Predict food trend using social media data

Objective is to predict total sales volume using social media data.

```
require(scales)
```

```
## Loading required package: scales
```

Read datasets for clementine sales and social media.

```
social <- read.csv("/Users/kevalshah/Keval_Backup/University/UChicago/Capstone/Data/clem
entine_social.csv")

# Check summary statistics
#summary(social)
head(social[1:9])
```

```
##   Analysis_Date Blogs Forums Reviews Facebook Twitter Comments
## 1        1/3/10   795    285       6        0     774        0
## 2       1/10/10  1166    253       3        0     661        0
## 3       1/17/10   692    212       4        0     595        0
## 4       1/24/10   638    174       2        0     664        0
## 5       1/31/10   670    223       3        0     416        0
## 6        2/7/10   710    361       1        0     428        0
##   Google.trends.data Total_mentions
## 1                 86           1946
## 2                 65           2148
## 3                 73           1576
## 4                 65           1543
## 5                 46           1358
## 6                 49           1549
```

```
sales <- read.csv("/Users/kevalshah/Keval_Backup/University/UChicago/Capstone/Data/cleme
ntine_sales.csv")

# Check summary statistics
#summary(sales)
head(sales)
```

```
##    Period Scan.Date       Geography    Product  Dollars  Volume
## 1  1/9/10    1/9/10 Total U.S. PROJ Clementine 27659560 5541236
## 2 1/16/10   1/16/10 Total U.S. PROJ Clementine 25247902 5035414
## 3 1/23/10   1/23/10 Total U.S. PROJ Clementine 25617283 5052122
## 4 1/30/10   1/30/10 Total U.S. PROJ Clementine 17500257 3249063
## 5  2/6/10    2/6/10 Total U.S. PROJ Clementine 20281201 3909343
## 6 2/13/10   2/13/10 Total U.S. PROJ Clementine 18320152 3416482
```

Let's make some time series plots for social media mentions and sales data for Clementines.

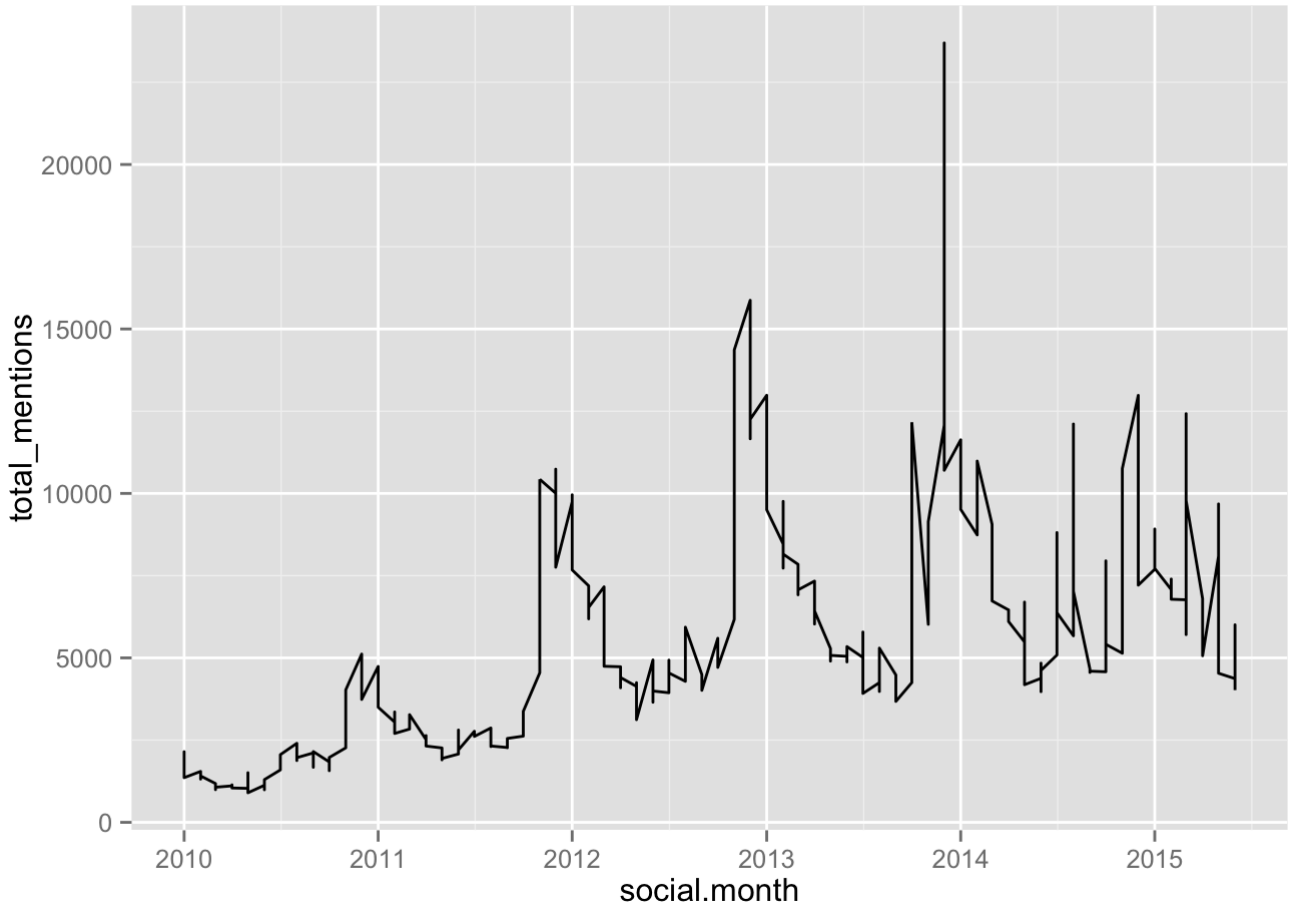
```
# Social media mentions plot

# Date formatting
analysis_date = as.Date(social[,1], "%m/%d/%y")
str(analysis_date)
```

```
##  Date[1:287], format: "2010-01-03" "2010-01-10" "2010-01-17" "2010-01-24" ...
```

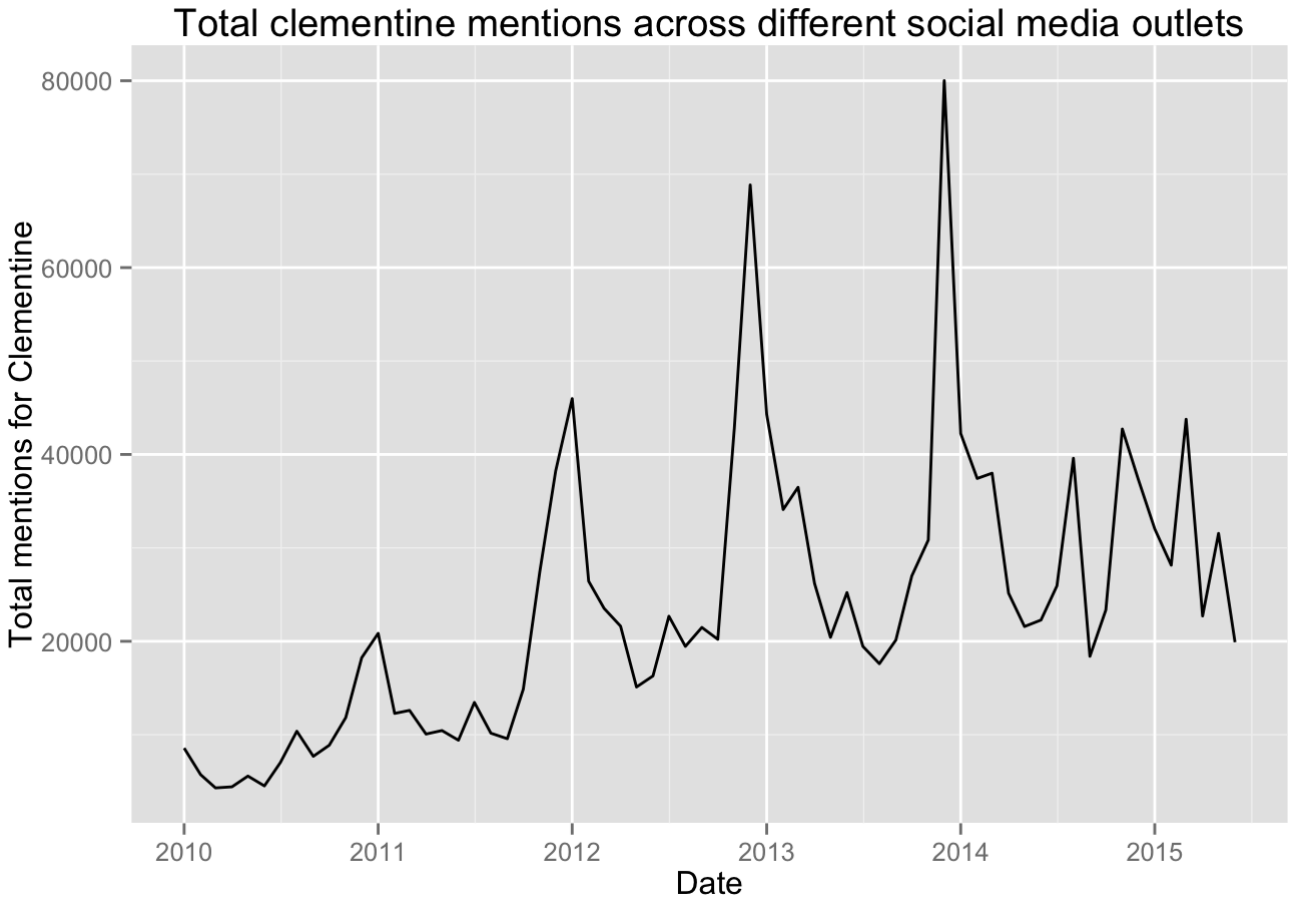
```
# Create a variable for month to aggregate / sum all observations for a particular month

social$month = as.Date(cut(analysis_date, breaks = "month"))
#social

total_mentions = social[,9]

# Create a dataset containing analysis date, month of analysis date and total social med
ia mentions for plotting a observations aggregated at a monthly level.
social1 <- data.frame(analysis_date, social$month, total_mentions)
#social1

require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
# Time series plot by monthly total observations
p <- ggplot(social1, aes(x=social.month, y=total_mentions))
p + geom_line()
```




```
# add up all observations(total_mentions) for the month
p = p + stat_summary(fun.y = sum, geom = "line")
p = p + ggtitle ("Total clementine mentions across different social media outlets")
p + xlab("Date") +  ylab ("Total mentions for Clementine")
```

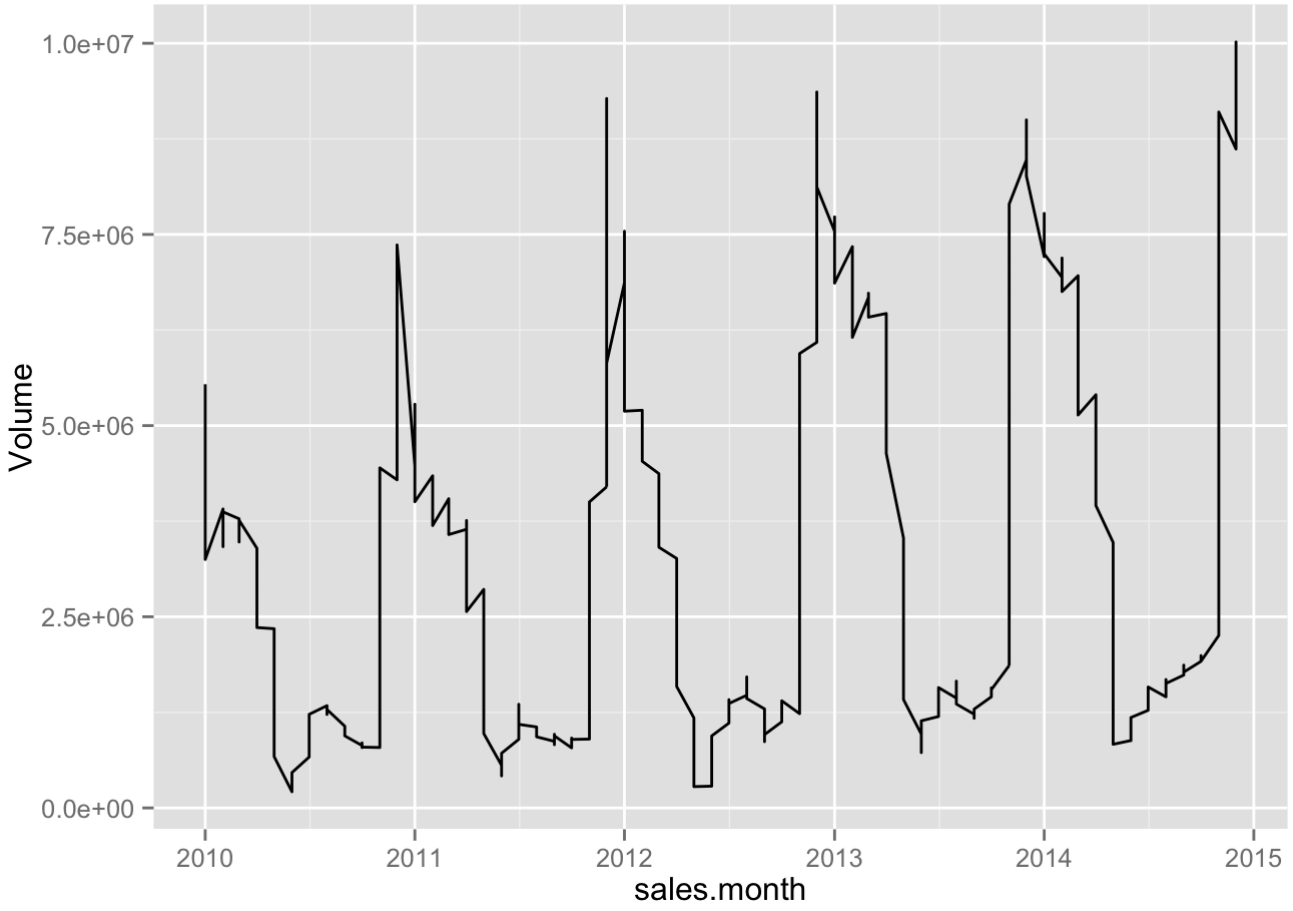



```
#p + scale_x_date(labels = date_format("%y/%m"), breaks = "1 month")

# Plotting time series with curve / trendline
#ggplot(socialts,aes(x=social.month, y=total_mentions)) +
#  geom_point(aes(colour = total_mentions)) +
#  scale_colour_gradient2(low = "blue", mid = "green" , high = "red", midpoint = 4738 )
 +
#  geom_smooth(color = "red",size = 1) +
#  scale_y_continuous(limits = c(899,23695)) +
#  ggtitle ("Total clementine mentions across different social media outlets") +
#  xlab("Date") +  ylab ("Total mentions for Clementine")
```

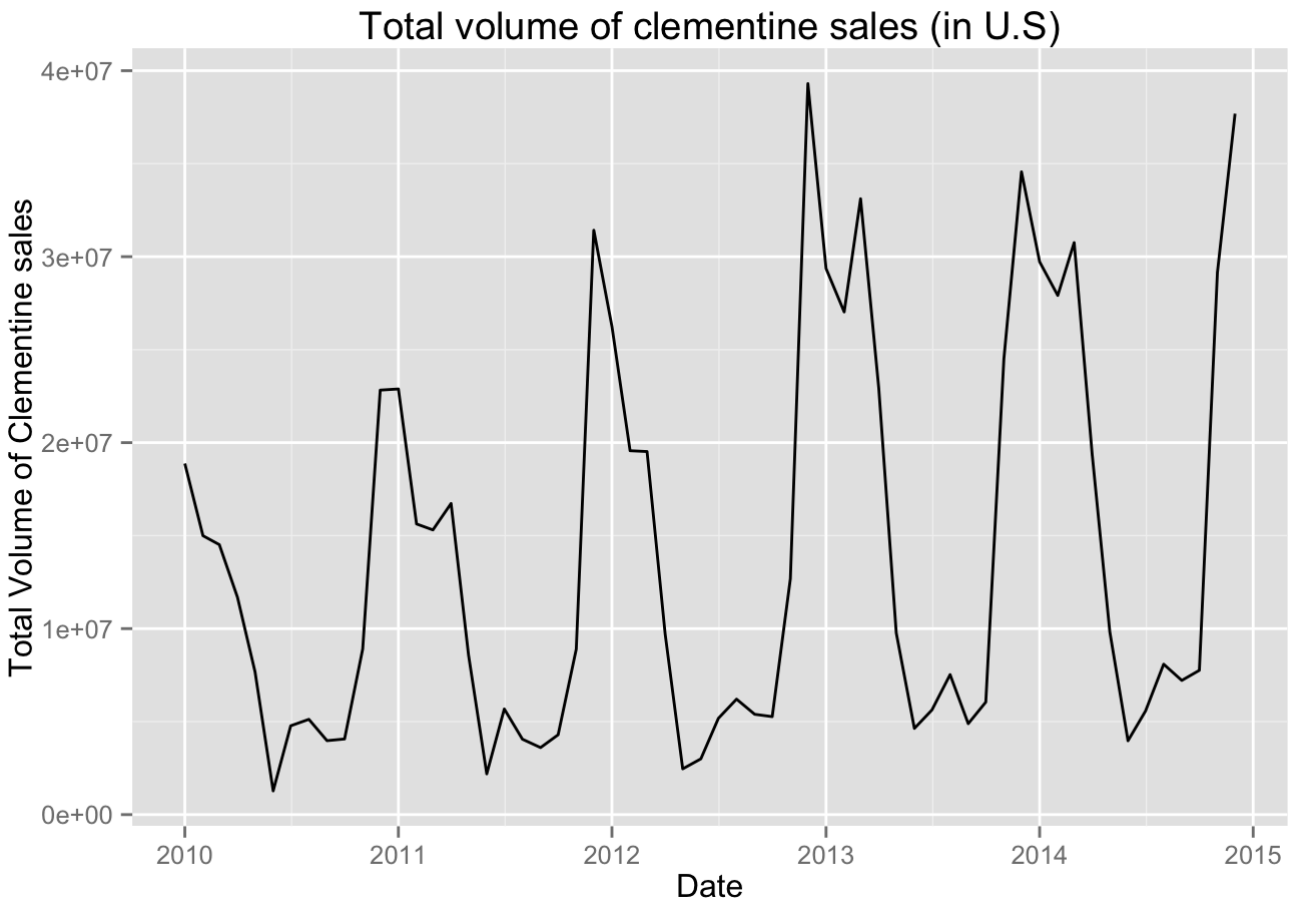
```
# Sales data plot

# Date formatting
period = as.Date(sales[,1], "%m/%d/%y")
str(period)
```

```
##  Date[1:260], format: "2010-01-09" "2010-01-16" "2010-01-23" "2010-01-30" ...
```

```
Volume = sales[,6]

# Create a variable for month to aggregate / sum all observations for a particular month

sales$month = as.Date(cut(period, breaks = "month"))
#sales

# Bind period, sales month and volume into a data frame
sales1 <- data.frame(period, sales$month, Volume)

# Now, let's plot time series of montly aggregated total Clementine sales volume
p <- ggplot(sales1, aes(x=sales.month, y=Volume))
p + geom_line()
```




```
# add up all observations(total_mentions) for the month
p = p + stat_summary(fun.y = sum, geom = "line")
#p = p + scale_y_continuous(limits = c(211151,10017414))
p = p + ggtitle ("Total volume of clementine sales (in U.S)")
p + xlab("Date") +  ylab ("Total Volume of Clementine sales")
```



Remove seasonal component. The remaining time series should be trend and random noise.

Why do we seasonally adjust data?

http://www.bls.gov/cps/seasfaq.htm (http://www.bls.gov/cps/seasfaq.htm)

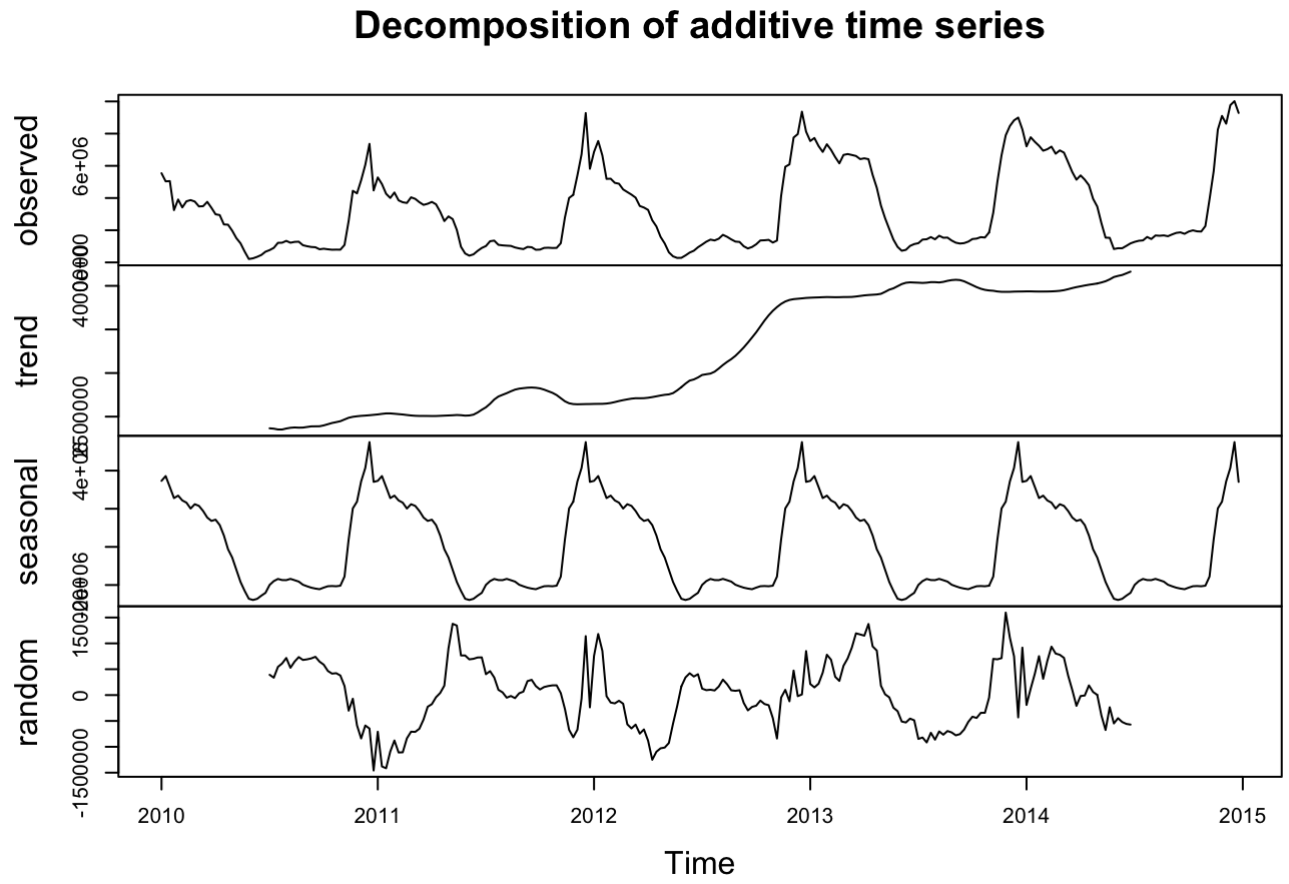
In order to obtain / understand the true underlying trend, we need to seasonally adjust the data. Upon, seasonally adjusting the data we get the trend component, seasonal component and irregular time series component.

Decomposing the time series into trend, seasonal and irregular components.

```
# Columns required time and volume

#Using zoo object
require(zoo)
```

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

```
salests = sales[, c(1,6)]
salestsz = read.zoo(salests, format = "%m/%d/%y", index.column = 1)
View(salestsz)


#Decompose zoo object
salestszDec <- ts(salestsz, frequency=52, start=c(2010, 1))
View(salestszDec)
salestsDec1 <- decompose(salestszDec, type=c("additive"))
#salestsDec1 <- stl(salestszDec, s.window = "periodic")
View(salestszDec)
plot(salestsDec1)
```




```
#plot(salestsDec1, col="darkorange",lwd=3,
#     main = "Clementine: Sales Data Decomposition")


# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series

```
# Check object type
class(salestsDec1$seasonal)
```

```
## [1] "ts"
```

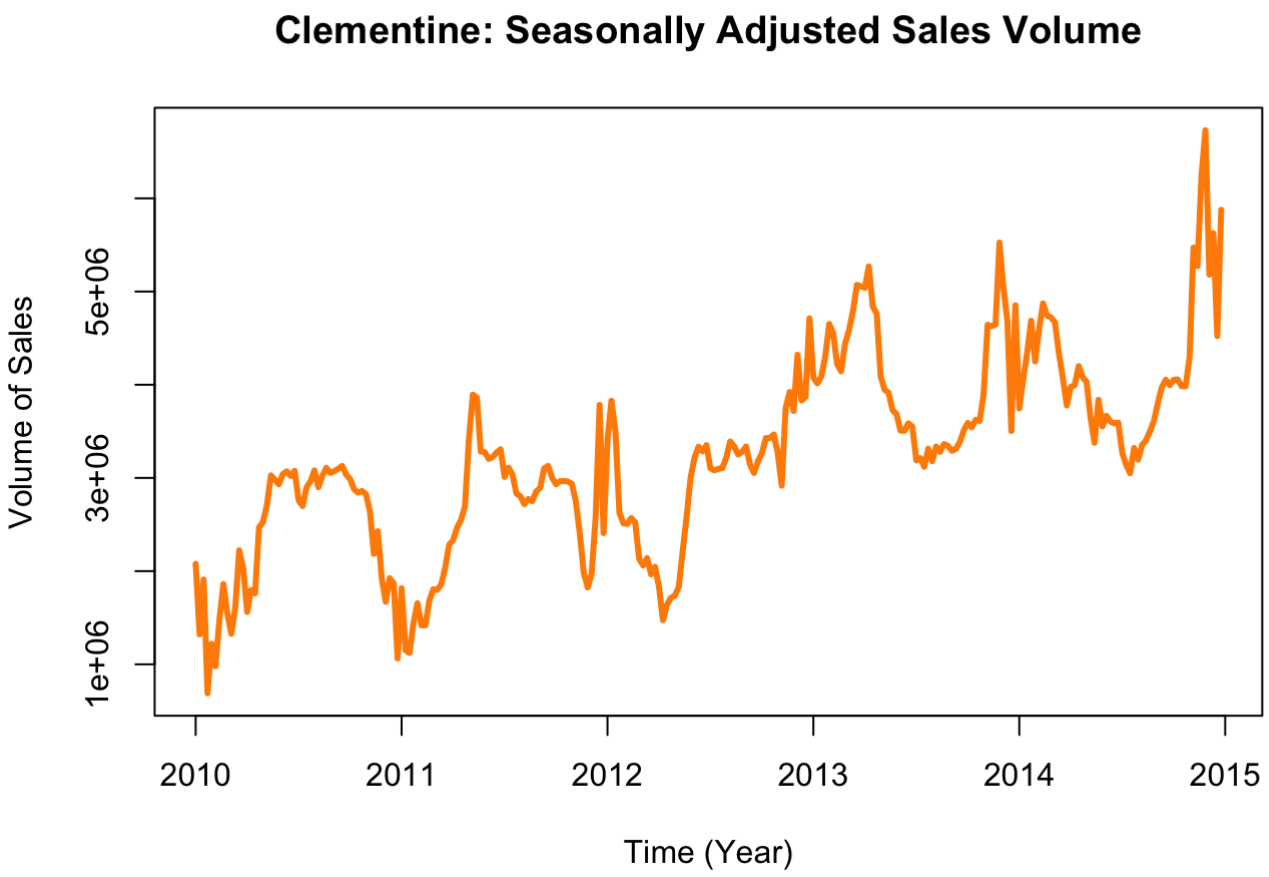
```
class(salestszDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


salestszDec <- as.numeric(unlist(salestszDec))
class(salestszDec)
```

```
## [1] "numeric"
```

```
salestszSeasonallyAdjusted <- salestszDec - salestsDec1$seasonal

#Plot seasonally adjusted time series
plot(salestszSeasonallyAdjusted,
     xlab="Time (Year)", ylab="Volume of Sales",
     main="Clementine: Seasonally Adjusted Sales Volume",
     col="darkorange", lwd=3)
```




```
clemSalesFinal <- cbind(index(salestsz),salestsDec1$x,salestsDec1$seasonal,salestszSeaso
nallyAdjusted)

require(xlsx)
```

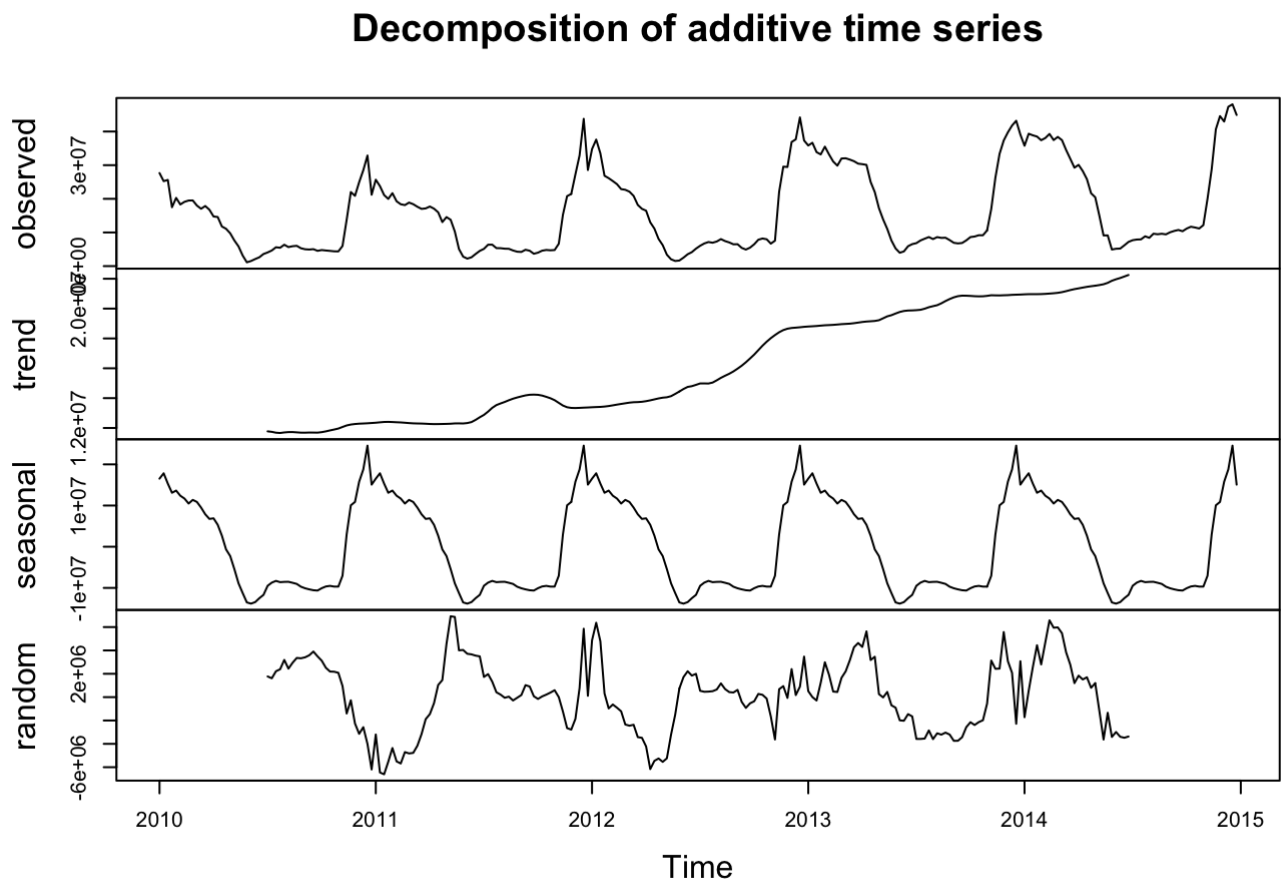
```
## Loading required package: xlsx
## Loading required package: rJava
## Loading required package: xlsxjars
```

```
# Don't re-run, the file will append the current combined file.
write.xlsx(clemSalesFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/Capst
one/Data/clem_sales_seasonally_adjusted_1.xlsx", sheetName="sheet1", col.names=TRUE, ro
w.names=FALSE, append=FALSE, showNA=TRUE)

index(salestsz)
```

```
##   [1] "2010-01-09" "2010-01-16" "2010-01-23" "2010-01-30" "2010-02-06"
##   [6] "2010-02-13" "2010-02-20" "2010-02-27" "2010-03-06" "2010-03-13"
##  [11] "2010-03-20" "2010-03-27" "2010-04-03" "2010-04-10" "2010-04-17"
##  [16] "2010-04-24" "2010-05-01" "2010-05-08" "2010-05-15" "2010-05-22"
##  [21] "2010-05-29" "2010-06-05" "2010-06-12" "2010-06-19" "2010-06-26"
##  [26] "2010-07-03" "2010-07-10" "2010-07-17" "2010-07-24" "2010-07-31"
##  [31] "2010-08-07" "2010-08-14" "2010-08-21" "2010-08-28" "2010-09-04"
##  [36] "2010-09-11" "2010-09-18" "2010-09-25" "2010-10-02" "2010-10-09"
##  [41] "2010-10-16" "2010-10-23" "2010-10-30" "2010-11-06" "2010-11-13"
##  [46] "2010-11-20" "2010-11-27" "2010-12-04" "2010-12-11" "2010-12-18"
##  [51] "2010-12-25" "2011-01-01" "2011-01-08" "2011-01-15" "2011-01-22"
##  [56] "2011-01-29" "2011-02-05" "2011-02-12" "2011-02-19" "2011-02-26"
##  [61] "2011-03-05" "2011-03-12" "2011-03-19" "2011-03-26" "2011-04-02"
##  [66] "2011-04-09" "2011-04-16" "2011-04-23" "2011-04-30" "2011-05-07"
##  [71] "2011-05-14" "2011-05-21" "2011-05-28" "2011-06-04" "2011-06-11"
##  [76] "2011-06-18" "2011-06-25" "2011-07-02" "2011-07-09" "2011-07-16"
##  [81] "2011-07-23" "2011-07-30" "2011-08-06" "2011-08-13" "2011-08-20"
##  [86] "2011-08-27" "2011-09-03" "2011-09-10" "2011-09-17" "2011-09-24"
##  [91] "2011-10-01" "2011-10-08" "2011-10-15" "2011-10-22" "2011-10-29"
##  [96] "2011-11-05" "2011-11-12" "2011-11-19" "2011-11-26" "2011-12-03"
## [101] "2011-12-10" "2011-12-17" "2011-12-24" "2011-12-31" "2012-01-07"
## [106] "2012-01-14" "2012-01-21" "2012-01-28" "2012-02-04" "2012-02-11"
## [111] "2012-02-18" "2012-02-25" "2012-03-03" "2012-03-10" "2012-03-17"
## [116] "2012-03-24" "2012-03-31" "2012-04-07" "2012-04-14" "2012-04-21"
## [121] "2012-04-28" "2012-05-05" "2012-05-12" "2012-05-19" "2012-05-26"
## [126] "2012-06-02" "2012-06-09" "2012-06-16" "2012-06-23" "2012-06-30"
## [131] "2012-07-07" "2012-07-14" "2012-07-21" "2012-07-28" "2012-08-04"
## [136] "2012-08-11" "2012-08-18" "2012-08-25" "2012-09-01" "2012-09-08"
## [141] "2012-09-15" "2012-09-22" "2012-09-29" "2012-10-06" "2012-10-13"
## [146] "2012-10-20" "2012-10-27" "2012-11-03" "2012-11-10" "2012-11-17"
## [151] "2012-11-24" "2012-12-01" "2012-12-08" "2012-12-15" "2012-12-22"
## [156] "2012-12-29" "2013-01-05" "2013-01-12" "2013-01-19" "2013-01-26"
## [161] "2013-02-02" "2013-02-09" "2013-02-16" "2013-02-23" "2013-03-02"
## [166] "2013-03-09" "2013-03-16" "2013-03-23" "2013-03-30" "2013-04-06"
## [171] "2013-04-13" "2013-04-20" "2013-04-27" "2013-05-04" "2013-05-11"
## [176] "2013-05-18" "2013-05-25" "2013-06-01" "2013-06-08" "2013-06-15"
## [181] "2013-06-22" "2013-06-29" "2013-07-06" "2013-07-13" "2013-07-20"
## [186] "2013-07-27" "2013-08-03" "2013-08-10" "2013-08-17" "2013-08-24"
## [191] "2013-08-31" "2013-09-07" "2013-09-14" "2013-09-21" "2013-09-28"
## [196] "2013-10-05" "2013-10-12" "2013-10-19" "2013-10-26" "2013-11-02"
## [201] "2013-11-09" "2013-11-16" "2013-11-23" "2013-11-30" "2013-12-07"
## [206] "2013-12-14" "2013-12-21" "2013-12-28" "2014-01-04" "2014-01-11"
## [211] "2014-01-18" "2014-01-25" "2014-02-01" "2014-02-08" "2014-02-15"
## [216] "2014-02-22" "2014-03-01" "2014-03-08" "2014-03-15" "2014-03-22"
## [221] "2014-03-29" "2014-04-05" "2014-04-12" "2014-04-19" "2014-04-26"
## [226] "2014-05-03" "2014-05-10" "2014-05-17" "2014-05-24" "2014-05-31"
## [231] "2014-06-07" "2014-06-14" "2014-06-21" "2014-06-28" "2014-07-05"
## [236] "2014-07-12" "2014-07-19" "2014-07-26" "2014-08-02" "2014-08-09"
## [241] "2014-08-16" "2014-08-23" "2014-08-30" "2014-09-06" "2014-09-13"
## [246] "2014-09-20" "2014-09-27" "2014-10-04" "2014-10-11" "2014-10-18"
## [251] "2014-10-25" "2014-11-01" "2014-11-08" "2014-11-15" "2014-11-22"
## [256] "2014-11-29" "2014-12-06" "2014-12-13" "2014-12-20" "2014-12-27"
```

Decompose the time series and adjust seasonality for $$ amounts from clementine sales.

```
# Columns required time and Price

#Using zoo object for time indexing
require(zoo)
clemSalesPrice = sales[, c(1,5)]
clemSalesPricez = read.zoo(clemSalesPrice, format = "%m/%d/%y", index.column = 1)


#Decompose zoo object
clemSalesPricezDec <- ts(clemSalesPricez, frequency=52, start=c(2010, 1))
clemSalesPriceDec1 <- decompose(clemSalesPricezDec, type=c("additive"))
plot(clemSalesPriceDec1)
```




```
# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series (Price)

```
# Check object type
class(clemSalesPriceDec1$seasonal)
```

```
## [1] "ts"
```

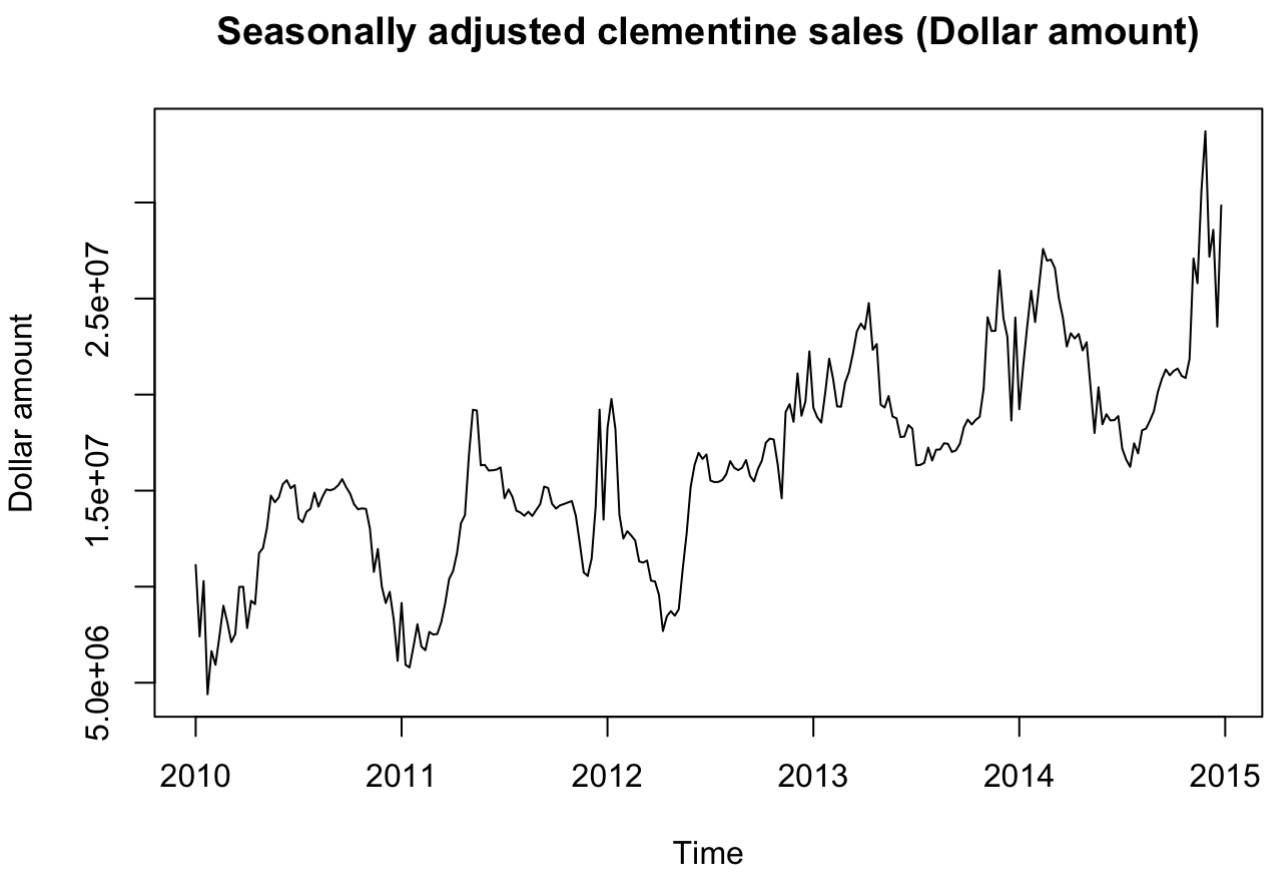
```
class(clemSalesPricezDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


clemSalesPricezDec <- as.numeric(unlist(clemSalesPricezDec))
class(clemSalesPricezDec)
```

```
## [1] "numeric"
```

```
clemSalesPriceSeasonallyAdjusted <- clemSalesPricezDec - clemSalesPriceDec1$seasonal

#Plot seasonally adjusted time series
plot(clemSalesPriceSeasonallyAdjusted, xlab="Time", ylab="Dollar amount", main="Seasonal
ly adjusted clementine sales (Dollar amount)")
```

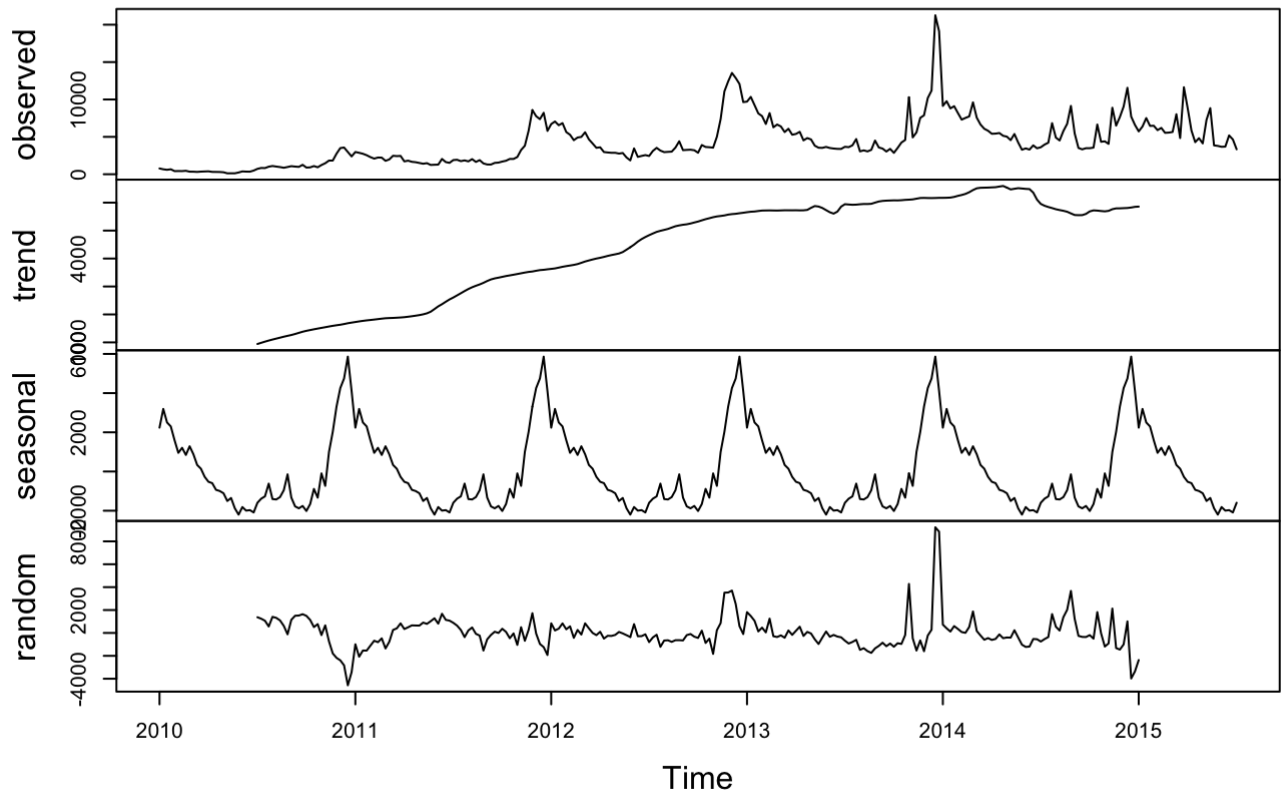


```
# Combine columns
clemsalesPriceFinal <- cbind(clemSalesPriceDec1$x,clemSalesPriceDec1$seasonal,clemSalesP
riceSeasonallyAdjusted)


require(xlsx)
# Don't re-run, the file will append the current combined file. Write original price, se
asonal component and seasonally adjusted values to en excel file

write.xlsx(clemsalesPriceFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/
Capstone/Data/clem_sales_seasonally_adjusted_price.xlsx", sheetName="sheet1", col.names=
TRUE, row.names=FALSE, append=FALSE, showNA=TRUE)
```

Decompose the time series and adjust seasonality for social media mentions.

Twitter mentions

```
# Columns required time and twitter mentions

#Using zoo object for time indexing
require(zoo)
clemTwitter = social[, c(1,6)]
clemTwitterz = read.zoo(clemTwitter, format = "%m/%d/%y", index.column = 1)
View(clemSalesPricez)


#Decompose zoo object
clemTwitterzDec <- ts(clemTwitterz, frequency=52, start=c(2010, 1))
clemTwitterDec1 <- decompose(clemTwitterzDec, type=c("additive"))
plot(clemTwitterDec1)
```




```
# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series (Twitter)

```
# Check object type
class(clemTwitterDec1$seasonal)
```

```
## [1] "ts"
```

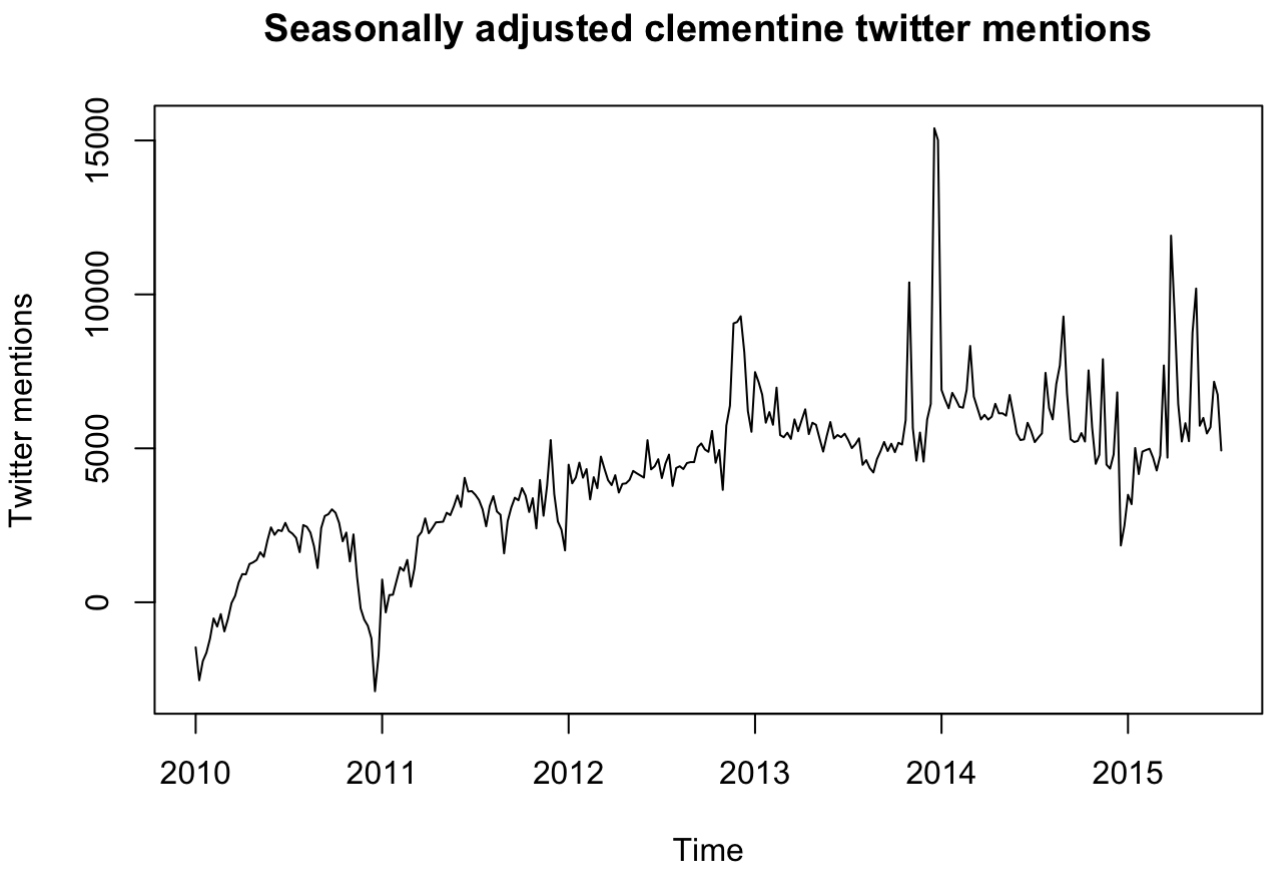
```
class(clemTwitterzDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


clemTwitterzDec <- as.numeric(unlist(clemTwitterzDec))
class(clemTwitterzDec)
```

```
## [1] "numeric"
```

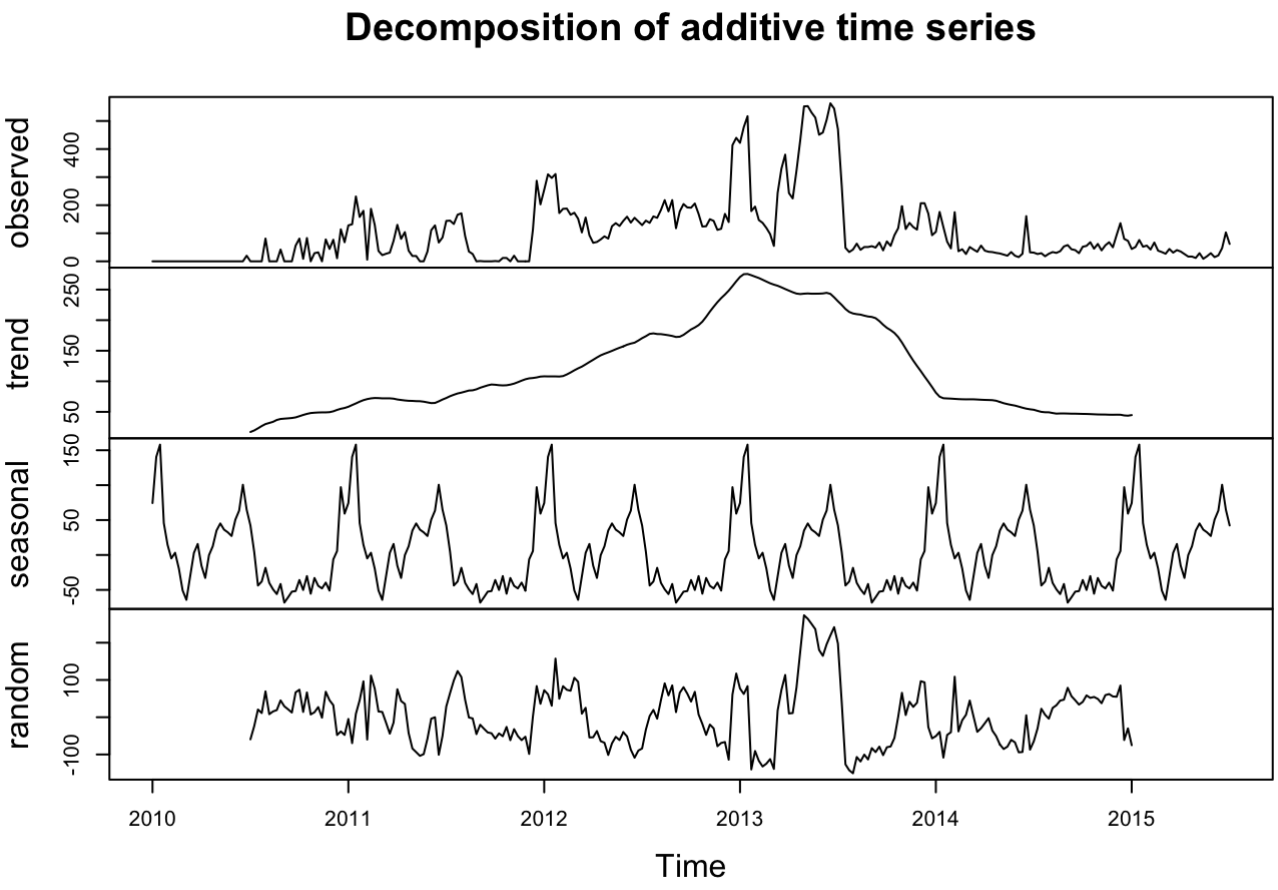
```
clemTwitterSeasonallyAdjusted <- clemTwitterzDec - clemTwitterDec1$seasonal

# Plot seasonally adjusted time series
plot(clemTwitterSeasonallyAdjusted, xlab="Time", ylab="Twitter mentions", main="Seasonal
ly adjusted clementine twitter mentions")
```




```
# Combine columns
clemTwitterFinal <- cbind(clemTwitterDec1$x,clemTwitterDec1$seasonal,clemTwitterSeasonal
lyAdjusted)

require(xlsx)
# Don't re-run, the file will append the current combined file. Write original price, se
asonal component and seasonally adjusted values to en excel file

write.xlsx(clemTwitterFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/Cap
stone/Data/clem_social_seasonally_adjusted_twitter.xlsx", sheetName="sheet1", col.names=
TRUE, row.names=FALSE, append=FALSE, showNA=TRUE)
```

Facebook mentions

```
# Columns required time and facebook mentions

#Using zoo object for time indexing
require(zoo)
clemFacebook = social[, c(1,5)]
clemFacebookz = read.zoo(clemFacebook, format = "%m/%d/%y", index.column = 1)
View(clemFacebookz)


#Decompose zoo object
clemFacebookzDec <- ts(clemFacebookz, frequency=52, start=c(2010, 1))
clemFacebookDec1 <- decompose(clemFacebookzDec, type=c("additive"))
plot(clemFacebookDec1)
```




```
# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series (Facebook)

```
# Check object type
class(clemFacebookDec1$seasonal)
```

```
## [1] "ts"
```

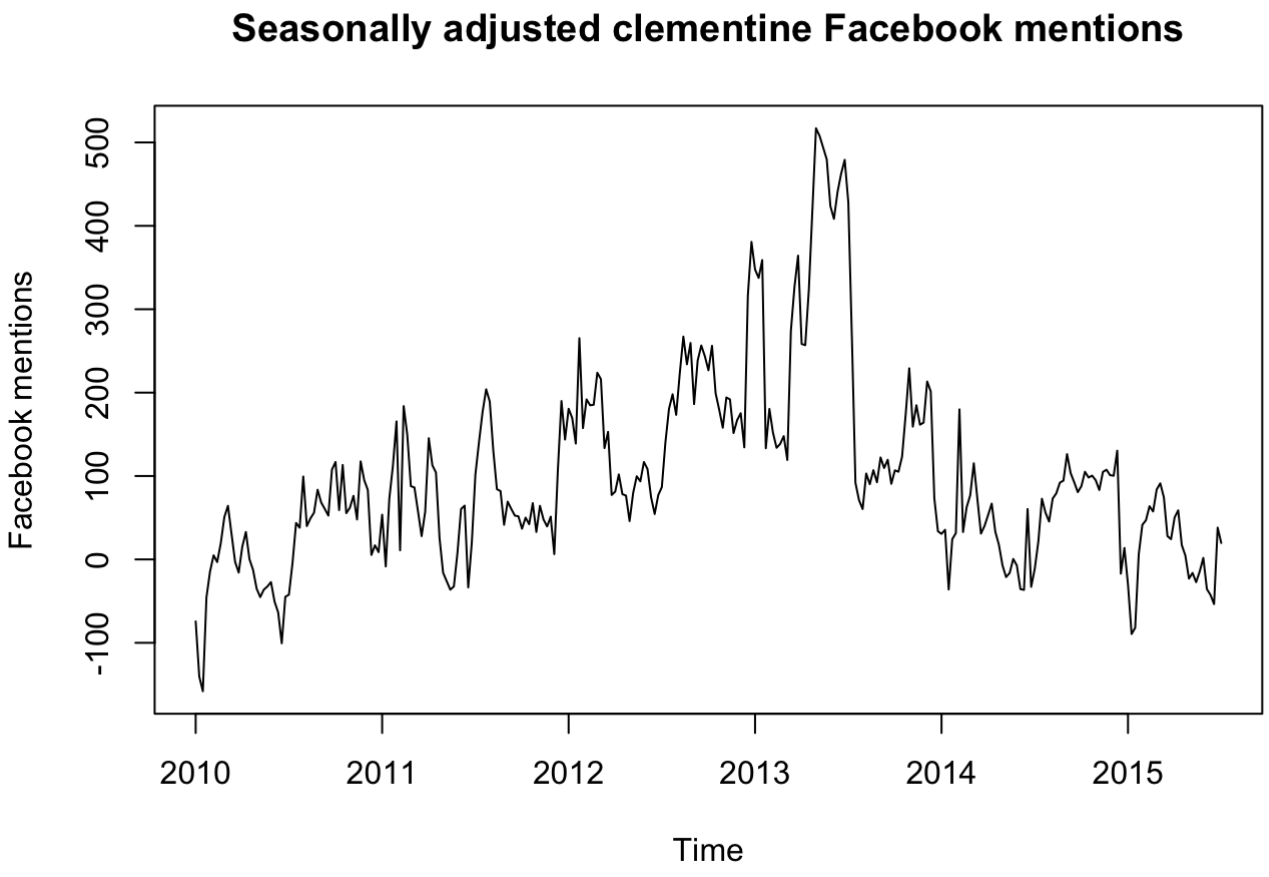
```
class(clemFacebookzDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


clemFacebookzDec <- as.numeric(unlist(clemFacebookzDec))
class(clemFacebookzDec)
```

```
## [1] "numeric"
```

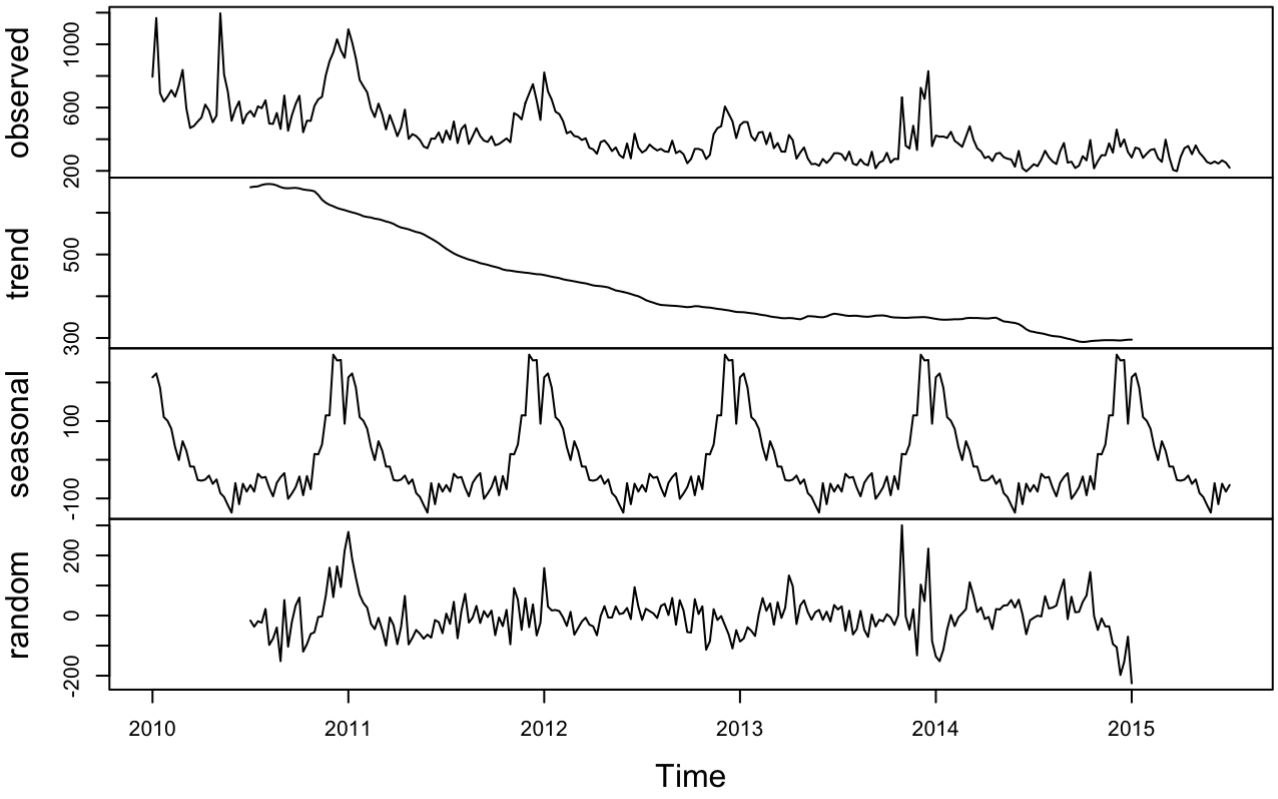
```
clemFacebookSeasonallyAdjusted <- clemFacebookzDec - clemFacebookDec1$seasonal

#Plot seasonally adjusted time series
plot(clemFacebookSeasonallyAdjusted, xlab="Time", ylab="Facebook mentions", main="Season
ally adjusted clementine Facebook mentions")
```



```
#Combine data
clemFacebookFinal <- cbind(clemFacebookDec1$x,clemFacebookDec1$seasonal,clemFacebookSeas
onallyAdjusted)

require(xlsx)
# Don't re-run, the file will append the current combined file. Write original price, se
asonal component and seasonally adjusted values to en excel file

write.xlsx(clemFacebookFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/Ca
pstone/Data/clem_social_seasonally_adjusted_fb.xlsx", sheetName="sheet1", col.names=TRUE
, row.names=FALSE, append=FALSE, showNA=TRUE)
```

Blogs mentions

```
# Columns required time and blog mentions

#Using zoo object for time indexing
require(zoo)
clemBlog = social[, c(1,2)]
clemBlogz = read.zoo(clemBlog, format = "%m/%d/%y", index.column = 1)
View(clemBlogz)


#Decompose zoo object
clemBlogzDec <- ts(clemBlogz, frequency=52, start=c(2010, 1))
clemBlogDec1 <- decompose(clemBlogzDec, type=c("additive"))
plot(clemBlogDec1)
```




```
# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series (Blog)

```
# Check object type
class(clemBlogDec1$seasonal)
```

```
## [1] "ts"
```

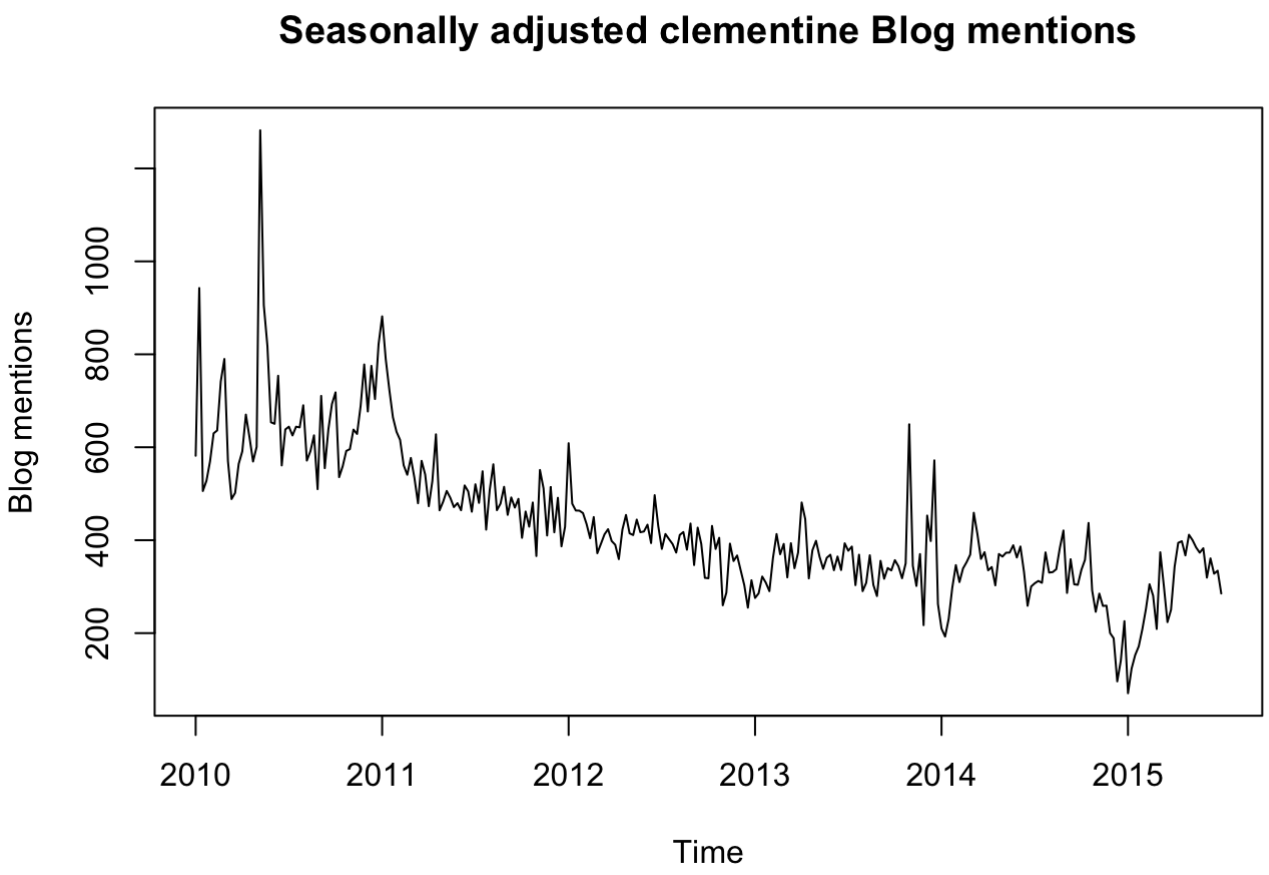
```
class(clemBlogzDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


clemBlogzDec <- as.numeric(unlist(clemBlogzDec))
class(clemBlogzDec)
```

```
## [1] "numeric"
```

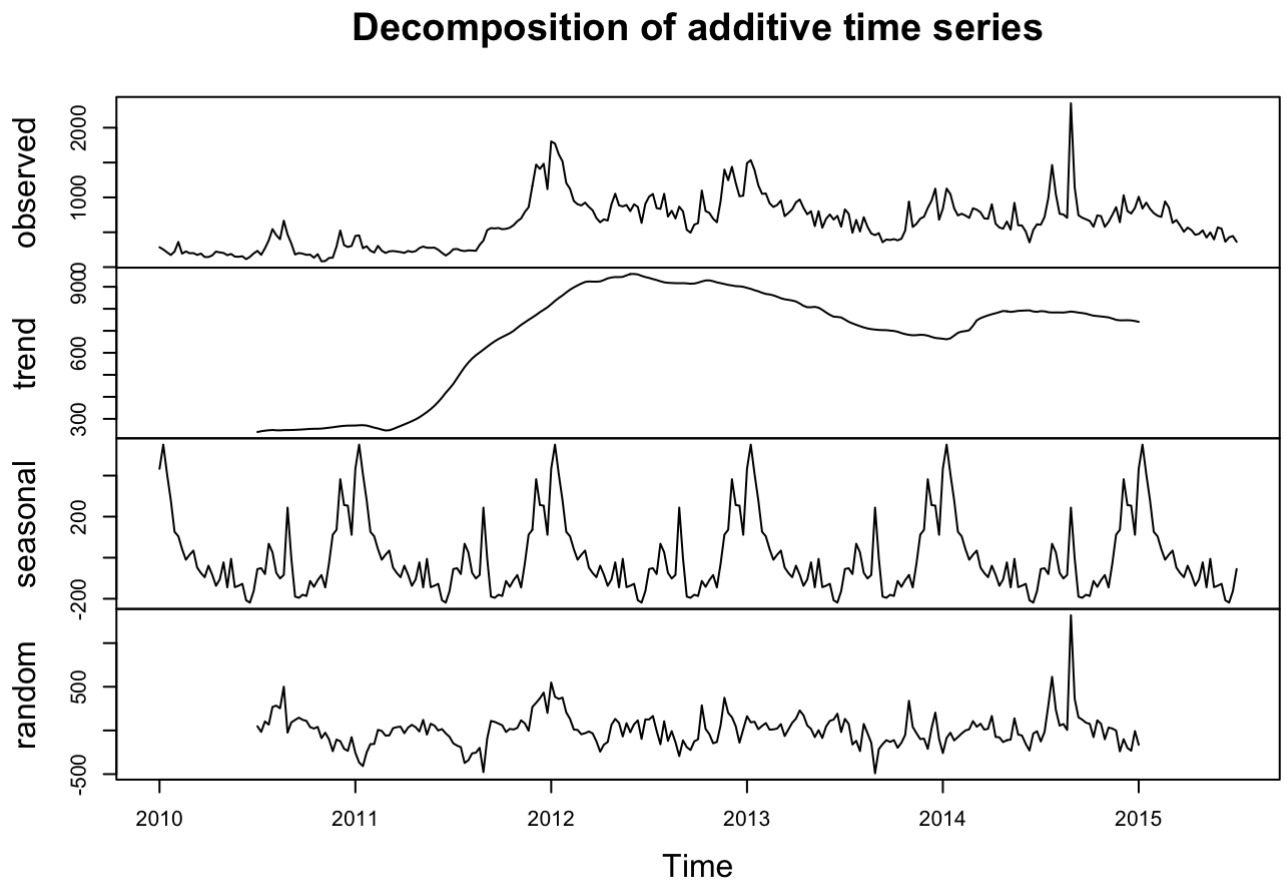
```
clemBlogSeasonallyAdjusted <- clemBlogzDec - clemBlogDec1$seasonal

#Plot seasonally adjusted time series
plot(clemBlogSeasonallyAdjusted, xlab="Time", ylab="Blog mentions", main="Seasonally adj
usted clementine Blog mentions")
```




```
# Combine data
clemBlogFinal <- cbind(clemBlogDec1$x,clemBlogDec1$seasonal,clemBlogSeasonallyAdjusted)


require(xlsx)
# Don't re-run, the file will append the current combined file. Write original price, se
asonal component and seasonally adjusted values to en excel file

write.xlsx(clemBlogFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/Capsto
ne/Data/clem_social_seasonally_adjusted_blog.xlsx", sheetName="sheet1", col.names=TRUE,
row.names=FALSE, append=FALSE, showNA=TRUE)
```

Forums mentions

```
# Columns required time and forums mentions

#Using zoo object for time indexing
require(zoo)
clemForums = social[, c(1,3)]
clemForumsz = read.zoo(clemForums, format = "%m/%d/%y", index.column = 1)
View(clemForumsz)


#Decompose zoo object
clemForumszDec <- ts(clemForumsz, frequency=52, start=c(2010, 1))
clemForumsDec1 <- decompose(clemForumszDec, type=c("additive"))
plot(clemForumsDec1)
```




```
# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series (Forums)

```
# Check object type
class(clemForumsDec1$seasonal)
```

```
## [1] "ts"
```

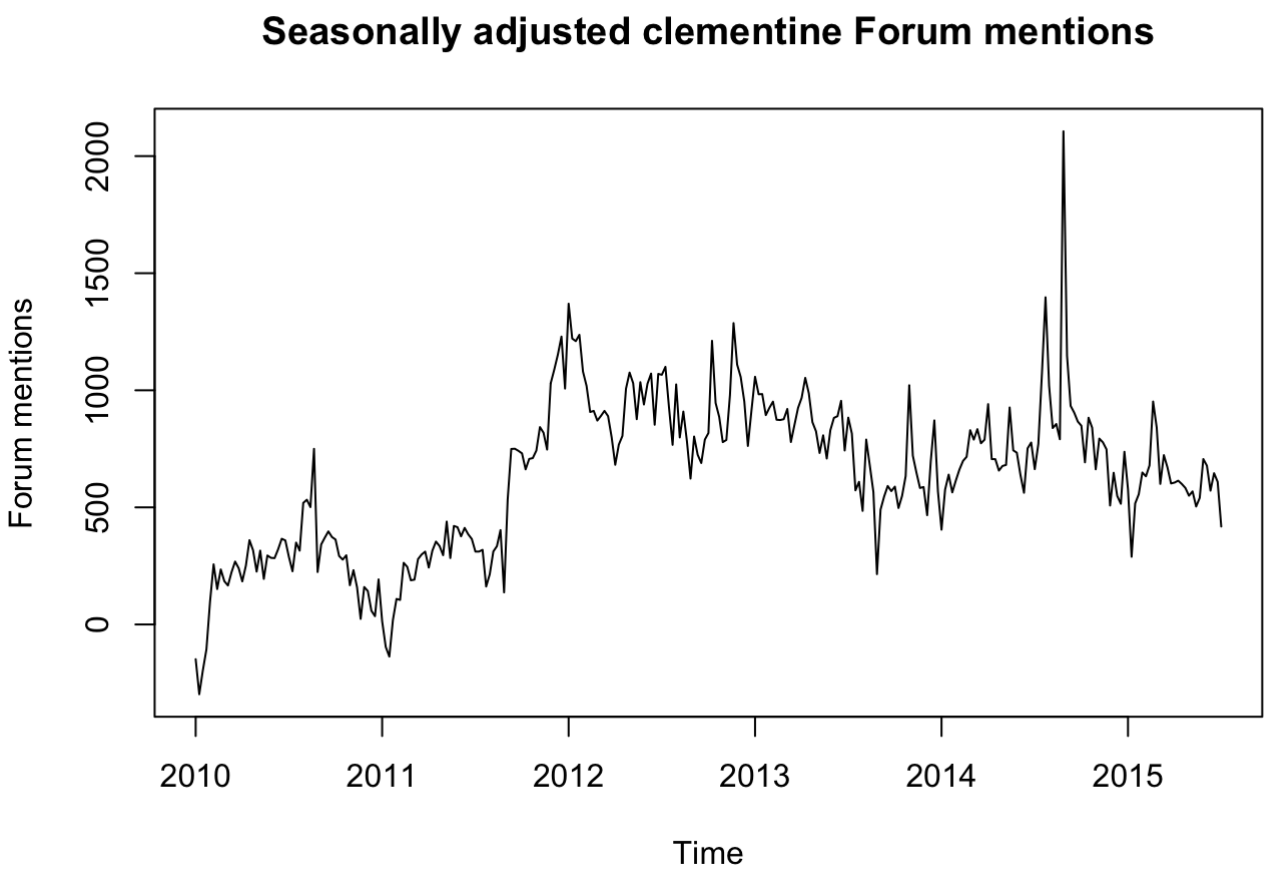
```
class(clemForumszDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


clemForumszDec <- as.numeric(unlist(clemForumszDec))
class(clemForumszDec)
```

```
## [1] "numeric"
```

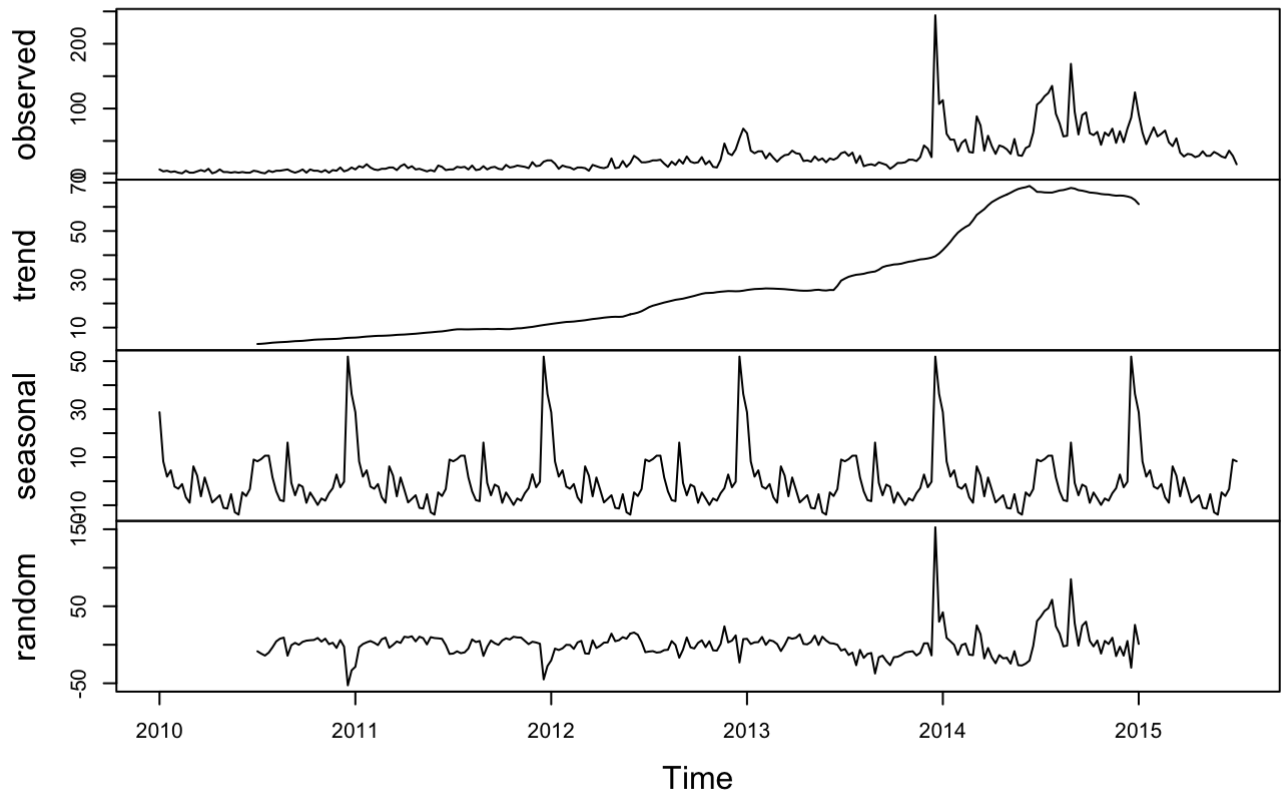
```
clemForumsSeasonallyAdjusted <- clemForumszDec - clemForumsDec1$seasonal

#Plot seasonally adjusted time series
plot(clemForumsSeasonallyAdjusted, xlab="Time", ylab="Forum mentions", main="Seasonally
 adjusted clementine Forum mentions")
```



```
#Combine data
clemForumsFinal <- cbind(clemForumsDec1$x,clemForumsDec1$seasonal,clemForumsSeasonallyAd
justed)
require(xlsx)
# Don't re-run, the file will append the current combined file. Write original price, se
asonal component and seasonally adjusted values to en excel file

write.xlsx(clemForumsFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/Caps
tone/Data/clem_social_seasonally_adjusted_forums.xlsx", sheetName="sheet1", col.names=TR
UE, row.names=FALSE, append=FALSE, showNA=TRUE)
```

Reviews mention

```
# Columns required time and blog mentions

#Using zoo object for time indexing
require(zoo)
clemReviews = social[, c(1,4)]
clemReviewsz = read.zoo(clemReviews, format = "%m/%d/%y", index.column = 1)
View(clemReviewsz)


#Decompose zoo object
clemReviewszDec <- ts(clemReviewsz, frequency=52, start=c(2010, 1))
clemReviewsDec1 <- decompose(clemReviewszDec, type=c("additive"))
plot(clemReviewsDec1)
```




```
# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series (Reviews)

```
# Check object type
class(clemReviewsDec1$seasonal)
```

```
## [1] "ts"
```

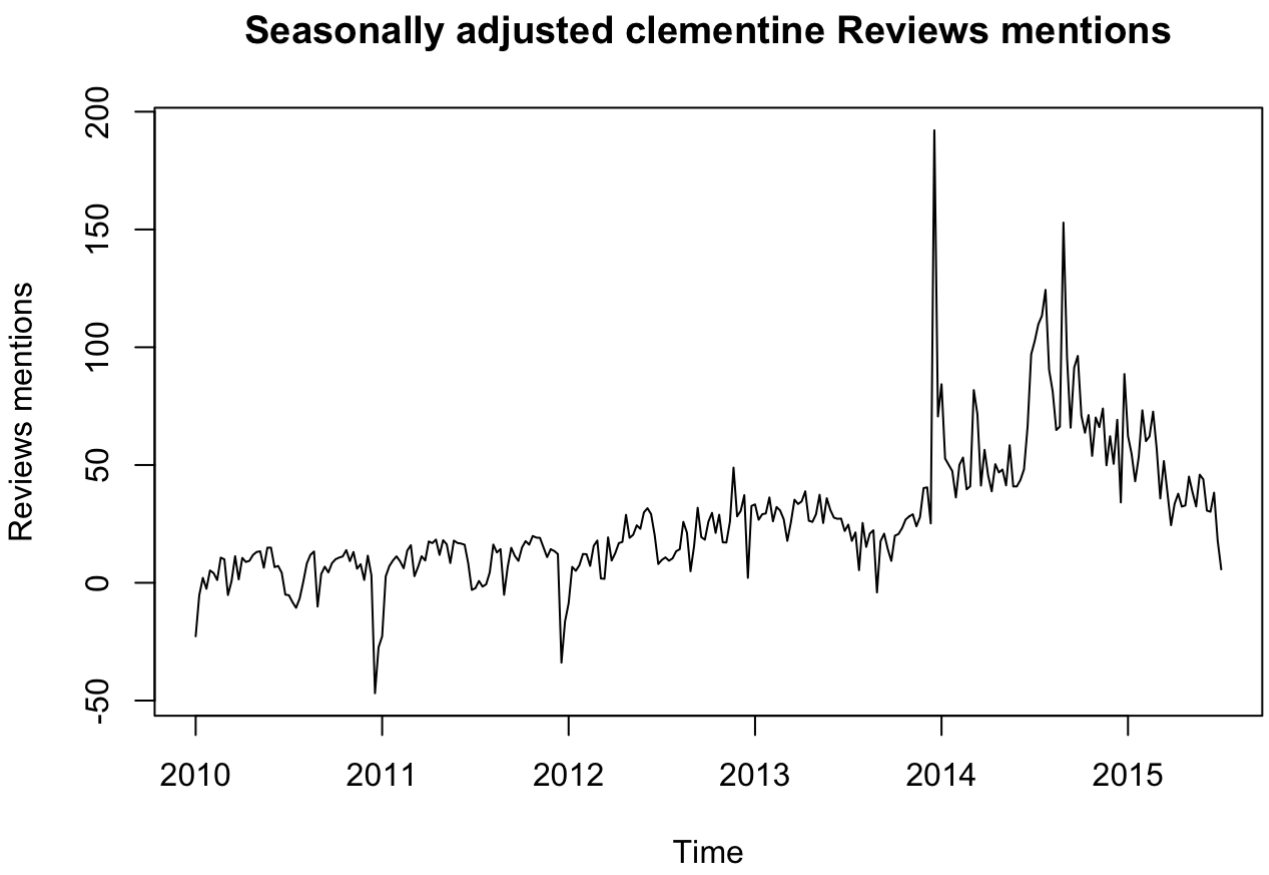
```
class(clemReviewszDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


clemReviewszDec <- as.numeric(unlist(clemReviewszDec))
class(clemReviewszDec)
```

```
## [1] "numeric"
```

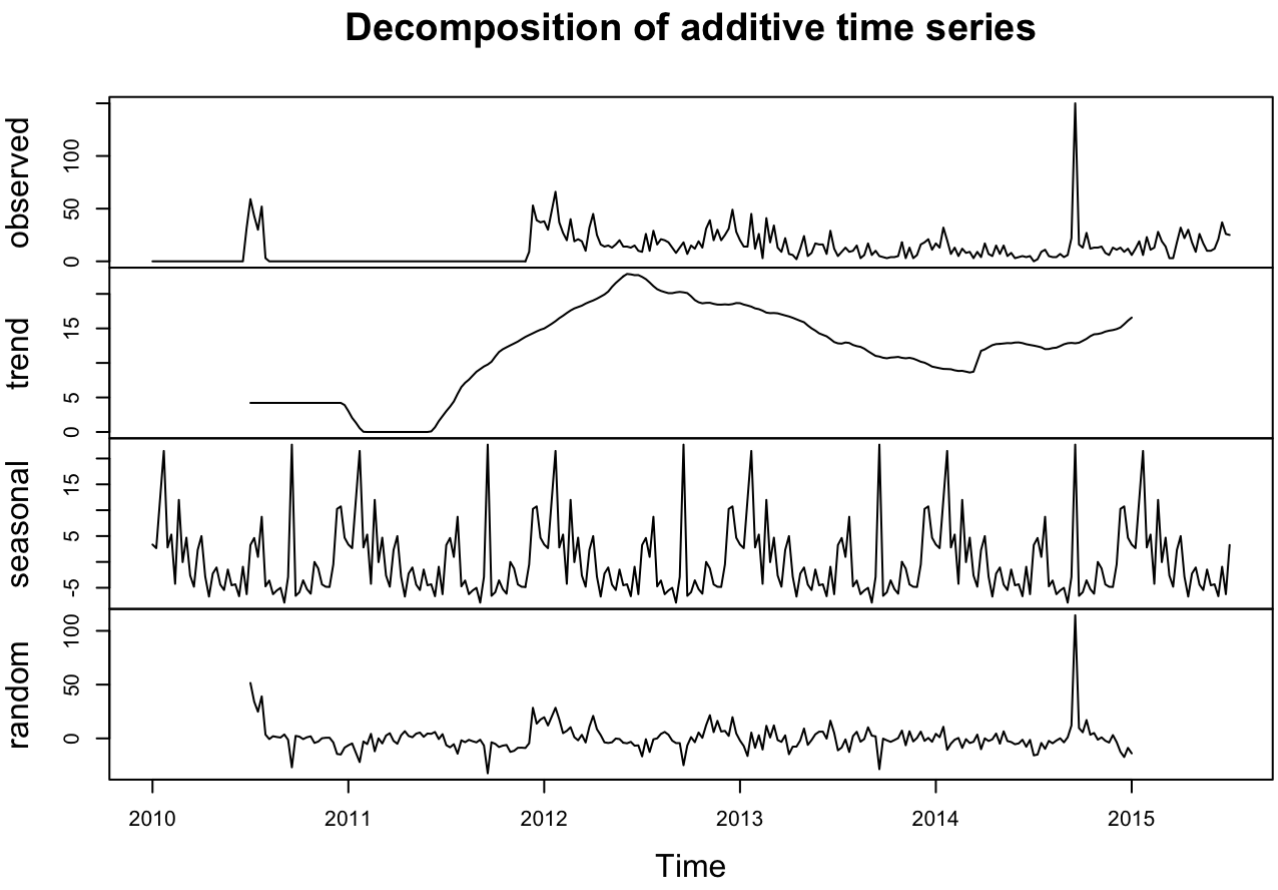
```
clemReviewsSeasonallyAdjusted <- clemReviewszDec - clemReviewsDec1$seasonal

#Plot seasonally adjusted time series
plot(clemReviewsSeasonallyAdjusted, xlab="Time", ylab="Reviews mentions", main="Seasonal
ly adjusted clementine Reviews mentions")
```




```
#Combine data
clemReviewsFinal <- cbind(clemReviewsDec1$x,clemReviewsDec1$seasonal,clemReviewsSeasonal
lyAdjusted)

require(xlsx)
# Don't re-run, the file will append the current combined file. Write original price, se
asonal component and seasonally adjusted values to en excel file

write.xlsx(clemReviewsFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/Cap
stone/Data/clem_social_seasonally_adjusted_reviews.xlsx", sheetName="sheet1", col.names=
TRUE, row.names=FALSE, append=FALSE, showNA=TRUE)
```

Comments mention

```
# Columns required time and comments mentions

#Using zoo object for time indexing
require(zoo)
clemComments = social[, c(1,7)]
clemCommentsz = read.zoo(clemComments, format = "%m/%d/%y", index.column = 1)
View(clemCommentsz)


#Decompose zoo object
clemCommentszDec <- ts(clemCommentsz, frequency=52, start=c(2010, 1))
clemCommentsDec1 <- decompose(clemCommentszDec, type=c("additive"))
plot(clemCommentsDec1)
```




```
# Frequency = 52 weeks in year. i.e 52 # of observations in one year.
#salestsDec <- ts(salests[,2], frequency=52, start=c(2010, 1), end=c(2014,12))
#salestsDec <- decompose(salestsDec, type=c("additive"))
#plot(salestsDec)
```

Plot seasonally adjusted time series (Comments)

```
# Check object type
class(clemCommentsDec1$seasonal)
```

```
## [1] "ts"
```

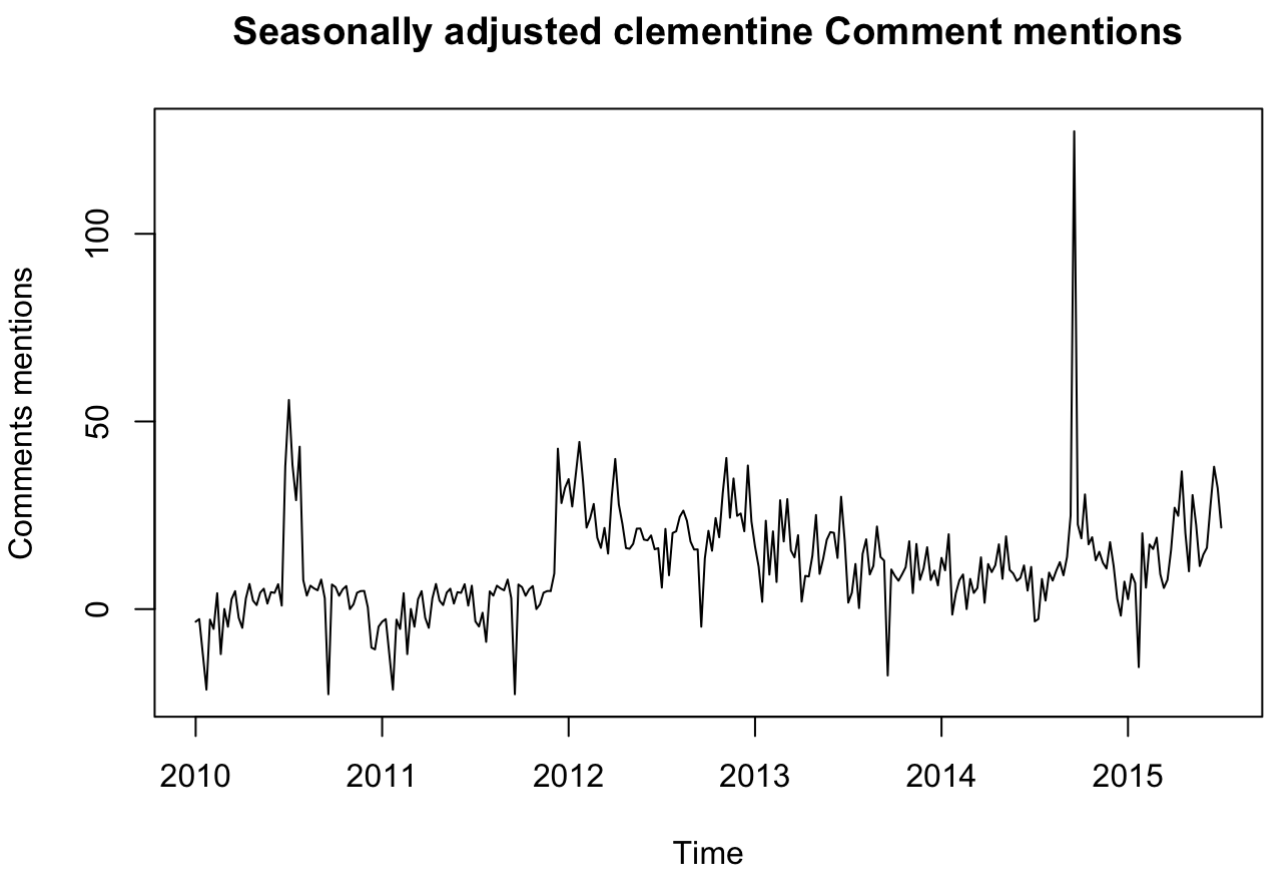
```
class(clemCommentszDec)
```

```
## [1] "ts"
```

```
# Both are 'ts' object, so we need to convert to numeric to be able to perform math oper
ation of subtracting seasonal component from the original series.


clemCommentszDec <- as.numeric(unlist(clemCommentszDec))
class(clemCommentszDec)
```

```
## [1] "numeric"
```

```
clemCommentsSeasonallyAdjusted <- clemCommentszDec - clemCommentsDec1$seasonal

#Plot seasonally adjusted time series
plot(clemCommentsSeasonallyAdjusted, xlab="Time", ylab="Comments mentions", main="Season
ally adjusted clementine Comment mentions")
```



```
#Combine data
clemCommentsFinal <- cbind(clemCommentsDec1$x,clemCommentsDec1$seasonal,clemCommentsSeas
onallyAdjusted)

require(xlsx)
# Don't re-run, the file will append the current combined file. Write original price, se
asonal component and seasonally adjusted values to en excel file

write.xlsx(clemCommentsFinal, file="/Users/kevalshah/Keval_Backup/University/UChicago/Ca
pstone/Data/clem_social_seasonally_adjusted_comments.xlsx", sheetName="sheet1", col.name
s=TRUE, row.names=FALSE, append=FALSE, showNA=TRUE)
```

Indexing can be an effective means of normalizing data to a common starting point and observing how variables change over time relative to each other.

Indexing: Modifying two or more numeric data series so that the resulting series start at the same value and change at the same rate as the unmodified series.

Normalizing data for Platform Growth - The total volume of tweets / FB posts and so on for each of the social media platforms. In the case of the twitter data here, the search term was actually the total # of times the word “the” was tweeted (so as to provide a search term), but can be considered a proxy for total activity.

Read platform growth data

```
platform <- read.csv("/Users/kevalshah/Keval_Backup/University/UChicago/Capstone/Data/Pl
atform_growth.csv")
```

Steps taken in Excel to Index / Normalize seasonally adjusted sales volume and social media mentions data.

- Platform Dynamics data (Volume of activity on each social media platform) indexed to a more recent data where the data looks reasonable for each of the social media platforms. So, it was indexed to an average of observations for the Month of January of 2012.
- Pivot the data to Filter out 2008 and 2009 data so that we’re left with data from beginning of 2010 to match with our seasonally adjusted social media mentions data.
- Since the data the original social media mentions data and sales datas are at weekly level, Indexed PD data was rolled up to weekly level and an average of that week was calculated and used as an observation.
- The seasonally adjusted social media mentions data was then divided by an average of weekly Platform dynamics data calculated in previous step to obtain a normalized value.
- Percent values of normalized data obtained from previous step i.e dividing seasonally adjusted social media mentions were calculated.
- Seasonally adjusted sales volume was indexed to an average on weekly (count 4) observations for Jan’12 (to keep it consistent with PD)
- Calculate 7 Day Moving Average of final percent values of indexed sales volume and seasonally adjusted indexed normalized social media mentions data and plot the trend lines.
- Something to consider: feature scaling techniques so that social media mentions for each of the series are within a particular range.

Next steps:

Read final cleaned up dataset of sales volume + social media mentions data and build a linear regression model in time series context (?) to derive the coefficients.

Pseudo code:

Sales volume ~ twitter + fb + blogs + forums + reviews + comments