4825124 4595462
```

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


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


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

```
## [1] "19.79%"
```

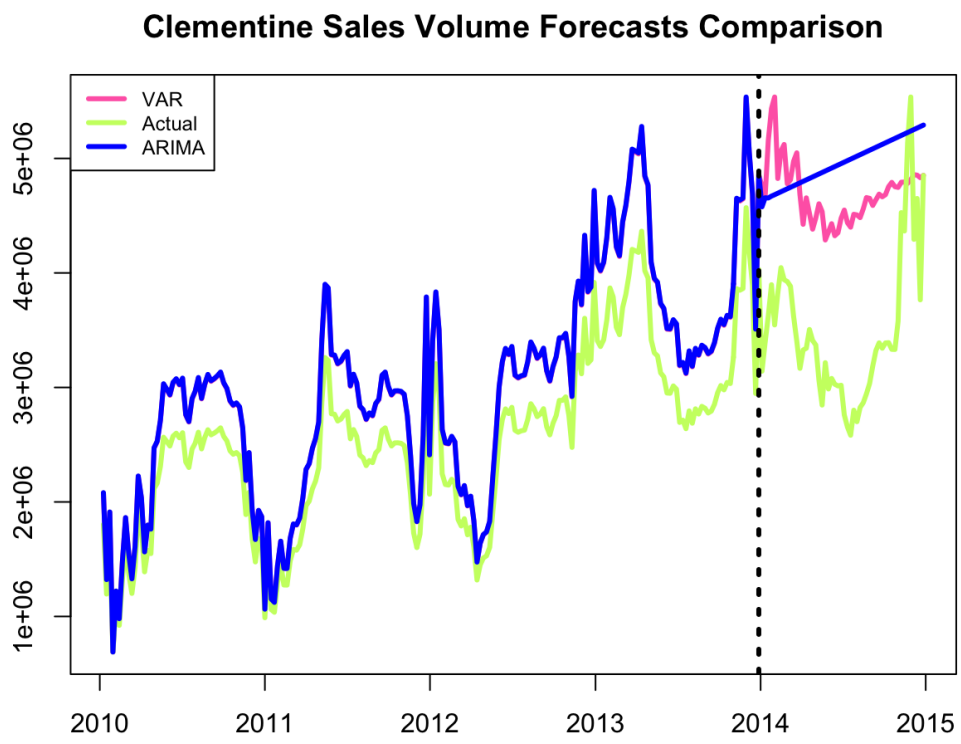
```
paste(round((((ar.clem.sales.mae - var.clem.sales.mae)/ar.clem.sales.mae))*100, digits = 2), "%", sep = "")
```

```
## [1] "16.36%"
```

```
# Create a table to compare RMSE and MAE
accuracy_table <- matrix(c(1082234,790863.8,"26.92%",934652.2,672441,"28.05%"),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 "1082234"  "790863.8" "26.92%"
## MAE  "934652.2" "672441"   "28.05%"
```

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

clem.sales.actual.All <- append(Train_clem_sales, Test_clem_sales)

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

clem_df_total_final <- cbind.data.frame(clem.week.All, clem.sales.VAR.All, clem.sales.actual.All, clem.sale
s.arima.All)


mar.default <- c(3,3,3,3) + 0.1
par(mar = mar.default + c(0, 1, 0, 0))
plot(clem_df_total_final[,1:2], type="l",
     ylab="", xlab="Time (Year)",
     lwd=3, main="Clementine Sales Volume Forecasts Comparison", col="hotpink")
par(new=TRUE)
# Actual data train + test
plot(clem_df_total_final[,3], type="l", col="darkolivegreen1", axes=FALSE,
     ylab="", xlab="", lwd=3)
par(new=TRUE)
# ARIMA Forecast
plot(clem_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 VAR model which includes social media and lagged sales volume as endogenours predictors and google trends as exogenous variables does predict better than ARIMA model series predicting itself.

```
print(accuracy_table)
```

```
##      ARIMA      VAR        Lift in Prediction accuracy
## RMSE "1082234"  "790863.8" "26.92%"
## MAE  "934652.2" "672441"   "28.05%"
```

Based on Root mean squared error and Mean Absolute Error which calculates the prediction error (predicted - actual), from our train and test split, we see increased accuracy of more than 25% when using VAR models.

Predictions for 2015

```
var_clem_forecasts_2015 <- predict(Train_VAR_model_clem, n.ahead = 104, ci = 0.95)

plot(var_clem_forecasts_2015, type = "l", main = "2015 Cupcakes sales + social forecast using train model on
 test set")
```