SalesForecasting using TimeSeries Model

Thomas K John

November 18, 2017

Problem Statement:

Forecasting for next 12 months ie., from Jan 2016 to Dec 2106 using Time series model for the three categories - MenClothing, WomenClothing and OthersClothing.

Preprocessing

Clearing the environment variabes

```
rm(list = ls(all = TRUE))
```

Setting the working directory

```
setwd("I:\\DATA-SCIENCE\\SalesForecasting")
```

Libraries used

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.1
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(imputeTS)
```

```
## Warning: package 'imputeTS' was built under R version 3.4.1
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.4.1
```

Reading the train and test data

```
sales_data= read.csv("Train.csv")
```

Understanding the dataset with str() and summary()

```
str(sales_data)
```

```
## 'data.frame':    252 obs. of  4 variables:
##  $ Year                       : int  2009 2009 2009 2009 2009 2009 2009 2009 2009 2009 ...
##  $ Month                      : int  1 1 1 2 2 2 3 3 3 4 ...
##  $ ProductCategory            : Factor w/ 3 levels "MenClothing",..: 3 1 2 3 1 2 3 1 2 3 ...
##  $ Sales.In.ThousandDollars.: int  1755 524 936 1729 496 859 2256 542 921 2662 ...
```

```
summary(sales_data)
```

```
##       Year          Month            ProductCategory
##  Min.   :2009   Min.   : 1.00   MenClothing  :84
##  1st Qu.:2010   1st Qu.: 3.75   OtherClothing:84
##  Median :2012   Median : 6.50   WomenClothing:84
##  Mean   :2012   Mean   : 6.50
##  3rd Qu.:2014   3rd Qu.: 9.25
##  Max.   :2015   Max.   :12.00
##
##  Sales.In.ThousandDollars.
##  Min.   : 471
##  1st Qu.: 714
##  Median :1136
##  Mean   :1747
##  3rd Qu.:2804
##  Max.   :5874
##  NA's   :13
```

Viewing the first 10 rows

```
head(sales_data)
```

```
##   Year Month ProductCategory Sales.In.ThousandDollars.
## 1 2009     1   WomenClothing                      1755
## 2 2009     1     MenClothing                       524
## 3 2009     1   OtherClothing                       936
## 4 2009     2   WomenClothing                      1729
## 5 2009     2     MenClothing                       496
## 6 2009     2   OtherClothing                       859
```

Converting the date into Date datatype

```
sales_data$Date = paste(sales_data$Year, sales_data$Month,"1", sep ="/")
sales_data$Date= as.Date(sales_data$Date,"%Y/%m/%d")
```

Splitting the data into training and validation

```
train_data=sales_data[which(sales_data$Date <="2014/12/1"),]
validation_data = sales_data[which(sales_data$Date > "2014/12/1"),]
```

Verifying the train and validation

```r
summary(train_data)
```

```
##       Year          Month          ProductCategory
##  Min.   :2009   Min.   : 1.00   MenClothing  :72
##  1st Qu.:2010   1st Qu.: 3.75   OtherClothing:72
##  Median :2012   Median : 6.50   WomenClothing:72
##  Mean   :2012   Mean   : 6.50
##  3rd Qu.:2013   3rd Qu.: 9.25
##  Max.   :2014   Max.   :12.00
##
##  Sales.In.ThousandDollars.      Date
##  Min.   : 471.0            Min.   :2009-01-01
##  1st Qu.: 710.8            1st Qu.:2010-06-23
##  Median :1080.5            Median :2011-12-16
##  Mean   :1702.4            Mean   :2011-12-16
##  3rd Qu.:2750.8            3rd Qu.:2013-06-08
##  Max.   :5664.0            Max.   :2014-12-01
##  NA's   :12
```

```r
summary(validation_data)
```

```
##       Year          Month          ProductCategory
##  Min.   :2015   Min.   : 1.00   MenClothing  :12
##  1st Qu.:2015   1st Qu.: 3.75   OtherClothing:12
##  Median :2015   Median : 6.50   WomenClothing:12
##  Mean   :2015   Mean   : 6.50
##  3rd Qu.:2015   3rd Qu.: 9.25
##  Max.   :2015   Max.   :12.00
##
##  Sales.In.ThousandDollars.      Date
##  Min.   : 560              Min.   :2015-01-01
##  1st Qu.: 758              1st Qu.:2015-03-24
##  Median :1300              Median :2015-06-16
##  Mean   :2005              Mean   :2015-06-16
##  3rd Qu.:3732              3rd Qu.:2015-09-08
##  Max.   :5874              Max.   :2015-12-01
##  NA's   :1
```

Viewing the first and last rows in train and validation data

```r
head(train_data)
```

```
##   Year Month ProductCategory Sales.In.ThousandDollars.       Date
## 1 2009     1   WomenClothing                      1755 2009-01-01
## 2 2009     1     MenClothing                       524 2009-01-01
## 3 2009     1   OtherClothing                       936 2009-01-01
## 4 2009     2   WomenClothing                      1729 2009-02-01
## 5 2009     2     MenClothing                       496 2009-02-01
## 6 2009     2   OtherClothing                       859 2009-02-01
```

```r
tail(train_data)
```

```
##     Year Month ProductCategory Sales.In.ThousandDollars.       Date
## 211 2014    11   WomenClothing                      4525 2014-11-01
## 212 2014    11     MenClothing                       803 2014-11-01
## 213 2014    11   OtherClothing                      1468 2014-11-01
## 214 2014    12   WomenClothing                      5664 2014-12-01
## 215 2014    12     MenClothing                      1070 2014-12-01
## 216 2014    12   OtherClothing                      1967 2014-12-01
```

```
head(validation_data)
```

```
##     Year Month ProductCategory Sales.In.ThousandDollars.       Date
## 217 2015     1   WomenClothing                      3041 2015-01-01
## 218 2015     1     MenClothing                       560 2015-01-01
## 219 2015     1   OtherClothing                      1190 2015-01-01
## 220 2015     2   WomenClothing                      3646 2015-02-01
## 221 2015     2     MenClothing                       602 2015-02-01
## 222 2015     2   OtherClothing                      1210 2015-02-01
```

```
tail(validation_data)
```

```
##     Year Month ProductCategory Sales.In.ThousandDollars.       Date
## 247 2015    11   WomenClothing                      4401 2015-11-01
## 248 2015    11     MenClothing                       643 2015-11-01
## 249 2015    11   OtherClothing                      1478 2015-11-01
## 250 2015    12   WomenClothing                      5874 2015-12-01
## 251 2015    12     MenClothing                       967 2015-12-01
## 252 2015    12   OtherClothing                      1680 2015-12-01
```

Spliting the Train data into three categories : 1. MenClothing, 2. WomenClothing, 3. OthersClothing

```
train_data_men = train_data[which(train_data$ProductCategory == "MenClothing"),]
train_data_women = train_data[which(train_data$ProductCategory == "WomenClothing"),]
train_data_others = train_data[which(train_data$ProductCategory == "OtherClothing"),]
```

Spliting the validation data into three categories: 1. MenClothing, 2. WomenClothing, 3. OthersClothing

```
validation_data_men = validation_data[which(validation_data$ProductCategory == "MenClothing"),]
validation_data_women = validation_data[which(validation_data$ProductCategory == "WomenClothing"),]
validation_data_others = validation_data[which(validation_data$ProductCategory == "OtherClothing"),]
```

Spliting the whole data into three categories: This will be used for the final prediction

```
total_data_men = sales_data[which(sales_data$ProductCategory == "MenClothing"),]
total_data_women = sales_data[which(sales_data$ProductCategory == "WomenClothing"),]
total_data_others = sales_data[which(sales_data$ProductCategory == "OtherClothing"),]
```

Interpolation of the sales data

Interpolation of the Train data with "linear"

```
train_data_men_linear = na.interpolation(train_data_men$Sales.In.ThousandDollars.,option="linear")
train_data_women_linear = na.interpolation(train_data_women$Sales.In.ThousandDollars.,option="linear")
train_data_others_linear = na.interpolation(train_data_others$Sales.In.ThousandDollars.,option="linear"
```

Interpolation of the total data with linear

```
total_data_men_linear = na.interpolation(total_data_men$Sales.In.ThousandDollars.,option="linear")
total_data_women_linear = na.interpolation(total_data_women$Sales.In.ThousandDollars.,option="linear")
total_data_others_linear = na.interpolation(total_data_others$Sales.In.ThousandDollars.,option="linear"
```

Converting the sales price from total data into Timeseries

```
total_data_men_linear_ts = ts(total_data_men_linear, frequency = 12, start = c(2009,1,1))
total_data_women_linear_ts = ts(total_data_women_linear, frequency = 12, start = c(2009,1,1))
total_data_others_linear_ts = ts(total_data_others_linear, frequency = 12, start = c(2009,1,1))
```

Converting the sales price from train data into Timeseries (Without interpolation)

```
train_data_men_ts = ts(train_data_men$Sales.In.ThousandDollars., frequency = 12, start = c(2009,1,1))
train_data_women_ts = ts(train_data_women$Sales.In.ThousandDollars., frequency = 12, start = c(2009,1,1
train_data_others_ts = ts(train_data_others$Sales.In.ThousandDollars., frequency = 12, start = c(2009,1
```

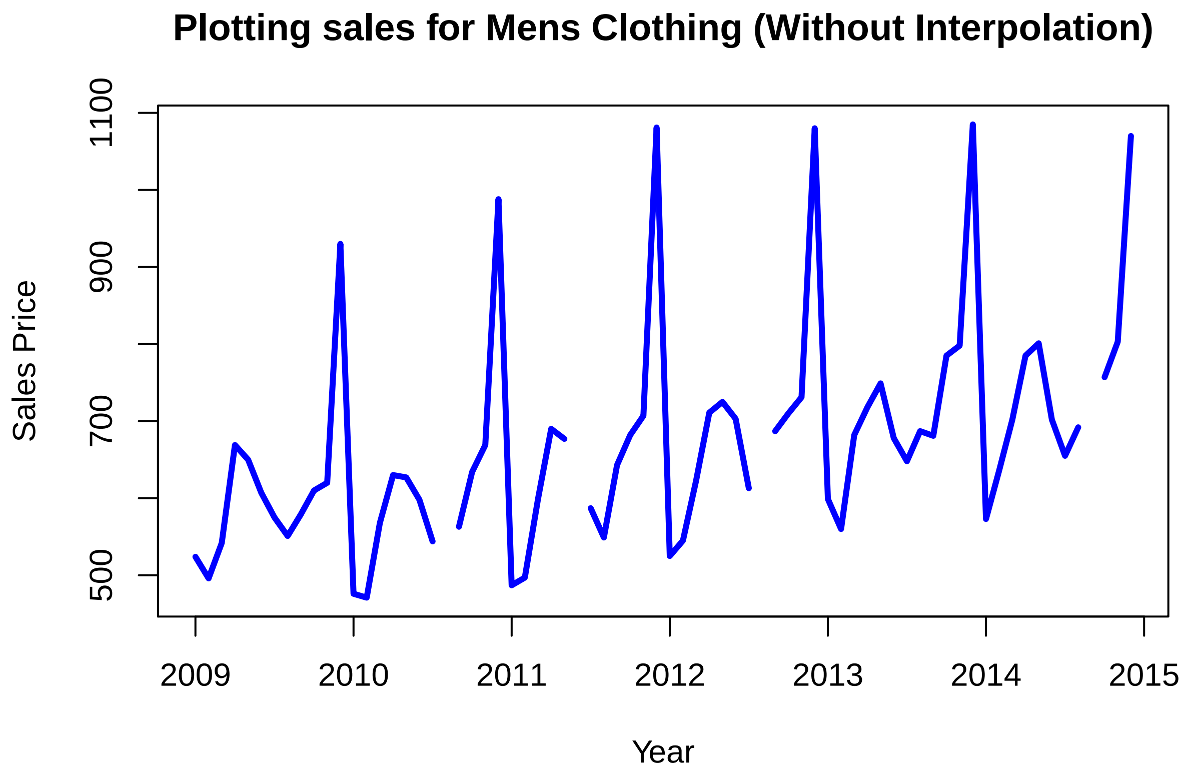
Converting the sales price from train data into Timeseries (linear interpolation)

```
train_data_men_linear_ts = ts(train_data_men_linear, frequency = 12, start = c(2009,1,1))
train_data_women_linear_ts = ts(train_data_women_linear, frequency = 12, start = c(2009,1,1))
train_data_others_linear_ts = ts(train_data_others_linear, frequency = 12, start = c(2009,1,1))
```

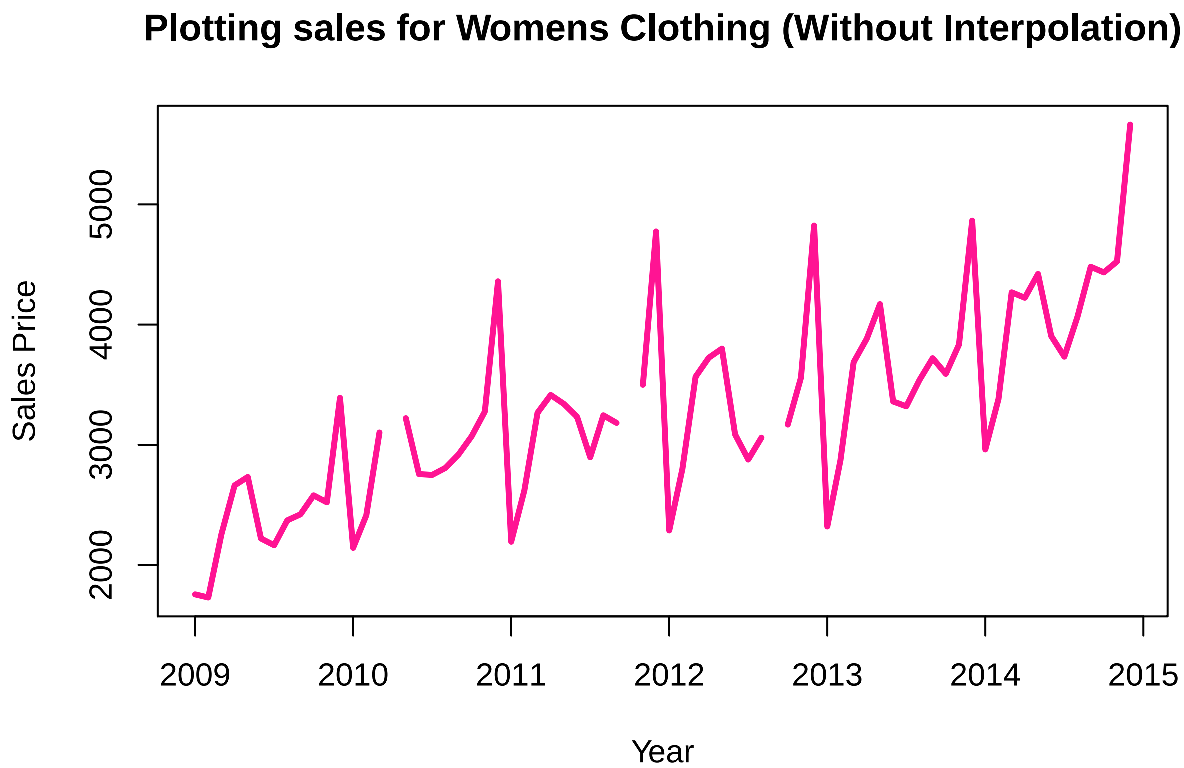
Plotting the times series of train data

Plotting of the Training data without any imputation of missing values

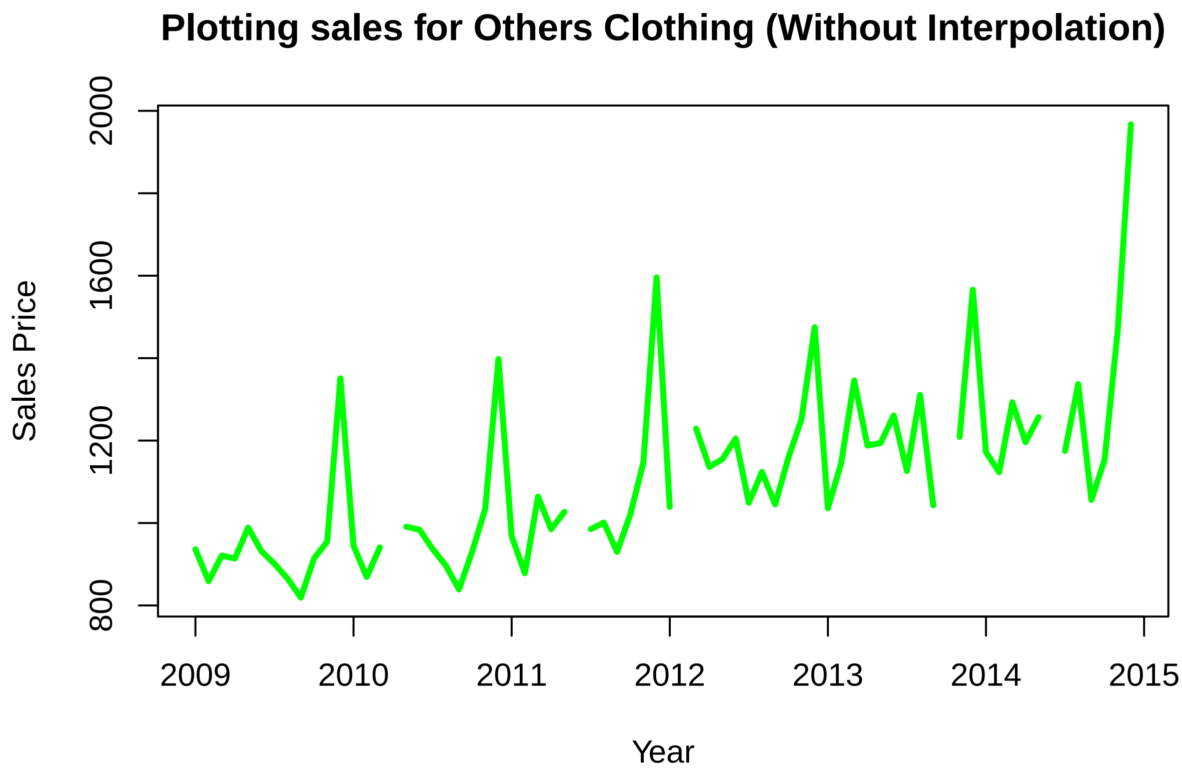
```
plot(train_data_men_ts, type="l",lwd=3, col="blue", xlab = "Year", ylab="Sales Price",
     main= "Plotting sales for Mens Clothing (Without Interpolation)")
```




```r
plot(train_data_women_ts, type="l",lwd=3, col="#FF1493",xlab = "Year", ylab="Sales Price",
     main="Plotting sales for Womens Clothing (Without Interpolation)")
```

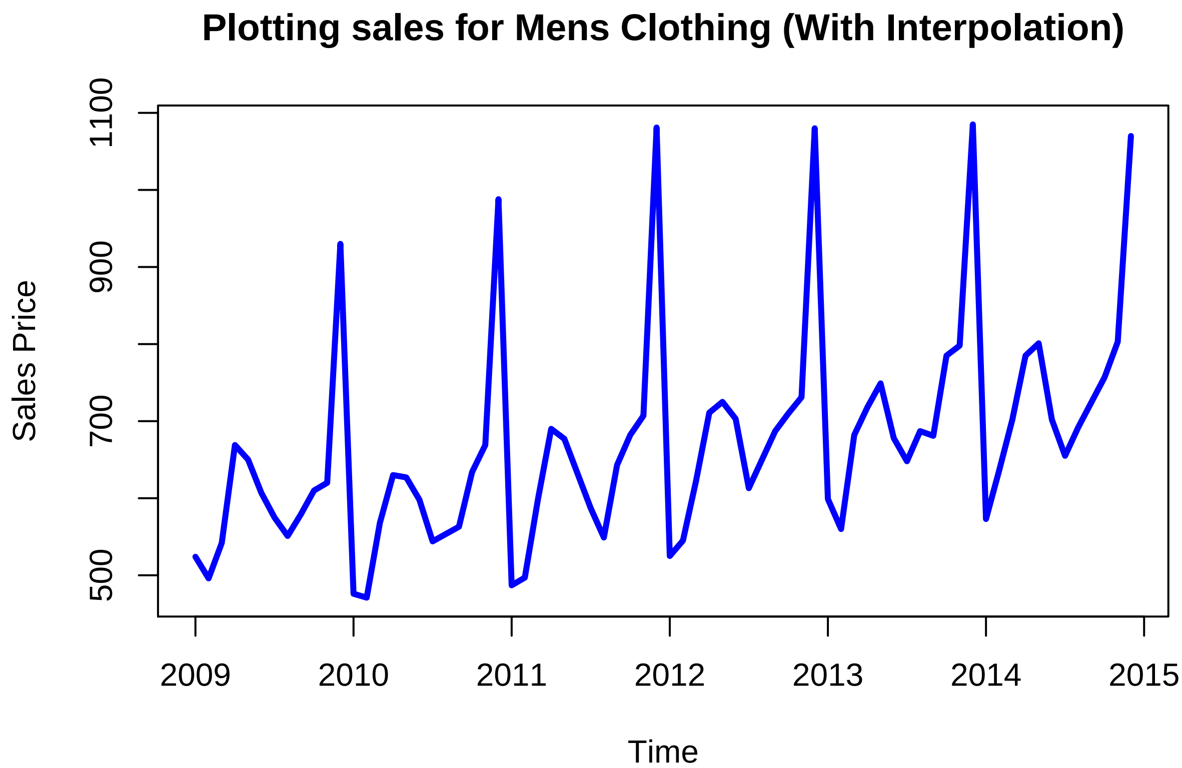



```
plot(train_data_others_ts, type="l",lwd=3, col="green",xlab = "Year", ylab="Sales Price",
     main="Plotting sales for Others Clothing (Without Interpolation)")
```

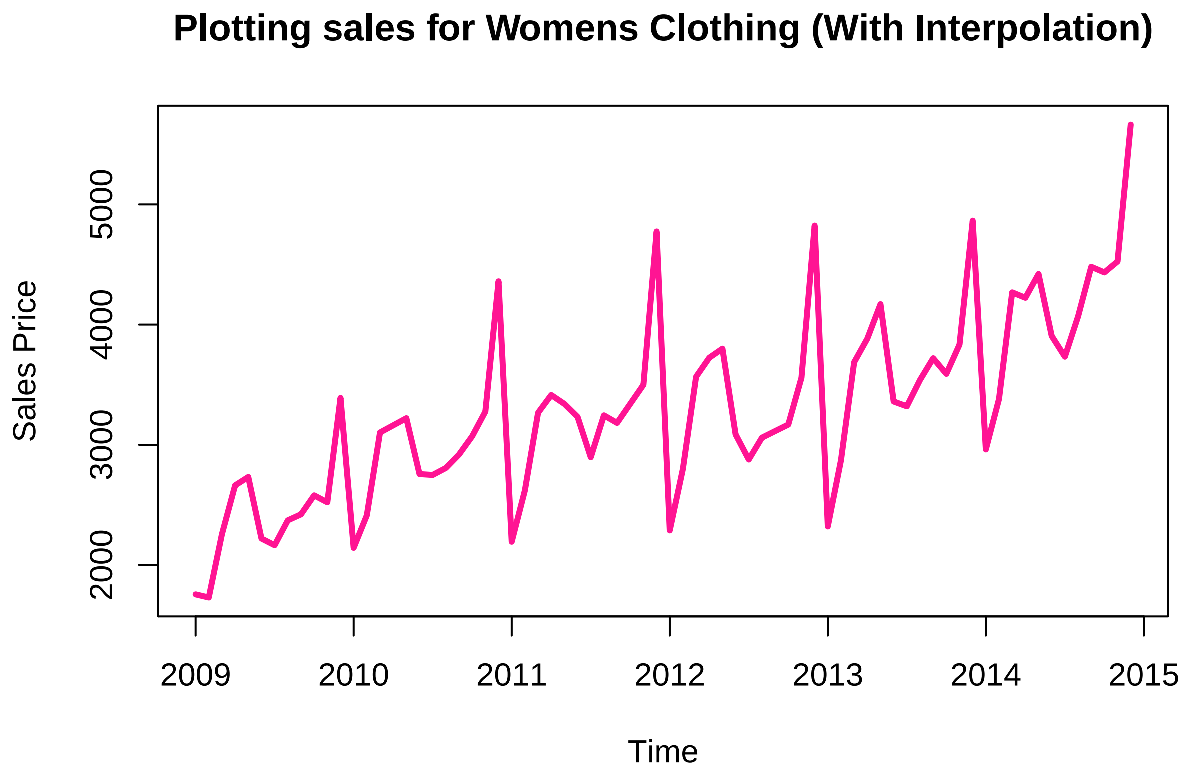


Plotting of the Traing data after interpolation (linear)

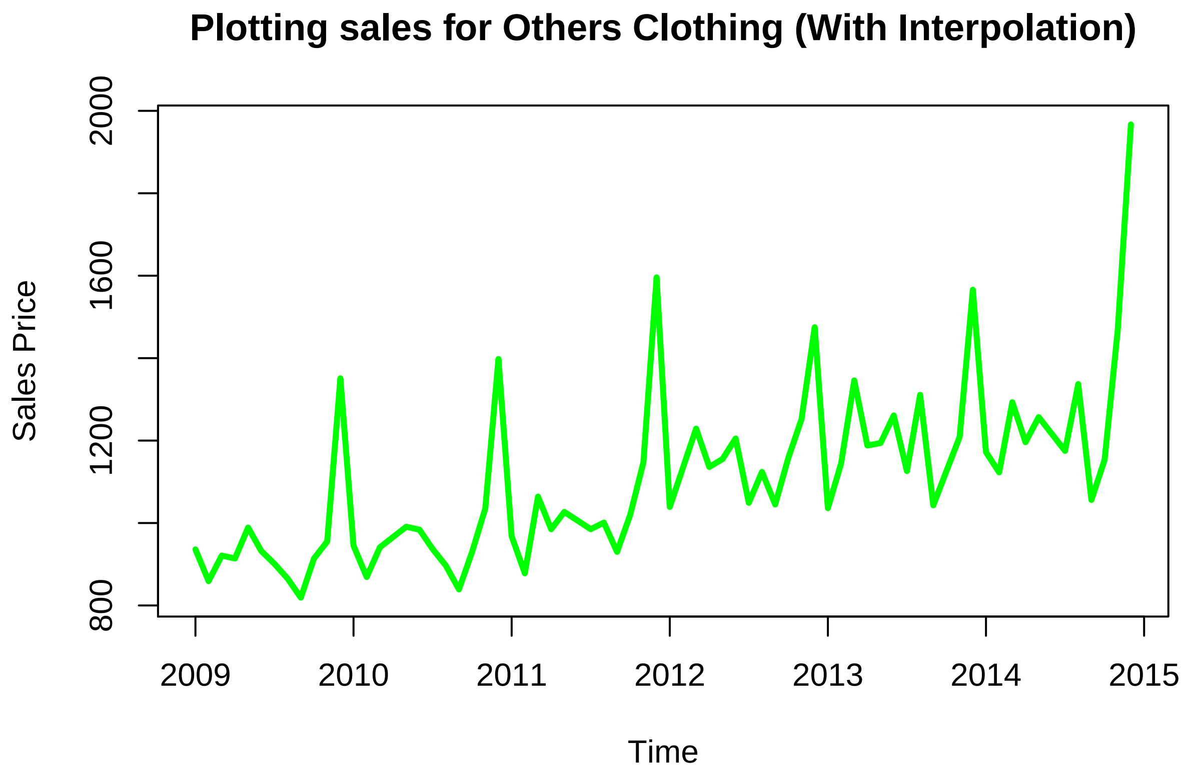
```
plot(train_data_men_linear_ts, type="l",lwd=3, col="blue", ylab="Sales Price",
     main="Plotting sales for Mens Clothing (With Interpolation)")
```




```
plot(train_data_women_linear_ts, type="l",lwd=3, col="#FF1493", ylab="Sales Price",
     main="Plotting sales for Womens Clothing (With Interpolation)")
```



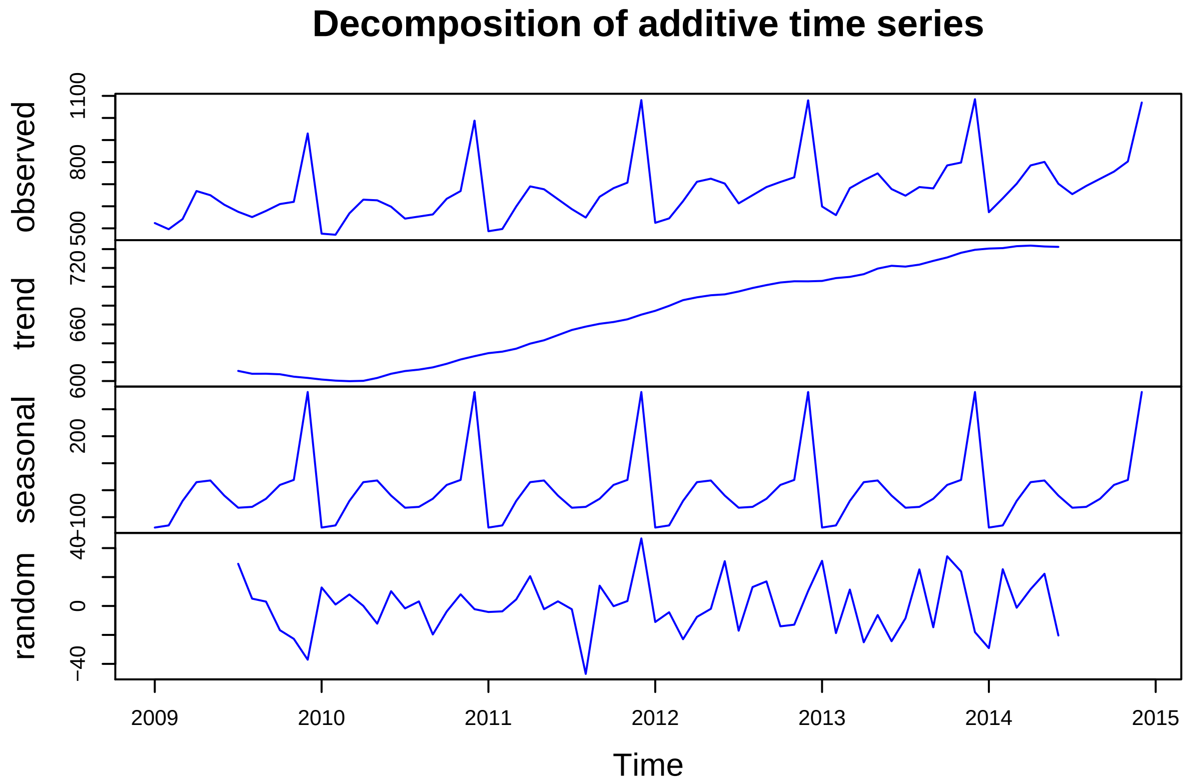

```
plot(train_data_others_linear_ts, type="l",lwd=3, col="green", ylab="Sales Price",
     main="Plotting sales for Others Clothing (With Interpolation)")
```



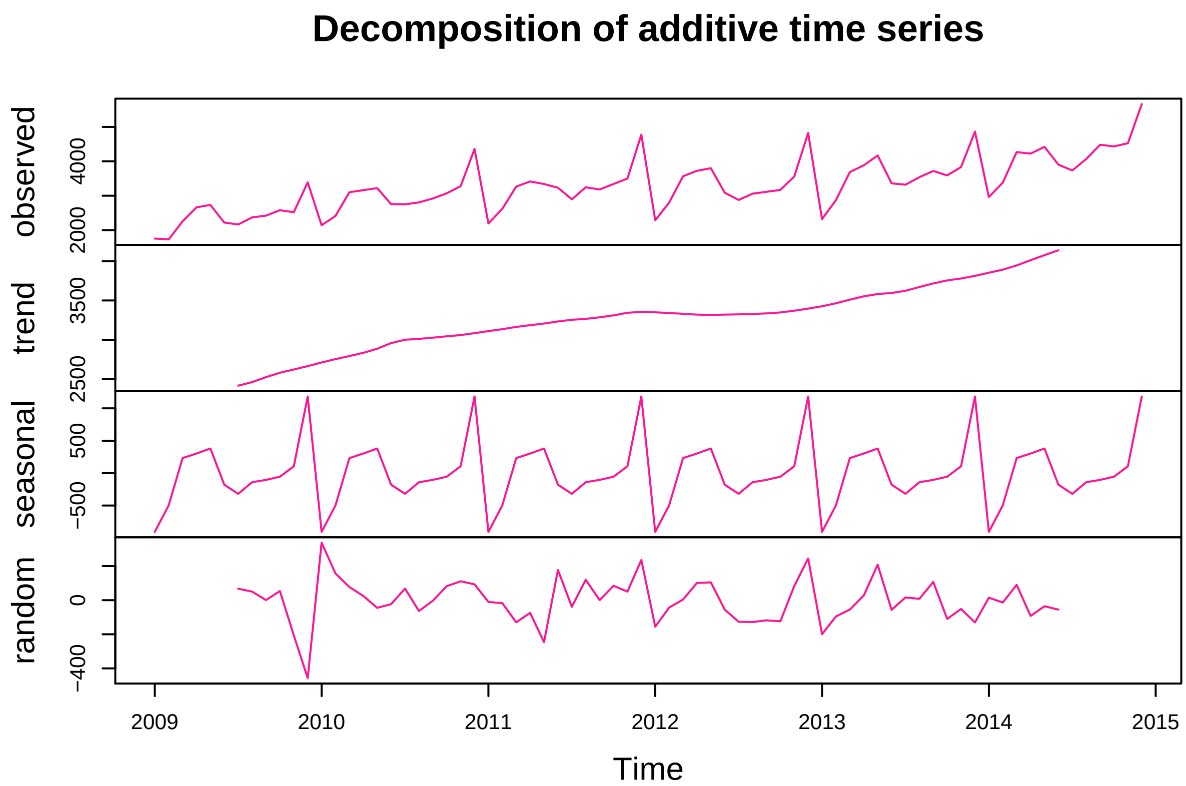
Decomposing the Train data to check the Trend, Seasonality and Noise

Decomposing the Time Series created with linear imputation

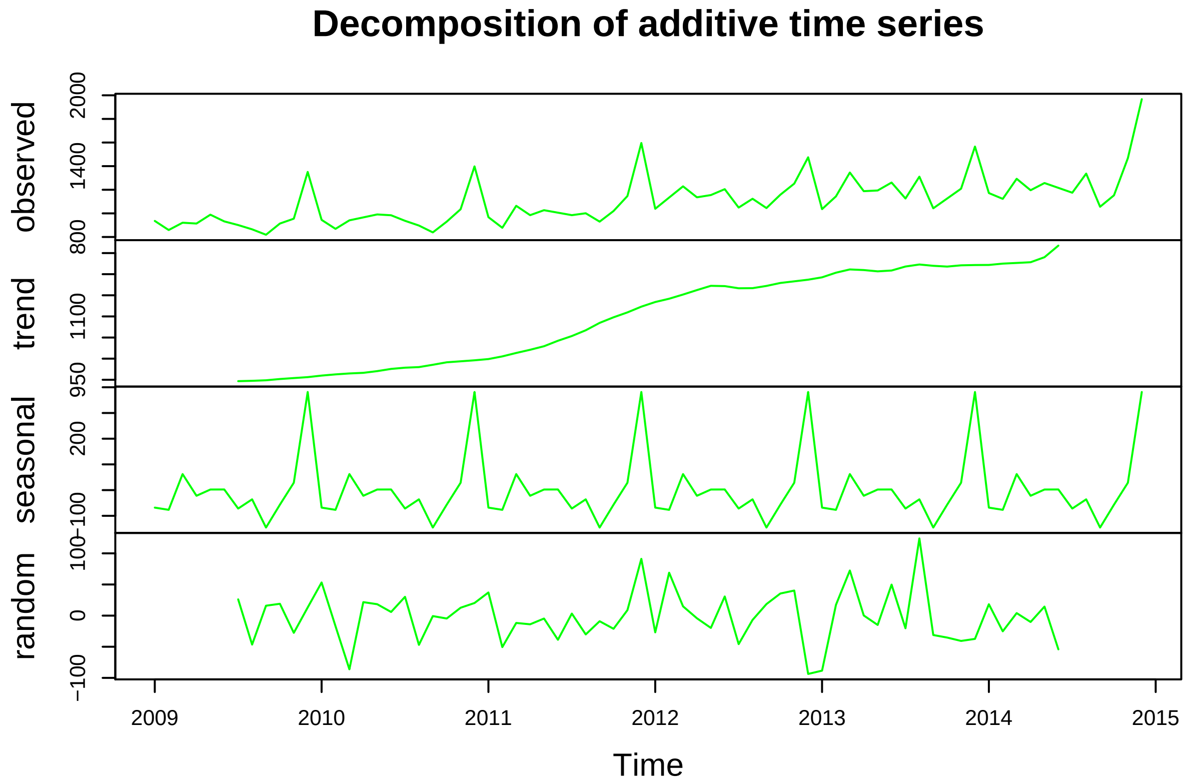
```
train_data_men_spl_decomposed=decompose(train_data_men_linear_ts)
train_data_women_spl_decomposed=decompose(train_data_women_linear_ts)
train_data_others_spl_decomposed=decompose(train_data_others_linear_ts)
par(mfrow=c(1,3))
plot(train_data_men_spl_decomposed,col="blue")
```




```
plot(train_data_women_spl_decomposed,col="#FF1493")
```




```
plot(train_data_others_spl_decomposed,col="green")
```



Holt-Winters Model

HoltWinters Model for Mens Category

```r
hw_men = HoltWinters(train_data_men_linear_ts,alpha = 0.6, beta=TRUE, gamma=TRUE, seasonal = "additive"
hw_men
```

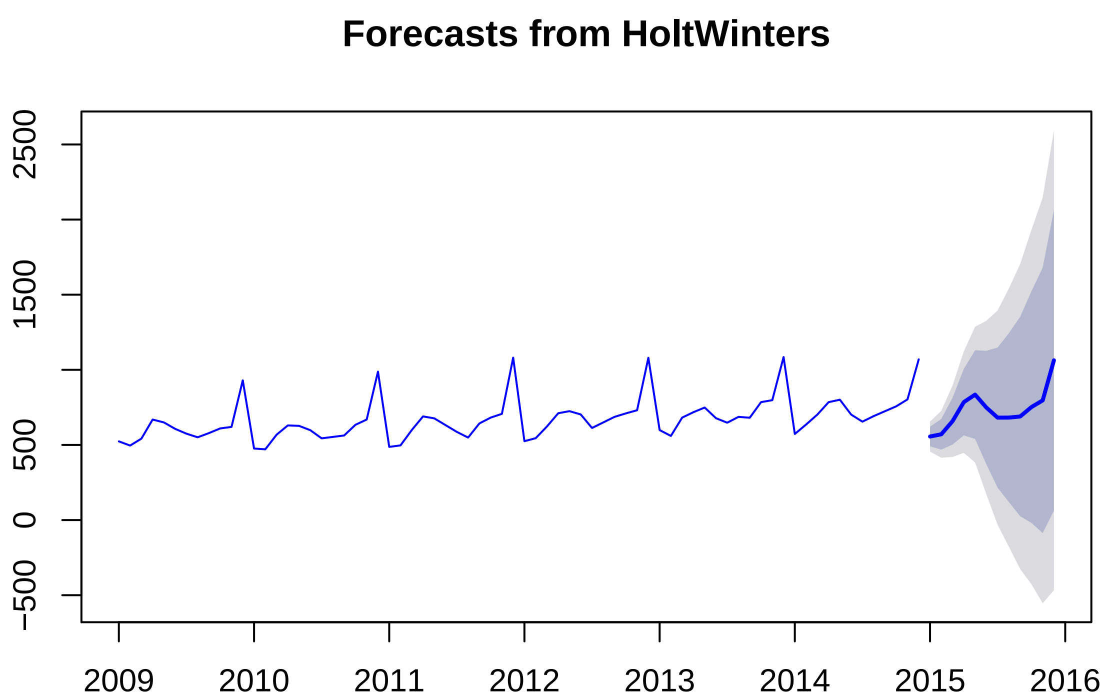
```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train_data_men_linear_ts, alpha = 0.6, beta = TRUE,     gamma = TRUE, seasonal = "ad
##
## Smoothing parameters:
##  alpha: 0.6
##  beta : TRUE
##  gamma: TRUE
##
## Coefficients:
##             [,1]
## a     738.415654
## b      -0.530101
## s1  -181.696280
## s2  -166.476048
## s3   -78.527038
```

```
## s4    48.633098
## s5    99.267378
## s6    13.869755
## s7   -52.977014
## s8   -51.988413
## s9   -44.481638
## s10   18.797428
## s11   64.054245
## s12  331.584346
```

```r
forecast_hw_men = forecast(hw_men, h=12)
hw_acc_men =accuracy(forecast_hw_men,validation_data_men$Sales.In.ThousandDollars.)
hw_acc_men
```

```
##                         ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   0.002492418 50.53613 38.22271 -0.3102968 5.771822 0.3812605
## Test set     -26.891648617 62.65156 45.83660 -3.6575856 6.352519 0.4572069
##                     ACF1
## Training set -0.04928946
## Test set              NA
```

```r
plot(forecast_hw_men,col="blue", )
```

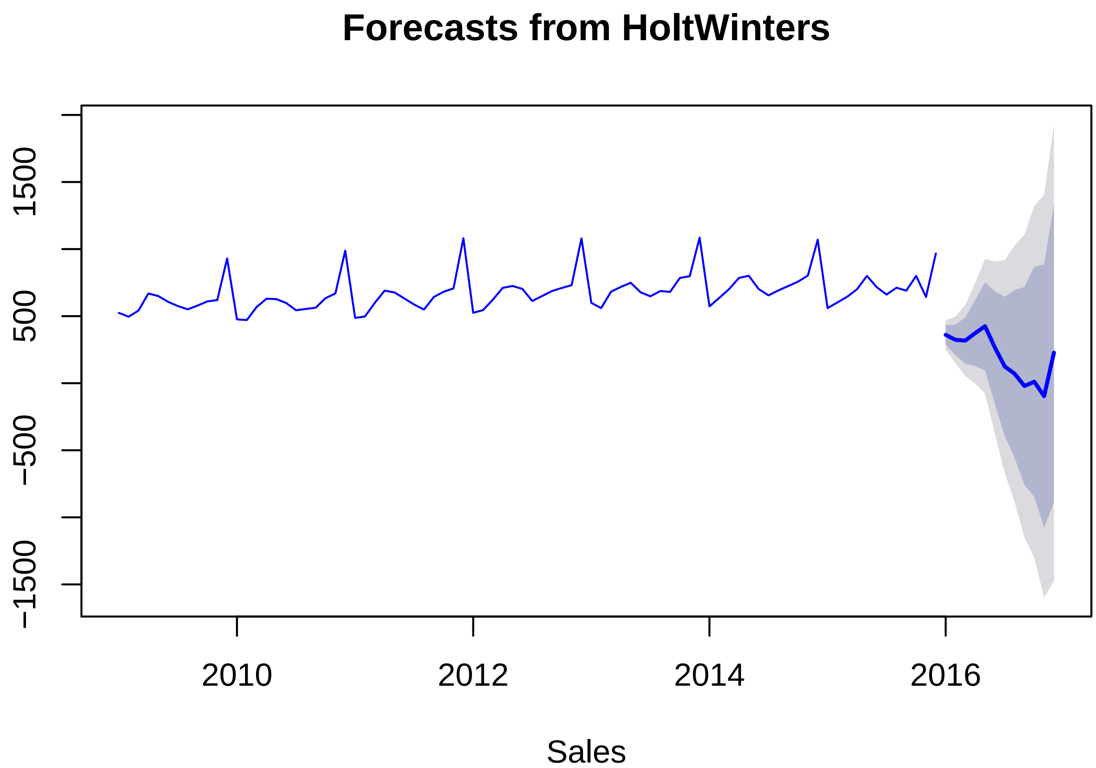



```r
# Creating the model on the entire dataset and predicting for mens category
hw_men_total = HoltWinters(total_data_men_linear_ts,alpha = 0.6, beta=TRUE, gamma=TRUE, seasonal = "add
hw_men_total
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
```

```
##
## Call:
## HoltWinters(x = total_data_men_linear_ts, alpha = 0.6, beta = TRUE,     gamma = TRUE, seasonal = "ad
##
## Smoothing parameters:
##  alpha: 0.6
##  beta : TRUE
##  gamma: TRUE
##
## Coefficients:
##           [,1]
## a    602.47593
## b    -61.62107
## s1  -180.17199
## s2  -155.85696
## s3   -99.33299
## s4    17.25625
## s5   130.29862
## s6    32.45825
## s7   -46.33906
## s8   -38.84759
## s9   -68.43144
## s10   24.87235
## s11  -21.09700
## s12  364.52407
```

```
forcast_hw_men_total = forecast(hw_men_total, h = 12)
plot(forcast_hw_men_total, col = "blue", xlab = "Sales")
```




```r
forcast_hw_men_total$mean
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2016 360.68287 323.37683 318.27973 373.24790 424.66921 265.20777 124.78939
##            Aug       Sep       Oct       Nov       Dec
## 2016  70.65978 -20.54513  11.13758 -96.45284 227.54716
```

```r
print("Lower: 80%")
```

```
## [1] "Lower: 80%"
```

```r
forcast_hw_men_total$lower[,1]
```

```
##  [1]   288.04151   209.90740   145.15538   127.55251    96.25311
##  [6]  -154.60653  -394.17711  -554.56254  -758.63287  -846.03051
## [11] -1078.58987  -885.17059
```

```r
print("Lower: 90%")
```

```
## [1] "Lower: 90%"
```

```r
forcast_hw_men_total$lower[,2]
```

```
##  [1]   249.587494   149.840304    53.508883    -2.510768   -77.599873
##  [6]  -376.842795  -668.901380  -885.535251 -1149.352910 -1299.787895
## [11] -1598.501806 -1474.207781
```

```r
print("Upper: 80% - Preffered")
```

```
## [1] "Upper: 80% - Preffered"
```

```
forcast_hw_men_total$upper[,1]
```

```
##  [1]  433.3242  436.8463  491.4041  618.9433  753.0853  685.0221  643.7559
##  [8]  695.8821  717.5426  868.3057  885.6842 1340.2649
```

```
print("Upper: 90%")
```

```
## [1] "Upper: 90%"
```

```
forcast_hw_men_total$upper[,2]
```

```
##  [1]  471.7783  496.9134  583.0506  749.0066  926.9383  907.2583  918.4802
##  [8] 1026.8548 1108.2626 1322.0631 1405.5961 1929.3021
```

Holtwinters Model for WomenCategory

```
hw_women = HoltWinters(train_data_women_linear_ts, alpha = 0.6, beta=TRUE, gamma=TRUE, seasonal = "addi
hw_women
```

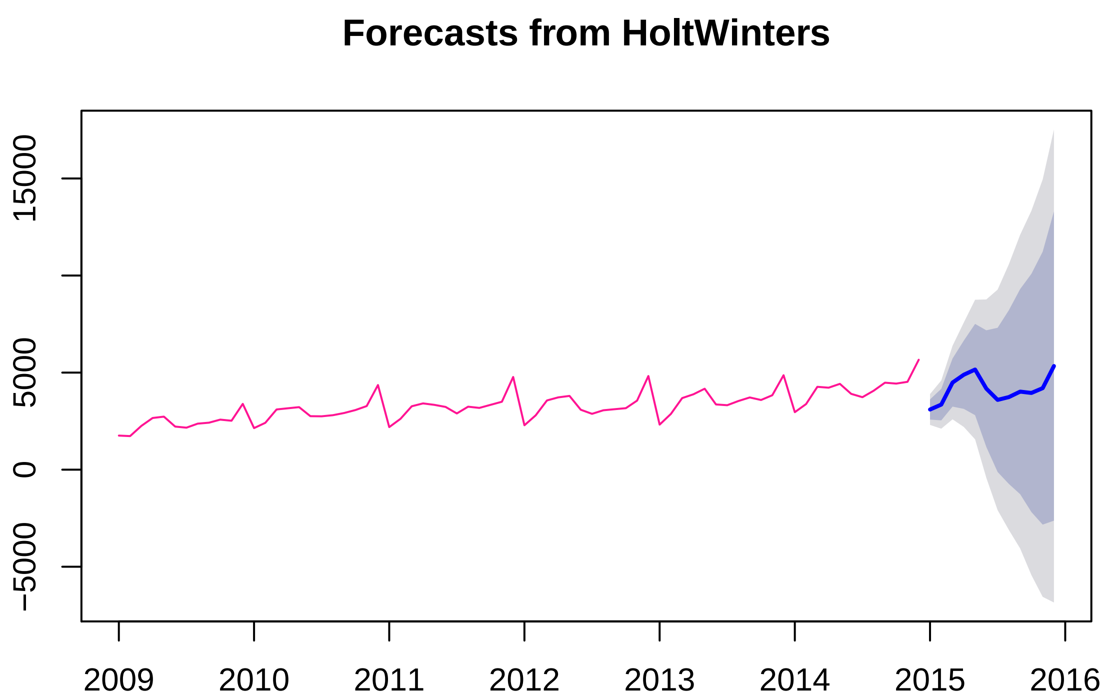
```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train_data_women_linear_ts, alpha = 0.6, beta = TRUE,     gamma = TRUE, seasonal = "a
##
## Smoothing parameters:
##  alpha: 0.6
##  beta : TRUE
##  gamma: TRUE
##
## Coefficients:
##             [,1]
## a    4347.89949
## b     -27.17041
## s1  -1221.22705
## s2   -940.88408
## s3    221.39009
## s4    650.62227
## s5    946.10773
## s6    -11.18528
## s7   -562.69901
## s8   -391.12005
## s9    -82.92805
## s10  -123.57670
## s11   149.93011
## s12  1316.10051
```

```
forecast_hw_women = forecast(hw_women, h=12)
hw_acc_women =accuracy(forecast_hw_women,validation_data_women$Sales.In.ThousandDollars.)
hw_acc_women
```

```
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -2.06123 402.2797 310.5280 -1.0184086 9.523478 0.6619475
## Test set     38.07341 351.0151 294.7216  0.6594328 6.833982        NA
##                    ACF1
## Training set 0.09116328
```

```
## Test set                NA
```

```r
plot(forecast_hw_women,col="#FF1493")
```

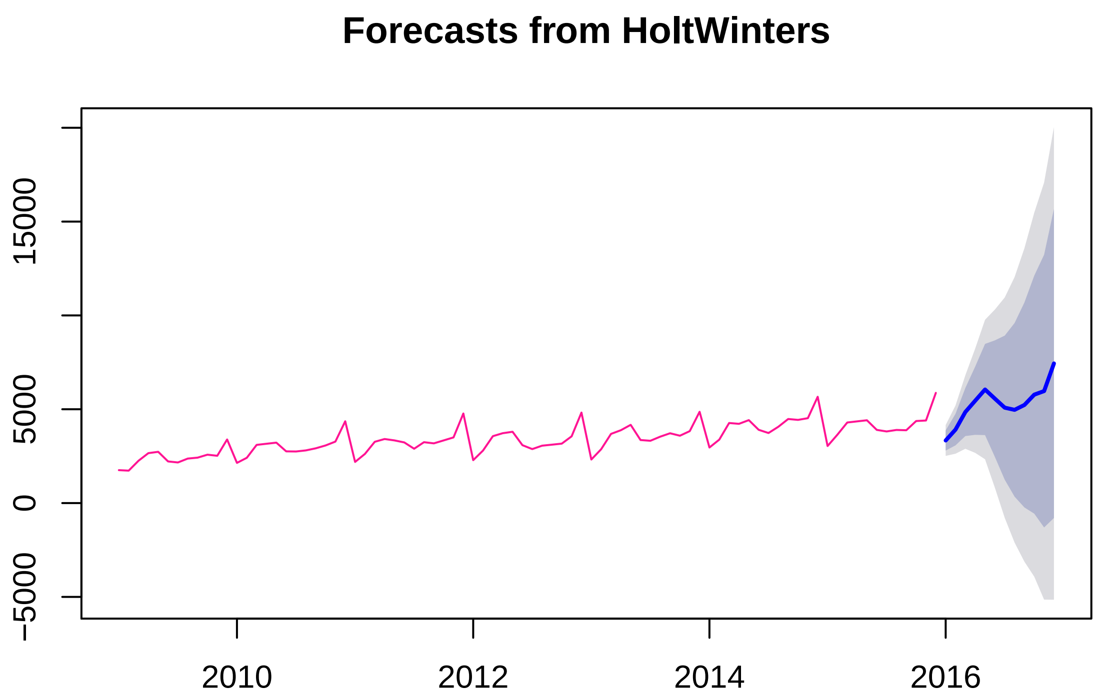



```r
# Creating the model on the entire dataset and predicting for Womens category
hw_women_total = HoltWinters(total_data_women_linear_ts,alpha = 0.6, beta=TRUE, gamma=TRUE, seasonal =
hw_women_total
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = total_data_women_linear_ts, alpha = 0.6, beta = TRUE,     gamma = TRUE, seasonal = "a
##
## Smoothing parameters:
##  alpha: 0.6
##  beta : TRUE
##  gamma: TRUE
##
## Coefficients:
##             [,1]
## a    4450.41270
## b     130.59620
## s1  -1244.62786
## s2   -795.47295
## s3     11.50684
## s4    482.36801
## s5    948.95631
```

```
## s6     330.10668
## s7    -280.31659
## s8    -526.16797
## s9    -396.96457
## s10     21.54951
## s11     81.18350
## s12   1423.58730
```

```r
forcast_hw_women_total = forecast(hw_women_total, h = 12)
plot(forcast_hw_women_total, col = "#FF1493")
```




```r
print("mean-Preffered")
```

```
## [1] "mean-Preffered"
```

```r
forcast_hw_women_total$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2016 3336.381 3916.132 4853.708 5455.165 6052.350 5564.097 5084.269
##           Aug      Sep      Oct      Nov      Dec
## 2016 4969.014 5228.814 5777.924 5968.154 7441.154
```

```r
print("Lower: 80%")
```

```
## [1] "Lower: 80%"
```

```r
forcast_hw_women_total$lower[,1]
```

```
##  [1]  2798.9238  3076.5971  3572.7997  3637.3201  3622.4725  2457.9842
```

```
##  [7]  1244.5521   343.1338  -232.1326  -564.0714 -1298.4584  -791.5960
```

```r
print("Lower: 90%")
```

```
## [1] "Lower: 90%"
```

```r
forcast_hw_women_total$lower[,2]
```

```
##  [1]  2514.4111  2632.1741  2894.7277  2675.0109  2336.1731   813.7077
##  [7]  -788.0715 -2105.6594 -3122.9831 -3921.3210 -5145.1709 -5149.7509
```

```r
print("Upper: 80%")
```

```
## [1] "Upper: 80%"
```

```r
forcast_hw_women_total$upper[,1]
```

```
##  [1]  3873.838  4755.667  6134.617  7273.011  8482.227  8670.209  8923.987
##  [8]  9594.895 10689.760 12119.920 13234.767 15673.905
```

```r
print("Upper: 90%")
```

```
## [1] "Upper: 90%"
```

```r
forcast_hw_women_total$upper[,2]
```

```
##  [1]  4158.351  5200.090  6812.689  8235.320  9768.527 10314.485 10956.610
##  [8] 12043.688 13580.611 15477.169 17081.480 20032.060
```

HoltWinters Model for OthersCategory

```r
hw_others = HoltWinters(train_data_others_linear_ts, alpha = 0.2, beta=TRUE, gamma=TRUE, seasonal = "ad
hw_others
```

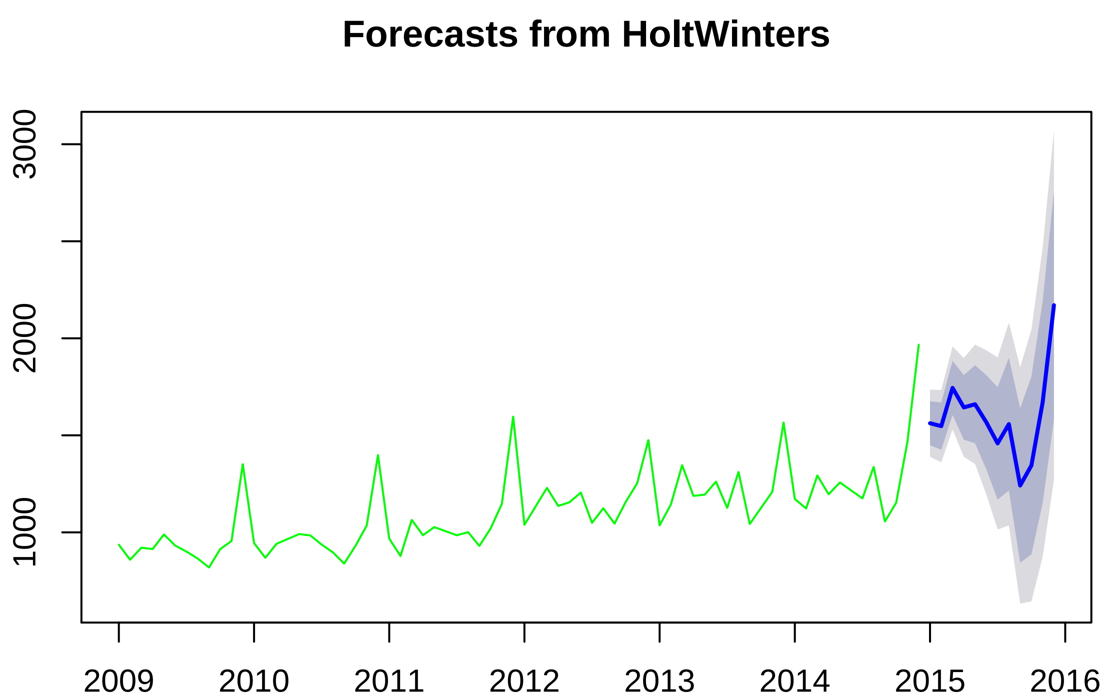
```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train_data_others_linear_ts, alpha = 0.2, beta = TRUE,     gamma = TRUE, seasonal =
##
## Smoothing parameters:
##  alpha: 0.2
##  beta : TRUE
##  gamma: TRUE
##
## Coefficients:
##            [,1]
## a   1482.399388
## b     16.951391
## s1    62.853984
## s2    30.969400
## s3   211.053844
## s4    93.567473
## s5    92.744075
## s6   -17.207118
## s7  -142.548292
## s8   -60.198699
## s9  -393.711982
```

```
## s10 -307.284655
## s11    2.552004
## s12  484.600612
```

```
forecast_hw_others = forecast(hw_others, h=12)
hw_acc_others =accuracy(forecast_hw_others,validation_data_others$Sales.In.ThousandDollars.)
hw_acc_others
```

```
##                        ME      RMSE       MAE         MPE      MAPE
## Training set    1.195638  87.60679  62.38197  -0.1261985  5.348396
## Test set     -253.532650 299.30637 256.82433 -18.9829609 19.243998
##                   MASE      ACF1
## Training set 0.4516284 0.3713851
## Test set     1.8593380        NA
```

```
plot(forecast_hw_others,col="green")
```

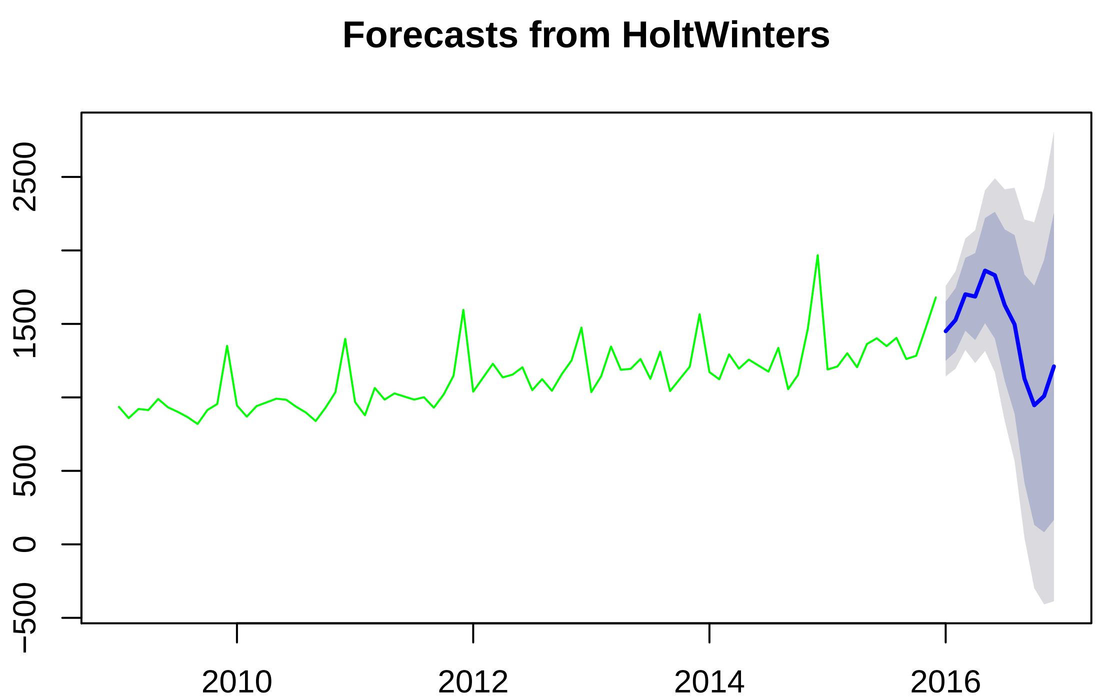



```
# Creating the model on the entire dataset and predicting for Others category
hw_others_total = HoltWinters(total_data_others_linear_ts,alpha = 0.2, beta=TRUE, gamma=TRUE, seasonal =
hw_others_total
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = total_data_others_linear_ts, alpha = 0.2, beta = TRUE,     gamma = TRUE, seasonal =
##
## Smoothing parameters:
##  alpha: 0.2
```

```
##   beta : TRUE
##   gamma: TRUE
##
## Coefficients:
##            [,1]
## a    1724.31719
## b     -39.08542
## s1  -234.90983
## s2  -119.74233
## s3    94.55090
## s4   117.78880
## s5   333.77017
## s6   341.15981
## s7   176.72991
## s8    85.00026
## s9  -244.12591
## s10 -387.25859
## s11 -285.40262
## s12  -44.31719
```

```
forcast_hw_others_total = forecast(hw_others_total, h = 12)
plot(forcast_hw_others_total, col = "green")
```




```
print("mean")
```

```
## [1] "mean"
```

```r
forcast_hw_others_total$mean
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2016 1450.3219 1526.4040 1701.6118 1685.7643 1862.6602 1830.9645 1627.4491
##            Aug       Sep       Oct       Nov       Dec
## 2016 1496.6341 1128.4225  946.2044 1008.9749 1210.9749
```

```r
print("Lower: 80%")
```

```
## [1] "Lower: 80%"
```

```r
forcast_hw_others_total$lower[,1]
```

```
##  [1] 1249.04617 1309.62337 1453.46238 1389.95053 1504.86461 1399.27603
##  [7] 1111.93162  888.79456  420.80143  132.09425   82.23166  165.89032
```

```r
print("Lower: 90%")
```

```
## [1] "Lower: 90%"
```

```r
forcast_hw_others_total$lower[,2]
```

```
##  [1] 1142.49721 1194.86663 1322.10001 1233.35619 1315.45904 1170.75397
##  [7]  839.03314  567.02377   46.20948 -298.86961 -408.35658 -387.34404
```

```r
print("Upper: 80%")
```

```
## [1] "Upper: 80%"
```

```r
forcast_hw_others_total$upper[,1]
```

```
##  [1] 1651.598 1743.185 1949.761 1981.578 2220.456 2262.653 2142.967
##  [8] 2104.474 1836.044 1760.314 1935.718 2256.060
```

```r
print("Upper: 90%")
```

```
## [1] "Upper: 90%"
```

```r
forcast_hw_others_total$upper[,2]
```

```
##  [1] 1758.147 1857.941 2081.124 2138.172 2409.861 2491.175 2415.865
##  [8] 2426.244 2210.635 2191.278 2426.306 2809.294
```

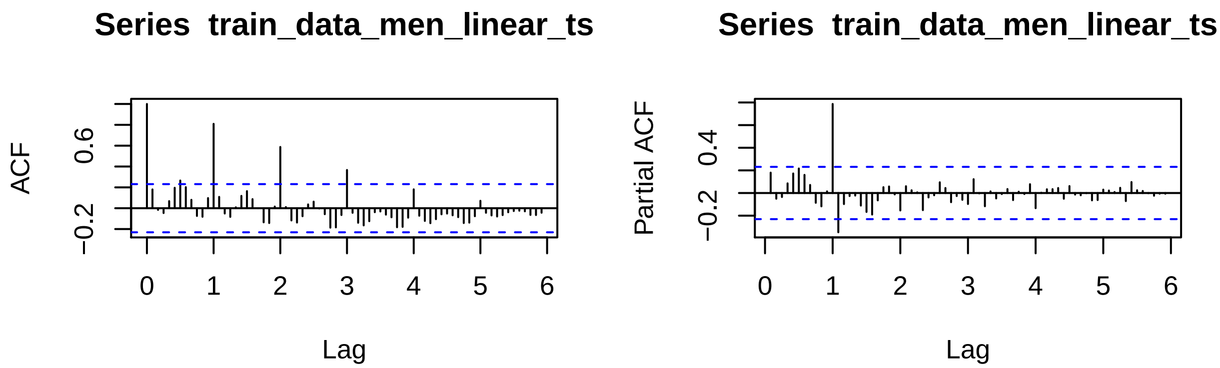
ACF and PACF

- Autocorrelation is the linear dependence of a variable with itself at two points in time
- For stationary processes, autocorrelation between any two observations only depends on the time lag h between them
- Partial autocorrelation is the autocorrelation between yt and yt(h) after removing any linear dependence on y1,y2, . . . , yt(h+1)

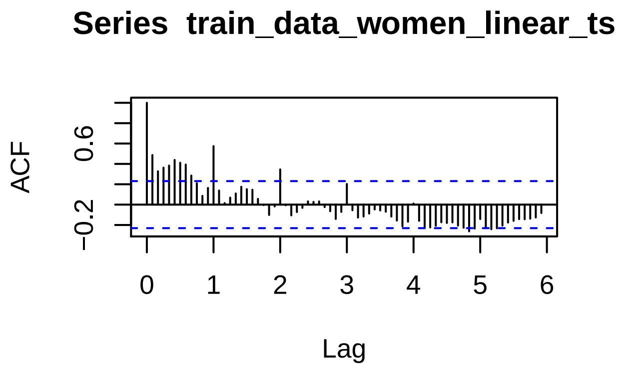
Verifying the ACF and PACF values

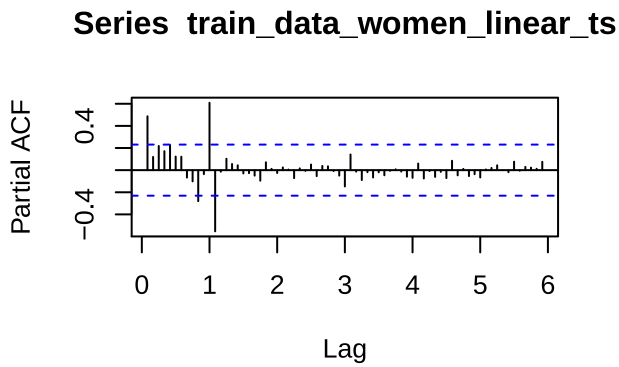
```r
par(mfrow=c(2,2))
acf(train_data_men_linear_ts,lag.max =120)
pacf(train_data_men_linear_ts,lag.max =120)
```

```r
par(mfrow=c(2,2))
```

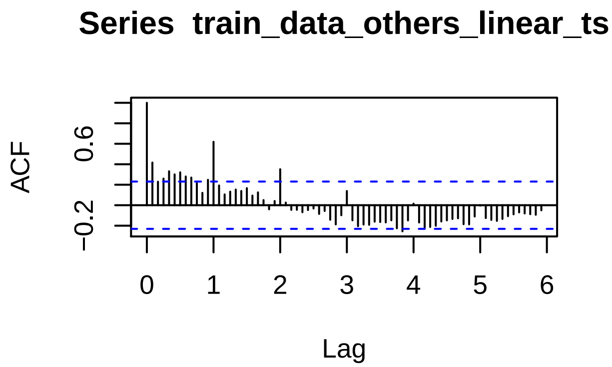


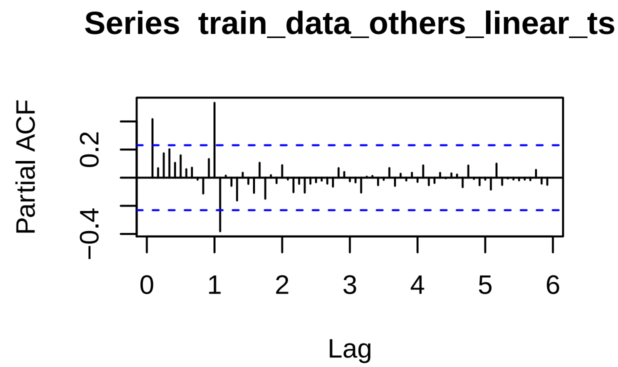

```r
acf(train_data_women_linear_ts,lag.max =120)
pacf(train_data_women_linear_ts,lag.max =120)


par(mfrow=c(2,2))
```






```
acf(train_data_others_linear_ts,lag.max =120)
pacf(train_data_others_linear_ts,lag.max =120)
```





AUTO ARIMA

```
auto_arima_men = auto.arima(train_data_men_linear_ts, ic='aic')
auto_arima_women = auto.arima(train_data_women_linear_ts, ic='aic')
auto_arima_others = auto.arima(train_data_others_linear_ts, ic='aic')
```

Summary of the Auto arima model

```
summary(auto_arima_men)
```

```
## Series: train_data_men_linear_ts
## ARIMA(1,0,1)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1      ma1     sma1   drift
##       0.9432  -0.7060  -0.7751  1.9181
## s.e.  0.0717   0.1065   0.3325  0.6076
##
## sigma^2 estimated as 739.7:  log likelihood=-286.36
## AIC=582.72   AICc=583.83   BIC=593.19
##
## Training set error measures:
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.7238923 23.98609 18.47167 -0.1019525 2.771932 0.5177763
##                    ACF1
## Training set -0.1148345
```

```r
summary(auto_arima_women)
```

```
## Series: train_data_women_linear_ts
## ARIMA(0,1,1)(0,1,0)[12]
##
## Coefficients:
##           ma1
##       -0.5664
## s.e.   0.1218
##
## sigma^2 estimated as 43269:  log likelihood=-398.32
## AIC=800.65   AICc=800.86   BIC=804.8
##
## Training set error measures:
##                    ME     RMSE     MAE         MPE     MAPE      MASE
## Training set 9.105897 186.6967 132.436 -0.07783658 4.044671 0.3539965
##                    ACF1
## Training set 0.05254706
```

```r
summary(auto_arima_others)
```

```
## Series: train_data_others_linear_ts
## ARIMA(0,0,1)(0,1,1)[12] with drift
##
## Coefficients:
##          ma1     sma1   drift
##       0.3956  -0.5064  5.9778
## s.e.  0.1137   0.1713  0.6729
##
## sigma^2 estimated as 5339:  log likelihood=-342.94
## AIC=693.89   AICc=694.61   BIC=702.27
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -1.989613 65.01565 45.01269 -0.5419672 3.824652 0.5749972
##                    ACF1
## Training set 0.03508462
```

Forecasting for Men clothing, Women Clothing and Others Clothing on train and validation

```r
forecast_a_arima_men = forecast(auto_arima_men, h=12)
forecast_a_arima_women = forecast(auto_arima_women, h=12)
forecast_a_arima_others = forecast(auto_arima_others, h=12)
acc_men =accuracy(forecast_a_arima_men,validation_data_men$Sales.In.ThousandDollars.)
acc_women =accuracy(forecast_a_arima_women,validation_data_women$Sales.In.ThousandDollars.)
acc_others =accuracy(forecast_a_arima_others,validation_data_others$Sales.In.ThousandDollars.)
acc_men
```

```
##                       ME     RMSE      MAE        MPE     MAPE      MASE
```

```
## Training set   0.7238923 23.98609 18.47167 -0.1019525 2.771932 0.1842496
## Test set     -47.5273688 74.91789 53.85209 -6.7553972 7.575423 0.5371590
##                    ACF1
## Training set -0.1148345
## Test set             NA
```

```
acc_women
```

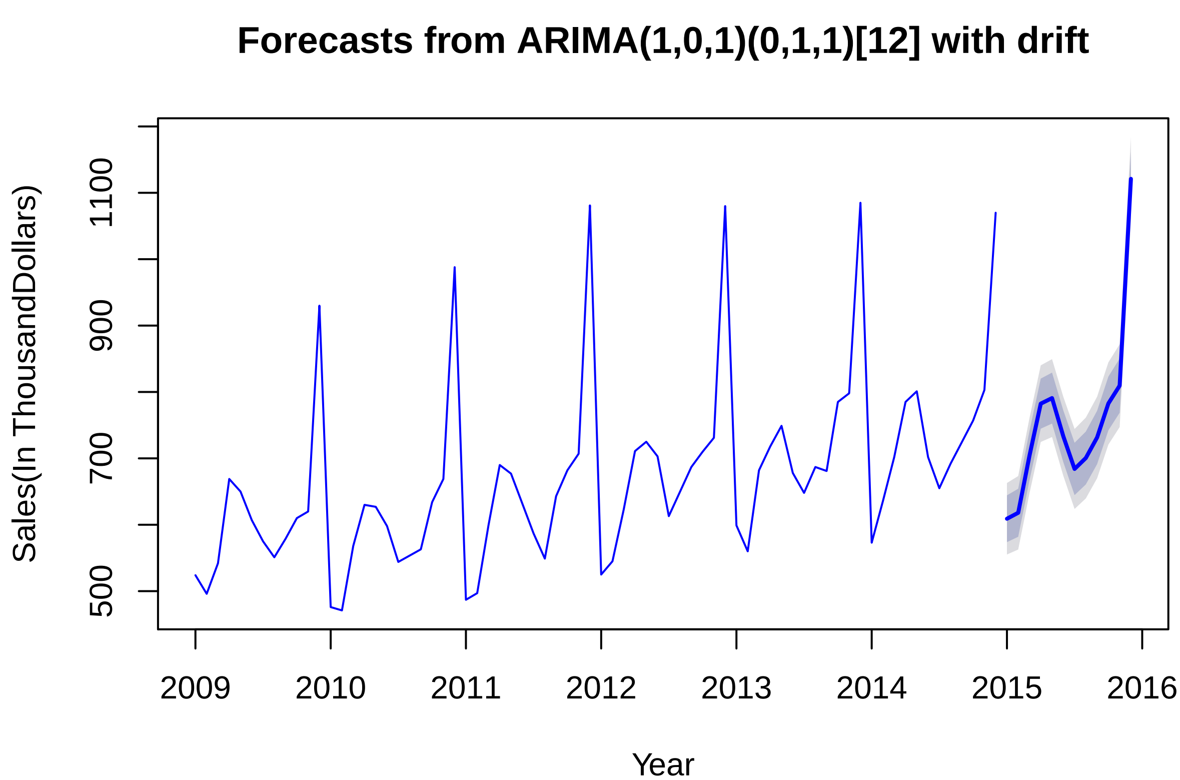
```
##                       ME     RMSE      MAE          MPE      MAPE
## Training set    9.105897 186.6967 132.4360  -0.07783658  4.044671
## Test set     -769.693687 800.2802 769.6937 -19.09462591 19.094626
##                   MASE       ACF1
## Training set 0.2823117 0.05254706
## Test set            NA         NA
```

```
acc_others
```

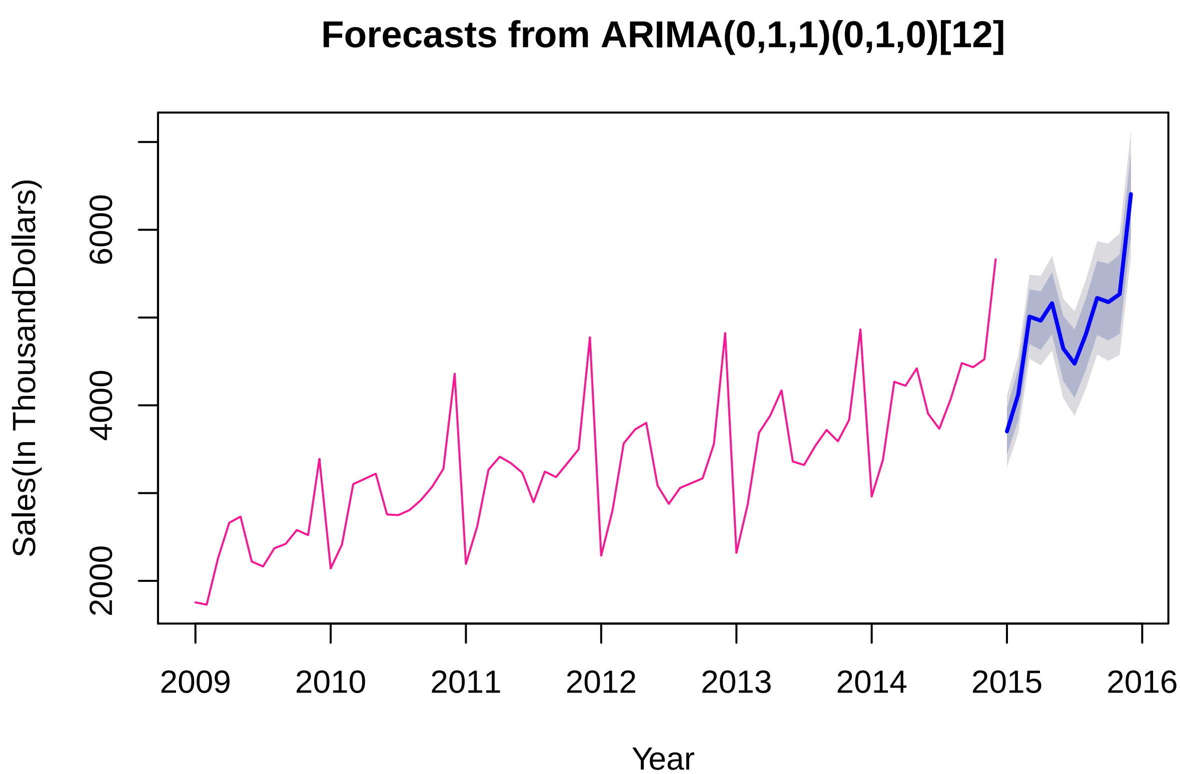
```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  -1.989613 65.01565 45.01269 -0.5419672 3.824652 0.3258796
## Test set     -22.063661 92.63618 73.00588 -1.5956867 5.411289 0.5285426
##                    ACF1
## Training set 0.03508462
## Test set             NA
```

Plotting the values

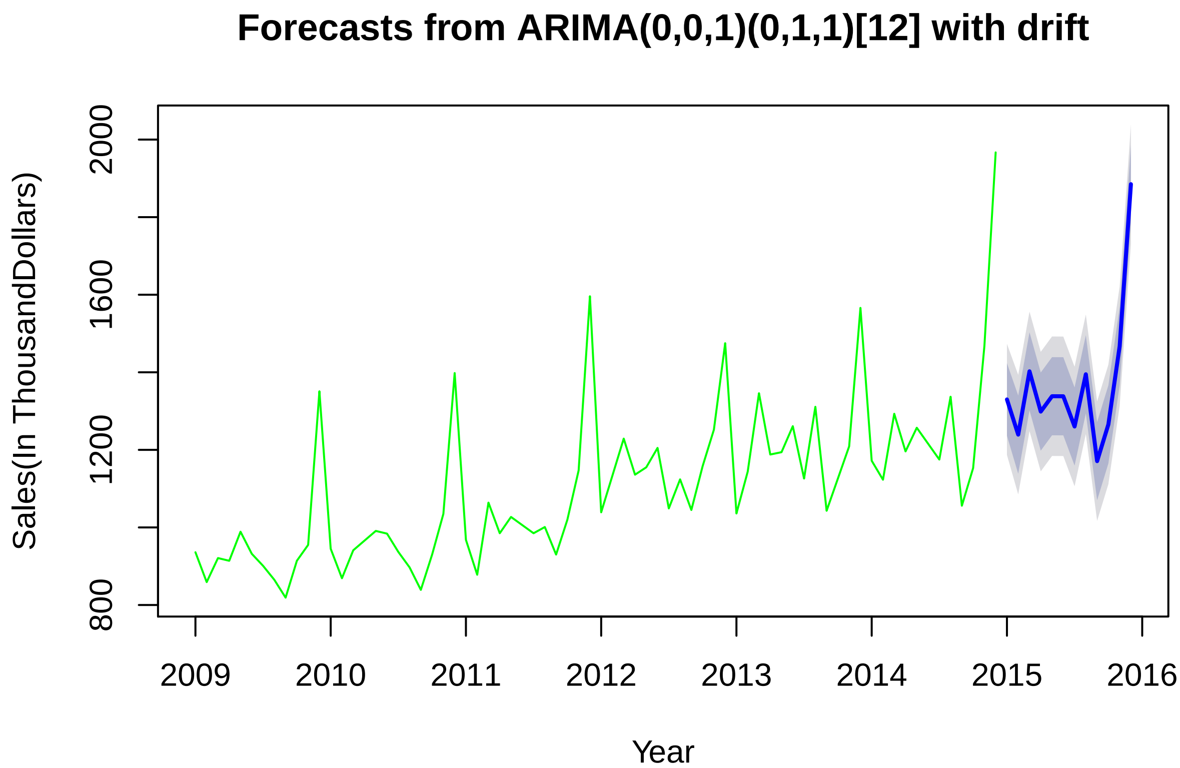
```
plot(forecast_a_arima_men,col="blue", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



```r
plot(forecast_a_arima_women,col="#FF1493", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```




```r
plot(forecast_a_arima_others,col="green", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



Forecasting (Auto ARIMA):

```
# Creating an auto arima model
auto_arima_total_men = auto.arima(total_data_men_linear_ts, ic='aic')
auto_arima_total_women = auto.arima(total_data_women_linear_ts, ic='aic')
auto_arima_total_others = auto.arima(total_data_others_linear_ts, ic='aic')

# On Combined Data
summary(auto_arima_total_men)
```

```
## Series: total_data_men_linear_ts
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.6454  -0.4626
## s.e.   0.1149   0.1641
##
## sigma^2 estimated as 1359:  log likelihood=-357.56
## AIC=721.11   AICc=721.47   BIC=727.9
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.4499339 33.40999 23.66468 -0.1198056 3.450731 0.6287294
##                    ACF1
```

```
## Training set -0.001875814
```

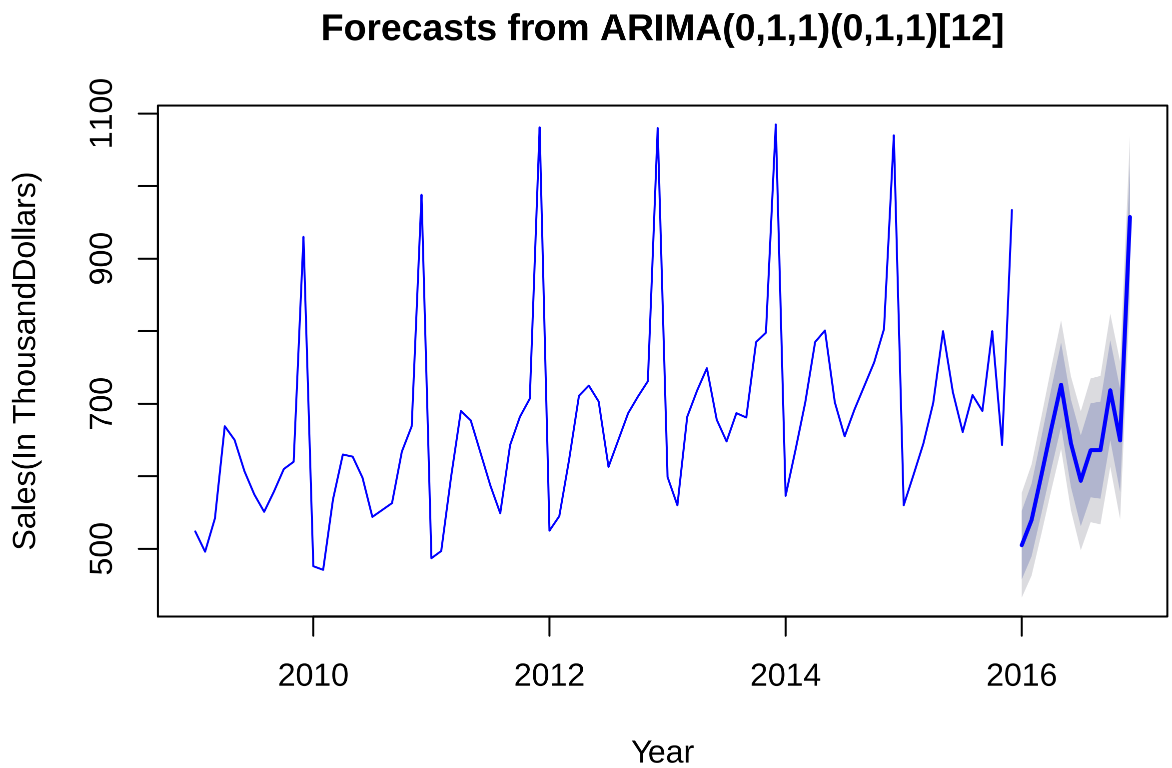
```r
summary(auto_arima_total_women)
```

```
## Series: total_data_women_linear_ts
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.5382  -0.5683
## s.e.   0.1202   0.1803
##
## sigma^2 estimated as 45928:  log likelihood=-483.32
## AIC=972.65   AICc=973.01   BIC=979.44
##
## Training set error measures:
##                      ME     RMSE     MAE        MPE     MAPE      MASE
## Training set -8.48645 194.2339 136.468 -0.7281371 4.081784 0.4058108
##                     ACF1
## Training set -0.02042828
```

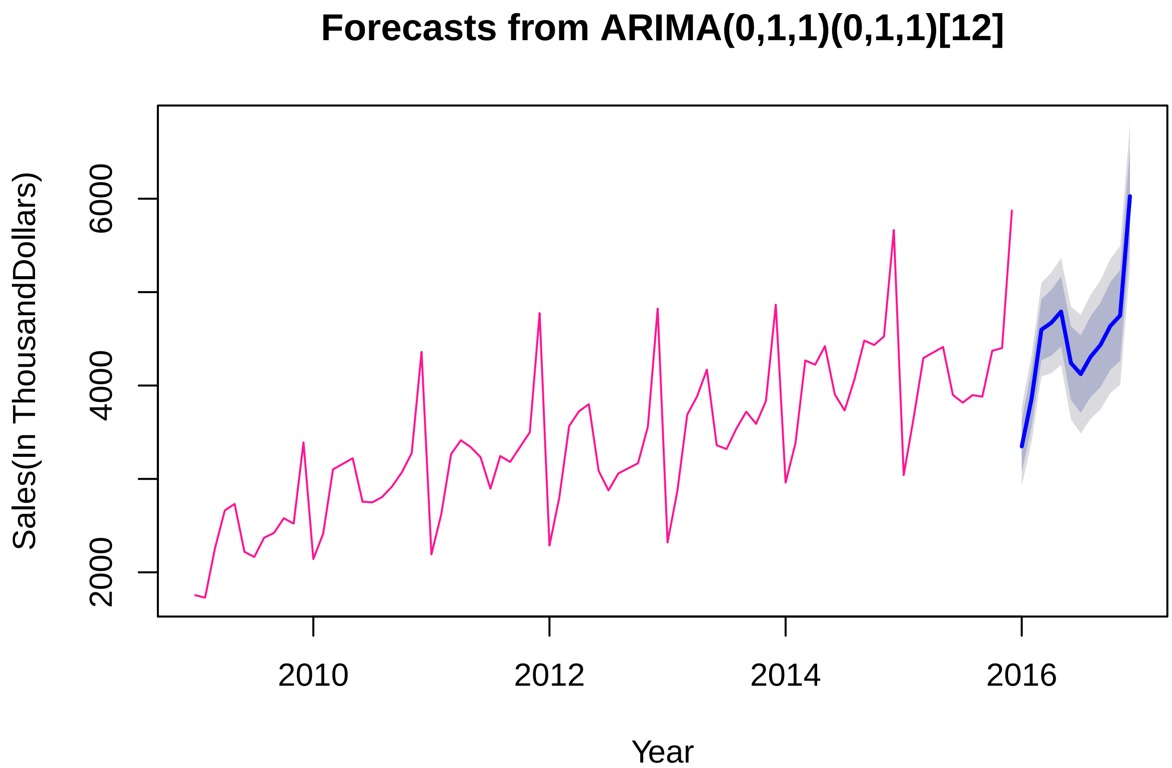
```r
summary(auto_arima_total_others)
```

```
## Series: total_data_others_linear_ts
## ARIMA(0,0,1)(0,1,1)[12] with drift
##
## Coefficients:
##          ma1     sma1   drift
##       0.2793  -0.7029  5.7726
## s.e.  0.1181   0.1511  0.4418
##
## sigma^2 estimated as 5287:  log likelihood=-413.34
## AIC=834.69   AICc=835.29   BIC=843.8
##
## Training set error measures:
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -1.477429 65.90174 47.56879 -0.4329567 3.917721 0.5723518
##                      ACF1
## Training set 0.0008476278
```

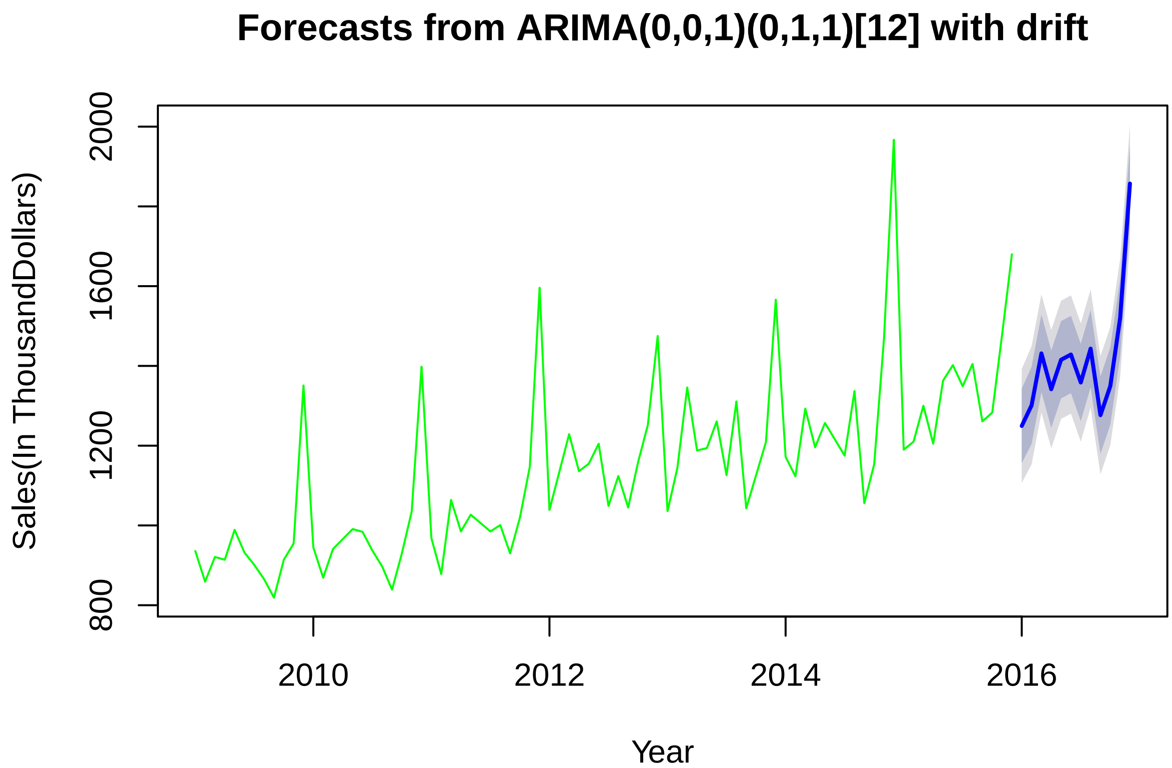
```r
# Forecasting
forecast_auto_arima_men = forecast(auto_arima_total_men, h=12)
forecast_auto_arima_women = forecast(auto_arima_total_women, h=12)
forecast_auto_arima_others = forecast(auto_arima_total_others, h=12)

# Plotting the forecasted results
plot(forecast_auto_arima_men,col="blue", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```




```
plot(forecast_auto_arima_women,col="#FF1493", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```




```
plot(forecast_auto_arima_others,col="green", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



Forecasted results for each category

Select the values from either mean , or lower or upper confidence values based on the plot.

```
forecast_auto_arima_men$upper[,1]
```

```
##             Jan       Feb       Mar       Apr       May       Jun       Jul
## 2016  552.1939  589.8645  654.4571  720.6724  784.1354  705.5750  656.2969
##             Aug       Sep       Oct       Nov       Dec
## 2016  700.5545  702.9290  787.5001  720.3351 1030.2746
```

```
forecast_auto_arima_women$lower[,1]
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2016 3073.368 3557.138 4269.758 4319.133 4417.653 3845.442 3707.821
##           Aug      Sep      Oct      Nov      Dec
## 2016 3874.131 3982.189 4167.292 4263.557 5524.443
```

```
forecast_auto_arima_others$lower[,2]
```

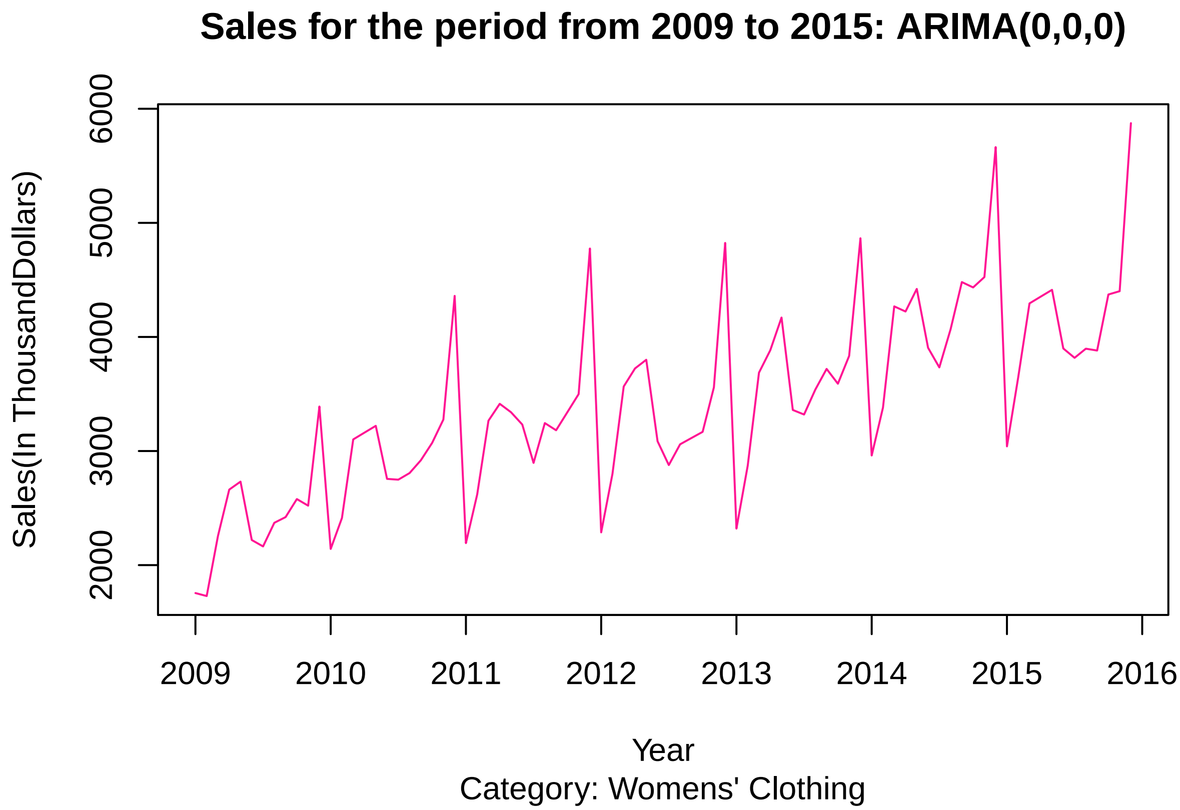
```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2016 1106.704 1152.899 1283.141 1193.310 1267.173 1280.334 1210.174
##           Aug      Sep      Oct      Nov      Dec
## 2016 1295.228 1128.258 1201.094 1371.712 1709.052
```

Manual ARIMA model

Manual AARIMA model for women

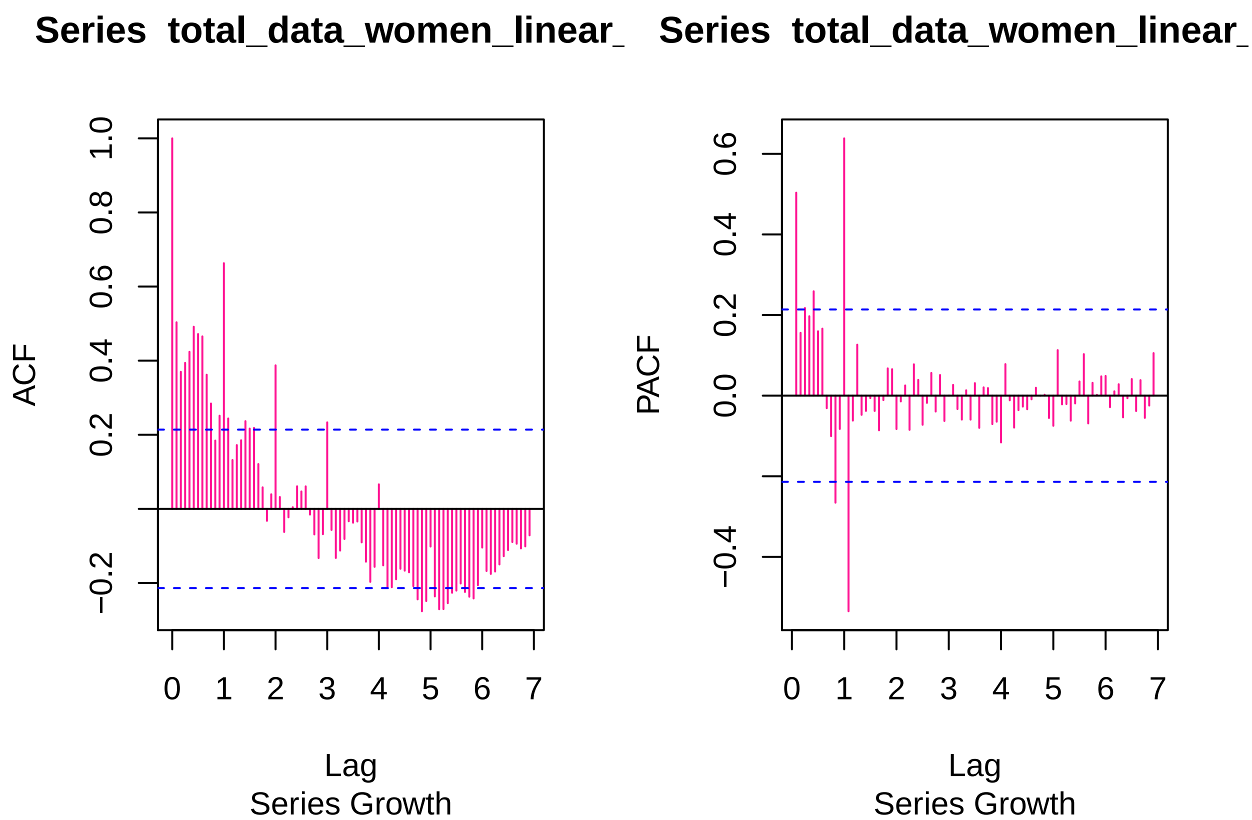
Step 1: Plot the Sales Forecasting data

```
plot(total_data_women_linear_ts, col = "#FF1493", main = "Sales for the period from 2009 to 2015: ARIMA
     sub = "Category: Womens' Clothing", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



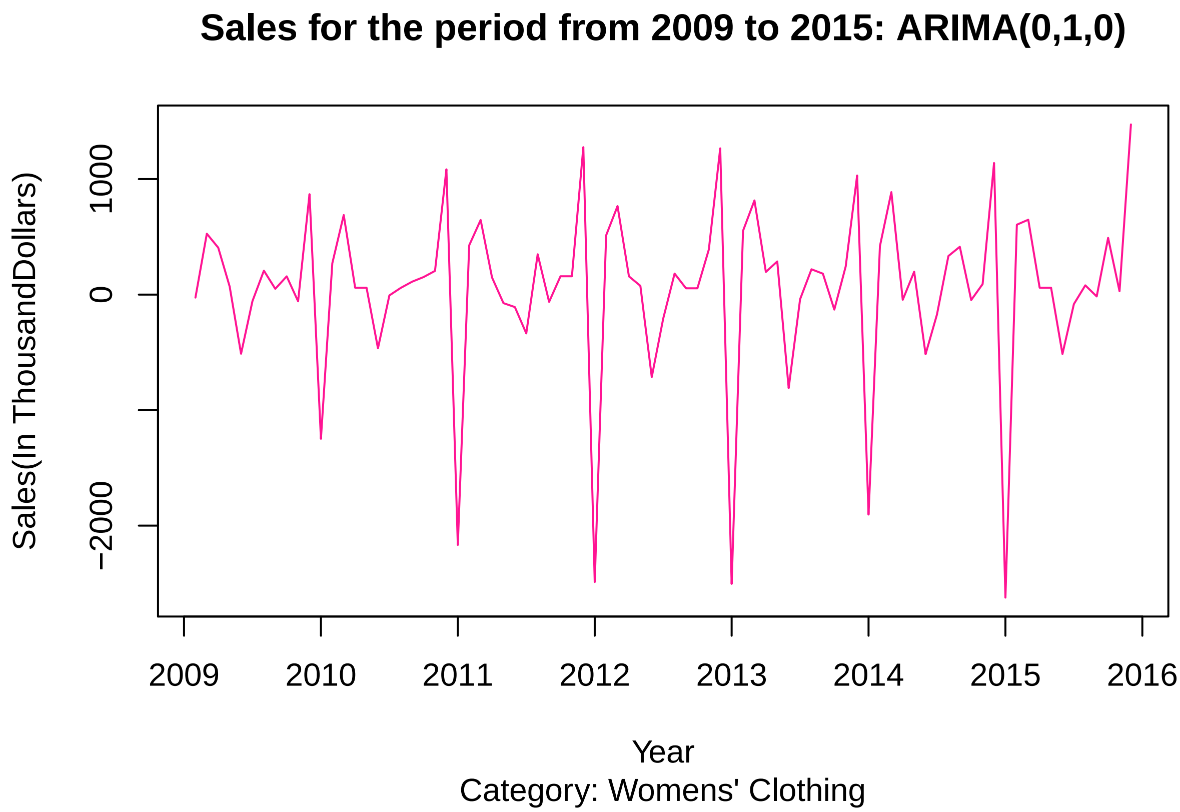
Step 2: Plotting ACF and PACF to get preliminary understanding of the process

```
acf = acf(total_data_women_linear_ts, lag.max =120, plot = FALSE)
pacf = pacf(total_data_women_linear_ts, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acf, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacf, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



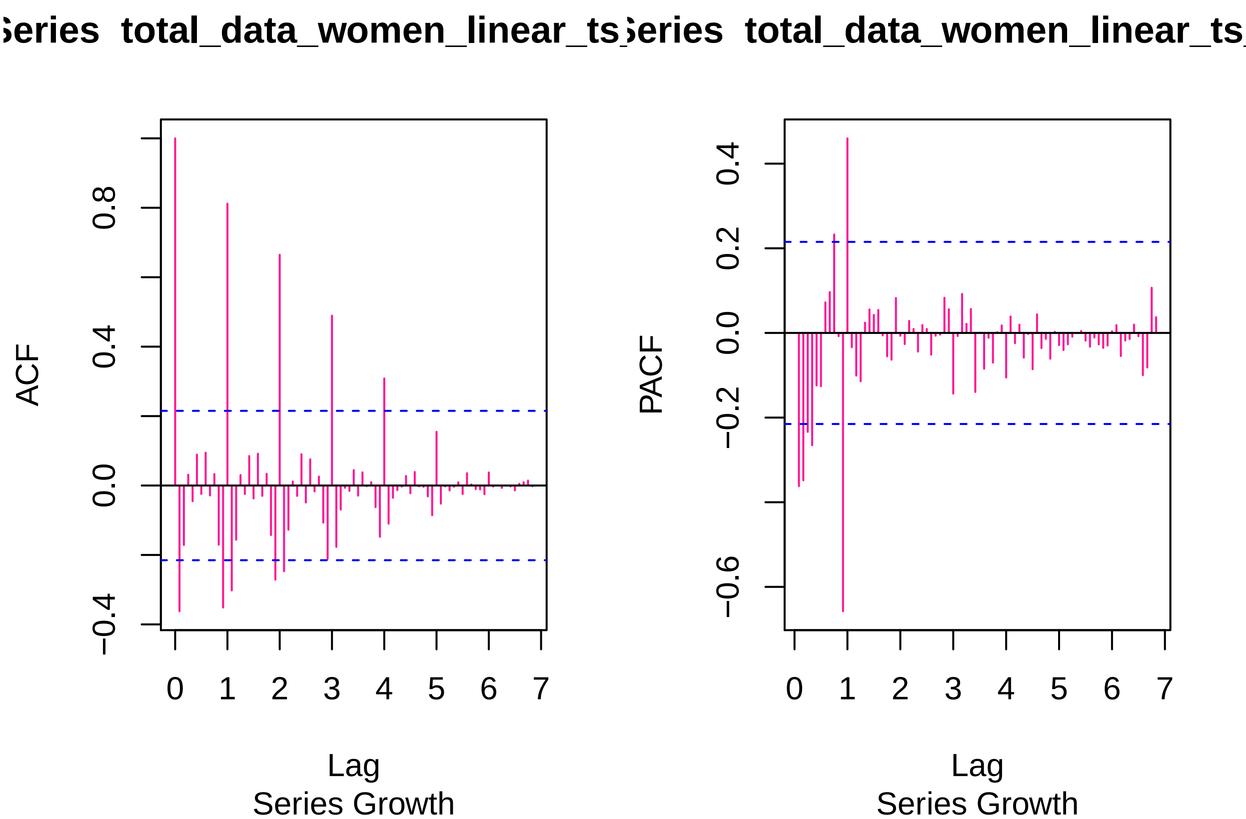
Step 3: The suspension bridge pattern in ACF suggests both nonstationarity and strong seasonality. Perform a non-seasonal difference to give an ARIMA(0,1,0) model.

```
par(mfrow = c(1, 1), bg = "white")
total_data_women_linear_ts_diff1 = diff(total_data_women_linear_ts, differences = 1)
plot(total_data_women_linear_ts_diff1, col = "#FF1493", main = "Sales for the period from 2009 to 2015:
     sub = "Category: Womens' Clothing", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



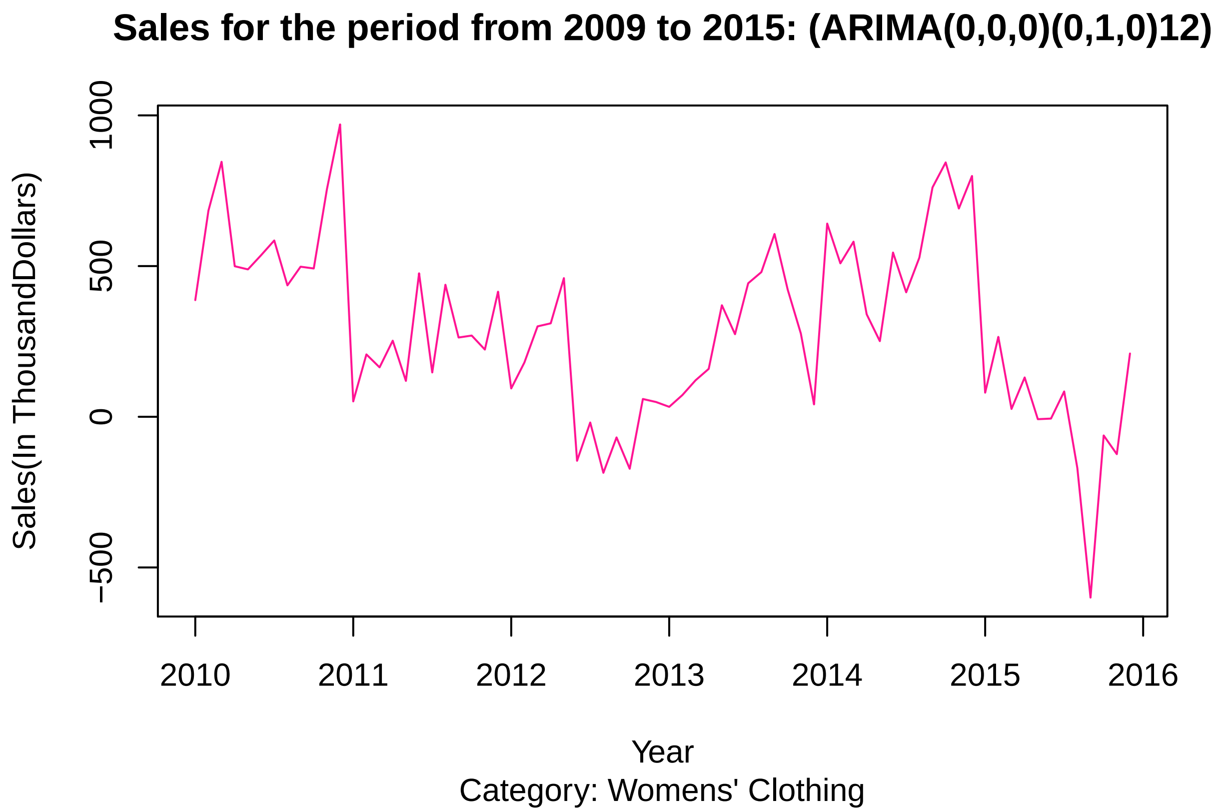
Step 4: Check ACF and PACF to explore remaining dependencies

```
acf_1 = acf(total_data_women_linear_ts_diff1, lag.max =120, plot = FALSE)
pacf_1 = pacf(total_data_women_linear_ts_diff1, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acf_1, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacf_1, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



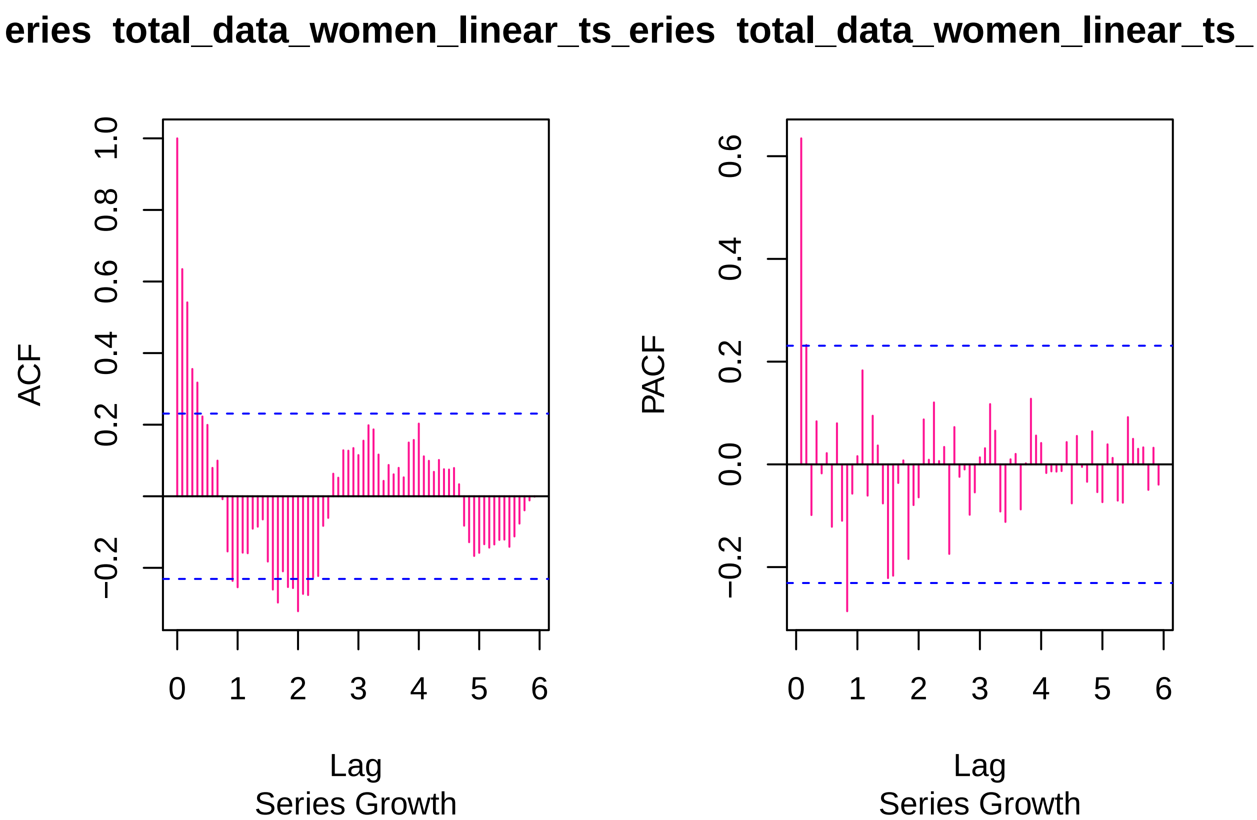
Step 5: The differenced series looks stationary but has strong seasonal lags. Perform a seasonal differencing on the original time series (ARIMA(0,0,0)(0,1,0)12)

```
par(mfrow = c(1, 1), bg = "white")
total_data_women_linear_ts_sdiff1 = diff(total_data_women_linear_ts, lag = 12, differences = 1)
plot(total_data_women_linear_ts_sdiff1, col = "#FF1493", main = "Sales for the period from 2009 to 2015
     xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



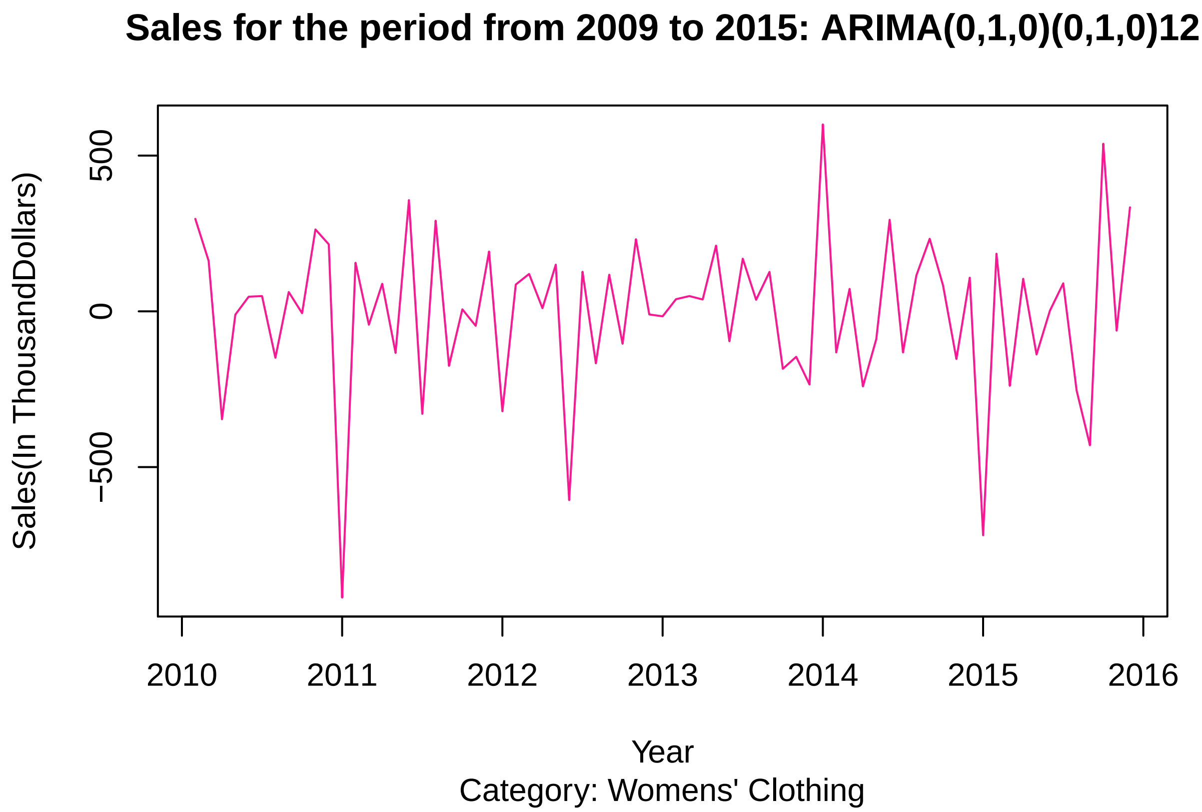
Step 6: Check ACF and PACF for seasonally differenced data to explore remaining dependencies

```
acf_s1 = acf(total_data_women_linear_ts_sdiff1, lag.max =120, plot = FALSE)
pacf_s1 = pacf(total_data_women_linear_ts_sdiff1, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acf_s1, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacf_s1, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



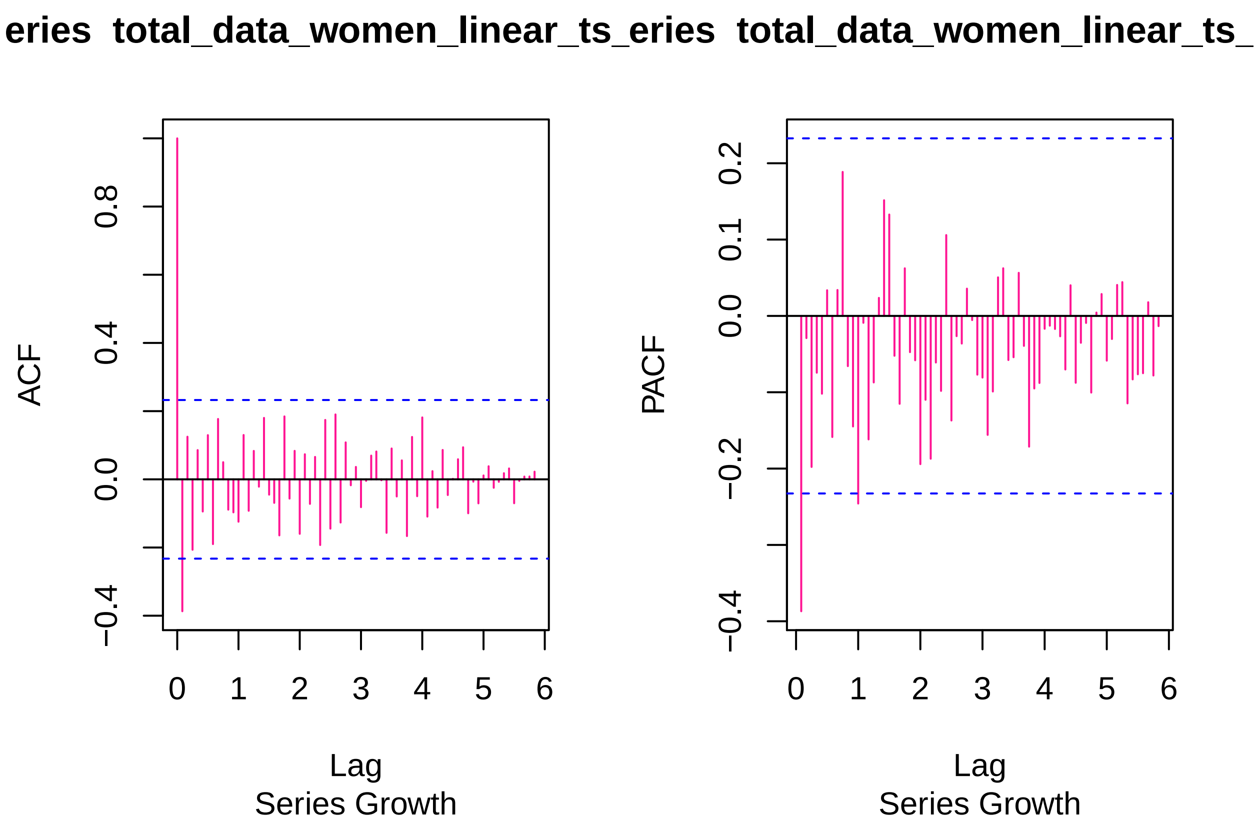
Step 7: Strong positive autocorrelation indicates need for either an AR component or a non-seasonal differencing. Perform a non-seasonal differencing on a seasonal differenced data.

```
par(mfrow = c(1, 1), bg = "white")
total_data_women_linear_ts_sdiff2 = diff(total_data_women_linear_ts_sdiff1, differences = 1)
plot(total_data_women_linear_ts_sdiff2, col = "#FF1493", main = "Sales for the period from 2009 to 2015
```



Step 8: Check ACF and PACF to explore remaining dependencies

```
acf_s1d2 = acf(total_data_women_linear_ts_sdiff2, lag.max =120, plot = FALSE)
pacf_s1d2 = pacf(total_data_women_linear_ts_sdiff2, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acf_s1d2, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacf_s1d2, col = "#FF1493", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



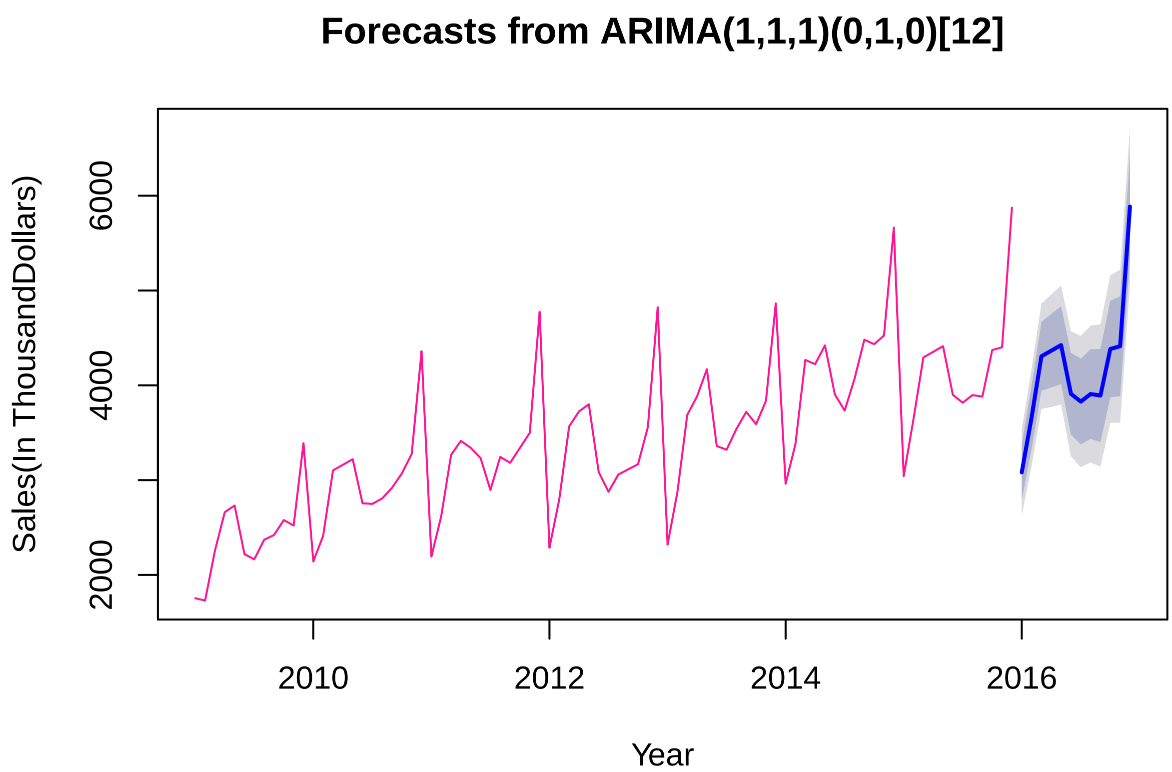
Step 9: ACF and PACF shows that we need to use an AR(1) and an MA(1) term.

```r
sales_women_arima = Arima(total_data_women_linear_ts, order = c(1,1,1), seasonal = c(0,1,0), include.dri
summary(sales_women_arima)
```

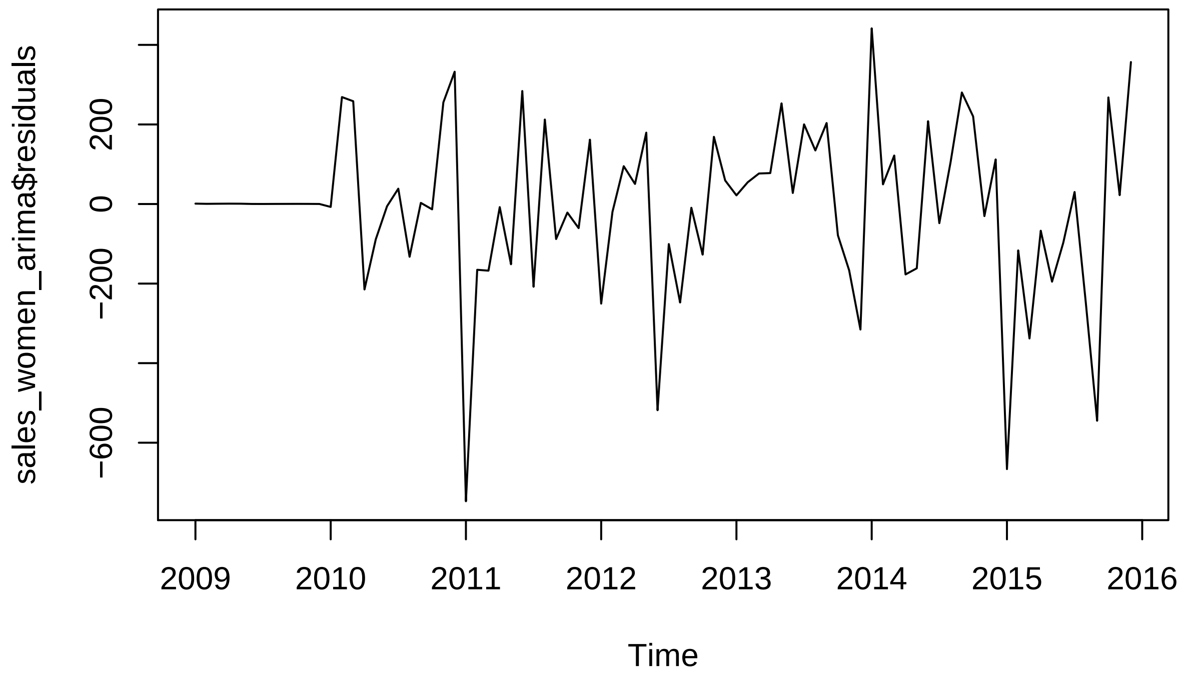
```
## Series: total_data_women_linear_ts
## ARIMA(1,1,1)(0,1,0)[12]
##
## Coefficients:
##          ar1      ma1
##       0.1486  -0.6129
## s.e.  0.2892   0.2423
##
## sigma^2 estimated as 53488:  log likelihood=-486.38
## AIC=978.75   AICc=979.11   BIC=985.54
##
## Training set error measures:
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -11.48128 209.6105 145.7723 -0.6406699 4.285231 0.4334787
##                    ACF1
## Training set -0.01486991
```

Step 10: Forcasting the sales for women category for the next year

```
forecast_manual_arima_women = forecast(sales_women_arima, h = 12)
plot(forecast_manual_arima_women,col="#FF1493", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



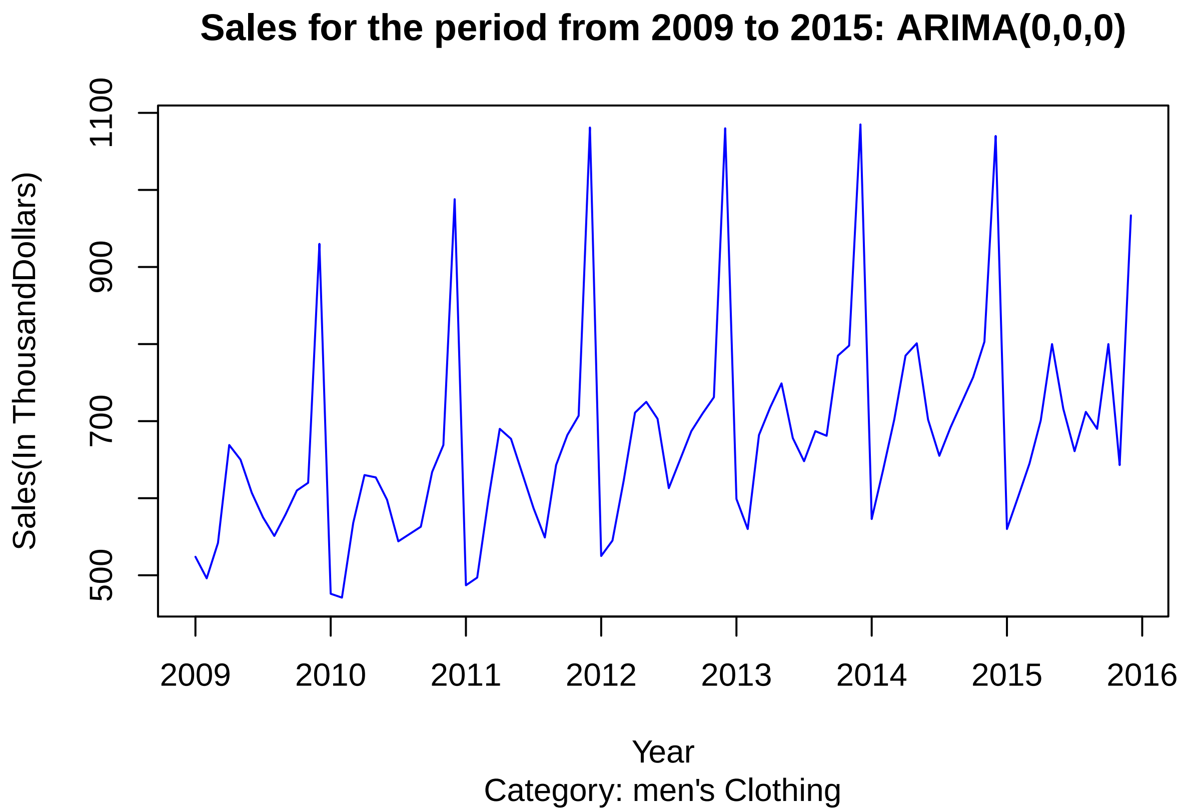

```
plot(sales_women_arima$residuals)
```



Manual AARIMA model for men

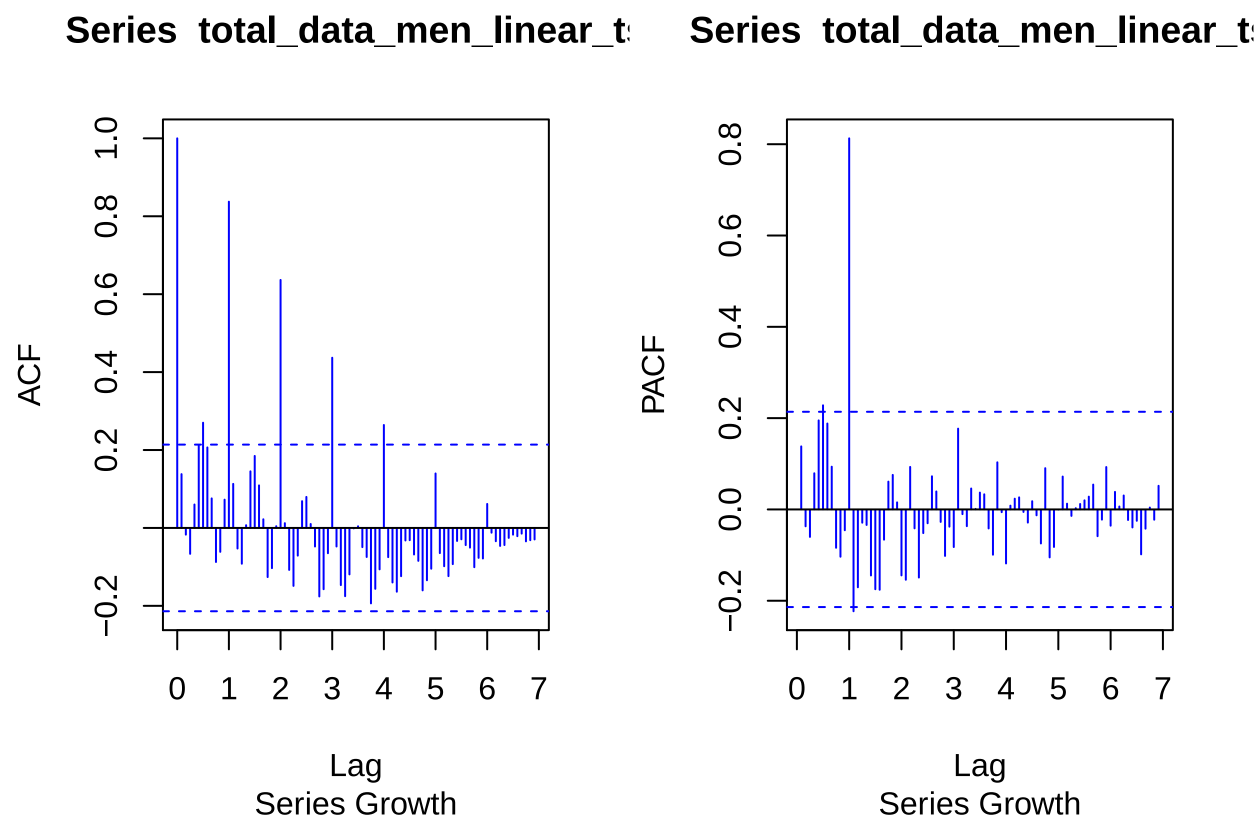
Step 1: Plot the Sales Forecasting data

```
plot(total_data_men_linear_ts, col = "blue", main = "Sales for the period from 2009 to 2015: ARIMA(0,0,(
```



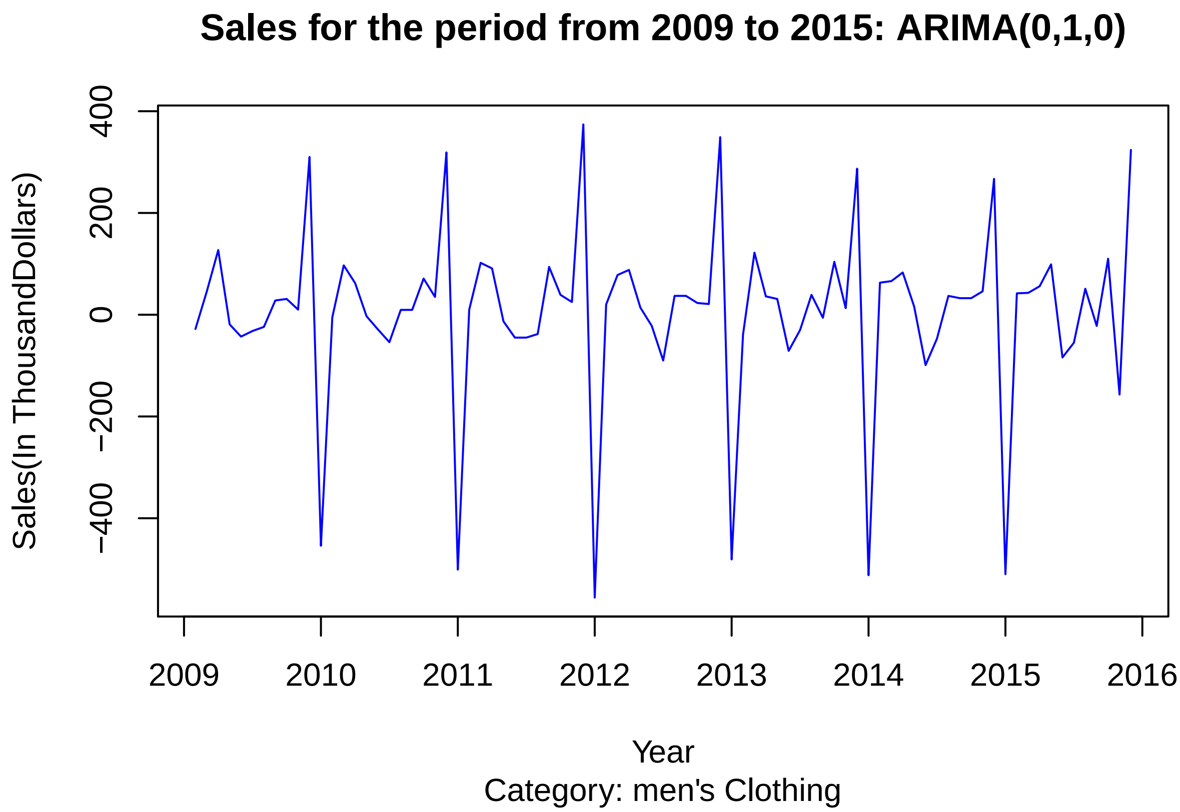
Step 2: Plotting ACF and PACF to get preliminary understanding of the process

```
acfm = acf(total_data_men_linear_ts, lag.max =120, plot = FALSE)
pacfm = pacf(total_data_men_linear_ts, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acfm, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacfm, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



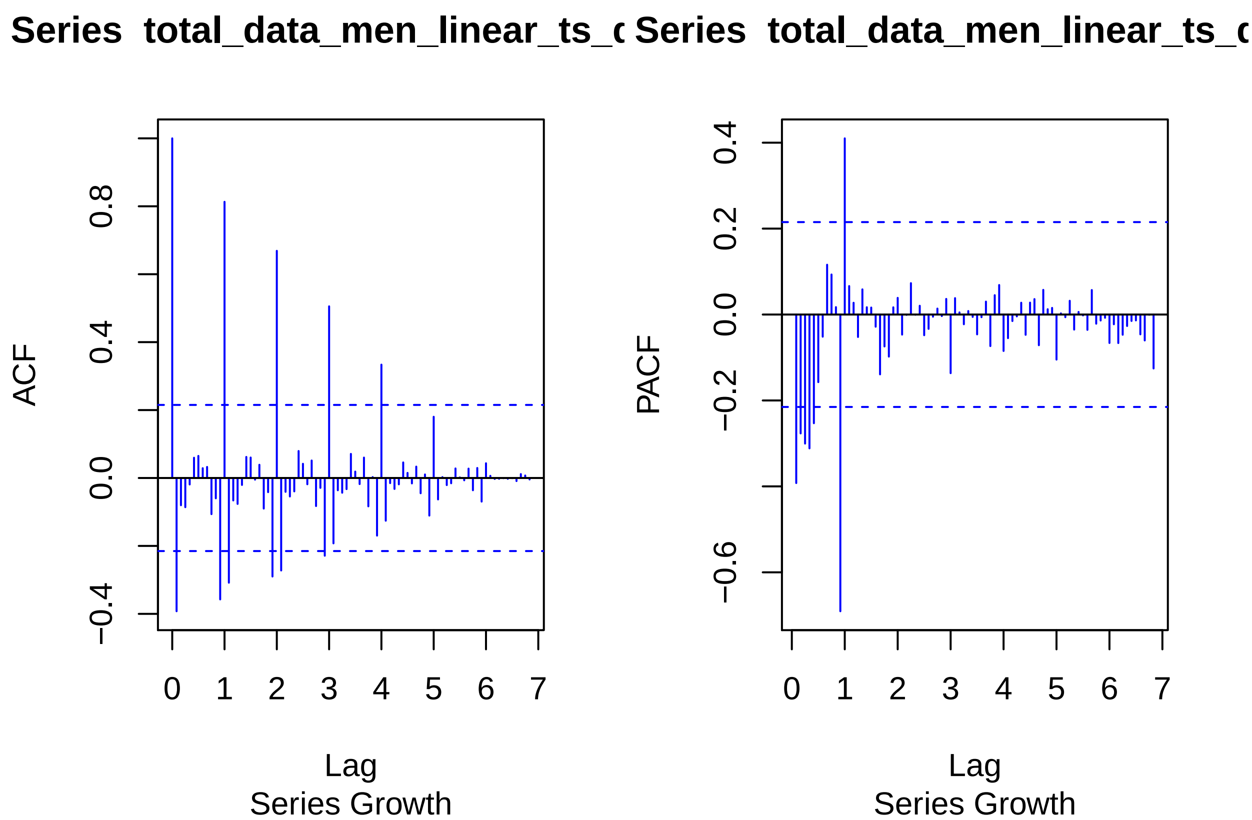
Step 3: The suspension bridge pattern in ACF suggests both nonstationarity and strong seasonality. Perform a non-seasonal difference to give an ARIMA(0,1,0) model.

```
par(mfrow = c(1, 1), bg = "white")
total_data_men_linear_ts_diff1 = diff(total_data_men_linear_ts, differences = 1)
plot(total_data_men_linear_ts_diff1, col = "blue", main = "Sales for the period from 2009 to 2015: ARIMA
     sub = "Category: men's Clothing", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



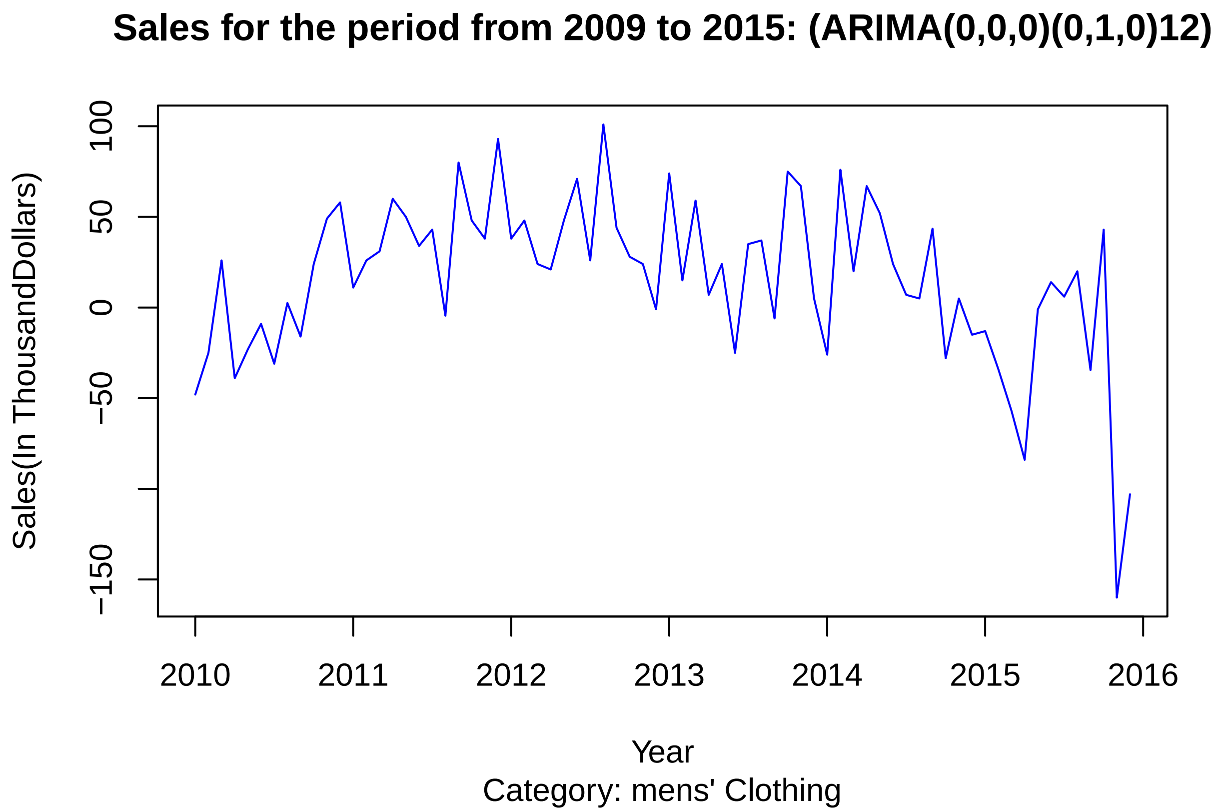
Step 4: Check ACF and PACF to explore remaining dependencies

```
acfm_1 = acf(total_data_men_linear_ts_diff1, lag.max =120, plot = FALSE)
pacfm_1 = pacf(total_data_men_linear_ts_diff1, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acfm_1, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacfm_1, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



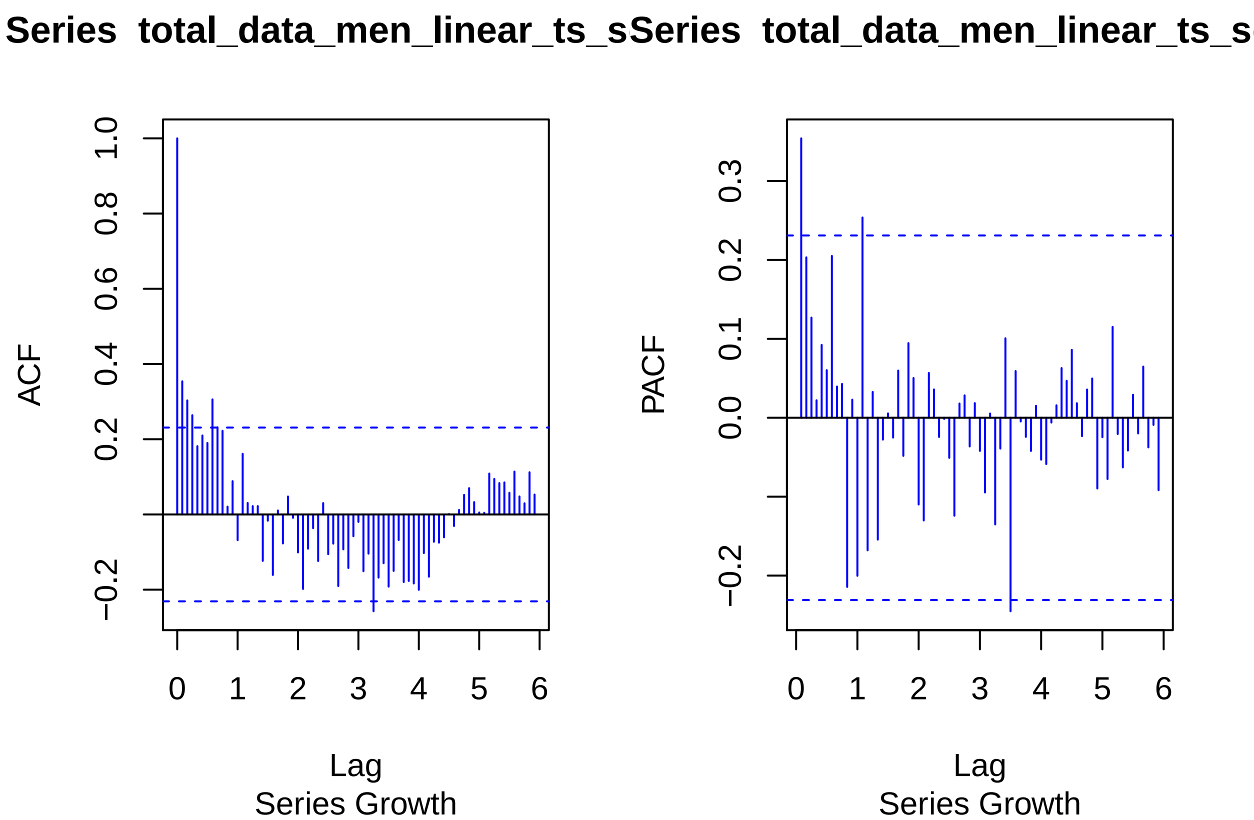
Step 5: The differenced series looks stationary but has strong seasonal lags. Perform a seasonal differencing on the original time series (ARIMA(0,0,0)(0,1,0)12)

```
par(mfrow = c(1, 1), bg = "white")
total_data_men_linear_ts_sdiff1 = diff(total_data_men_linear_ts, lag = 12, differences = 1)
plot(total_data_men_linear_ts_sdiff1, col = "blue", main = "Sales for the period from 2009 to 2015: (AR
     xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



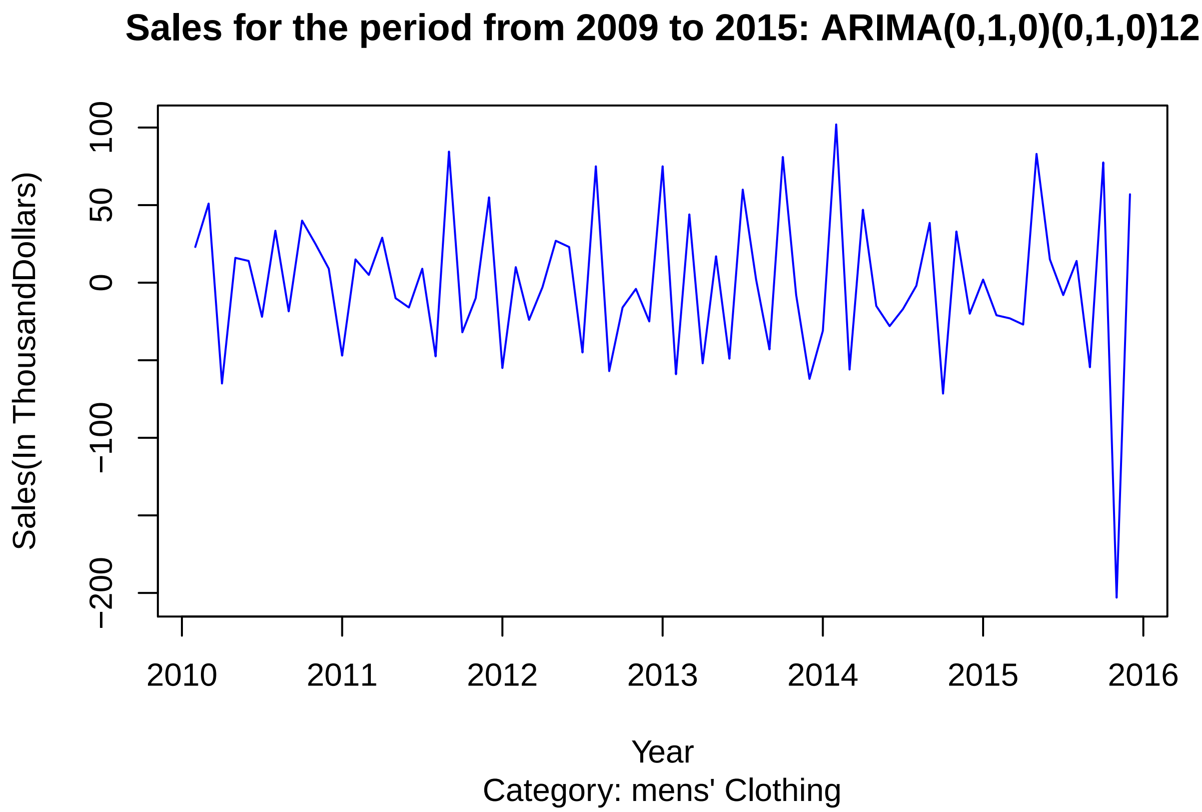
Step 6: Check ACF and PACF for seasonally differenced data to explore remaining dependencies

```
acfm_s1 = acf(total_data_men_linear_ts_sdiff1, lag.max =120, plot = FALSE)
pacfm_s1 = pacf(total_data_men_linear_ts_sdiff1, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acfm_s1, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacfm_s1, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



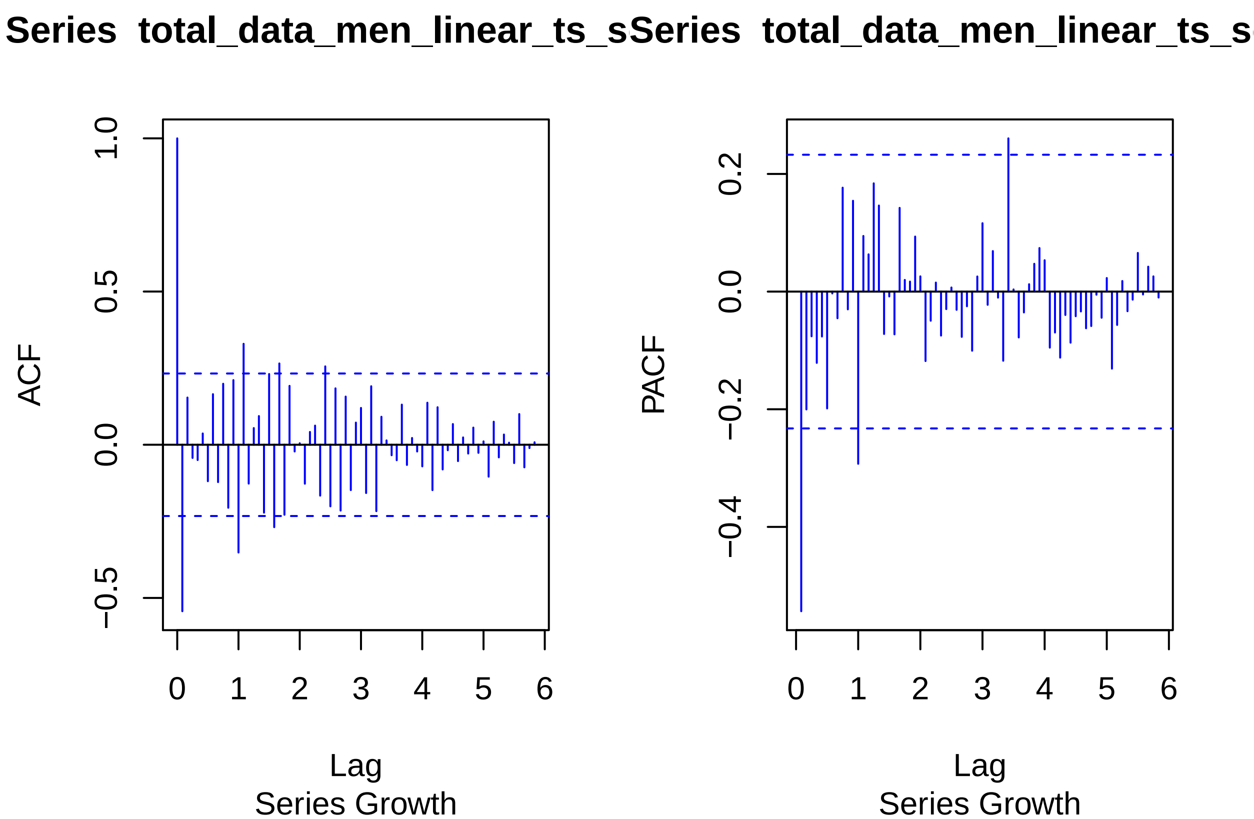
Step 7: Strong positive autocorrelation indicates need for either an AR component or a non-seasonal differencing. Perform a non-seasonal differencing on a seasonal differenced data.

```
par(mfrow = c(1, 1), bg = "white")
total_data_men_linear_ts_sdiff2 = diff(total_data_men_linear_ts_sdiff1, differences = 1)
plot(total_data_men_linear_ts_sdiff2, col = "blue", main = "Sales for the period from 2009 to 2015: ARI
```



Step 8: Check ACF and PACF to explore remaining dependencies

```
acfm_s1d2 = acf(total_data_men_linear_ts_sdiff2, lag.max =120, plot = FALSE)
pacfm_s1d2 = pacf(total_data_men_linear_ts_sdiff2, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acfm_s1d2, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacfm_s1d2, col = "blue", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



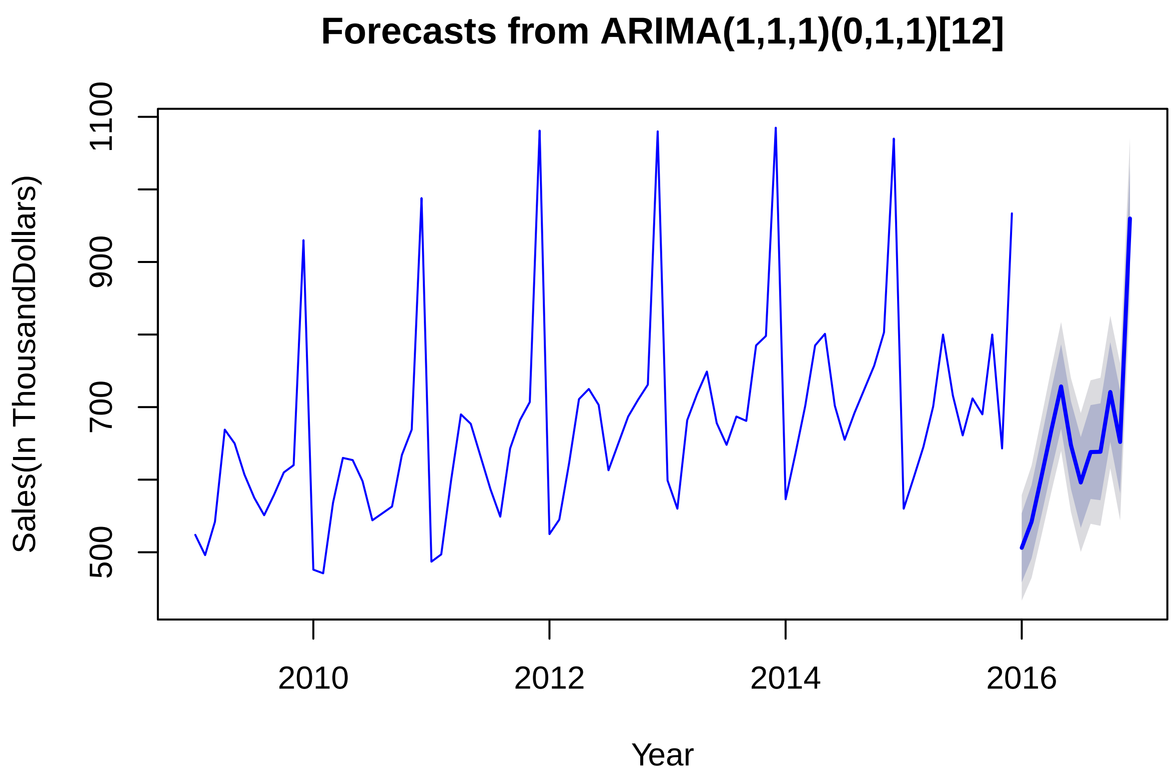
Step 9: ACF and PACF shows that we need to use an AR(1) and an MA(1) and a negative seasonal term.

```
sales_men_arima = Arima(total_data_men_linear_ts, order = c(1,1,1), seasonal = c(0,1,1), include.drift =
summary(sales_men_arima)
```

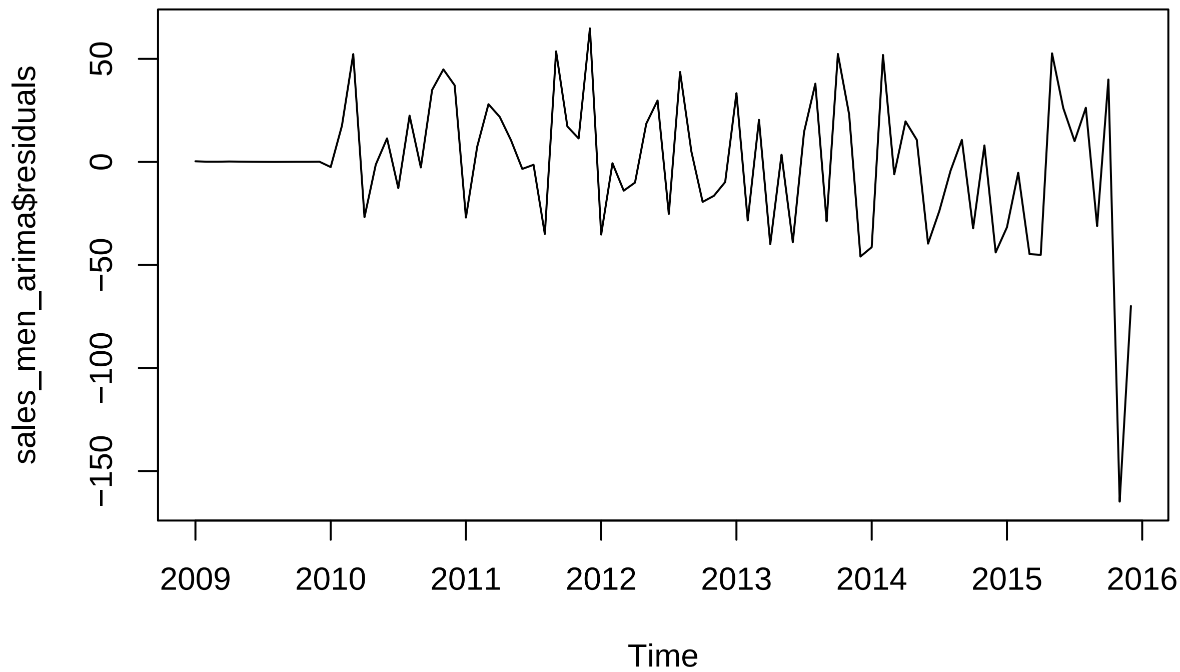
```
## Series: total_data_men_linear_ts
## ARIMA(1,1,1)(0,1,1)[12]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.0262  -0.6637  -0.4641
## s.e.  0.1997   0.1720   0.1641
##
## sigma^2 estimated as 1378:  log likelihood=-357.55
## AIC=723.1   AICc=723.7   BIC=732.15
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.4057578 33.39886 23.62888 -0.1121924 3.445873 0.6277785
##                     ACF1
## Training set -0.01114166
```

Step 10: Forcasting the sales for men category for the next year

```
forecast_manual_arima_men = forecast(sales_men_arima, h = 12)
plot(forecast_manual_arima_men,col="blue", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



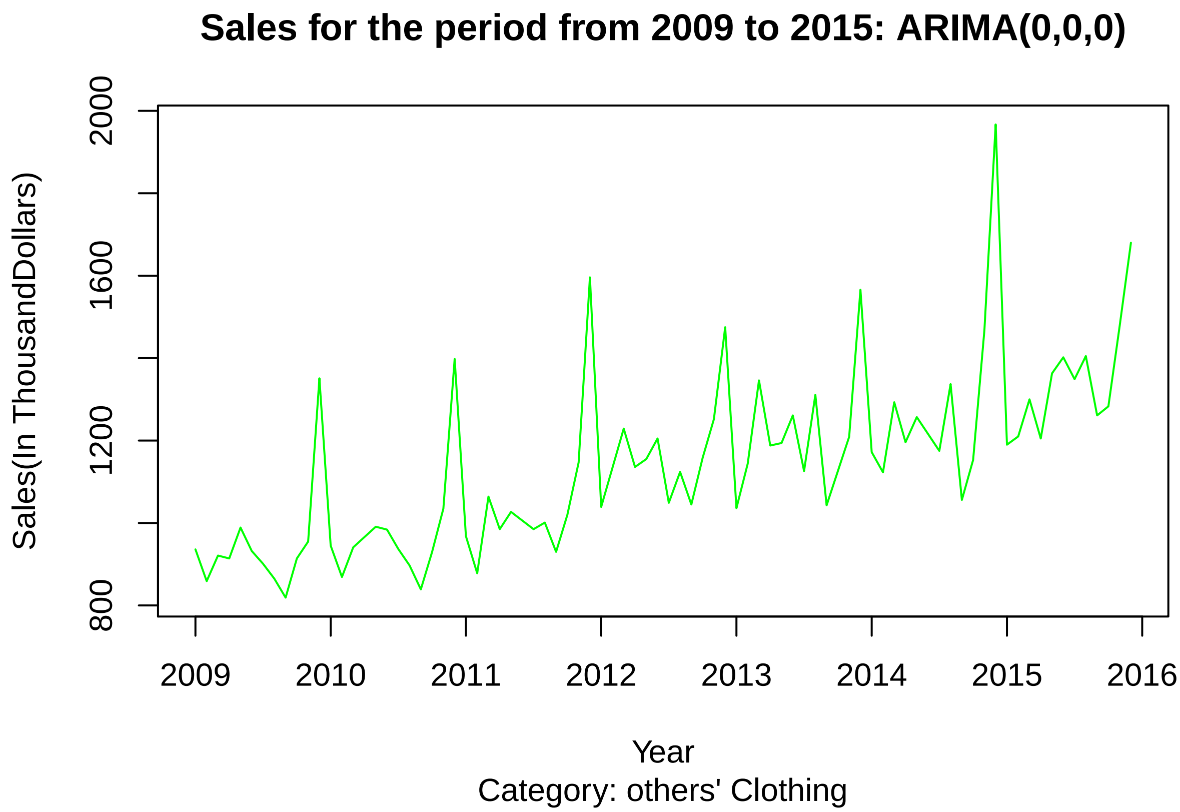

```
plot(sales_men_arima$residuals)
```



Manual AARIMA model for others

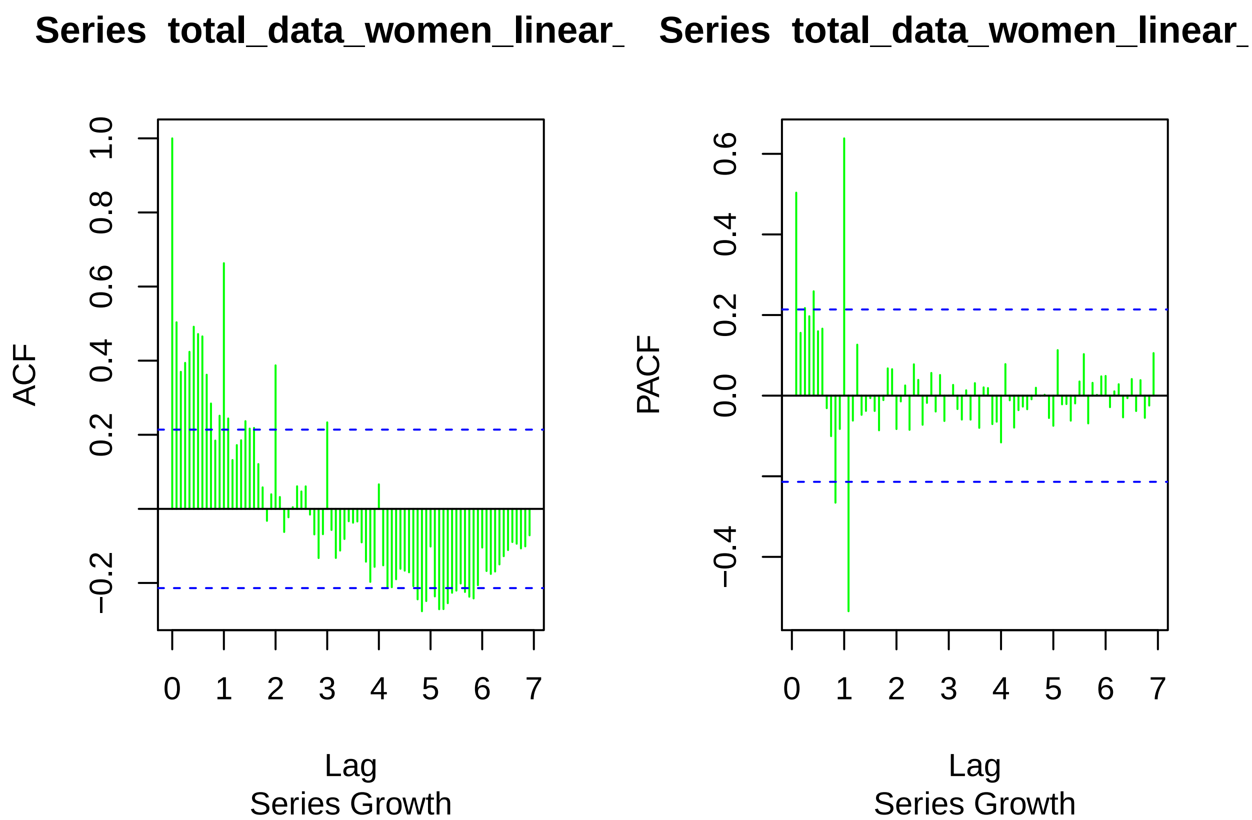
Step 1: Plot the Sales Forecasting data

```
plot(total_data_others_linear_ts, col = "green", main = "Sales for the period from 2009 to 2015: ARIMA(
```



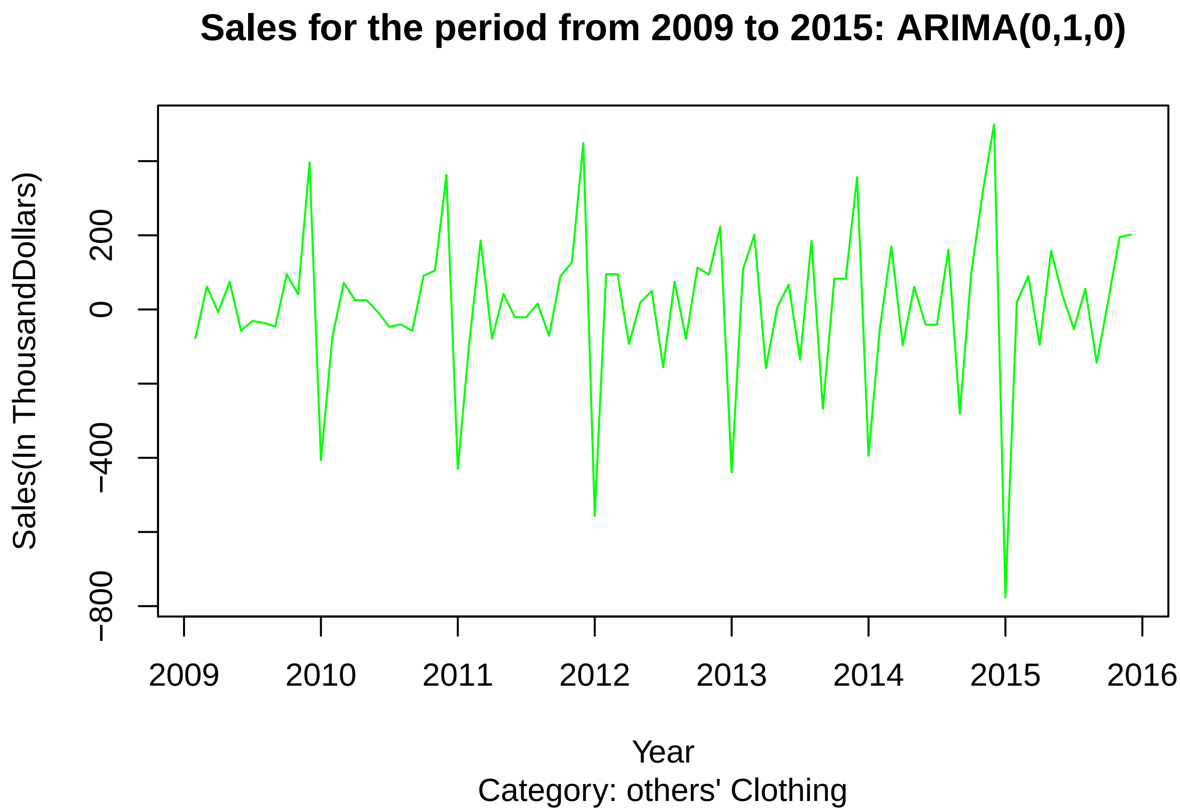
Step 2: Plotting ACF and PACF to get preliminary understanding of the process

```
acfo = acf(total_data_others_linear_ts, lag.max =120, plot = FALSE)
pacfo = pacf(total_data_others_linear_ts, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acf, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacf, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



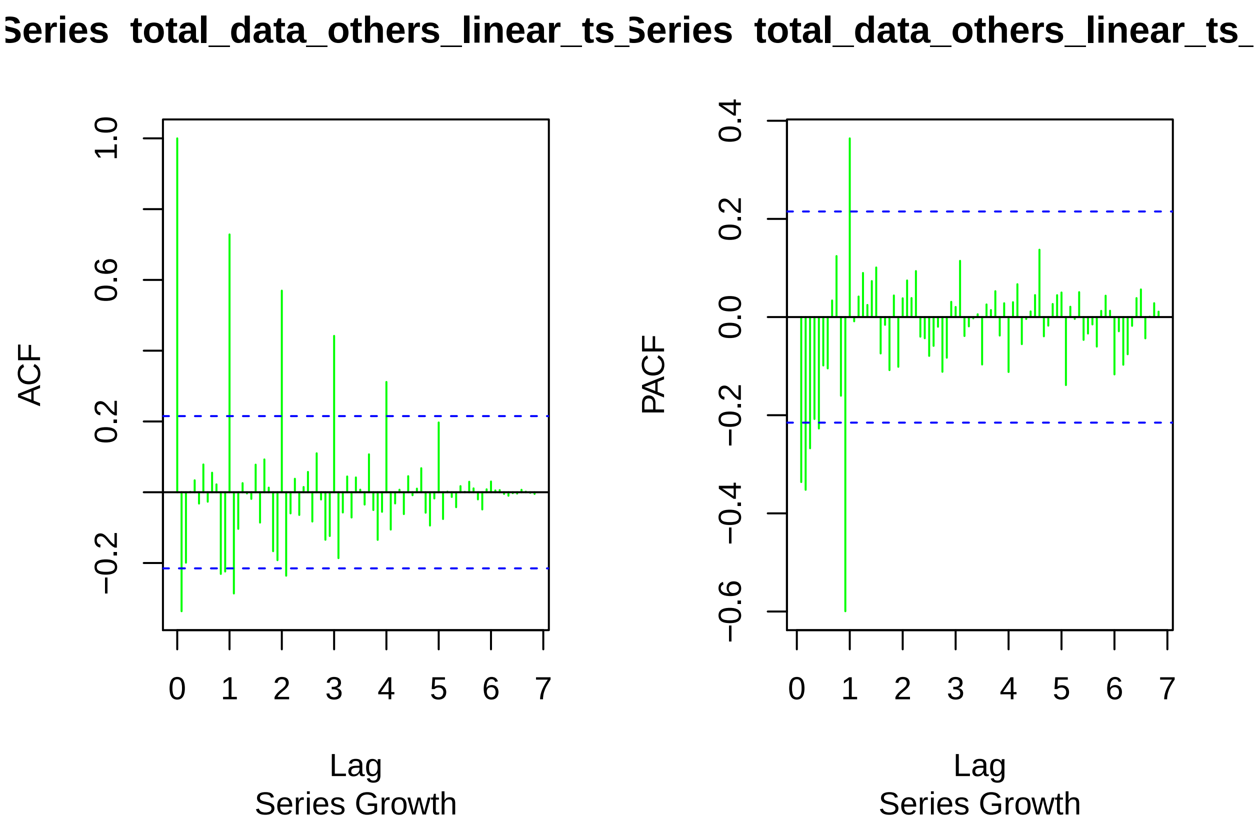
Step 3: The suspension bridge pattern in ACF suggests both nonstationarity and strong seasonality. Perform a non-seasonal difference to give an ARIMA(0,1,0) model.

```
par(mfrow = c(1, 1), bg = "white")
total_data_others_linear_ts_diff1 = diff(total_data_others_linear_ts, differences = 1)
plot(total_data_others_linear_ts_diff1, col = "green", main = "Sales for the period from 2009 to 2015: /
     sub = "Category: others' Clothing", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



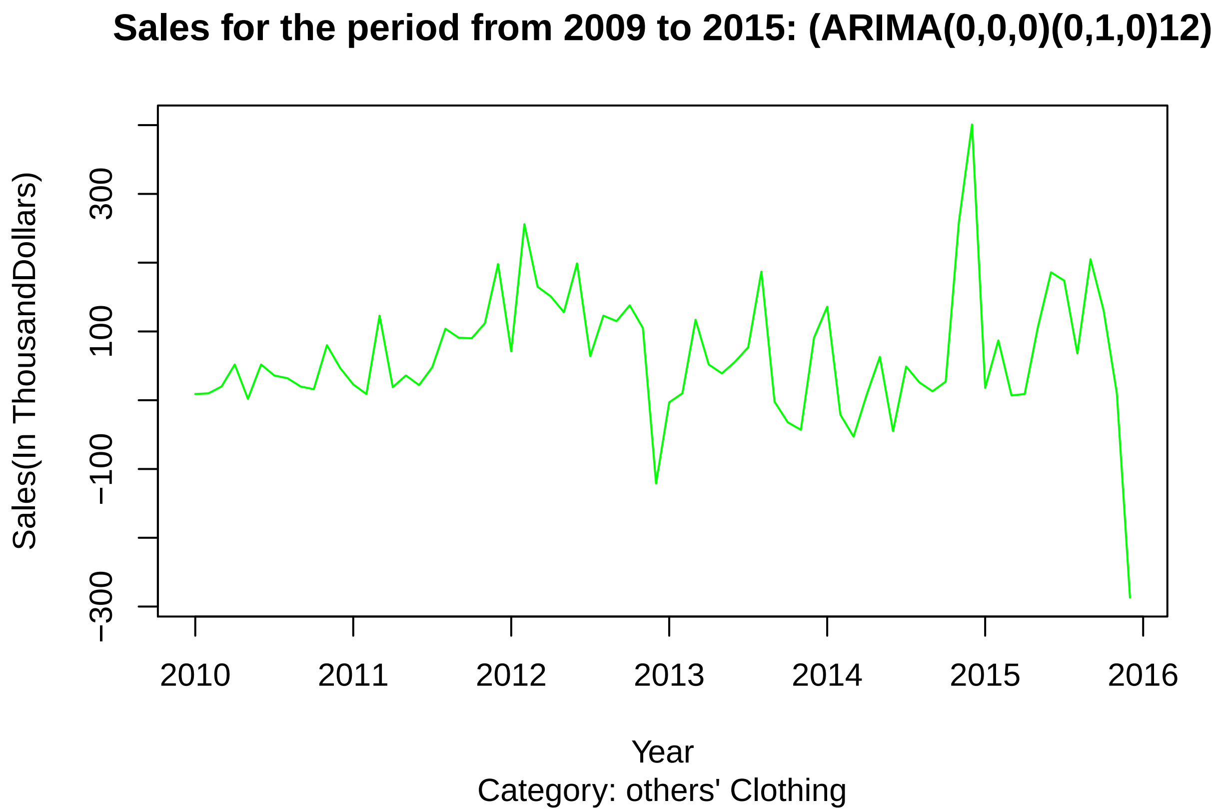
Step 4: Check ACF and PACF to explore remaining dependencies

```
acfo_1 = acf(total_data_others_linear_ts_diff1, lag.max =120, plot = FALSE)
pacfo_1 = pacf(total_data_others_linear_ts_diff1, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acfo_1, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacfo_1, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



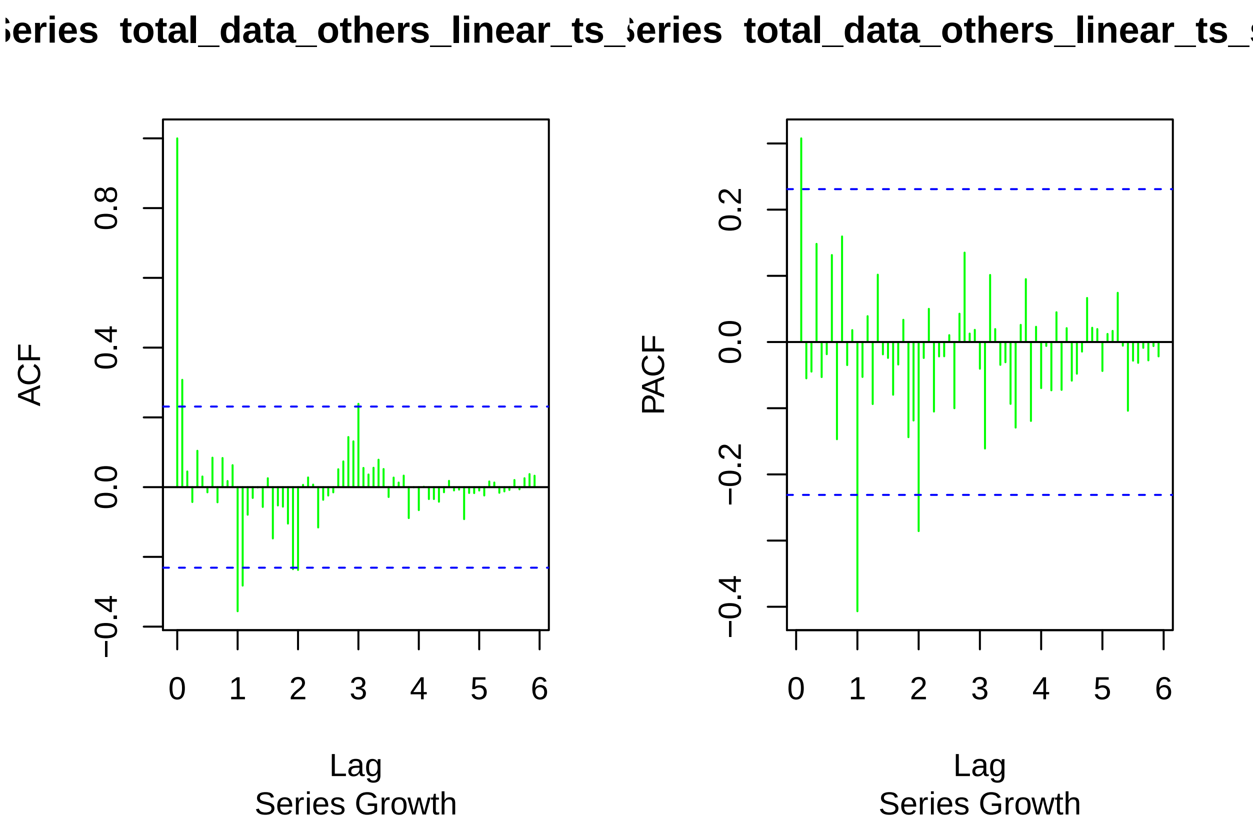
Step 5: The differenced series looks stationary but has strong seasonal lags. Perform a seasonal differencing on the original time series (ARIMA(0,0,0)(0,1,0)12)

```
par(mfrow = c(1, 1), bg = "white")
total_data_others_linear_ts_sdiff1 = diff(total_data_others_linear_ts, lag = 12, differences = 1)
plot(total_data_others_linear_ts_sdiff1, col = "green", main = "Sales for the period from 2009 to 2015:
     xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



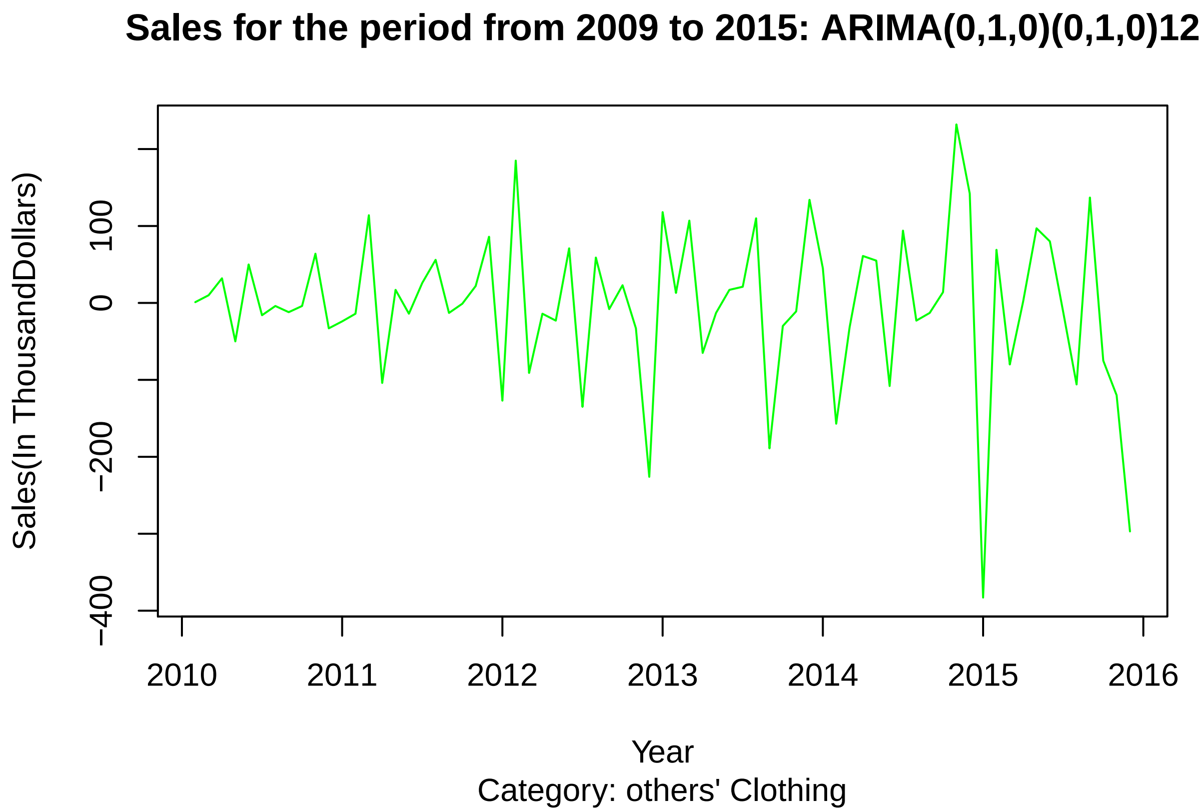
Step 6: Check ACF and PACF for seasonally differenced data to explore remaining dependencies

```
acfo_s1 = acf(total_data_others_linear_ts_sdiff1, lag.max =120, plot = FALSE)
pacfo_s1 = pacf(total_data_others_linear_ts_sdiff1, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acfo_s1, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacfo_s1, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



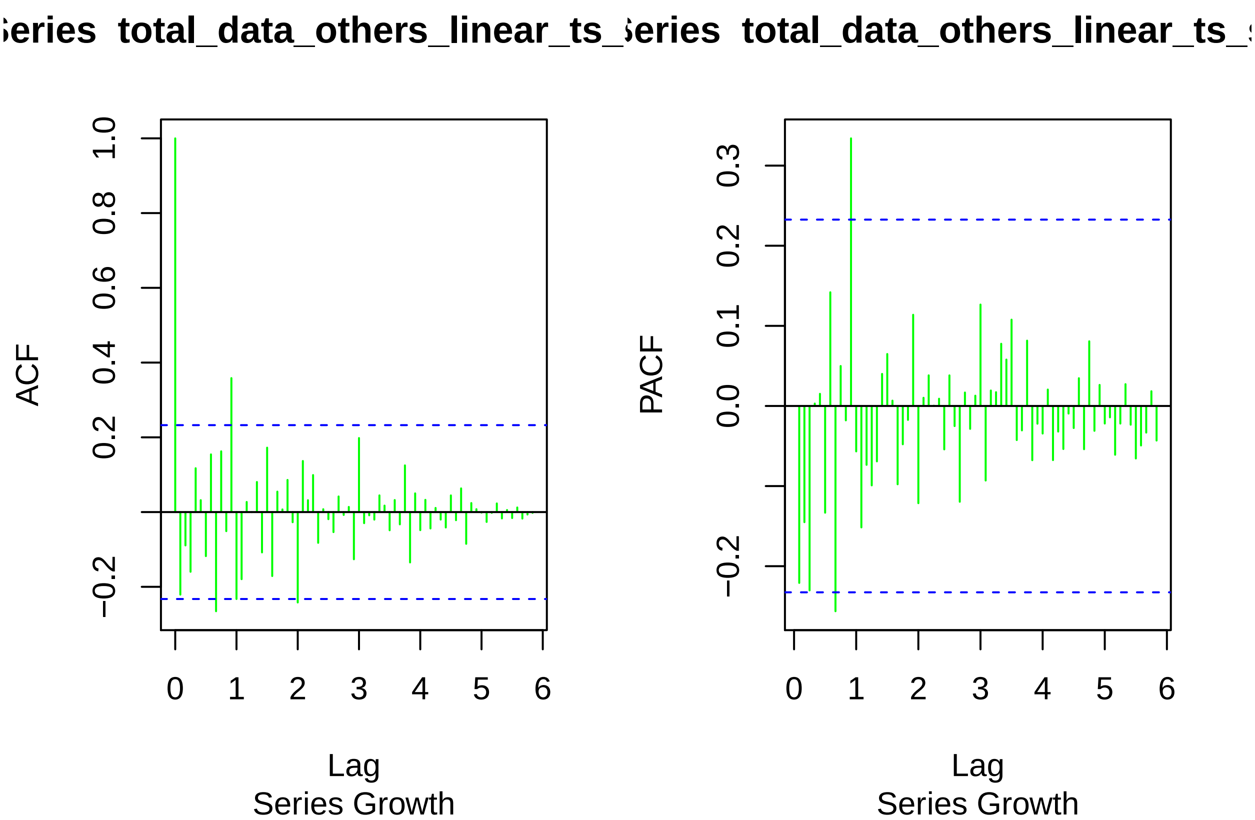
Step 7: Strong positive autocorrelation indicates need for either an AR component or a non-seasonal differencing. Perform a non-seasonal differencing on a seasonal differenced data.

```
par(mfrow = c(1, 1), bg = "white")
total_data_others_linear_ts_sdiff2 = diff(total_data_others_linear_ts_sdiff1, differences = 1)
plot(total_data_others_linear_ts_sdiff2, col = "green", main = "Sales for the period from 2009 to 2015:
```



Step 8: Check ACF and PACF to explore remaining dependencies

```
acfo_s1d2 = acf(total_data_others_linear_ts_sdiff2, lag.max =120, plot = FALSE)
pacfo_s1d2 = pacf(total_data_others_linear_ts_sdiff2, lag.max =120, plot = FALSE)
par(mfrow = c(1, 2), bg = "white")
plot(acfo_s1d2, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "ACF")
plot(pacfo_s1d2, col = "green", sub = "Series Growth", xlab = "Lag", ylab = "PACF")
```



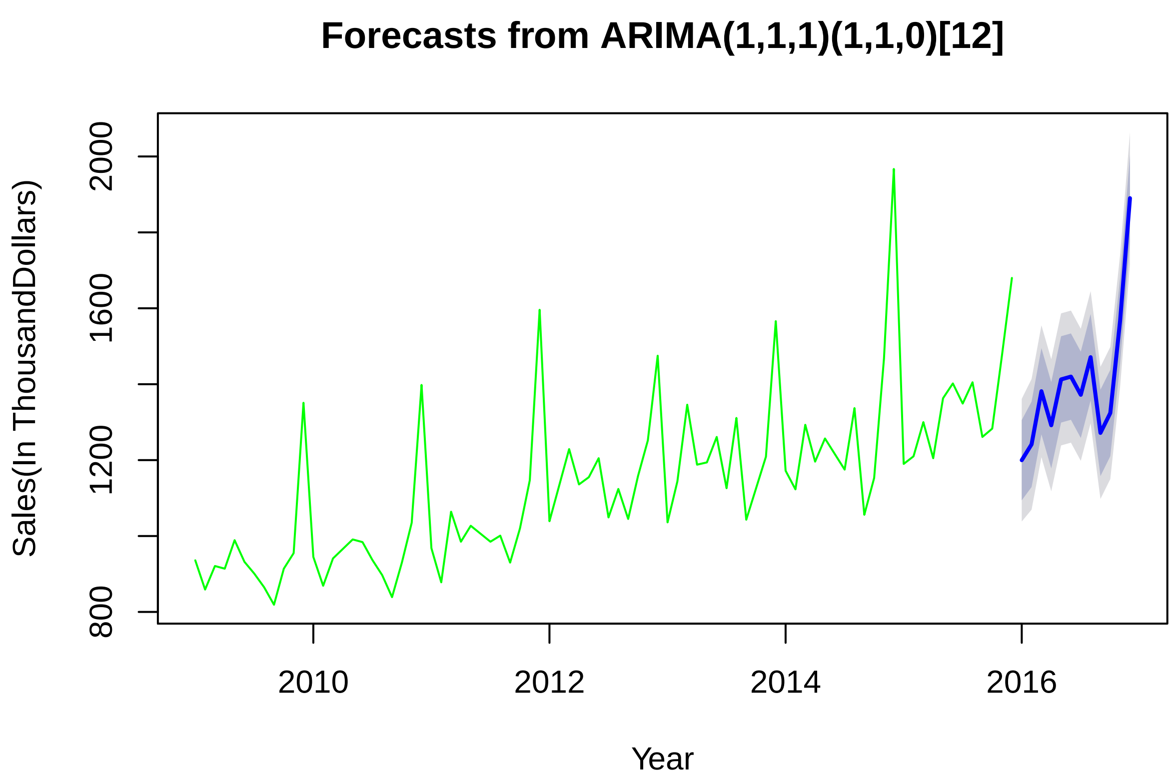
Step 9: ACF and PACF shows that we need to use an AR(1) and an MA(1) term and a positive seasonal term.

```
sales_others_arima = Arima(total_data_others_linear_ts, order = c(1,1,1), seasonal = c(1,1,0), include.
summary(sales_others_arima)
```

```
## Series: total_data_others_linear_ts
## ARIMA(1,1,1)(1,1,0)[12]
##
## Coefficients:
##          ar1      ma1     sar1
##       0.3513  -1.0000  -0.4053
## s.e.  0.1208   0.0552   0.1195
##
## sigma^2 estimated as 6696:  log likelihood=-415.09
## AIC=838.17   AICc=838.78   BIC=847.23
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE    MAPE      MASE
## Training set 6.403103 73.62628 48.23506 0.4268211 3.85797 0.5803683
##                     ACF1
## Training set 0.002272278
```

Step 10: Forcasting the sales for others category for the next year

```
forecast_manual_arima_others = forecast(sales_others_arima, h = 12)
plot(forecast_manual_arima_others,col="green", xlab = "Year", ylab = "Sales(In ThousandDollars)")
```



Writing the results into the excel sheet

```
template = read.csv("template.csv", header = TRUE)
```

```
women_results = data.frame(forecast_manual_arima_women$lower[,1])
colnames(women_results) = c("target")
men_results = data.frame(forecast_manual_arima_men$upper[,1])
colnames(men_results) = c("target")
others_results = data.frame(forecast_manual_arima_others$lower[,1])
colnames(others_results) = c("target")
results = bind_rows(women_results,men_results,others_results)
```

```
## Warning in bind_rows_(x, .id): Vectorizing 'ts' elements may not preserve
## their attributes

## Warning in bind_rows_(x, .id): Vectorizing 'ts' elements may not preserve
## their attributes

## Warning in bind_rows_(x, .id): Vectorizing 'ts' elements may not preserve
```

```
## their attributes
```

```r
test_result = cbind(template$Year,template$Month, template$ProductCategory, results)
colnames(test_result) = c("Year","Month","ProductCategory","target")
write.csv(x = test_result, file = "prediction.csv", row.names = FALSE )
```